

|  | **Projeto de Desenvolvimento de Software - Projetão** |
| --- | --- |
| Nome do Projeto **Shaypado** | |
| 06/12/2023 | **Apresentação da Arquitetura** |

Mobile

Arquitetura:

A arquitetura MVVM (Model-View-ViewModel) é amplamente recomendada pela Google para o desenvolvimento de aplicativos Android devido à sua eficiência e separação de responsabilidades. Nesse padrão, a lógica de apresentação reside na ViewModel, promovendo um código mais limpo e modular. Os dados são gerenciados na ViewModel, expondo-os à camada de visualização por meio do LiveData. Essa abordagem facilita a atualização dinâmica da interface do usuário, mantendo-a sincronizada com as alterações nos dados. Além disso, a MVVM simplifica os testes unitários, melhorando a robustez do código. Ao seguir essa recomendação, criaremos um app mais escalável, mantendo uma estrutura organizada e de fácil manutenção.

Riscos:

* Utilizaremos a biblioteca Jetpack Compose para a View, como não temos experiência com ela, podemos encontrar contratempos.
* Maria Virgínia não possui experiência com desenvolvimento mobile nativo.
* O uso da câmera para o BodyFat pode envolver o uso de recursos nativos avançados no Android.
* Terão animações diversas no uso de pets, isso pode gerar imprevistos.

Métricas

* Telas entregues
* Percentual feitas pela api que estão devidamente integradas com o mobile

Back-End:

A escolha arquitetura para o back foi a de MVC (Model-Controller) sem Views, combinada com design patterns, ela foi baseada no tamanho do projeto, nas necessidades específicas do sistema, e na experiência da equipe de desenvolvimento. Para o projeto esse tipo de arquitetura facilita a manutenção e a escalabilidade ao separar a lógica de negócios (Controller) da lógica de acesso a dados (Model) e também cada parte do sistema tem uma responsabilidade clara, o que simplifica a compreensão e a manutenção do código. A integração de design patterns específicos (como Singleton, Factory, Strategy, etc.) dentro do projeto o torna mais estruturada e organizada, fazendo assim uma melhora da eficiência do projeto, além de até mesmo reduzir repetição de código e aumentar a flexibilidade.

Riscos:

* Pode ser mais complexo para configurar e criar a estruturar inicial do projeto
* Requer um entendimento claro da separação de responsabilidades e do funcionamento dos design patterns.
* Há o risco de sobre-engenharia, onde a aplicação dos design patterns pode tornar o código mais complexo do que o necessário para a funcionalidade desejada.
* Dependendo de como o MVC for implementado dentro do sistema, pode haver uma dependência forte em frameworks específicos, o que pode limitar a flexibilidade em algumas situações.

Métricas:

* Funcionalidades escritas e entregas que sigam as diretrizes de padrão de código em FastAPI e da arquitetura do sistema
* Cada funcionalidade possui pelo menos um teste unitário.