# Arquitetura de Software

Este documento destina-se a descrever a arquitetura de software utilizada no aplicativo “My Health”, sendo dividido em três seções, a primeira, descreve a arquitetura utilizado para o desenvolvimento do aplicativo móvel, a segunda, a arquitetura utilizada nos serviços de back-end de suporte ao aplicativo, e a terceira seção descreve brevemente a arquitetura de banco de dados da aplicação e o serviço de back-end. Ao final as referências são descritas na quarta seção.

1. Arquitetura do aplicativo móvel

O projeto de arquitetura de software que será utilizado no aplicativo móvel “My Health’ é o MVVM (Model View ViewModel), devido a utilização do tookit (kit de ferramentas) JetPack Compose, que permite simplificar e acelerar o desenvolvimento de interface de usuário nativa no sistema Android, essa arquitetura é um requisito para a utilização dessas ferramentas. Apesar de bastante similar a arquitetura MVC (Model View Controller), o MVVM tem algumas peculiaridades que serão discutidas a seguir:

* 1. Model

O Model (modelo, ou classes de modelo) refere-se as classes de determinam as regras de negócio da aplicação, classes de dados que se adequam as exigências do banco de dados ou qualquer classe para representar alguma informação estática ou dinâmica, como configurações do aplicativo ou informações de banco de dados. São classes independentes utilizadas pelo ViewModel para enviar as informações a View adequada.

* 1. View

A View (visão, ou classes de visão) refere-se as classes responsáveis por desenhar a interface de usuário na tela do aplicativo, elas são usadas exclusivamente para descrever as regras para a construção dos elementos visuais, não contendo qualquer tipo de dados, esses serão trazidas do domínio de dados (as classes modelos) através da ViewModel.

* 1. ViewModel

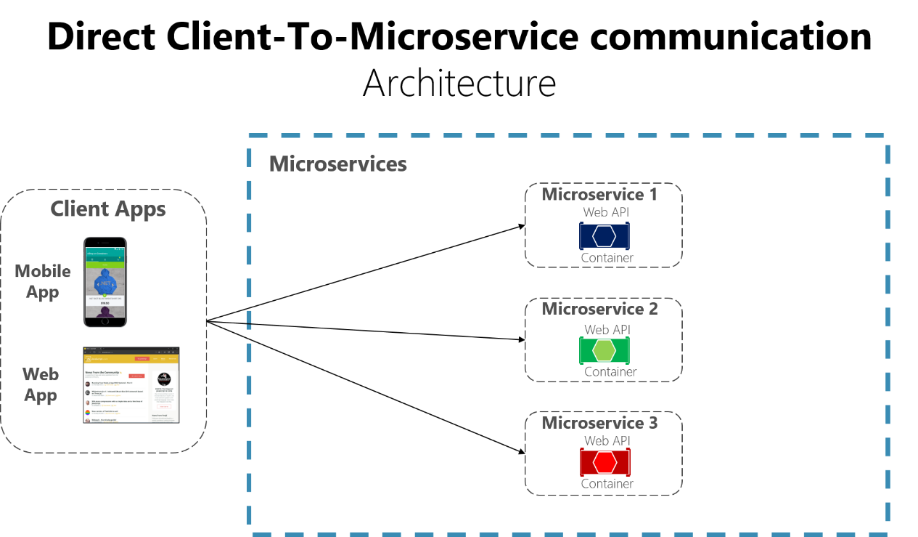
A ViewModel (modelo de visão, ou classes de modelo de visão) são as classes que utilizam e manipulam as classes de modelo para trazer as informações relevantes para as classes de visão exibi-las na interface de usuário. Ela também é responsável pelo gerenciar o estado da interface de usuário e a utilização das regras de negócio para o funcionamento adequado da aplicação. Cada classe de visão deve possui uma classe de modelo de visão correspondente.

Em suma, a arquitetura MVVM pode separada em dois domínios diferentes, o domínio da UI (user interface, ou interface de usuário) e o domínio de dados, as classes de visão pertencem ao domínio da UI, e as classes de modelo ao domínio de dados, as classes de modelo de visão são a interface entre os domínios, essas podem receber requisições das classes de visão para a atualização de componentes específicos, requisitar dados as classes de dados e gerenciar o estado geral do aplicativo. O diagrama abaixo mostra o fluxo de informações da arquitetura.



1. Arquitetura de Back-End

A arquitetura dos serviços de back-end será baseada em microsserviços, utilizando o padrão de comunicação direta de cliente para microsserviço. Cada funcionalidade é separada em um microsserviço independente, como um microsserviço para login de usuários, outro para cadastro de usuários, um outro para pesquisa e assim por diante. A aplicação cliente é pré-configurada para solicitar informações e recursos aos microsserviços adequados. Abaixo um diagrama da arquitetura:



1. Arquitetura de banco de dados.

A arquitetura de banco de dados será do modelo de dados relacional, um banco de dados será composto por diversas tabelas contendo informações altamente coesas, isto é, contendo apenas as informações relevantes a um domínio específico da aplicação, o cruzamento de informações em tabelas diferentes se faz através de chaves estrangeiras que referenciam informações em outras tabelas. Tanto a aplicação móvel como os microsserviços faram uso do banco de dados, com o diferencial de que a aplicação móvel terá uma cópia limitada no armazenamento do dispositivo, a fim de agilizar o processamento dos dados e reduzir a dependência de uma conexão constante com a internet. A sincronização entre a aplicação móvel e os microsserviços acontecem à medida que são detectadas discrepâncias entre os dados locais e remotos.

1. Referências

<https://developer.android.com/jetpack/compose?hl=pt-br>. Acesso 15/05/2023

<https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/>. Acesso 15/05/2023

<https://learn.microsoft.com/pt-br/dotnet/architecture/microservices/architect-microservice-container-applications/direct-client-to-microservice-communication-versus-the-api-gateway-pattern>. Acesso 15/05/2023