# Teste Maximatech

**Resumo do teste**

**Dificuldades encontradas**

- Compreender as regras de negócio da empresa através do json
- Alguns ícones não disponíveis, tendo que baixar pelo androidStudio (SVGs)
- Alinhamento de itemView um pouco complicado (itemView um do lado do outro sem comprometer a visualização para outros devices com telas pequenas - layout não ficando responsivo)
- Alguns layouts não ficaram responsivo, sendo assim, não adaptando para telas de tamanho menor ou igual que 3 polegadas
- Configuraçao de banco de dados
    - Resolvi utilizar o RoomDB, mas fiquei entravado com algumas coisas simples, como relacionamento e atributo como lista (RoomDB exige que seja arrayList em alguns casos e da problema de build quando é interface list)
- A documentação não deixou claro que “horas” seria para pegar para mostrar no snackBar. se do pedido ou do device. Se for do pedido, deveria ser executado o serviço também nessa tela de dados do cliente quando clicasse no botão.
- Dificuldade em integrar o navigation component com o bottomNavigation (é a forma mais correta). Tive que fazer de uma forma (replace fragments) menos regular que apenas usar navigation com o bottomNavigation. Tenho experiência com NavigationComponent, mas com bottomNavigation nunca cheguei a mexer, só dar manutenção.
- Algumas frases estão em inglês e outras em PTBR. Deveria ser mono linguagem.
- Alguns adapters ficaram um pouco acoplado pelo grande nível de detalhes em cada tela. Não gerenciei da melhor forma para evitar esse acoplamento.
- Perdi muito tempo com construção de layout. Gostaria de deixar tudo responsivo.
- Não consegui salvar legendas no BD, tive que retirar ela

Sobre arquitetura:

- Modularização
  - Resolvi fazer modularizado para isolar a camada em módulos de database, network e baseTest do APP. Motivo de ter feito isso é que acredito que o teste tem como base fazer algum comportamento em algum aplicativo já existente da maximatech. Imagino que o projeto seja grande e por isso multimodulos (microfrontEnd - separar por modulos) é uma arquitetura que desacopla, reaproveita e organiza melhor o código/projeto. Exemplo de reaproveitação: Poderíamos ter um módulo que está em um repositório (projeto multirepo) e que é usado em 2 projetos, maximatechClient e maximatechMotorista. Com o multimodulo e esse modulo em um repositório (fazendo também uma library desse repositório e subindo no maven), consigo reaproveitar muito o código para mais de um projeto
- MVVM
  - Desacoplamento de camadas e maior facilidade para aplicar testes unitarios, visto que quanto mais desacoplado, mais facilmente é feito o teste.
  - Aplicado camada de domain, mas sem useCase, pois nenhuma regra de negócio foi declarada na documentação.
  - Camada de domain para ter a interface de repository e os modelos (mappers) para a camada de data(repositorio e mappers (dataClass)) e presenter (mappers(dataClass)) usufruir
- Testes
  - Testes feitos unitáriamente para cada viewModel e testes instrumentados unitario (testando metódo) de legendas, onde retorna a imagem para aquela legenda / critica. Precisa do contexto do Android para rodar, mas ainda continua como unitário, pois está testando apenas um pedaço do código (apenas um metódo)
  - Testes de UI não foram feitos por motivos de falta de conhecimento concreto. Testes de interface é algo que ainda não domino 100%. É importante, mas não domino ainda. Está como meta refinar esse conhecimento.