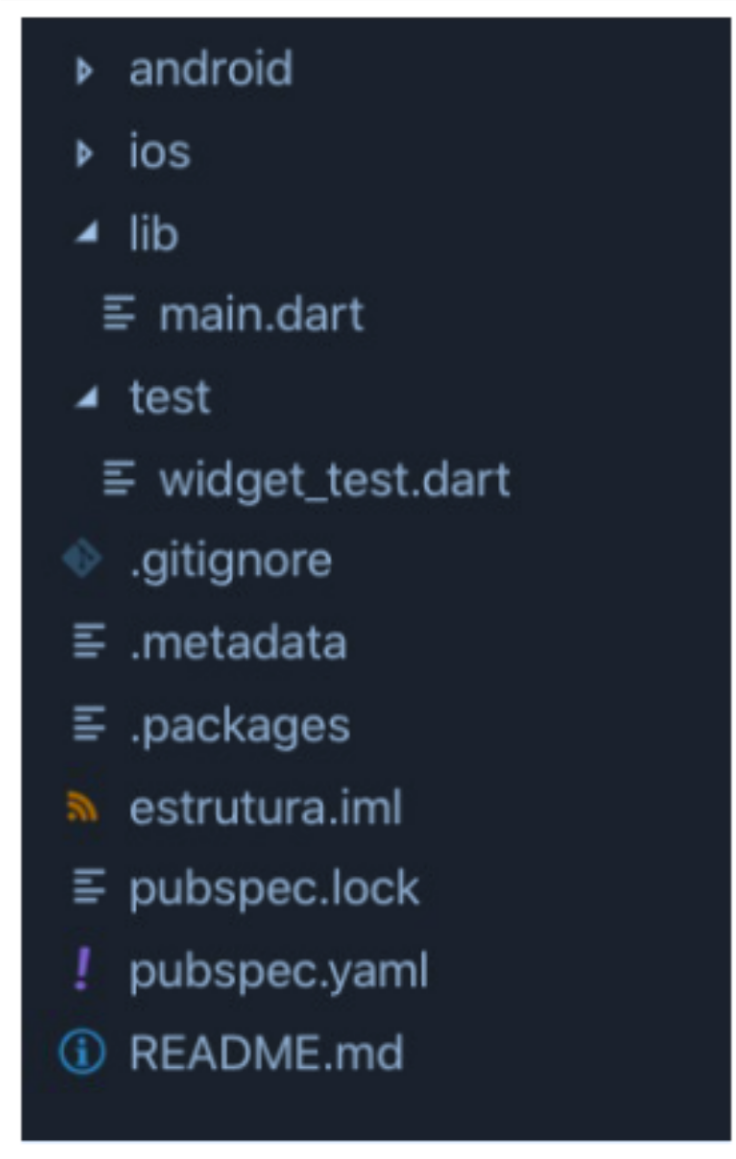
ESTRUTURA DE ARQUIVOS



android: nessa pasta, temos o projeto Android nativo, criado normalmente em Java. Quando o código Flutter for executado, ele automaticamente é compilado para Java e executado a partir dessa pasta.

Apesar de ser bom ter conhecimento na linguagem Java e Kotlin, não é obrigatório. Não precisamos obrigatoriamente conhecer as linguagens nativas, afinal, o Flutter está aí para isso! A seguir temos uma imagem do conteúdo da pasta android :

iOS: nessa pasta, temos o projeto iOS nativo, criado em Swift. Quando o código Flutter for executado ele automaticamente é compilado para esta linguagem e executado a partir da pasta iOS .

Assim como foi citado a respeito do Java e Kotlin, não é obrigatório ter conhecimento do Swift para criar seus aplicativos Flutter.

lib: nessa pasta, temos por padrão o arquivo main.dart , nele está o código que inicia a nossa aplicação. Dentro da pasta lib podemos criar outras pastas e organizar os arquivos escritos em Dart da forma que acharmos mais organizada e justa para o correto funcionamento do aplicativo.

Você pode brincar à vontade dentro dessa pasta, só não pode substituir o nome do arquivo main nem remover o método main de dentro dele, se não o aplicativo não executará.

test: nessa pasta, temos por padrão o arquivo Widget_test.dart , onde está o código desenvolvido especialmente para testar o funcionamento dos Widgets e interação com o usuário. A aplicação default que temos quando criamos um projeto Flutter já vem com os testes configurados para ela. Os testes vindos de presente no aplicativo padrão verificam se os Widgets especificados existem, se o contador está zerado e se o botão flutuante vai acrescer mais 1 no valor exibido na tela.

.gitignore: é nesse arquivo que nos comunicamos com o controle de versão, o que especificarmos no gitignore não será versionado. Não precisamos salvar no nosso repositório de códigos as pastas com as dependências que podem ser tranquilamente baixadas posteriormente, assim, economizamos espaço no disco do servidor responsável pelo repositório e tornamos muito mais rápido o envio e recebimento dos arquivos. Com o gitignore gerado pelo Flutter apenas os códigos Dart e coisas importantes do Flutter são salvos no repositório; arquivos de compilação, temporários e afins não entram no bolo.

.metadata: o arquivo de metadados mantém algumas propriedades específicas do nosso projeto Flutter, e ele é responsável por fornecer os dados para atualizações do framework, extensões e similares. É bastante importante versionar este arquivo (ou seja, não o colocar no gitignore) e principalmente não mexer em seu conteúdo, afinal, ele é autogerado e administrado pelo SDK Flutter.

.packages: é aqui que o SDK Flutter salva as urls das dependências que ele necessita mais frequentemente. Basicamente é um arquivo que indexa todos os arquivos principais de funcionamento do Flutter para caso houver a necessidade de executá-los não termos problemas com performance e demora no tempo de execução.

pubspec.yaml: podemos dizer que o arquivo pubspec é o coração das dependências e controle dos nossos aplicativos. Nele especificamos quais dependências/extensões queremos utilizar (http, lazy load, bloc), o nome do projeto e descrição, versão do SDK Flutter que queremos utilizar, o diretório dos nossos arquivos assets, isto é, nossos arquivos de fontes, imagens, vídeos e afins. Uma dica de ouro para se dar muito bem com esse arquivo é ler todos os comentários que vêm por padrão nele explicando cada funcionalidade do arquivo.

pubspec.lock: é efetivamente o arquivo que o Flutter lerá. Basicamente, ele é um compilado do pubspec.yaml só que de forma otimizada para a fácil leitura pelo SDK. O arquivo lock é gerado na primeira vez que você cria o projeto, e só é atualizado caso você o exclua e execute novamente a compilação. Ele é responsável por manter a compatibilidade das dependências.

README.md: como a extensão do próprio já diz, ele é escrito utilizando a linguagem de marcação Markdown. Este arquivo não é essencial para o funcionamento do projeto, é apenas um descritivo para você criar anotações sobre o aplicativo e dicas como as de instalação, por exemplo, caso outras pessoas possam baixar o projeto. Sinta-se livre para anotar o que achar pertinente neste arquivo, ou, até mesmo removê-lo.

estrutura.iml: se você não criou o seu projeto com o nome estrutura com certeza esse arquivo seu terá o nome do seu projeto com a extensão iml . Basicamente é um facilitador para o Dart se comunicar com o interpretador Java na hora de gerar o aplicativo para o android. Algumas versões do Flutter simplesmente não geram este arquivo. Caso o seu projeto não tenha, não se preocupe. Assim como o arquivo metadata, ele é autogerado e não deve ser editado manualmente.