Documentação do Sistema de Notificações (Swift)

# 📌 Visão Geral

Este projeto Swift implementa um sistema de envio de notificações com suporte a múltiplos canais: Email, SMS e Push Notification. O objetivo é permitir que diferentes formas de comunicação sejam tratadas de maneira uniforme por meio de um protocolo comum. O programa também permite que o usuário selecione prioridades e tipos de mensagens, agrupando e enviando notificações conforme sua importância.

# 📘 Protocolo Notificavel

O protocolo Notificavel define a estrutura base para qualquer tipo de notificação. Ele exige que cada notificação tenha uma mensagem, uma prioridade e implemente a função enviarNotificacao(). Isso permite o uso de polimorfismo para tratar todos os canais de forma unificada.

# 🏗️ Estruturas de Notificações

Foram implementadas três structs que seguem o protocolo Notificavel:

* - Email: representa o envio de mensagens via email.
* - SMS: representa o envio de mensagens via SMS.
* - PushNotification: representa notificações enviadas para dispositivos móveis.

# 🔠 Enumerações

Foram criadas duas enumerações para padronizar os dados do sistema:

* - Prioridade: baixa, média e alta.
* - TipoMensagem: promoção, lembrete e alerta.

# 📨 Struct Mensagem

A struct Mensagem representa o conteúdo de uma notificação, contendo seu tipo (usando TipoMensagem) e o texto da mensagem.

# 🛠️ Funções Auxiliares

- escolherPrioridade(): permite que o usuário defina a prioridade de uma notificação.

- escolherTipoMensagem(): permite que o usuário defina o tipo de mensagem.

- agruparPorPrioridade(): agrupa as notificações por prioridade para envio ordenado.

# 📋 Fluxo do Programa

O programa é executado em um loop interativo no terminal, onde o usuário pode adicionar notificações via email, SMS ou push, definir suas características, e posteriormente enviar todas agrupadas por prioridade. Notificações com prioridade alta são enviadas antes das médias e baixas.

# 🧠 Decisões de Design

- Uso de protocolo para garantir extensibilidade e reutilização.

- Enums para evitar valores inconsistentes e melhorar a legibilidade.

- Estrutura modular e funções separadas para responsabilidade única.

- Interface interativa via terminal para facilitar testes e simulações.