Documentação de Projeto

para o sistema

MRV

Versão 1.0

Projeto de sistema elaborado pelo(s) aluno(s) Fernanda Soares Oliveira Cunha
como parte da disciplina **Projeto de Software**.

10/11/2025

Tabela de Conteúdo

1. Introdução **1**
2. Modelos de Usuário e Requisitos **1**
2.1 Descrição de Atores 1
2.2 Modelo de Casos de Uso e Histórias de Usuários 1
2.3 Diagrama de Sequência do Sistema e Contrato de Operações 1
3. Modelos de Projeto **1**
3.1 Arquitetura 1
3.2 Diagrama de Componentes e Implantação. 2
3.3 Diagrama de Classes 2
3.4 Diagramas de Sequência 2
3.5 Diagramas de Comunicação 2
3.6 Diagramas de Estados 2
4. Modelos de Dados **2**

Histórico de Revisões

Nome	Data	Razões para Mudança	Versão

1. Introdução

Este documento agrega: 1) a elaboração e revisão de modelos de domínio e 2) modelos de projeto para o sistema MRV. A referência principal para a descrição geral do problema, domínio e requisitos do sistema é o documento de especificação que descreve a visão de domínio do sistema.

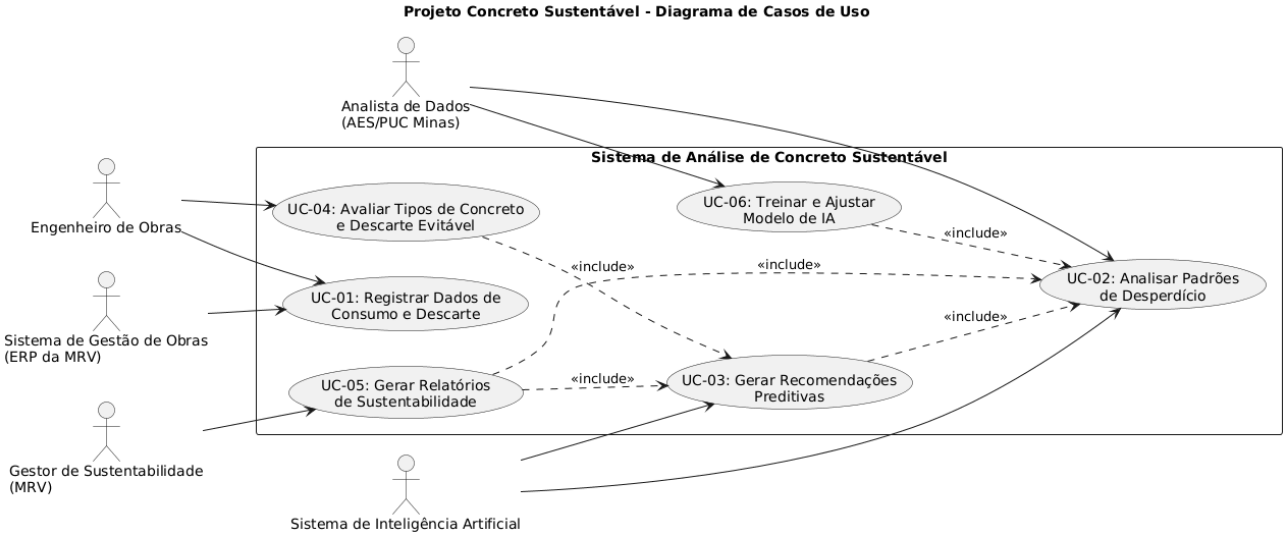
2. Modelos de Usuário e Requisitos

2.1 Descrição de Atores

Ator	Descrição
A1 - Engenheiro de Obras	Responsável por inserir, validar e consultar informações relacionadas ao consumo e descarte de concreto nas obras. Interage com o sistema para analisar relatórios e aplicar as recomendações sugeridas pela IA.
A2 - Analista de Dados (AES/PUC Minas)	Profissional que gerencia os modelos de IA, treina algoritmos, avalia os resultados e ajusta os parâmetros do sistema com base nos dados históricos.
A3 - Gestor de Sustentabilidade (MRV)	Acompanha os indicadores de eficiência e sustentabilidade, gera relatórios estratégicos e monitora o impacto ambiental e financeiro do uso do concreto.
A4 - Sistema de Inteligência Artificial (IA)	Ator secundário automatizado que processa dados históricos e operacionais, identifica padrões de desperdício, falhas e oportunidades de otimização, fornecendo previsões e recomendações.
A5 - Sistema de Gestão de Obras (ERP da MRV)	Sistema externo que fornece dados operacionais (como volume de concreto solicitado, recebido e descartado), e recebe recomendações da IA para integração com os fluxos produtivos.

2.2 Modelo de Casos de Uso

Nesta subseção é apresentado o diagrama de casos de uso do sistema.



@startuml

left to right direction

skinparam packageStyle rectangle

skinparam actorStyle awesome

skinparam usecase {

BackgroundColor #E8F0F8

BorderColor #2E5C9A

ArrowColor #2E5C9A

FontColor #1A1A1A

}

title Projeto Concreto Sustentável - Diagrama de Casos de Uso (com Include/Extend)

actor "Engenheiro de Obras" as Eng

```
actor "Analista de Dados\n(AES/PUC Minas)" as Analista

actor "Gestor de Sustentabilidade\n(MRV)" as Gestor

actor "Sistema de Gestão de Obras\n(ERP da MRV)" as ERP

actor "Sistema de Inteligência Artificial" as IA

rectangle "Sistema de Análise de Concreto Sustentável" {

  usecase "UC-01\nRegistrar Dados de Consumo e Descarte de Concreto" as UC01

  usecase "UC-02\nAnalisar Padrões de Desperdício" as UC02

  usecase "UC-03\nGerar Recomendações Preditivas" as UC03

  usecase "UC-04\nAvaliar Tipos de Concreto e Descarte Evitável" as UC04

  usecase "UC-05\nGerar Relatórios de Sustentabilidade" as UC05

  usecase "UC-06\nTreinar e Ajustar o Modelo de IA" as UC06

}


' --- Relações entre atores e casos de uso ---

Eng --> UC01

Eng --> UC03

ERP --> UC01

IA --> UC02

IA --> UC03

IA --> UC04

IA --> UC05

IA --> UC06

Analista --> UC02

Analista --> UC04
```

Analista --> UC06

Gestor --> UC05

' --- Relações include e extend ---

UC03 ..> UC02 : <<include>>

UC04 ..> UC02 : <<include>>

UC05 ..> UC02 : <<include>>

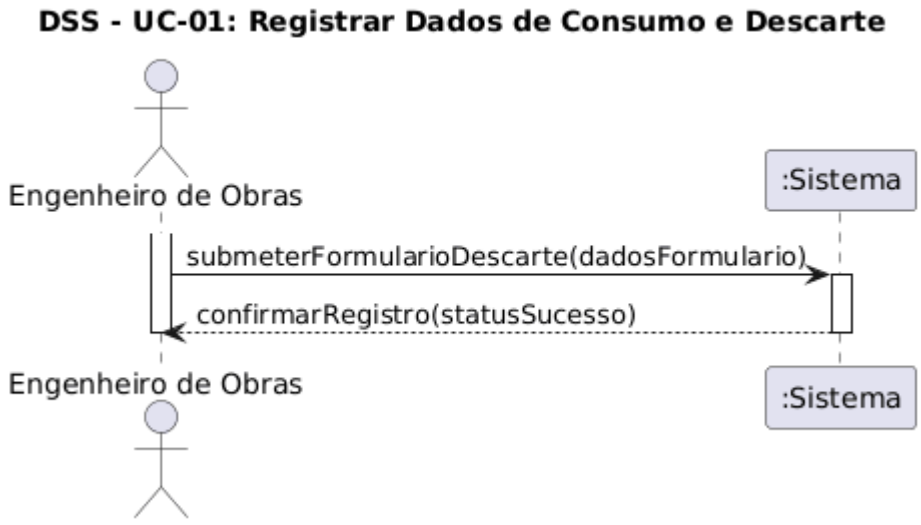
UC05 ..> UC03 : <<include>>

UC06 ..> UC02 : <<include>>

@enduml

2.3 Diagrama de Sequência do Sistema

Nesta subseção é apresentado o diagrama de sequência do sistema de pelo menos, 3 Casos de Uso ou Histórias de Usuário descritos na Seção 2.3.



Código Plantuml:

@startuml
title DSS - UC-01: Registrar Dados de Consumo e Descarte

actor "Engenheiro de Obras" as Engenheiro
participant ":Sistema" as Sistema

activate Engenheiro
Engenheiro -> Sistema : submeterFormularioDescarte(dadosFormulario)
activate Sistema
Sistema --> Engenheiro : confirmarRegistro(statusSucesso)
deactivate Sistema
deactivate Engenheiro
@enduml

CONTRATO – UC-01: Registrar Dados de Consumo e Descarte

Operação: registrarDadosDescarte(dadosFormulario)

Responsabilidade:

Armazenar no sistema um novo registro contendo dados de volume consumido, descartado e motivo do descarte em uma obra.

Tipo: Sistema / Serviço

Pré-condições:

1. Engenheiro deve estar autenticado.
2. Engenheiro deve estar vinculado a uma obra existente.
3. O formulário deve ter sido preenchido corretamente.

Pós-condições:

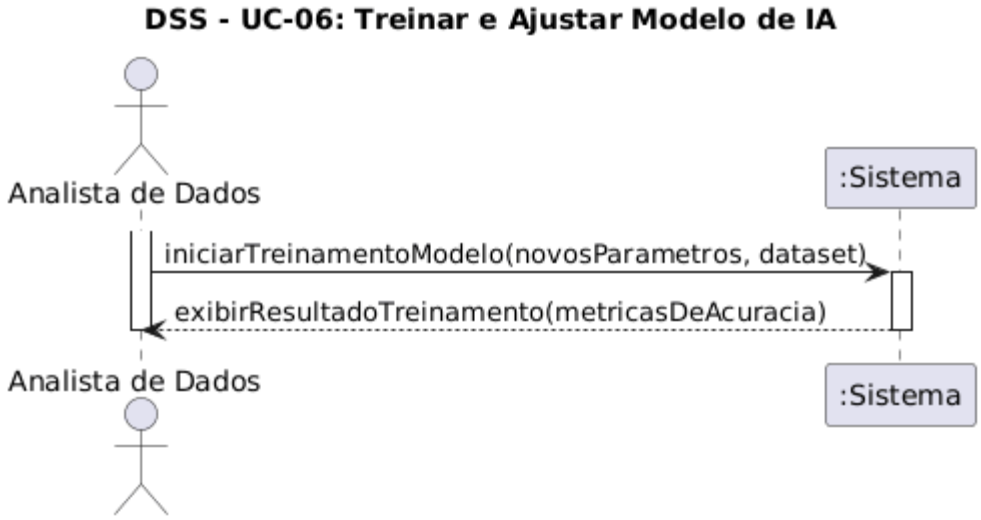
1. Um novo registro de consumo e descarte é criado e persistido no sistema.
2. O sistema retorna o status de sucesso ao Engenheiro.

Descrição:

Valida o formulário, registra o consumo e descarte no banco de dados e confirma a operação.

Exceções:

1. Dados inválidos → retornar erro “formulário incompleto”.
2. Obra não encontrada → retornar erro “obra inexistente”.
3. Falha no banco → retornar erro de persistência.



Código Plantuml:

@startuml
title DSS - UC-06: Treinar e Ajustar Modelo de IA
actor "Analista de Dados" as Analista
participant ":Sistema" as Sistema

activate Analista
Analista -> Sistema : iniciarTreinamentoModelo(novosParametros, dataset)
activate Sistema
Sistema --> Analista : exibirResultadoTreinamento(metricasDeAcuracia)
deactivate Sistema
deactivate Analista
@enduml

CONTRATO – UC-06: Treinar e Ajustar Modelo de IA

Operação: treinarModeloIA(parametros, dataset)

Responsabilidade:

Executar o processo de treino ou ajuste do modelo de IA do sistema e gerar métricas de performance.

Tipo: Serviço de IA / backend

Pré-condições:

1. Analista de Dados autenticado.
2. Dataset válido acessível pelo sistema.

3. Parâmetros de modelo definidos ou carregados.

Pós-condições:

1. O modelo é treinado com os novos dados.
2. As métricas são calculadas e retornadas ao Analista.
3. O modelo atualizado é salvo para uso posterior (predições e recomendações).

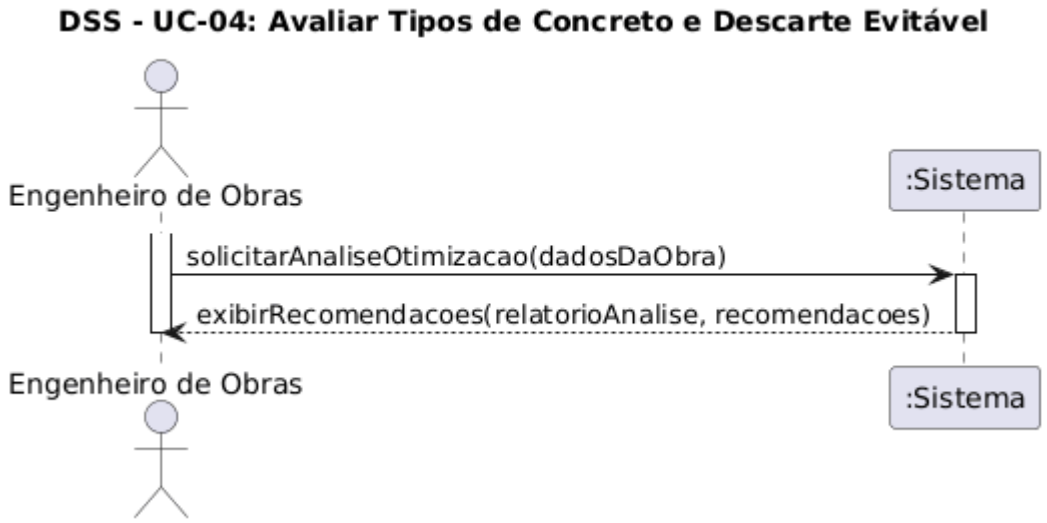
Descrição:

Executa pipeline de machine learning:

- pré-processamento,
- treino,
- validação,
- cálculo de métricas,
- registro do novo modelo.

Exceções:

- Dataset inválido → erro “dados inconsistentes”.
- Falha durante o treino → erro “não foi possível ajustar o modelo”.
- Parâmetros incorretos → erro “configuração inválida”.



Código Plantuml:

@startuml

title DSS - UC-04: Avaliar Tipos de Concreto e Descarte Evitável

actor "Engenheiro de Obras" as Engenheiro

participant ":Sistema" as Sistema

activate Engenheiro

Engenheiro -> Sistema : solicitarAnaliseOtimizacao(dadosDaObra)

activate Sistema

Sistema --> Engenheiro : exibirRecomendacoes(relatorioAnalise, recomendacoes)

deactivate Sistema

deactivate Engenheiro

@enduml

CONTRATO – UC-04: Avaliar Tipos de Concreto e Descarte Evitável

Operação: gerarAnaliseOtimizada(dadosDaObra)

Responsabilidade:

- Executar uma análise sobre os registros da obra e retornar recomendações de uso otimizado do concreto, além de possíveis reduções de descarte.

- **Tipo: Serviço de análise / módulo IA simples**

Pré-condições:

- Engenheiro autenticado.
- A obra deve possuir dados cadastrados.
- O módulo de análise deve estar disponível.

Pós-condições:

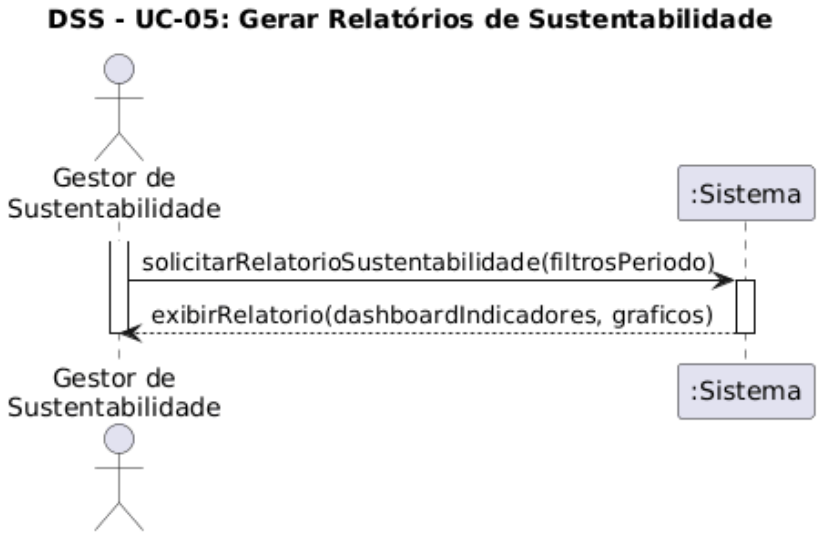
- Um relatório de análise é gerado.
- Recomendações são retornadas ao Engenheiro.

Descrição:

- O sistema coleta os dados da obra, aplica cálculos e heurísticas (ou IA simples) e devolve recomendações.

Exceções:

- Dados insuficientes para análise → retorna mensagem específica.
- Falha no módulo de análise → retorna erro técnico.



Código Plantuml:
@startuml
title DSS - UC-05: Gerar Relatórios de Sustentabilidade
actor "Gestor de\nSustentabilidade" as Gestor
participant ":Sistema" as Sistema

activate Gestor
Gestor -> Sistema : solicitarRelatorioSustentabilidade(filtrosPeriodo)
activate Sistema
Sistema --> Gestor : exibirRelatorio(dashboardIndicadores, graficos)
deactivate Sistema
deactivate Gestor
@enduml

CONTRATO – UC-05: Gerar Relatórios de Sustentabilidade

Operação: gerarRelatorioSustentabilidade(filtros)

Responsabilidade:

Criar um relatório contendo indicadores, métricas e gráficos de sustentabilidade da obra ou período solicitado.

Tipo: Serviço / módulo de relatórios

Pré-condições:

1. Gestor deve estar autenticado.
2. Dados suficientes devem existir para compor o relatório.

Pós-condições:

1. O sistema gera um relatório estruturado (JSON, PDF ou dashboard).
2. O relatório é disponibilizado ao Gestor.

Descrição:

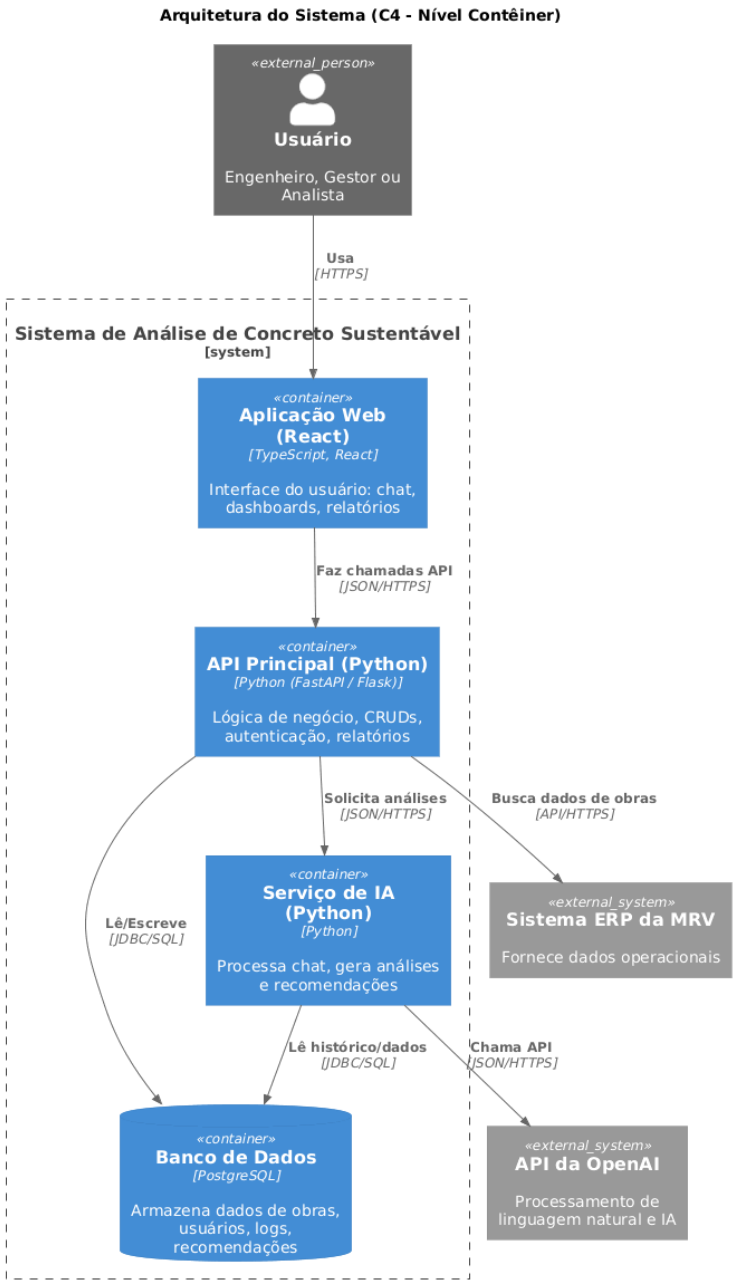
Processa indicadores, agregações, cálculos de impacto e monta o relatório com base nos filtros aplicados.

Exceções:

- Período sem dados → retorna relatório vazio ou aviso.
- Falha no módulo gráfico → retorna aviso e relatório sem gráficos.

3. Modelos de Projeto

3.1 Arquitetura



Código Plantuml:

@startuml !include https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4_Container.puml

title Arquitetura do Sistema (C4 - Nível Contêiner)

' Definição do Usuário Externo Person_Ext(usuario, "Usuário", "Engenheiro, Gestor ou Analista")

' Limite do Sistema System_Boundary(c1, "Sistema de Análise de Concreto Sustentável") {

```
Container(web_app, "Aplicação Web (React)", "TypeScript, React", "Interface do usuário: chat, dashboards, relatórios")

Container(api, "API Principal (Python)", "Python (FastAPI / Flask)", "Lógica de negócio, CRUDs, autenticação, relatórios")

Container(ai_service, "Serviço de IA (Python)", "Python", "Processa chat, gera análises e recomendações")

ContainerDb(db, "Banco de Dados", "PostgreSQL", "Armazena dados de obras, usuários, logs, recomendações")
```

}

' Sistemas Externos System_Ext(erp_mrv, "Sistema ERP da MRV", "Fornece dados operacionais") System_Ext(openai, "API da OpenAI", "Processamento de linguagem natural e IA")

' Relacionamentos Rel(usuario, web_app, "Usa", "HTTPS")

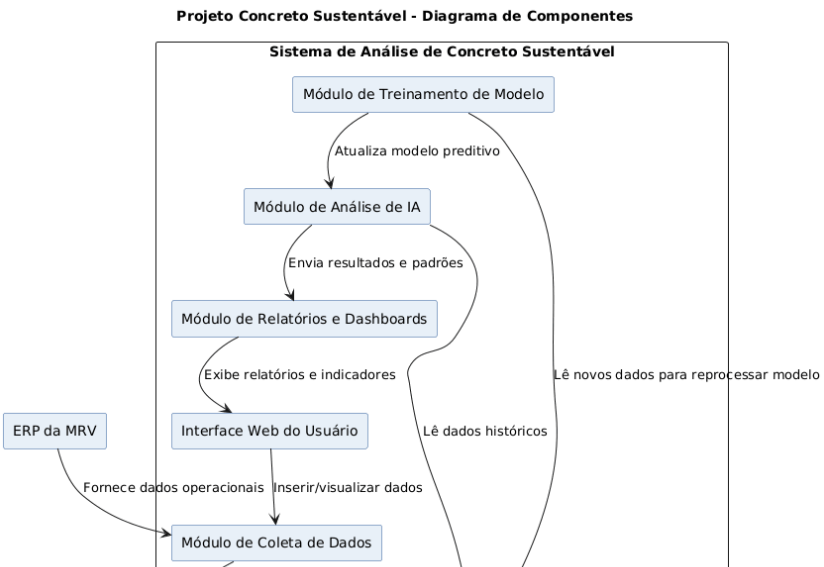
Rel(web_app, api, "Faz chamadas API", "JSON/HTTPS")

Rel(api, ai_service, "Solicita análises", "JSON/HTTPS") Rel(api, db, "Lê/Escreve", "JDBC/SQL") Rel(api, erp_mrv, "Busca dados de obras", "API/HTTPS")

Rel(ai_service, db, "Lê histórico/dados", "JDBC/SQL") Rel(ai_service, openai, "Chama API", "JSON/HTTPS") @enduml

3.2 Diagrama de Componentes e Implantação.

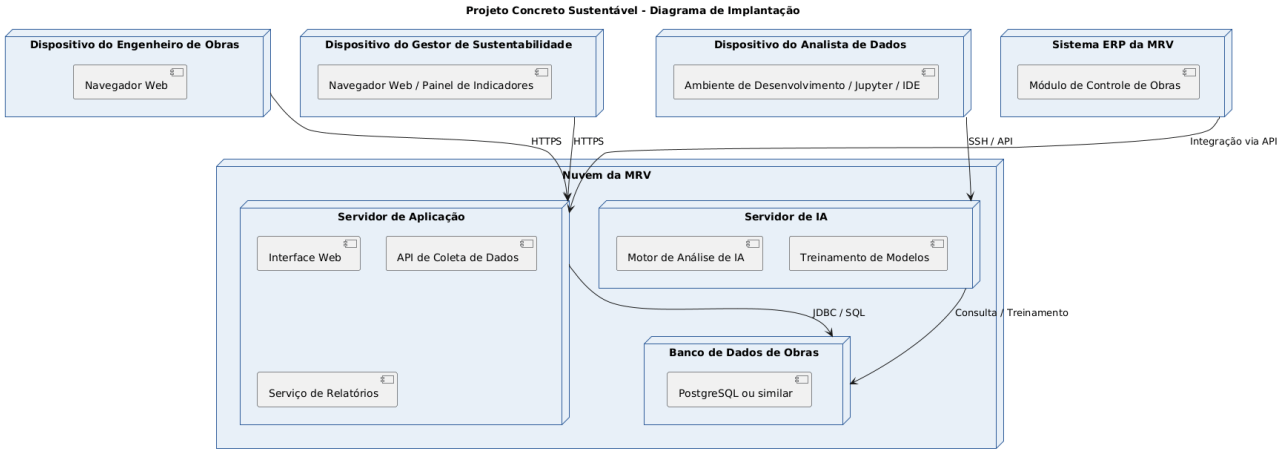
Diagrama de componentes do sistema:



Código Plantuml:

```
@startuml
title Projeto Concreto Sustentável - Diagrama de Componentes

skinparam componentStyle rectangle
skinparam component {
  BackgroundColor #E8F0F8
  BorderColor #2E5C9A
}

rectangle "Sistema de Análise de Concreto Sustentável" {

  [Interface Web do Usuário] as UI
  [Módulo de Coleta de Dados] as Coleta
  [Módulo de Análise de IA] as AnaliseIA
  [Módulo de Treinamento de Modelo] as Treinamento
  [Módulo de Relatórios e Dashboards] as Relatorios
  [Banco de Dados de Obras] as BD
}

' Componentes externos
[ERP da MRV] as ERP
[Serviço de Armazenamento em Nuvem] as Cloud

' Relações entre componentes
UI --> Coleta : Inserir/visualizar dados
Coleta --> BD : Registrar dados de consumo e descarte
ERP --> Coleta : Fornece dados operacionais
AnaliseIA --> BD : Lê dados históricos
AnaliseIA --> Relatorios : Envia resultados e padrões
Treinamento --> BD : Lê novos dados para reprocessar modelo
Treinamento --> AnaliseIA : Atualiza modelo preditivo
Relatorios --> UI : Exibe relatórios e indicadores
BD --> Cloud : Backup e persistência segura dos dados

@enduml
```

Diagrama de implantação do sistema:

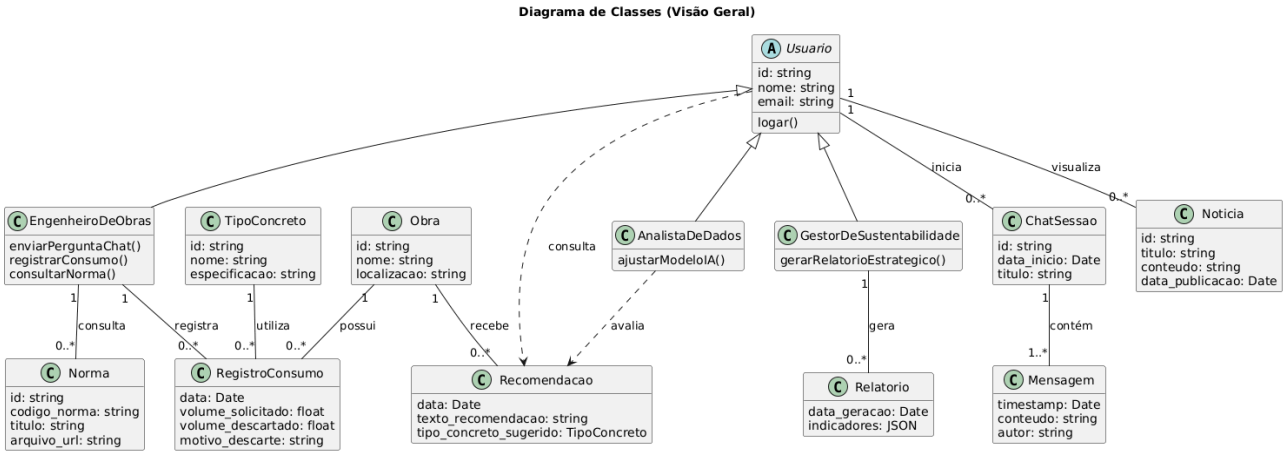


Código Plantuml:

```
@startuml
title Projeto Concreto Sustentável - Diagrama de Implantação

skinparam node {
  BackgroundColor #E8F0F8
  BorderColor #2E5C9A
}

node "Nuvem da MRV" {

  node "Servidor de Aplicação" {
    [Interface Web]
    [API de Coleta de Dados]
    [Serviço de Relatórios]
  }

  node "Servidor de IA" {
    [Motor de Análise de IA]
    [Treinamento de Modelos]
  }

  node "Banco de Dados de Obras" {
    [PostgreSQL ou similar]
  }
}

node "Dispositivo do Engenheiro de Obras" {
  [Navegador Web]
}

node "Dispositivo do Gestor de Sustentabilidade" {
```

```
  [Navegador Web / Painel de Indicadores]
}

node "Dispositivo do Analista de Dados" {
  [Ambiente de Desenvolvimento / Jupyter / IDE]
}

node "Sistema ERP da MRV" {
  [Módulo de Controle de Obras]
}

' Relações de comunicação
"Dispositivo do Engenheiro de Obras" --> "Servidor de Aplicação" : HTTPS
"Dispositivo do Gestor de Sustentabilidade" --> "Servidor de Aplicação" : HTTPS
"Dispositivo do Analista de Dados" --> "Servidor de IA" : SSH / API
"Servidor de Aplicação" --> "Banco de Dados de Obras" : JDBC / SQL
"Servidor de IA" --> "Banco de Dados de Obras" : Consulta / Treinamento
"Sistema ERP da MRV" --> "Servidor de Aplicação" : Integração via API
@enduml
```

3.3 Diagrama de Classes



Código Plantuml:

```
@startuml title Diagrama de Classes (Visão Geral) skinparam classAttributeIconSize 0

abstract class "Usuario" as Usuario { id: string nome: string email: string logar() }

class "EngenheiroDeObras" as Engenheiro { enviarPerguntaChat() registrarConsumo() consultarNorma() }

class "AnalistaDeDados" as Analista { ajustarModeloIA() }

class "GestorDeSustentabilidade" as Gestor { gerarRelatorioEstrategico() }

class "Norma" as Norma { id: string codigo_norma: string titulo: string arquivo_url: string }
```

class "TipoConcreto" as TipoConcreto { id: string nome: string especificacao: string }

class "RegistroConsumo" as RegistroConsumo { data: Date volume_solicitado: float volume_descartado: float motivo_descarte: string }

class "Obra" as Obra { id: string nome: string localizacao: string }

class "Recomendacao" as Recomendacao { data: Date texto_recomendacao: string tipo_concreto_sugerido: TipoConcreto }

class "Relatorio" as Relatorio { data_geracao: Date indicadores: JSON }

class "ChatSessao" as ChatSessao { id: string data_inicio: Date titulo: string }

class "Mensagem" as Mensagem { timestamp: Date conteudo: string autor: string }

class "Noticia" as Noticia { id: string titulo: string conteudo: string data_publicacao: Date }

' Relacionamentos de Herança Usuario <|-- Engenheiro Usuario <|-- Analista Usuario <|-- Gestor

' Relacionamentos Engenheiro Engenheiro "1" --> "0..*" Norma : consulta Engenheiro "1" --> "0..*" RegistroConsumo : registra Engenheiro ..> Recomendacao : consulta

' Relacionamentos RegistroConsumo TipoConcreto "1" -- "0..*" RegistroConsumo : utiliza Obra "1" -- "0..*" RegistroConsumo : possui

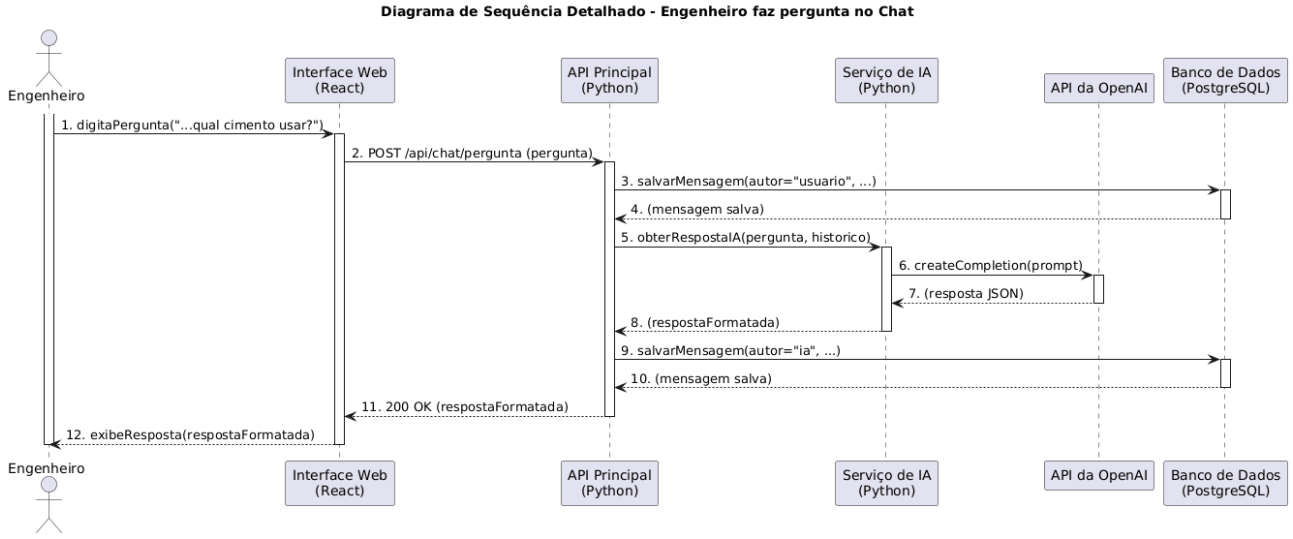
' Relacionamentos Obra/Recomendacao Obra "1" -- "0..*" Recomendacao : recebe Analista ..> Recomendacao : avalia

' Relacionamentos Gestor Gestor "1" --> "0..*" Relatorio : gera

' Relacionamentos Chat/Usuario Usuario "1" --> "0..*" ChatSessao : inicia ChatSessao "1" -- "1..*" Mensagem : contém

' Relacionamentos Noticia Usuario "1" --> "0..*" Noticia : visualiza @enduml

3.4 Diagramas de Sequência



Código Plantuml:

@startuml
title Diagrama de Sequência Detalhado - Engenheiro faz pergunta no Chat
actor Engenheiro
participant "Interface Web\n(React)" as Web
participant "API Principal\n(Python)" as API
participant "Serviço de IA\n(Python)" as IA
participant "API da OpenAI" as OpenAI
participant "Banco de Dados\n(PostgreSQL)" as DB

activate Engenheiro
Engenheiro -> Web : 1. digitaPergunta("...qual cimento usar?")
activate Web

Web -> API : 2. POST /api/chat/pergunta (pergunta)
activate API

API -> DB : 3. salvarMensagem(autor="usuario", ...)
activate DB
DB --> API : 4. (mensagem salva)
deactivate DB

API -> IA : 5. obterRespostaIA(pergunta, historico)
activate IA

IA -> OpenAI : 6. createCompletion(prompt)
activate OpenAI
OpenAI --> IA : 7. (resposta JSON)

deactivate OpenAI

IA --> API : 8. (respostaFormatada)
deactivate IA

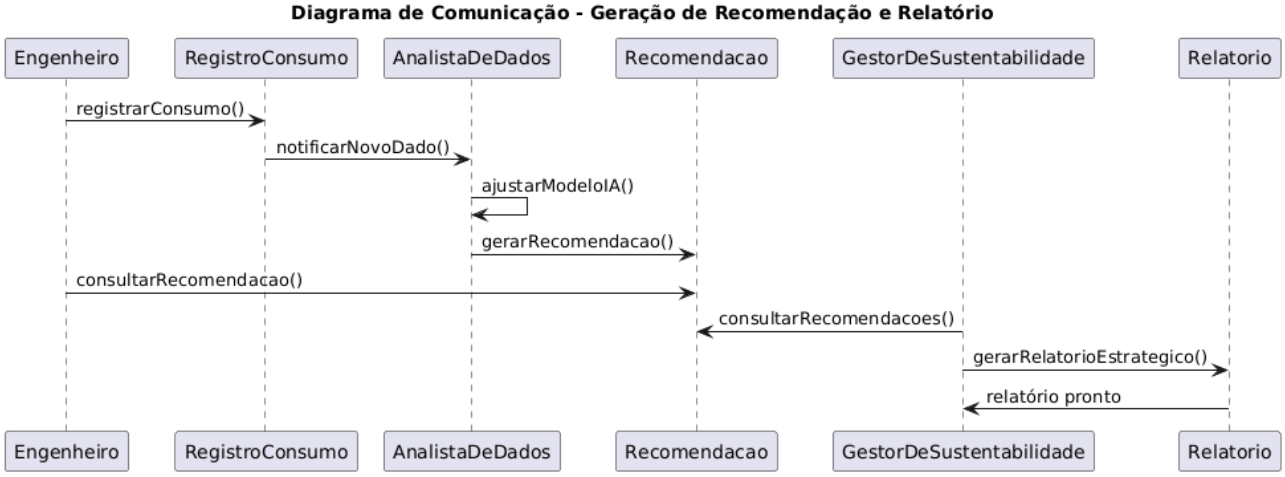
API -> DB : 9. salvarMensagem(autor="ia", ...)
activate DB
DB --> API : 10. (mensagem salva)
deactivate DB

API --> Web : 11. 200 OK (respostaFormatada)
deactivate API

Web --> Engenheiro : 12. exibeResposta(respostaFormatada)
deactivate Web
deactivate Engenheiro
@enduml

3.5 Diagramas de Comunicação

Diagramas de comunicação para realização do caso de uso: geração e recomendação de relatório



@startuml
title Diagrama de Comunicação - Geração de Recomendação e Relatório

participant Engenheiro as E
participant RegistroConsumo as RC
participant AnalistaDeDados as AD
participant Recomendacao as R
participant GestorDeSustentabilidade as G
participant Relatorio as Rel

E -> RC : registrarConsumo()
RC -> AD : notificarNovoDado()

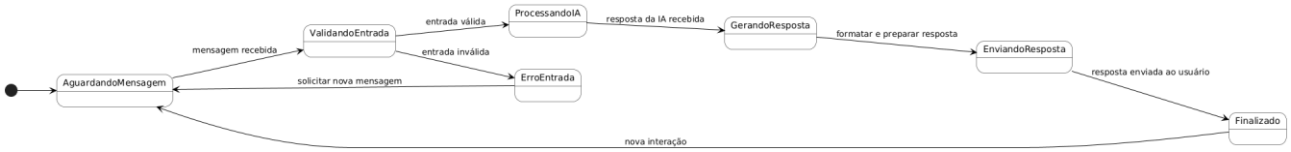
AD -> AD : ajustarModeloIA()
AD -> R : gerarRecomendacao()

E -> R : consultarRecomendacao()

G -> R : consultarRecomendacoes()
G -> Rel : gerarRelatorioEstrategico()
Rel -> G : relatório pronto

@enduml

3.6 Diagramas de Estados



@startuml
skinparam state {
  BackgroundColor White
  BorderColor Black
  ArrowColor Black
}
left to right direction

[*] --> AguardandoMensagem

state AguardandoMensagem {
}

AguardandoMensagem --> ValidandoEntrada : mensagem recebida

ValidandoEntrada --> ErroEntrada : entrada inválida
ErroEntrada --> AguardandoMensagem : solicitar nova mensagem

ValidandoEntrada --> ProcessandoIA : entrada válida

ProcessandoIA --> GerandoResposta : resposta da IA recebida

GerandoResposta --> EnviandoResposta : formatar e preparar resposta

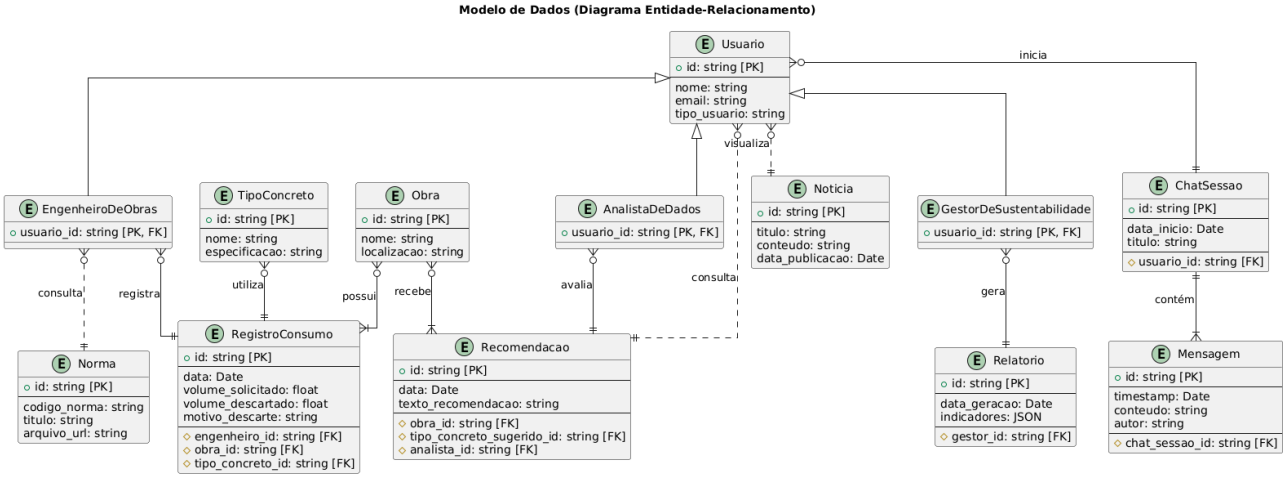
EnviandoResposta --> Finalizado : resposta enviada ao usuário

Finalizado --> AguardandoMensagem : nova interação

@enduml

4. Modelos de Dados

Deve-se apresentar os esquemas de banco de dados e as estratégias de mapeamento entre as representações de objetos e não-objetos.



Código Plantuml:

@startuml title Modelo de Dados (Diagrama Entidade-Relacionamento) hide circle skinparam linetype ortho

entity "Usuario" as Usuario { *id: string [PK] -- nome: string email: string tipo_usuario: string }

entity "EngenheiroDeObras" as Engenheiro { *usuario_id: string [PK, FK] }

entity "AnalistaDeDados" as Analista { *usuario_id: string [PK, FK] }

entity "GestorDeSustentabilidade" as Gestor { *usuario_id: string [PK, FK] }

entity "Norma" as Norma { *id: string [PK] -- codigo_norma: string titulo: string arquivo_url: string }

entity "TipoConcreto" as TipoConcreto { *id: string [PK] -- nome: string especificacao: string }

entity "Obra" as Obra { *id: string [PK] -- nome: string localizacao: string }

entity "RegistroConsumo" as RegistroConsumo { *id: string [PK] -- data: Date volume_solicitado: float volume_descartado: float motivo_descarte: string -- *engenheiro_id: string [FK] *obra_id: string [FK] *tipo_concreto_id: string [FK] }

entity "Recomendacao" as Recomendacao { *id: string [PK] -- data: Date texto_recomendacao: string -- *obra_id: string [FK] *tipo_concreto_sugerido_id: string [FK] *analista_id: string [FK] }

entity "ChatSessao" as ChatSessao { *id: string [PK] -- data_inicio: Date titulo: string -- *usuario_id: string [FK] }

entity "Mensagem" as Mensagem { *id: string [PK] -- timestamp: Date conteudo: string autor: string -- *chat_sessao_id: string [FK] }

entity "Noticia" as Noticia { *id: string [PK] -- titulo: string conteudo: string data_publicacao: Date }

entity "Relatorio" as Relatorio { *id: string [PK] -- data_geracao: Date indicadores: JSON -- *gestor_id: string [FK] }

' Relacionamentos Usuario <|-- Engenheiro Usuario <|-- Analista Usuario <|-- Gestor

Usuario ||--|{ ChatSessao : inicia ChatSessao ||--|{ Mensagem : contém Usuario ||--o{ Noticia : visualiza

Engenheiro |o..o{ Norma : consulta Engenheiro ||--o{ RegistroConsumo : registra

TipoConcreto ||--o{ RegistroConsumo : utiliza Obra ||--o{ RegistroConsumo : possui

Obra ||--o{ Recomendacao : recebe Analista ||--o{ Recomendacao : avalia Engenheiro |o..o{ Recomendacao : consulta

Gestor ||--o{ Relatorio : gera

@enduml