# Especificações UML

Henrique Lopes UP202308657
José Magalhães UP201809097

Abril 2024

## Introdução à UML

A UML (Unified Modeling Language ou Linguagem de Modelagem Unificada) é uma linguagem de modelagem padronizada que é usada para visualizar, especificar, construir e documentar os artefatos de sistemas de software. Ela foi desenvolvida para ajudar os engenheiros de software a definir as características do software, as interações entre os componentes do sistema, e outros aspectos importantes do que está a ser construído. Em resumo, a UML é essencial para a modelagem e documentação de sistemas de software, atuando como uma linguagem unificadora entre os diversos aspectos e stakeholders do projeto de software. Ela ajuda a equipa a visualizar o design e a funcionar eficientemente, garantindo que o produto final seja robusto, eficaz e alinhado com os requisitos iniciais. Em resumo, a UML não é apenas uma ferramenta para visualizar estruturas de software; ela é essencial para gerenciar a complexidade, promover a colaboração interdisciplinar, e garantir a qualidade e a confiabilidade de aplicativos complexos como o SafeGuard. A adoção da UML neste contexto não apenas melhora o processo de desenvolvimento, mas também ajuda a garantir que a aplicação final seja seguro, confiável e eficaz.

# Tipos de Diagramas UML Usados

2.1 Diagrama de Casos de Uso

Mostrar como os usuários (atores) interagem com o sistema. Incluir casos de uso como "Receber Alerta de Desastre", "Configurar Preferências de Alerta", e "Acessar Informações de Evacuação".

2.2 Diagrama de Classes

Especificar as classes, suas atribuições, métodos e relações dentro do sistema. Por exemplo, classes como Sensor, Alerta, Usuário, e Administração de Emergências.

2.3 Diagrama de Sequência

Detalhar a sequência de mensagens trocadas entre objetos para realizar uma função específica. Pode ser usado para descrever o processo de detecção de desastre e emissão de alertas.

2.4 Diagrama de Atividades

Ilustrar o fluxo de atividades para processos de negócio. Por exemplo, o processo de configuração inicial do usuário ou o processo de resposta a um alerta.

2.5 Diagrama de Componentes

Descrever a organização e dependências entre os componentes de software. Incluir componentes como módulos de coleta de dados, processador de machine learning, e sistema de notificação.

2.6 Diagrama de Implantação

Mostrar a configuração física dos artefatos de software em hardware. Muito útil para entender como o sistema é distribuído, especialmente em relação aos servidores e dispositivos dos usuários.

# Diagramas

Diagrama de Casos de Uso:

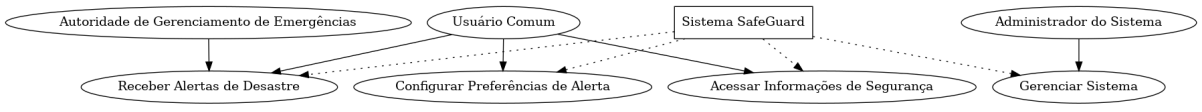


Diagrama de Classes:

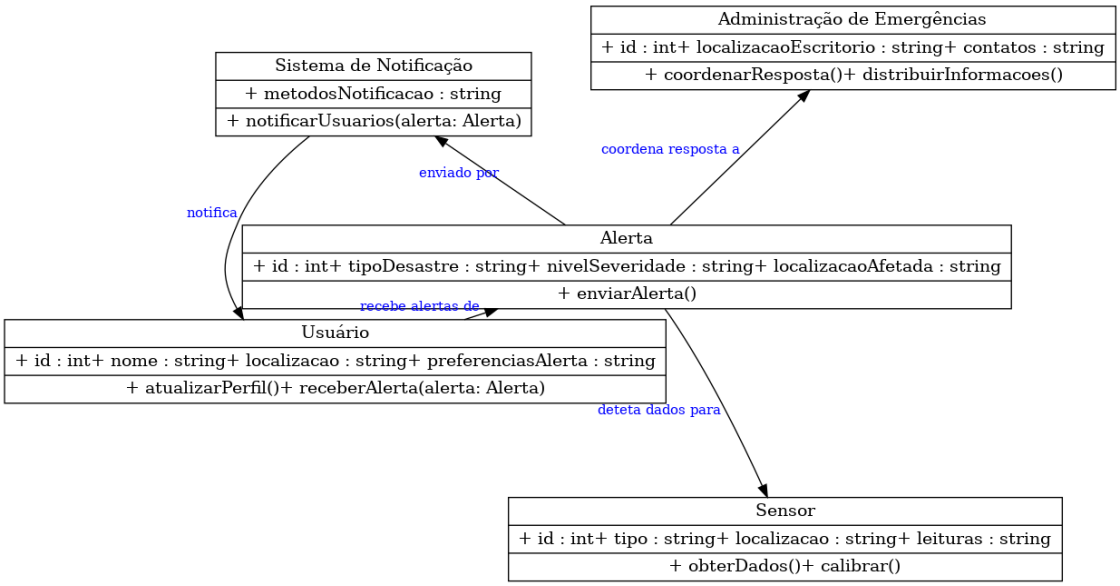



Diagrama de Sequência:

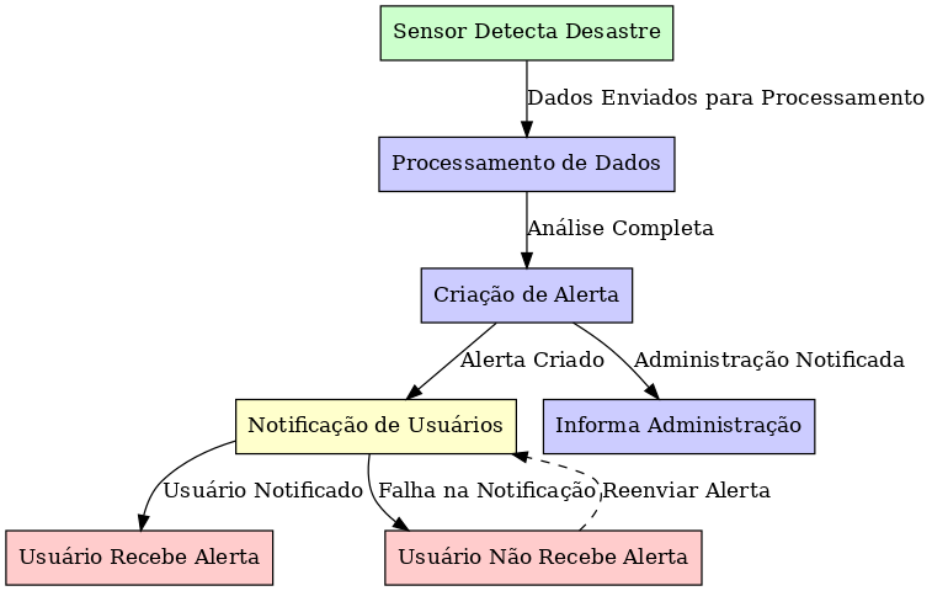


Diagrama de Atividades:

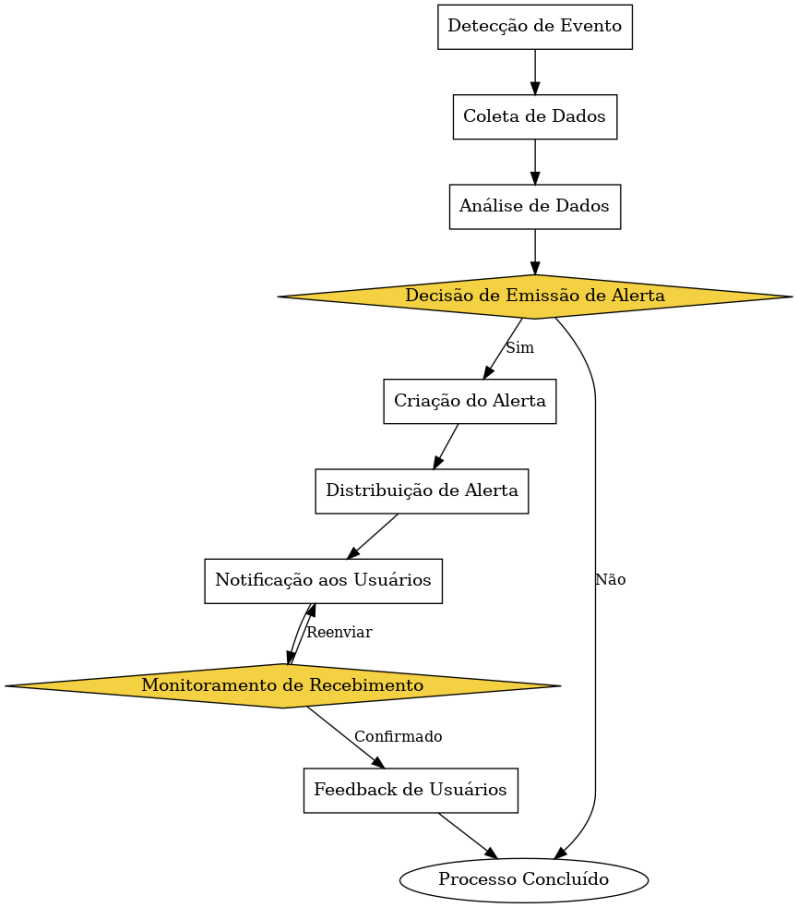



Diagrama de Componentes:

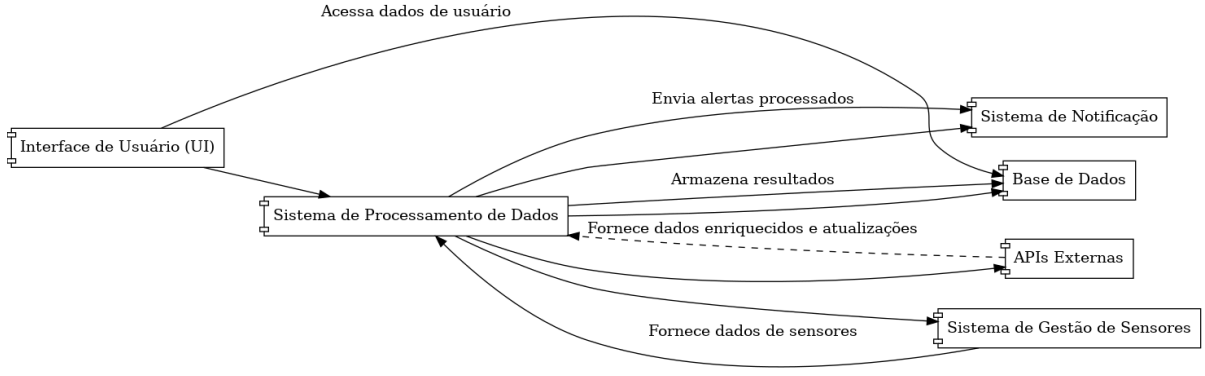
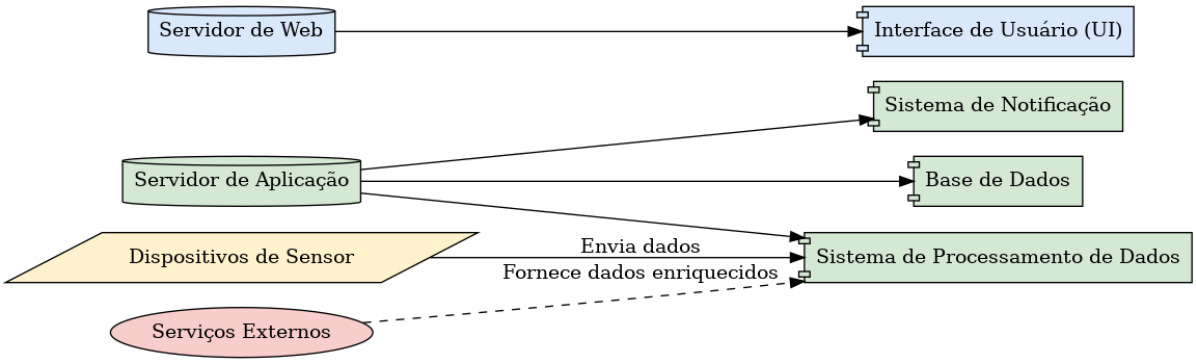


Diagrama de Implantação:




# Discussão

A UML é composta por vários tipos de diagramas, cada um proporcionando uma perspetiva única e valiosa sobre diferentes aspetos de um sistema de software. Para uma aplicação complexa como o SafeGuard, cada tipo de diagrama UML desempenha um papel crucial para entender o design e a funcionalidade do sistema:

1. Diagrama de Casos de Uso

Contribuição: Ilustra as interações entre os utilizadores (atores) e o sistema, facilitando a compreensão das funcionalidades oferecidas pela aplicação e como os utilizadores podem esperar interagir com ela. Identifica as várias ações que os utilizadores podem realizar e como o sistema responde a essas ações.

2. Diagrama de Classes

Contribuição: Descreve a estrutura interna do sistema em termos de classes e objetos, incluindo suas propriedades e métodos, bem como as relações entre eles. Este diagrama é fundamental para entender como os dados são organizados e manipulados dentro do sistema.

3. Diagrama de Sequência

Contribuição: Mostra como os objetos interagem entre si em termos de sequência de mensagens ao longo do tempo. É particularmente útil para visualizar o fluxo de um processo específico dentro do sistema, como o processo de alerta de desastre, desde a deteção até a notificação dos utilizadores.

4. Diagrama de Atividades

Contribuição: Foca no fluxo de atividades e na lógica de negócio do sistema. Este diagrama ajuda a mapear o fluxo de trabalho e as decisões que são tomadas, facilitando a visualização de operações complexas como a análise de dados para a emissão de alertas.

5. Diagrama de Componentes
Contribuição: Explica como o sistema é dividido em componentes de software, mostrando a organização e dependências entre esses componentes. É essencial para arquitetos de software que precisam entender ou explicar a estrutura geral do sistema.

6. Diagrama de Implantação
Contribuição: Fornece uma visão da configuração física do hardware e mostra como o software é distribuído entre os diferentes componentes da infraestrutura. Este diagrama é crucial para compreender como o sistema será mantido e gerido na prática. Cada um desses diagramas contribui de forma significativa para uma visão holística e detalhada do projeto SafeGuard, ajudando diferentes partes interessadas a compreender todas as nuances de como a aplicação funciona, como é construída, e como deve ser mantida ao longo do tempo.

Os diagramas UML interagem e complementam-se para fornecer uma visão abrangente de um projeto de software, como o SafeGuard, garantindo que todos os aspetos do sistema sejam completamente entendidos e integrados:

-Casos de Uso e Classes: Os diagramas de casos de uso delineiam as funções do sistema e as interações do usuário, enquanto os diagramas de classes detalham a estrutura necessária para implementar essas funções.

-Sequência e Atividades: Os diagramas de sequência mostram a interação entre objetos para executar processos, e os diagramas de atividades focam no fluxo de controle e decisões dentro desses processos.

-Componentes e Implantação: Os diagramas de componentes explicam como o software é organizado em partes menores e como essas partes se interligam, e os diagramas de implantação mostram como os componentes são fisicamente distribuídos e configurados.

-Classes e Componentes: Os diagramas de classes oferecem uma visão interna da lógica e estrutura de dados, enquanto os diagramas de componentes mostram as interações entre esses blocos maiores a um nível mais alto.

# Conclusão

Durante a elaboração da especificação UML para o SafeGuard, vários pontos principais foram destacados, mostrando a importância da UML no desenvolvimento de projetos de software complexos:

1. Visualização Integral do Sistema: A UML permitiu visualizar todas as componentes do SafeGuard, desde interações básicas do usuário até complexidades técnicas internas, como a arquitetura de dados e processos operacionais.

2. Clarificação de Requisitos: Os diagramas de casos de uso e de classes ajudaram a definir claramente os requisitos funcionais e as expectativas, garantindo que todas as funcionalidades necessárias fossem consideradas e adequadamente planeadas.

3. Detalhamento de Processos: Os diagramas de sequência e de atividades esclareceram os processos detalhados, como a coleta e análise de dados de sensores e a emissão de alertas, ajudando a identificar pontos críticos para a intervenção em tempo real.

4. Identificação de Dependências: Os diagramas de componentes e de implantação mostraram como os diferentes elementos do sistema interagem e dependem uns dos outros, facilitando a configuração da infraestrutura e a gestão de dependências.

5. Preparação para Expansão e Manutenção: A estrutura detalhada oferecida pelos diagramas UML preparou o projeto para futuras expansões e simplificou a manutenção, ao documentar meticulosamente cada parte do sistema.

Em suma, a UML foi essencial para descomplicar o design complexo do SafeGuard, permitindo uma compreensão clara e um desenvolvimento sistemático do projeto. Ao fornecer uma documentação detalhada e uma representação visual das operações do sistema, a UML tornou possível abordar eficazmente o desenvolvimento e a gestão de um aplicativo de alerta de desastres naturais.