Aplicativo para iOS - Detecção de Queimadas

# 1. Introdução

Este documento descreve o planejamento e o cronograma de desenvolvimento de um aplicativo iOS baseado em WebView, destinado à visualização de uma interface web para análise de imagens de satélite com foco na detecção de queimadas em áreas rurais. O objetivo principal é a criação de um aplicativo demonstrativo para uso em iPhones e iPads reais, sem necessidade de publicação oficial na App Store.

# 2. Estrutura da Equipe e Metodologia

O desenvolvimento será conduzido com base na metodologia ágil Scrum, com equipe composta por:  
  
- Product Owner  
- Scrum Master  
- Desenvolvedores iOS (6 membros no total)  
  
O projeto está previsto para ser concluído em 8 semanas, com entregas semanais (sprints).

# 3. Requisitos do Ambiente de Desenvolvimento iOS

Para o desenvolvimento e testes do aplicativo WebView para iOS, serão necessários os seguintes recursos:  
  
- Sistema operacional: macOS (Monterey ou superior)  
- Xcode (versão 14 ou superior)  
- Linguagem: Swift  
- Dispositivos de teste: iPhone (iOS 15 ou superior), iPad (iPadOS 15 ou superior)  
- Conta Apple ID (não é necessária conta de desenvolvedor paga para testes locais)  
- Cabos USB ou pareamento com Wi-Fi para instalação via Xcode  
- Provisionamento automático com Apple ID gratuito  
- Interface web hospedada em servidor com HTTPS (obrigatório para WebView no iOS)

# 4. Cronograma de Desenvolvimento (Sprints Semanais)

## Semana 1

Tarefa: Planejamento e Levantamento Técnico  
Início: 15/04/2025  
Término: 21/04/2025

## Semana 2

Tarefa: Protótipo da Interface Web  
Início: 22/04/2025  
Término: 28/04/2025

## Semana 3

Tarefa: Início do App Android e Backend  
Início: 29/04/2025  
Término: 05/05/2025

## Semana 4

Tarefa: Funcionalidade mínima da Web + App básico  
Início: 06/05/2025  
Término: 12/05/2025

## Semana 5

Tarefa: Integração de Satélite + Filtros  
Início: 13/05/2025  
Término: 19/05/2025

## Semana 6

Tarefa: Funcionalidades avançadas + Notificações  
Início: 20/05/2025  
Término: 26/05/2025

## Semana 7

Tarefa: Testes e Ajustes  
Início: 27/05/2025  
Término: 02/06/2025

## Semana 8

Tarefa: Publicação e Documentação  
Início: 03/06/2025  
Término: 09/06/2025

## Semana 9

Tarefa: Testes finais  
Início: 10/06/2025  
Término: 16/06/2025

## Semana 10

Tarefa: Apresentação final  
Início: 17/06/2025  
Término: 23/06/2025

# 5. Entregáveis por Fase

MVP até semana 4:  
- Aplicativo funcional em WebView carregando a interface web em dispositivos reais (iPhone/iPad).  
- Interface web responsiva com mapas e imagens de satélite.  
- Backend com dados simulados.  
  
Versão final até semana 9:  
- Aplicativo com melhorias de navegação, splash screen, suporte a push notification (simulado).  
- Integração real com imagens de satélite.  
- Testes em múltiplos dispositivos (iPhone e iPad).  
- Documento final de apresentação e preparação para versão pública (opcional).

# 6. Workflow Híbrido para Desenvolvimento Flutter/React Native

É possível desenvolver aplicações móveis com WebView para iOS e Android utilizando um workflow híbrido, combinando um ambiente local em Windows ou Linux com serviços de compilação remota para gerar o aplicativo iOS. Abaixo estão os passos recomendados para esse tipo de desenvolvimento.

## 6.1 Workflow com Flutter

1. Instalar Flutter SDK no ambiente local (Windows/Linux).  
2. Criar projeto Flutter com o plugin `webview\_flutter`.  
3. Desenvolver e testar o app localmente usando o emulador Android ou dispositivo físico.  
4. Realizar versionamento contínuo no GitHub ou GitLab.  
5. Integrar repositório ao serviço de build remoto, como Codemagic ou Bitrise.  
6. Configurar o pipeline de build para gerar o `.ipa` com Apple ID remoto (ou chave de desenvolvedor provisionada).  
7. Fazer download do `.ipa` e instalar em dispositivos iOS reais via TestFlight ou sideload (usando AltStore, diawi, etc.).

## 6.2 Workflow com React Native

1. Instalar Node.js, React Native CLI e Android SDK no ambiente local.  
2. Criar projeto React Native utilizando o componente `WebView` da comunidade (`react-native-webview`).  
3. Testar no Android localmente via emulador ou dispositivo.  
4. Subir o projeto para um repositório remoto.  
5. Conectar o repositório ao AppCenter, Bitrise ou EAS Build (Expo).  
6. Configurar as credenciais de build iOS e definir os parâmetros da build pipeline.  
7. Compilar e baixar o aplicativo `.ipa` para instalação ou distribuição via TestFlight.

## 6.3 Considerações Finais

- A interface web deve estar publicada em servidor HTTPS, pois o iOS bloqueia conexões não seguras por padrão.  
- Durante o desenvolvimento, é possível validar 100% da lógica da WebView no Android, replicando para o iOS posteriormente.  
- A compilação para iOS requer obrigatoriamente um ambiente macOS, mesmo usando automação.  
- O uso de Flutter ou React Native proporciona portabilidade e maior produtividade entre as plataformas.

# 7. Utilização de VM Remota com macOS

Para desenvolvedores que não possuem um Mac físico, é possível utilizar serviços de virtualização em nuvem com macOS para realizar as etapas finais do processo de desenvolvimento de aplicativos iOS. Esses serviços permitem acesso remoto a máquinas com macOS, Xcode e ferramentas necessárias pré-instaladas.

## 7.1 Principais Finalidades

As VMs com macOS remoto permitem:  
- Compilar e assinar o aplicativo iOS com Xcode.  
- Rodar testes funcionais e visuais no simulador iOS.  
- Instalar o aplicativo diretamente em iPhones ou iPads reais via cabo USB (quando o serviço permite pareamento físico ou remoto).  
- Exportar o aplicativo como arquivo `.ipa` sem a necessidade de publicar na App Store.

## 7.2 Serviços Recomendados

- \*\*MacStadium\*\*: Acesso completo a VMs com macOS dedicado. Permite Xcode, conexão USB e automação.  
- \*\*MacInCloud\*\*: Mais acessível, com opções de planos mensais ou por hora. Acesso ao Xcode pronto para build.  
- \*\*AWS EC2 Mac Instances\*\*: Nível empresarial, com instâncias macOS em infraestrutura da Amazon.

## 7.3 Etapas de Uso

1. Contratar um serviço com acesso remoto a macOS.  
2. Conectar-se via VNC ou navegador remoto.  
3. Clonar o repositório do app Flutter ou React Native.  
4. Abrir o projeto no Xcode.  
5. Configurar o bundle ID, certificados e provisões (Apple ID gratuito ou conta paga).  
6. Rodar testes no simulador iOS.  
7. Gerar o `.ipa` e instalar localmente ou exportar para envio por TestFlight ou sideload (AltStore, diawi etc.).