EstOkay!

O melhor estoque que você pode ter!

Descrição da demanda

Bonde do Godzilla

Descrição do produto:

Administração de estoque e financeiro de forma intuitiva e que seja possivel visualizar atraves do mobile.

Oportunidades e justificativa:

> Controle mais intuitivo do estoque;

> Não existem softwares intuitivos e mobile para controle de estoque e lojas em geral.

Benefícios:

> Organização dos produtos disponíveis para o uso;

> Controle de quando vai comprar um produto que está em falta;

> Comunicação entre as áreas da empresa: administrativo, produção, vendas.

Recursos:

> Banco de dados SQL
> REST API para backend
> Frontend web e mobile

Equipe:

- Gabriel Carvalho
- Guilherme Reis
- Maria Carolina Florio
- Vitor Grechi Kuninari

Etapas:

Estudo planejamento e desenvolvimento (CBL)

Riscos:

Custo do produto e concorrentes.

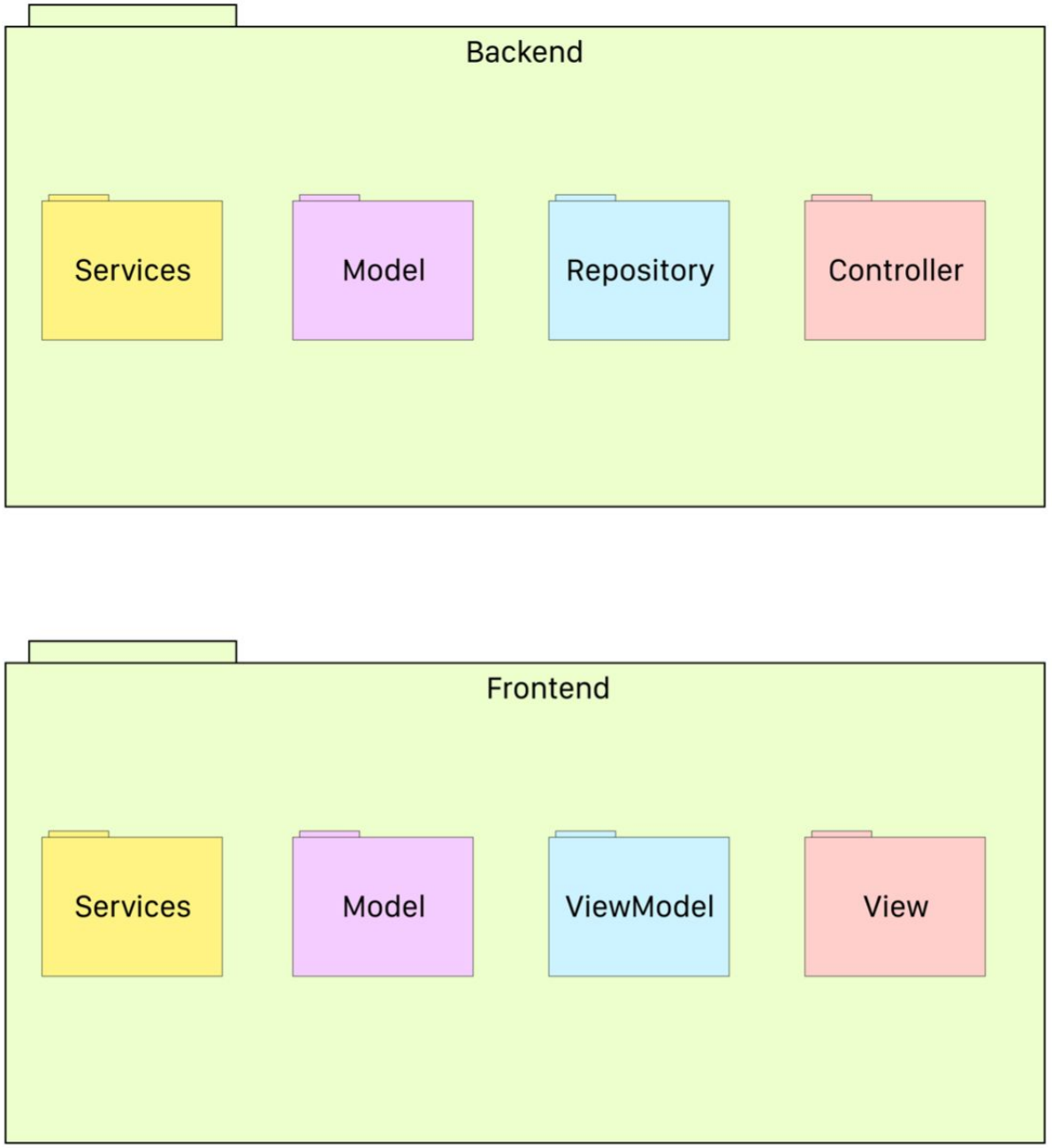
Critérios de qualidade:

Usabilidade, estabilidade e otimização

Clientes:

A parte administrativa e de produção de empresas voltadas para o comercio

Modelagem conceitual


Backend
Services
Model
Repository
Controller
Frontend
Services
Model
ViewModel
View

Personas

- João: Gerente de estoque
- Maria: Dona da loja

RF1 - O sistemas deve permitir adicionar novos itens

Como João, quero adicionar novos itens que chegam na loja, para que os novos itens da loja estejam sempre cadastrados.

RF2 - O sistema deve permitir fazer alterações nos itens: quantidade, preço, nome e etc.

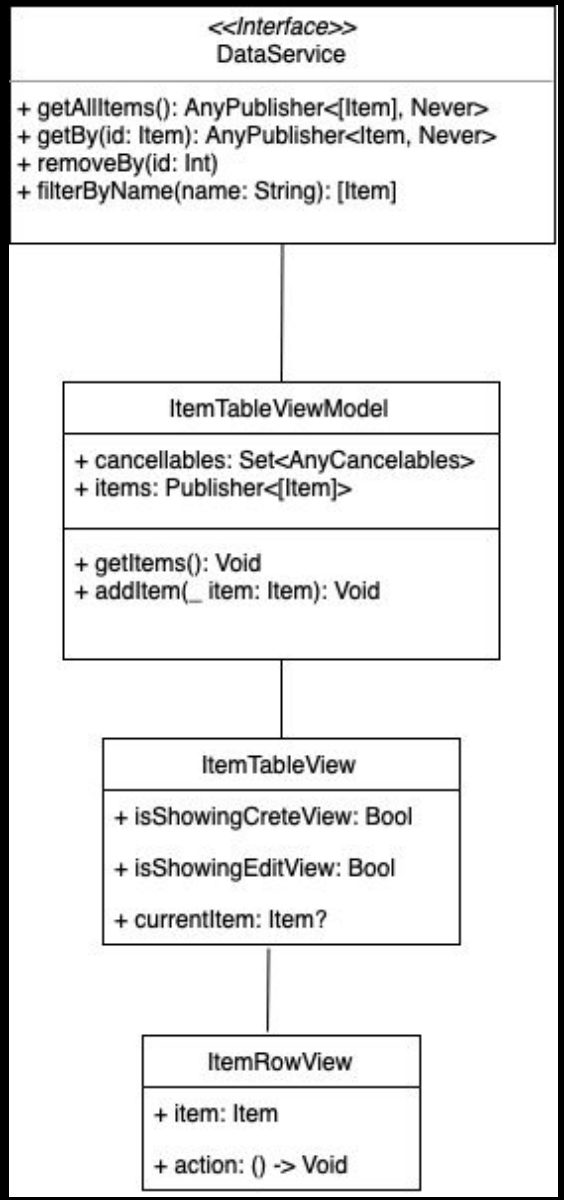
Como João, quero alterar quantidades, preços e nomes para manter o estoque dos produtos atualizados.

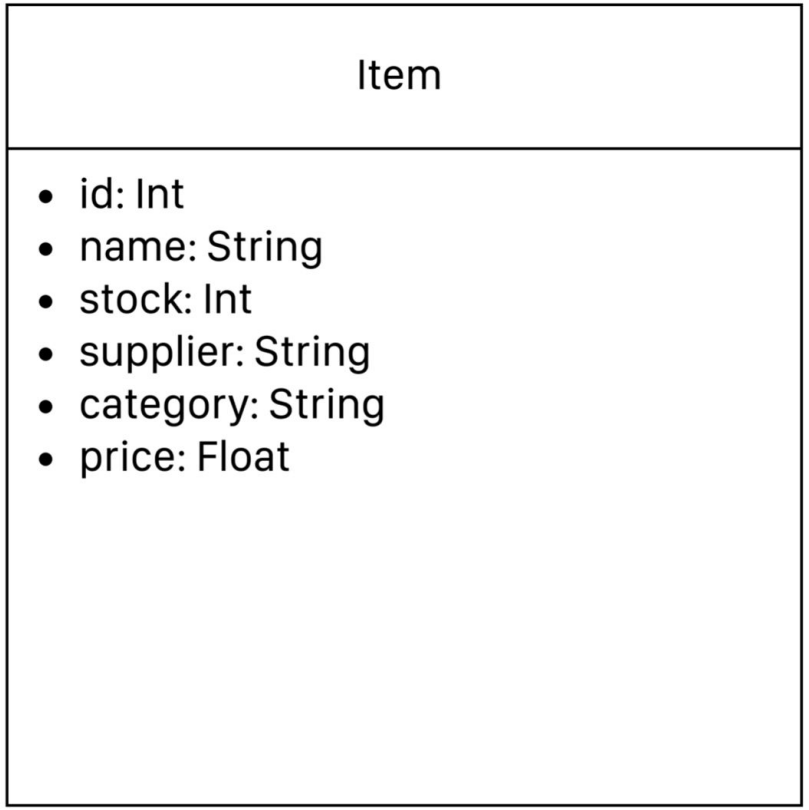
RF4 - O sistema deve permitir excluir produtos do estoque

Como João, quero excluir produtos do estoque para manter apenas os produtos que desejamos vender dentro de nossa loja.

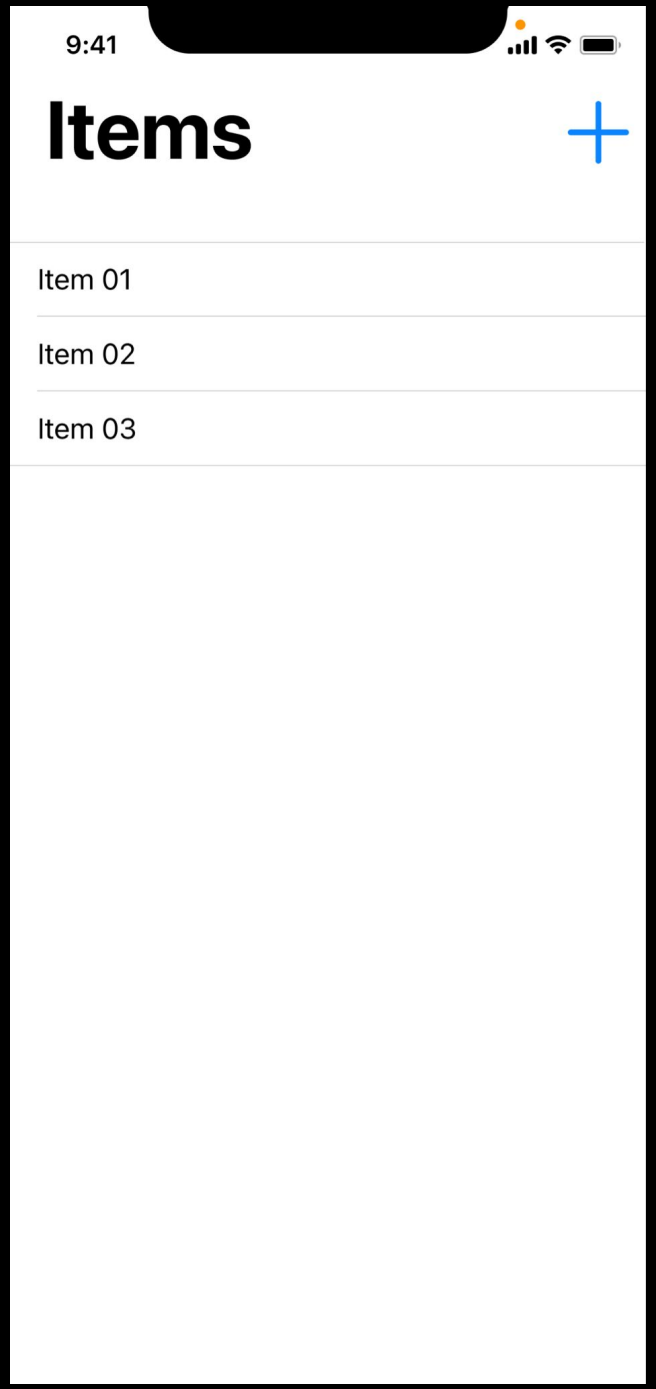
RNF1 - O sistema deve exibir os produtos do estoque.

Como Maria, quero ver os produtos que estão no estoque da minha loja para ter controle da minha empresa.



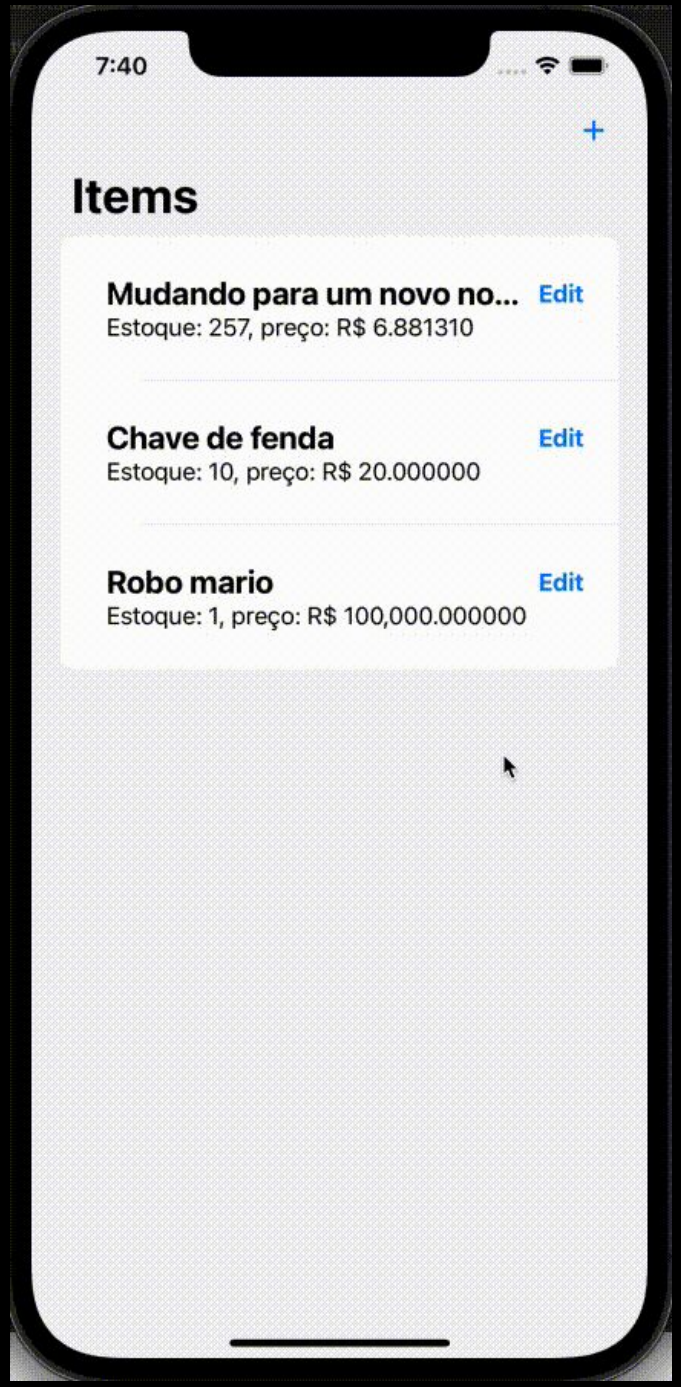

Item
id: Int
name: String
stock: Int
supplier: String
category: String
price: Float

Artefatos de design

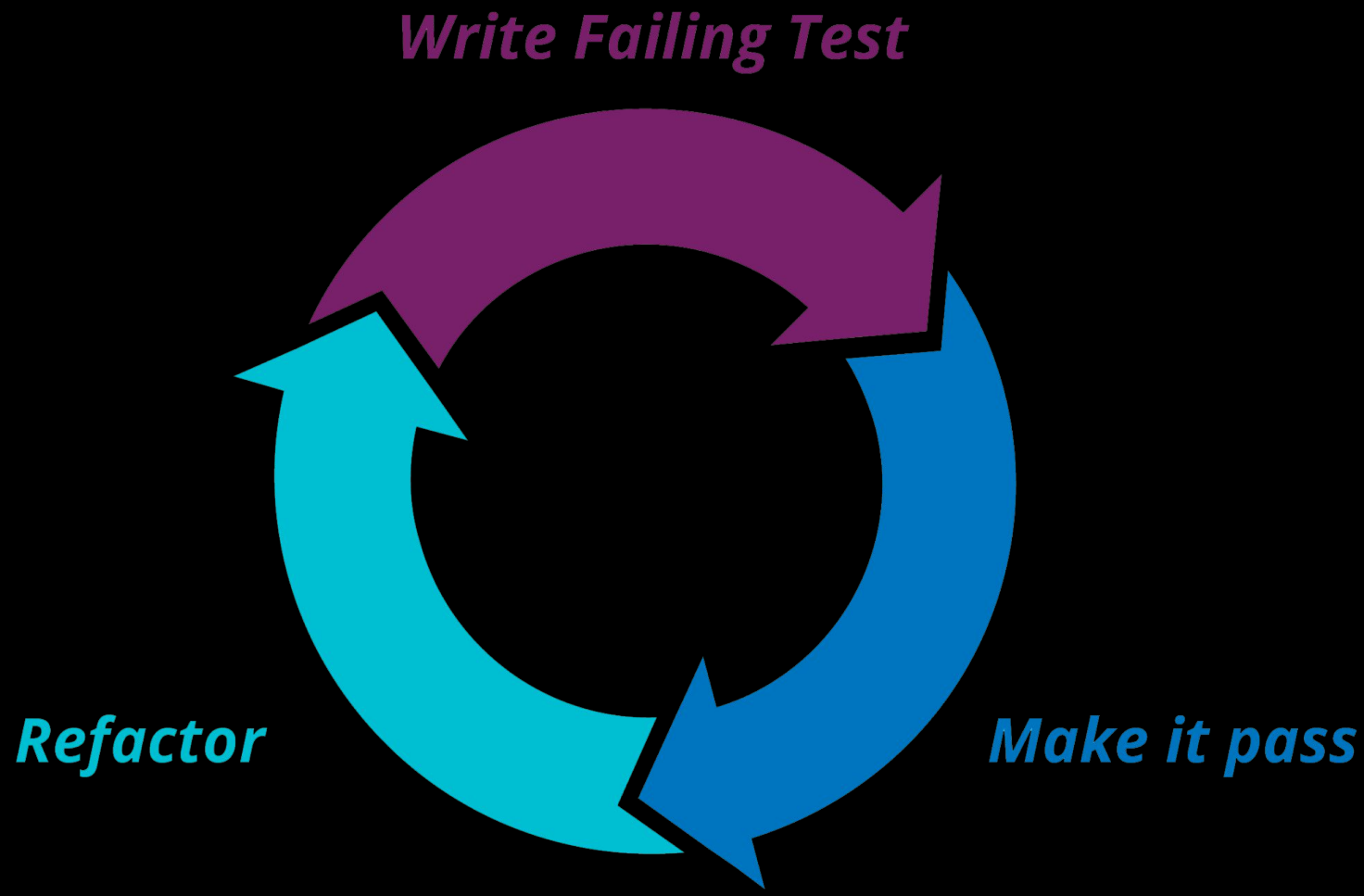

9:41
Items
Item 01
Item 02
Item 03

Código fonte

```swift
import Foundation
import Combine

final class ItemsTableViewModel: ObservableObject {

    private let dataService: DataService = DSMock()
    private var cancellables = Set<AnyCancellable>()

    @Published var items: [Item] = []

    init() {
        getItems()
    }

    private func getItems() {
        dataService
            .getAllItems()
            .assign(to: \.items, on: self)
            .store(in: &cancellables)
    }

    public func addItem(_ item: Item) {
        items.append(item)
    }

    public func replaceItem(in index: Int, for item: Item) {
        guard index < items.count , index >= 0 else {
            print(index)
            return
        }
        items[index] = item
    }

}
```


7:40
Items
Mudando para um novo no...
Edit
Estoque: 257, preço: R$ 6.881310
Chave de fenda
Edit
Estoque: 10, preço: R$ 20.000000
Robo mario
Edit
Estoque: 1, preço: R$ 100,000.000000

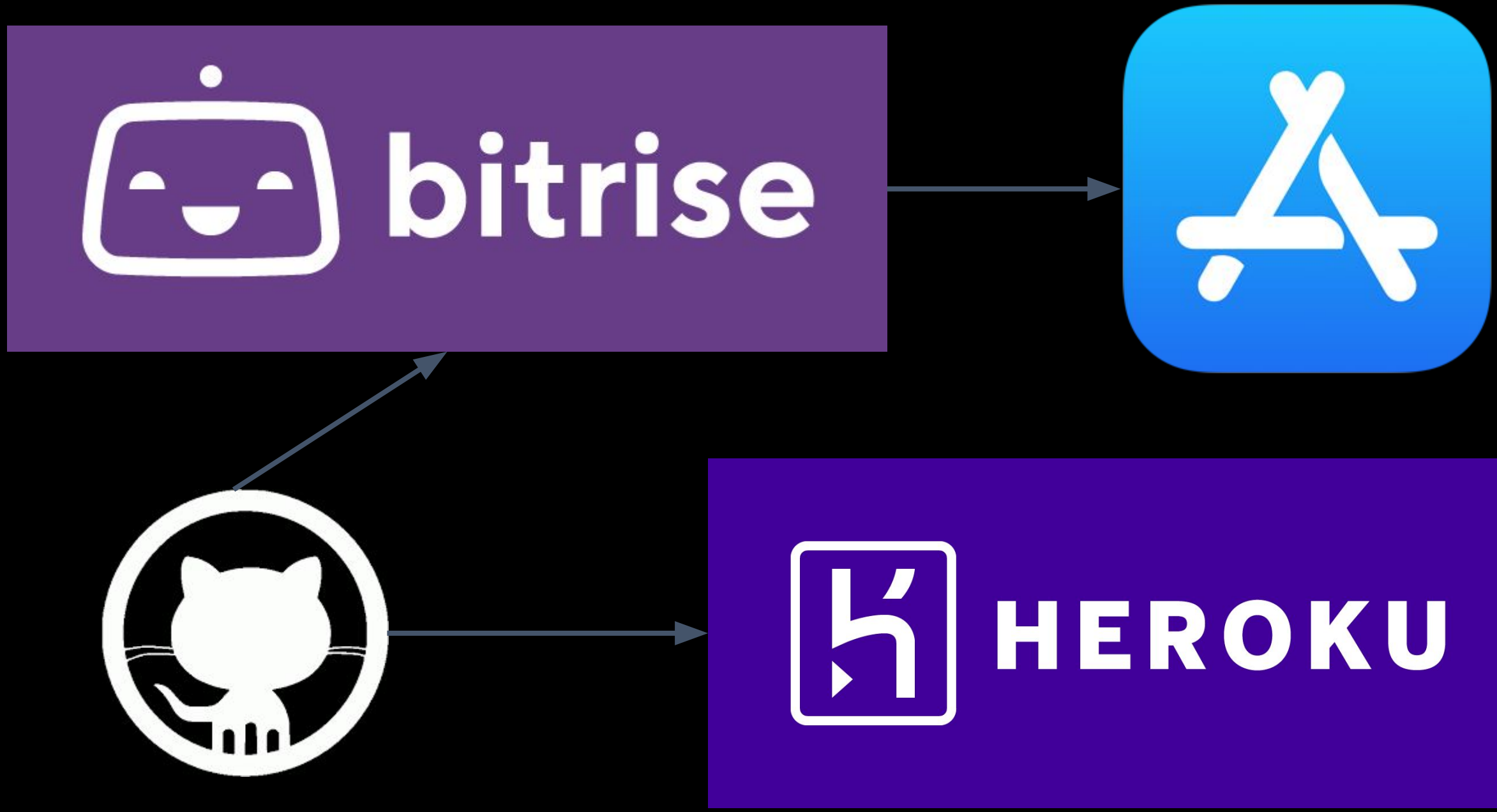
Casos de teste


Write Failing Test
Make it pass
Refactor

Lista de testes:

- getItem():
  - Um objeto da classe DSMock para passar dados "mocados" para a classe;
  - É esperado um retorno de uma lista específica.
- addItem():
  - É esperado um retorno da lista "mocada" mais o item adicionado.

Plano de Implantação


bitrise
HEROKU

$$$

Heroku: 25$ por mês
AppStore: 100$ por ano
Bitrise: 90$ por mês

Como a engenharia de software impactou o projeto?