# Documentação de Projeto

## para o sistema

# RPG Master

**Versão 1.0**

Projeto de sistema elaborado pelo aluno Miguel Figueiredo Diniz
como parte da disciplina **Projeto de Software**.

**12/11/2025**

# Tabela de Conteúdo



# Histórico de Revisões

| **Nome** | **Data** | **Razões para Mudança** | **Versão** |
|---|---|---|---|
| Primeira versão | 15/11 | Preenchimento inicial do documento | 1.0 |
| | | | |

# 1. Introdução

Este documento agrega a elaboração e revisão de modelos de domínio e modelos de projeto para o sistema RPG Master.

O sistema tem como objetivo centralizar o gerenciamento de campanhas, personagens e regras de Role-Playing Game, atendendo às necessidades de Mestres (GMs) e Jogadores (Players). Ele permite a criação e gerenciamento de:

- **Campanhas**
- **Fichas de personagens** (PCs e NPCs)
- **Sistemas de regras customizáveis**
- **Registro de sessões**

# 2. Modelos de Usuário e Requisitos

## 2.1 Descrição de Atores

O sistema possui dois perfis principais de usuários com diferentes níveis de permissão e funcionalidades:

**Mestre (GM)**

O Mestre é o administrador da campanha e possui o conjunto mais amplo de permissões no sistema. É responsável por:

1. Criar, editar e deletar campanhas
2. Escolher e configurar Sistemas de Regras para cada campanha (ex: D&D 5e, Tormenta)
3. Gerenciar a lista de Jogadores e seus Personagens participantes
4. Criar, editar e deletar NPCs (Personagens Não-Jogadores)
5. Acessar e editar fichas completas de todos os NPCs e PCs da campanha
6. Criar e gerenciar registros de sessões (data, resumo, notas)

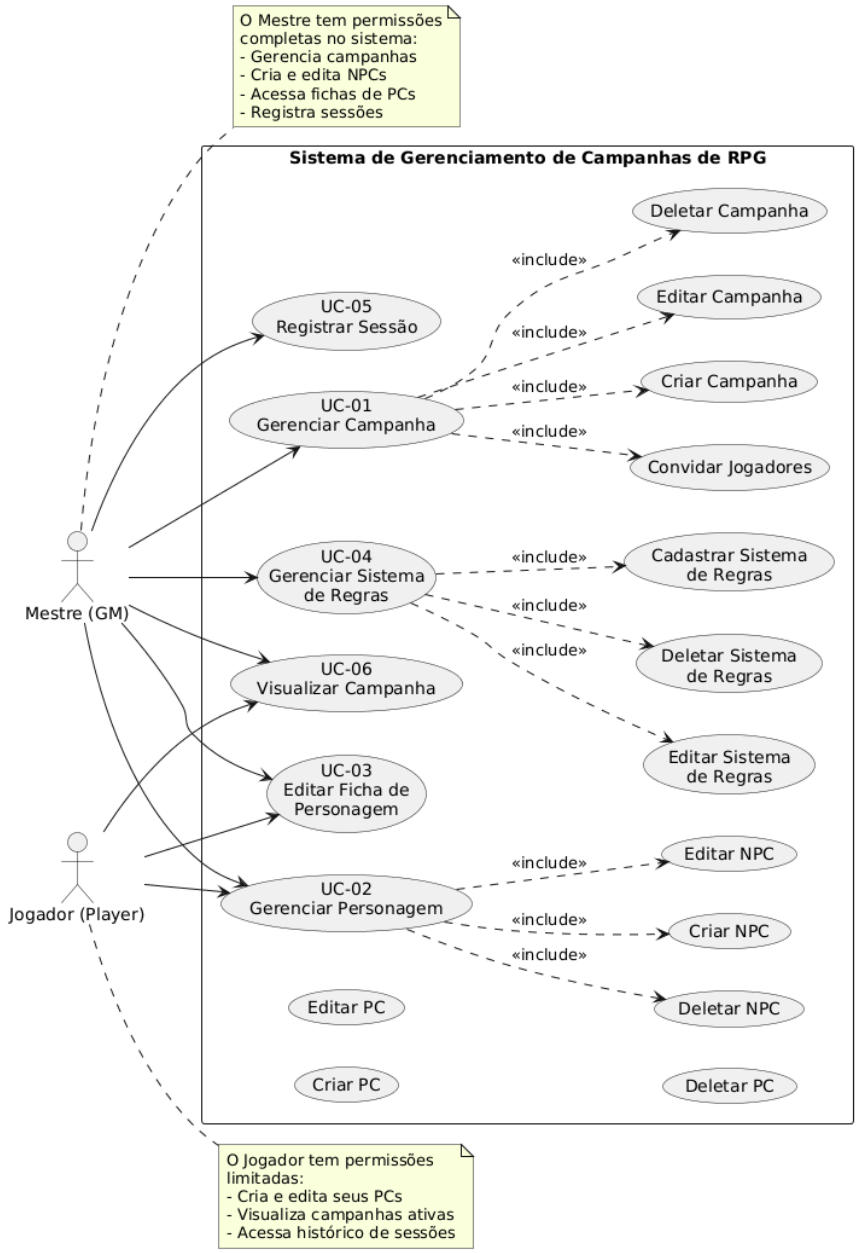
**Jogador (Player)**

O Jogador é o participante das campanhas e possui funcionalidades focadas no gerenciamento de seus próprios personagens:

1. Criar, editar e deletar seus Personagens Jogadores (PCs)
2. Gerenciar fichas de seus PCs com campos dinâmicos baseados no Sistema de Regras
3. Visualizar campanhas ativas nas quais está participando
4. Acessar informações básicas das campanhas e histórico de sessões

## 2.2 Modelo de Casos de Uso

O sistema é estruturado em torno dos seguintes Casos de Uso (UCs):




| ID | Caso de Uso | Descrição | Ator(es) |
|---|---|---|---|
| **UC-01** | **Gerenciar Campanha** | Permite ao GM criar, editar, visualizar e | Mestre (GM) |

| ID | Caso de Uso | Descrição | Ator(es) |
|---|---|---|---|
| | | deletar campanhas, escolhendo sistema de regras e gerenciando jogadores participantes. | |
| UC-01.1 | Criar Campanha | Permite ao GM criar uma nova campanha, definindo nome, descrição e selecionando um sistema de regras. | Mestre (GM) |
| UC-01.2 | Editar Campanha | Permite ao GM modificar informações de uma campanha existente, como nome, descrição ou status. | Mestre (GM) |
| UC-01.3 | Deletar Campanha | Permite ao GM remover permanentemente uma campanha do sistema, incluindo todos os dados associados. | Mestre (GM) |
| UC-01.4 | Convidar Jogadores | Permite ao GM adicionar jogadores à campanha através de convites por email ou username. | Mestre (GM) |
| **UC-02** | **Gerenciar Personagem** | Permite criar, editar e deletar personagens. Players gerenciam seus PCs, GM gerencia NPCs e tem acesso aos PCs. | Mestre (GM), Jogador (Player) |
| UC-02.1 | Criar NPC | Permite ao GM criar um Personagem Não-Jogador (NPC) associado a uma campanha específica. | Mestre (GM) |

| ID | Caso de Uso | Descrição | Ator(es) |
|---|---|---|---|
| UC-02.2 | Editar NPC | Permite ao GM modificar informações e ficha de um NPC existente. | Mestre (GM) |
| UC-02.3 | Deletar NPC | Permite ao GM remover permanentemente um NPC do sistema. | Mestre (GM) |
| UC-02.4 | Criar PC | Permite ao Player criar um Personagem Jogador (PC) vinculado a uma campanha, com ficha baseada no sistema de regras. | Jogador (Player) |
| UC-02.5 | Editar PC | Permite ao Player modificar informações e ficha de seu próprio PC. | Jogador (Player) |
| UC-02.6 | Deletar PC | Permite ao Player remover permanentemente seu próprio PC do sistema. | Jogador (Player) |
| **UC-03** | **Editar Ficha de Personagem** | Permite editar dados dinâmicos da ficha (estatísticas, habilidades, inventário, história) conforme o sistema de regras. Validação de permissões. | Mestre (GM), Jogador (Player) |
| **UC-04** | **Gerenciar Sistema de Regras** | Permite cadastrar, editar e deletar sistemas de regras com templates de fichas customizáveis em formato JSON. | Mestre (GM), Administrador |

| ID | Caso de Uso | Descrição | Ator(es) |
|---|---|---|---|
| UC-04.1 | Cadastrar Sistema de Regras | Permite criar um novo sistema de regras definindo nome, descrição e template de ficha em JSON. | Mestre (GM), Administrador |
| UC-04.2 | Editar Sistema de Regras | Permite modificar um sistema de regras existente, incluindo seu template de ficha. | Mestre (GM), Administrador |
| UC-04.3 | Deletar Sistema de Regras | Permite remover um sistema de regras do sistema, se não estiver sendo utilizado por campanhas ativas. | Mestre (GM), Administrador |
| **UC-05** | **Registrar Sessão** | Permite ao GM criar e gerenciar registros de sessões de jogo com data, resumo e notas, mantendo histórico da campanha. | Mestre (GM) |
| **UC-06** | **Visualizar Campanha** | Permite visualizar informações da campanha, lista de participantes, sistema de regras utilizado e histórico de sessões. | Mestre (GM), Jogador (Player) |

## 2.3 Diagrama de Sequência do Sistema

Os contratos de operação definem as pré e pós-condições das principais operações do sistema.

**1. Contrato: Criar Campanha**

| | |
|---|---|
| **Operação** | criarCampanha(nome: String, descrição: String, ruleset_id: Integer, gm_user_id: Integer) |
| **Referências Cruzadas** | UC-01: Gerenciar Campanha, UC-01.1: Criar Campanha |

| | |
|---|---|
| **Pré-condições** | • O usuário está autenticado no sistema<br>• O usuário possui role de Mestre (GM)<br>• Existe pelo menos um Sistema de Regras cadastrado no banco de dados<br>• O ruleset_id fornecido é válido e existe no banco de dados |
| **Pós-condições** | • Uma nova instância de Campanha foi criada no banco de dados<br>• A campanha está associada ao GM através do atributo gm_user_id<br>• A campanha está vinculada ao Sistema de Regras especificado<br>• O status da campanha foi definido como 'Ativa'<br>• Um campaign_id único foi gerado e retornado<br>• O sistema exibe mensagem de confirmação ao usuário |

**2. Contrato: Criar Personagem (Player)**

| | |
|---|---|
| **Operação** | criarPersonagem(nome: String, campaign_id: Integer, owner_user_id: Integer, ficha_data: JSON) |
| **Referências Cruzadas** | UC-02: Gerenciar Personagem, UC-02.4: Criar PC |
| **Pré-condições** | • O usuário está autenticado no sistema<br>• O usuário possui role de Jogador (Player)<br>• O campaign_id fornecido é válido e existe no banco de dados<br>• O jogador tem permissão para criar personagens na campanha especificada<br>• A campanha possui um Sistema de Regras associado<br>• Os dados da ficha (ficha_data) estão em formato JSON válido |
| **Pós-condições** | • Uma nova instância de Character foi criada no banco de dados<br>• O atributo is_npc foi definido como FALSE (indicando PC)<br>• O personagem está associado ao jogador através do atributo owner_user_id<br>• O personagem está vinculado à campanha especificada<br>• Os dados da ficha foram armazenados no campo ficha_data em formato JSON<br>• Um character_id único foi gerado e retornado |

| | |
|---|---|
| | • Uma associação foi criada na tabela CampaignCharacter<br>• O sistema exibe mensagem de confirmação ao usuário |

**3. Contrato: Editar Ficha de Personagem**

| | |
|---|---|
| **Operação** | atualizarFicha(character_id: Integer, user_id: Integer, ficha_data: JSON) |
| **Referências Cruzadas** | UC-03: Editar Ficha de Personagem |
| **Pré-condições** | • O usuário está autenticado no sistema<br>• O character_id fornecido é válido e existe no banco de dados<br>• O personagem existe e está ativo no sistema<br>• O usuário possui permissão para editar: é o owner (owner_user_id = user_id) OU é o GM da campanha<br>• Os dados da ficha (ficha_data) estão em formato JSON válido<br>• Os campos modificados respeitam o template do Sistema de Regras |
| **Pós-condições** | • O campo ficha_data do personagem foi atualizado no banco de dados<br>• As modificações foram persistidas com sucesso<br>• Um timestamp de última modificação foi registrado<br>• O sistema retorna confirmação de atualização<br>• O sistema exibe mensagem de sucesso ao usuário<br>• Em caso de permissão negada: nenhuma alteração é feita e mensagem de erro é exibida |

**4. Contrato: Registrar Sessão**

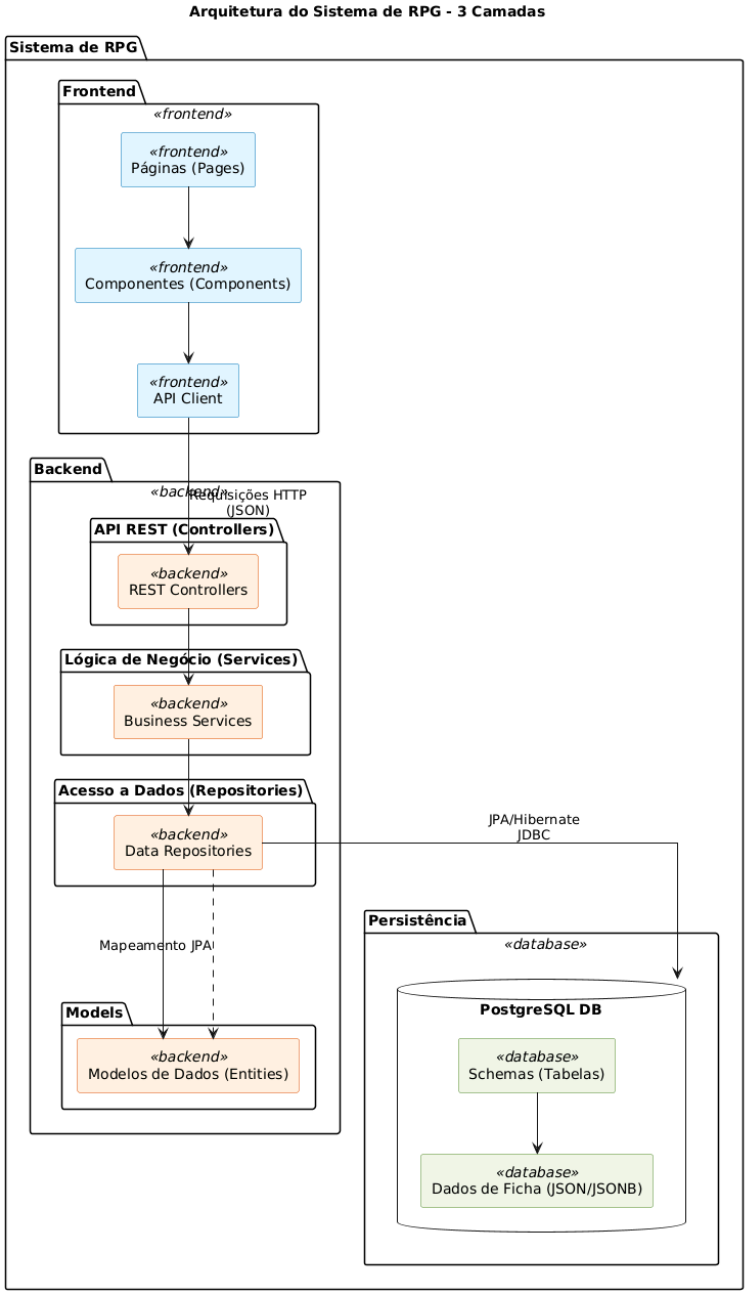
| | |
|---|---|
| **Operação** | registrarSessao(campaign_id: Integer, gm_user_id: Integer, data: Date, resumo: String, notas: Text) |
| **Referências Cruzadas** | UC-05: Registrar Sessão |
| **Pré-condições** | • O usuário está autenticado no sistema<br>• O usuário possui role de Mestre (GM)<br>• O campaign_id fornecido é válido e existe no banco de dados<br>• A campanha está com status 'Ativa' |

| | |
|---|---|
| | • O GM é o proprietário da campanha (gm_user_id da campanha = user_id do solicitante)<br>• Os campos obrigatórios (data, resumo) foram preenchidos<br>• A data da sessão é válida |
| **Pós-condições** | • Uma nova instância de Session foi criada no banco de dados<br>• A sessão está associada à campanha através do atributo campaign_id<br>• Os campos data, resumo e notas foram armazenados<br>• Um timestamp de criação (created_at) foi registrado automaticamente<br>• Um session_id único foi gerado e retornado<br>• O campo last_session da campanha foi atualizado com a data atual<br>• O histórico de sessões da campanha foi incrementado<br>• O sistema exibe mensagem de confirmação ao GM<br>• Em caso de permissão negada: nenhum registro é criado e mensagem de erro é exibida |

# 3. Modelos de Projeto

## 3.1 Arquitetura

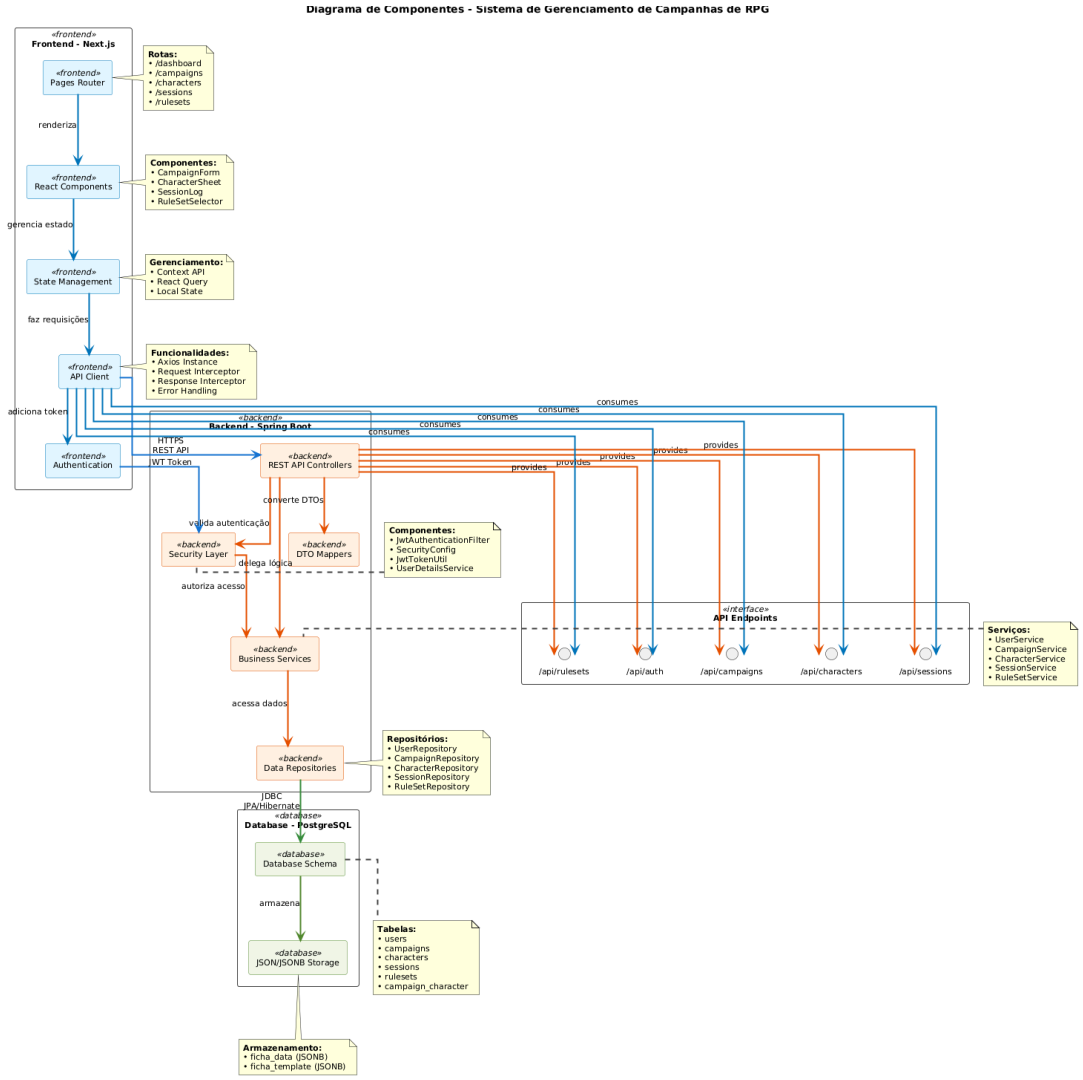
O sistema de RPG Master utiliza uma Arquitetura em 3 Camadas, conforme o diagrama abaixo:

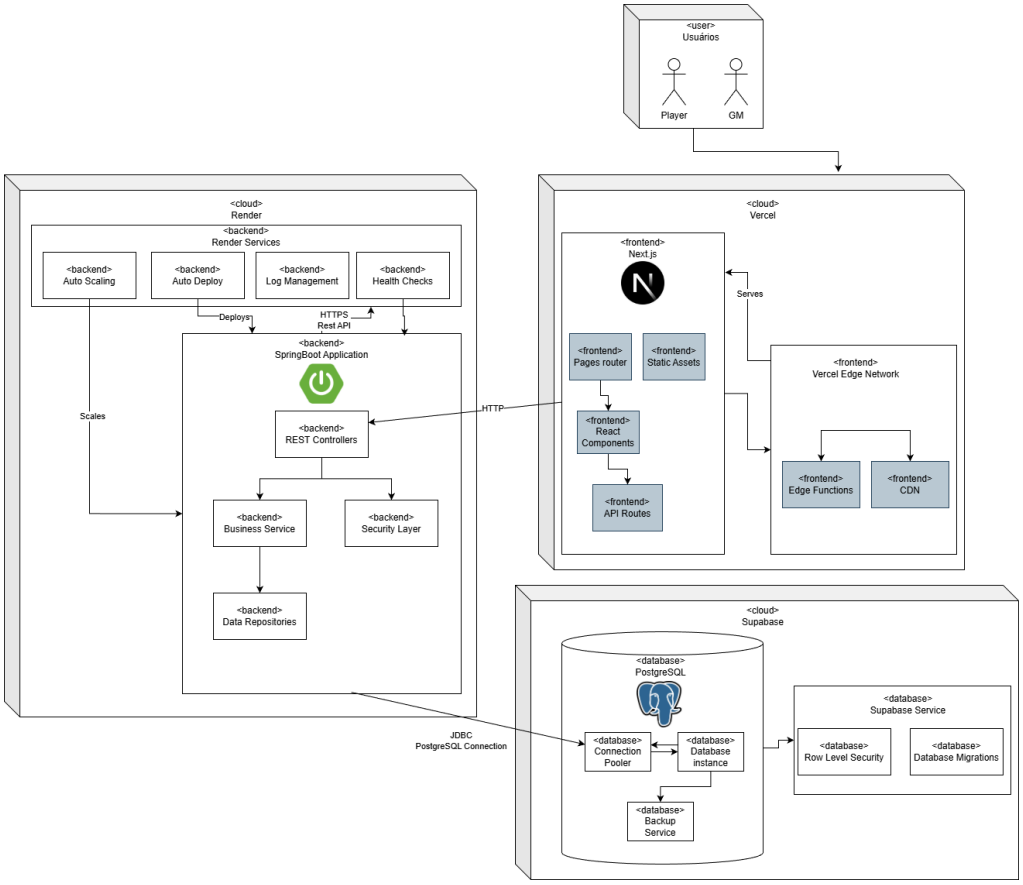



- **Frontend**: Contém as Páginas (Pages) e Componentes (Components), interagindo com o Backend através de um API Client.

- **Backend**: Divide-se em API REST (Controllers), Lógica de Negócio (Services) e Acesso a Dados (Repositories).
- **Persistência**: Utiliza PostgreSQL DB para Schemas/Tabelas e armazena Dados de Ficha em formato JSON/JSONB.
- **Models**: Representam os Modelos de Dados (Entities), mapeados para o banco de dados via JPA.

## 3.2 Diagrama de Componentes e Implantação.

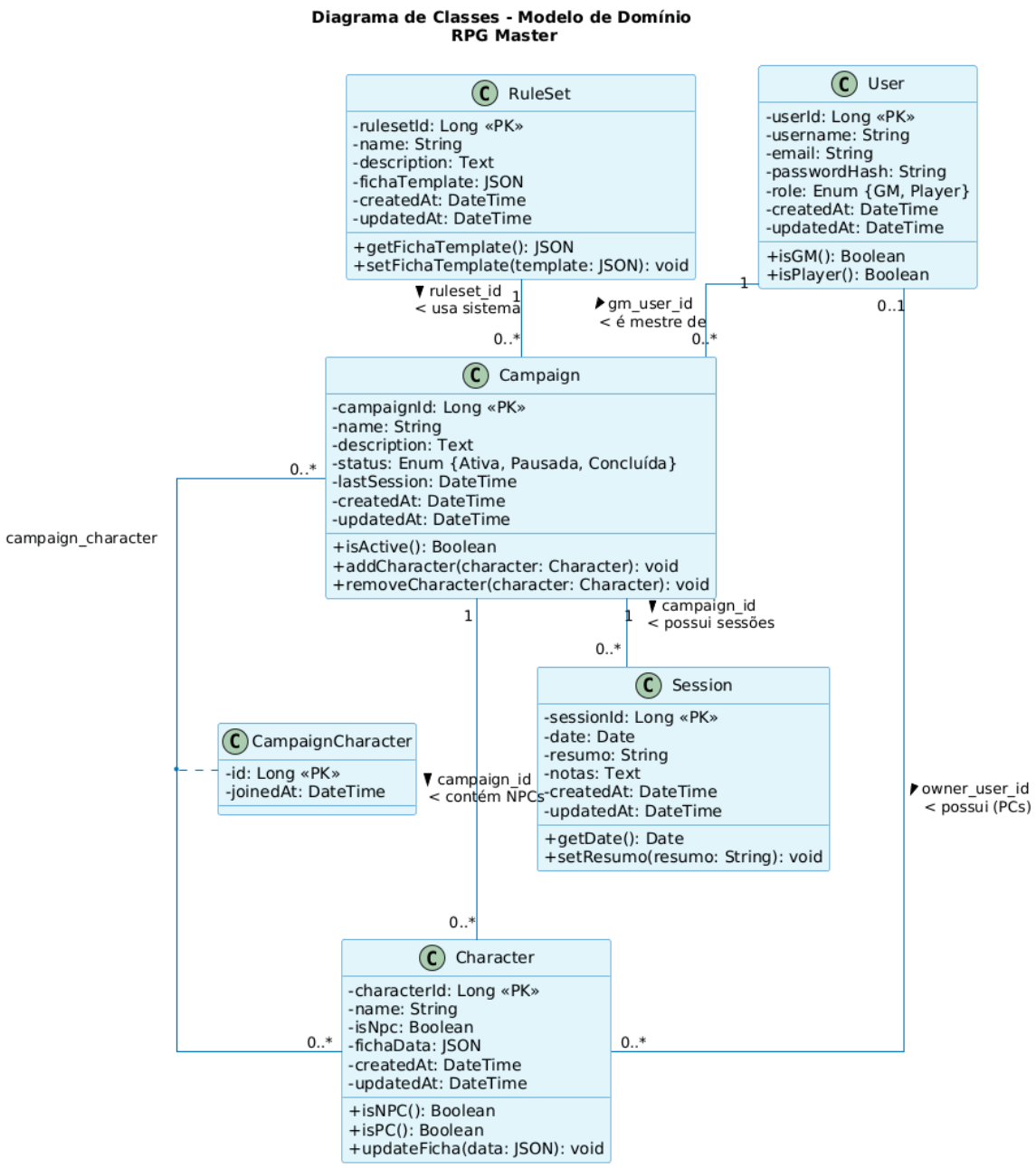





- **Frontend (Next.js)**: Servido no Vercel.
- **Backend (Spring Boot Application)**: Implantado no Render com Auto Scaling e Auto Deploy.
- **Database (PostgreSQL)**: Gerenciado como serviço no Supabase.

## 3.3 Diagrama de Classes

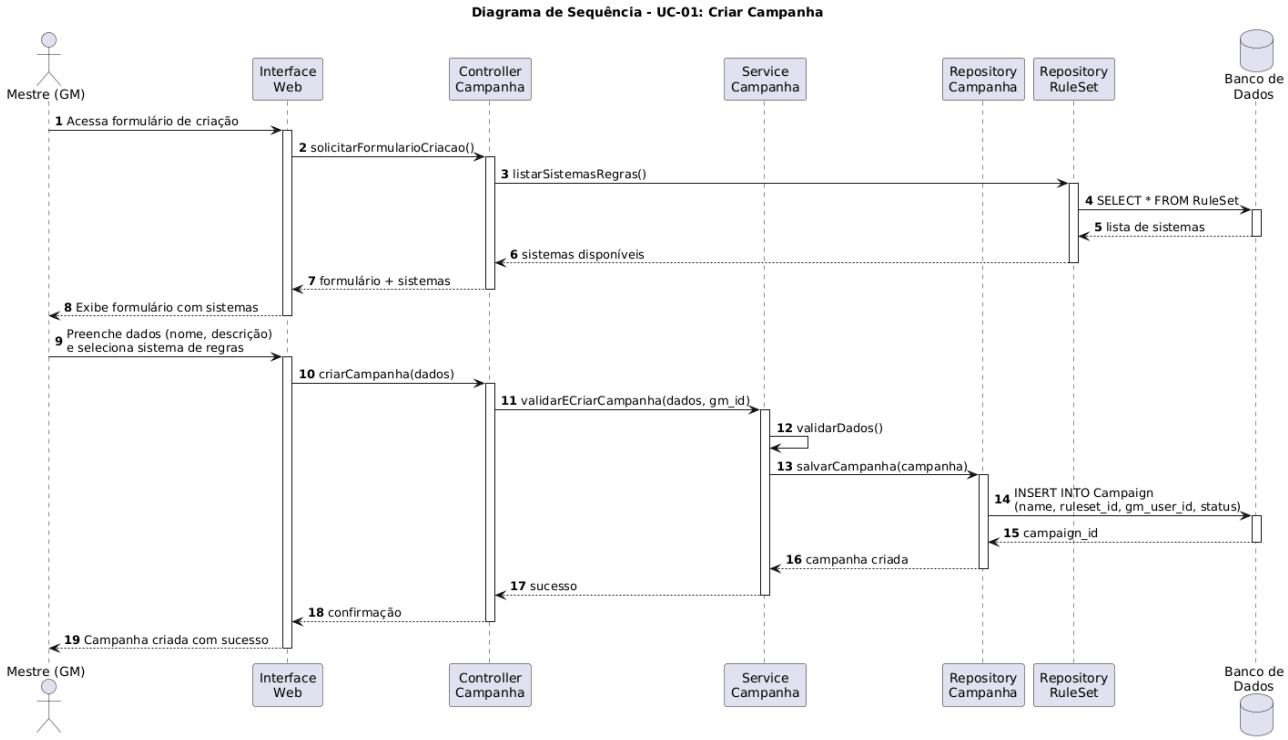
Diagrama de classes simplificado (Dominios). Diagrama completo na documentação do projeto no GitHub.




- **User**: role (GM, Player), isGM, isPlayer.
- **RuleSet**: Contém o fichaTemplate em JSON.
- **Campaign**: Possui status (Ativa, Pausada, Concluída), gm_user_id e associação com RuleSet.
- **Character**: Armazena fichaData em JSON e tem o atributo isNPC.
- **Session**: Registra data, resumo e notas para uma campanha.
- **CampaignCharacter**: Tabela de associação N:N entre Campaign e Character.

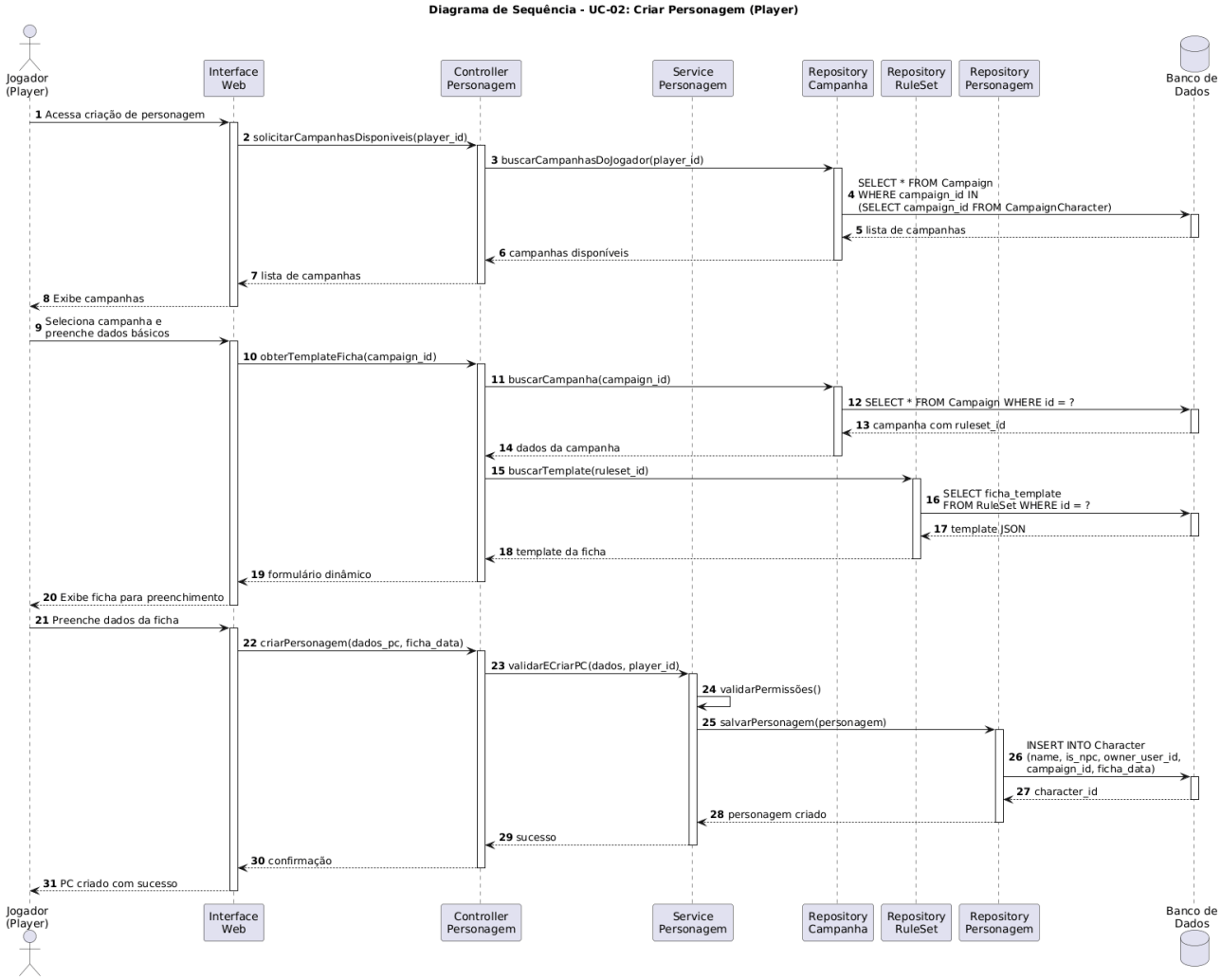
## 3.4 Diagramas de Sequência

# UC-01:



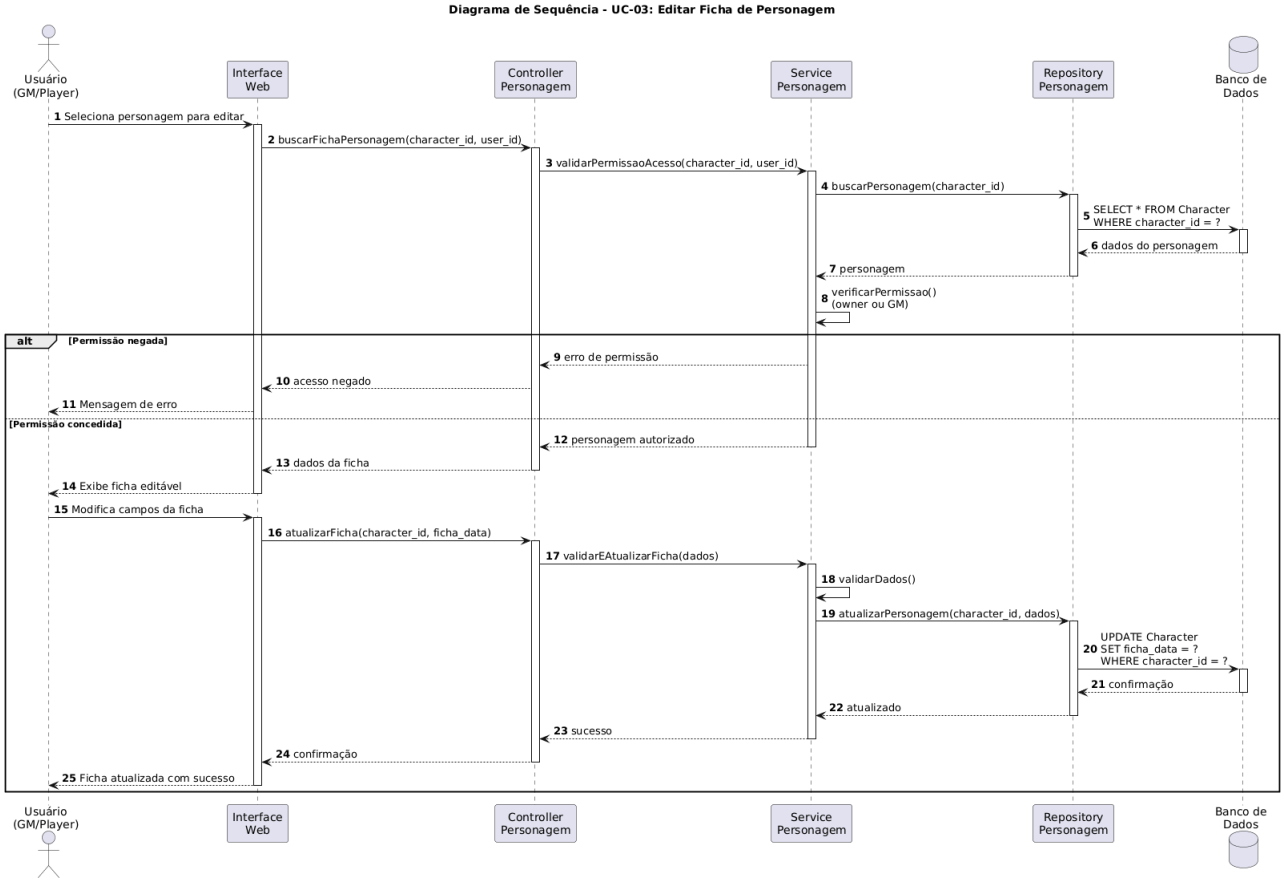

**Fluxo: GM solicita formulário, Controller lista sistemas (RepositoryRuleSet), exibe formulário. GM preenche, Controller chama Service, que valida e salva a Campaign (RepositoryCampaign).**

# UC-02:



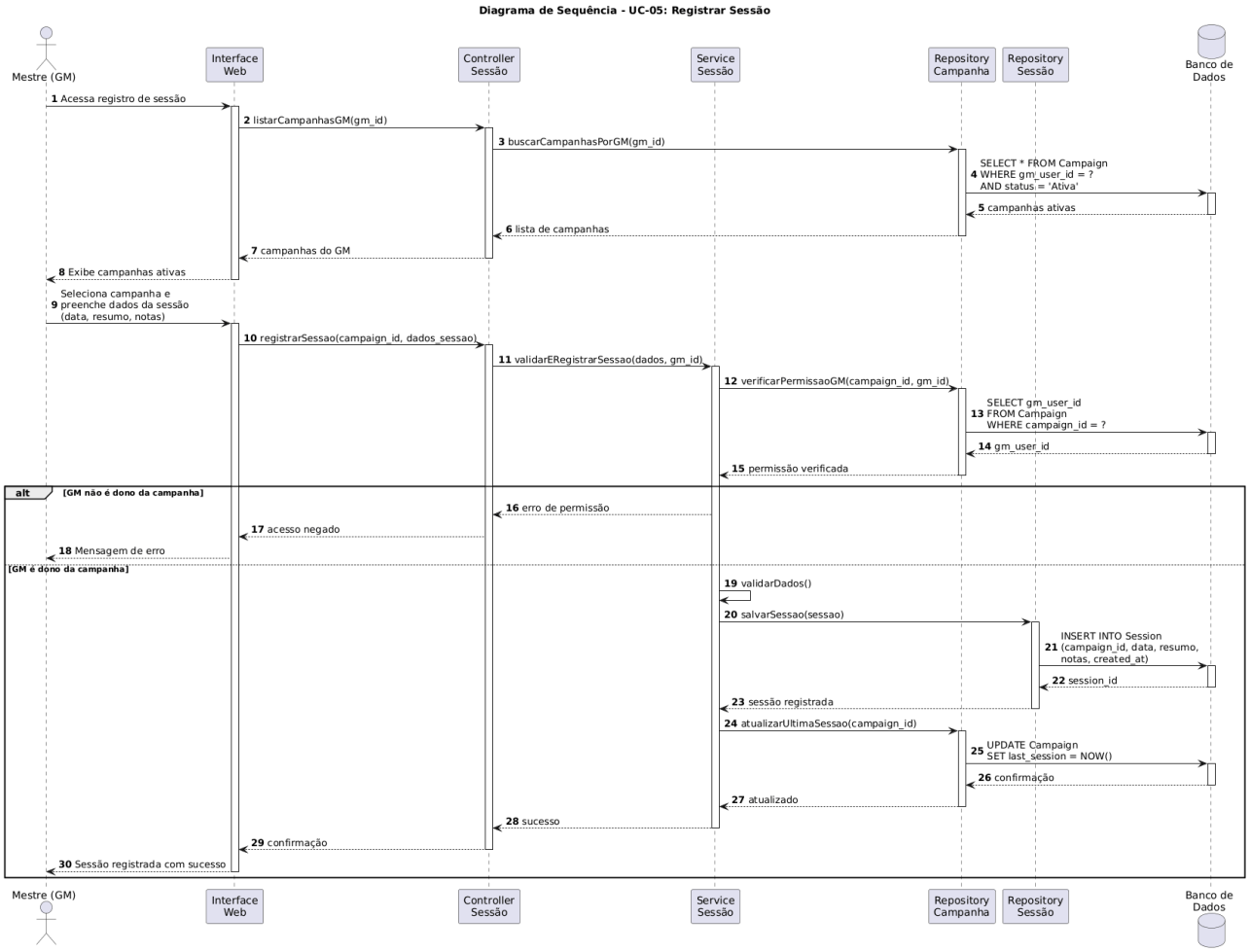

**Fluxo: Player acessa criação, Controller busca campanhas disponíveis. Player seleciona e solicita template, Controller busca RuleSet da campanha. Exibe formulário dinâmico. Player salva, Controller valida, Service persiste o Character.**

# UC-03:




**Fluxo: Usuário (GM/Player) seleciona personagem. Controller solicita validação de acesso (Service). Se Permissão Negada, exibe erro. Se Permissão Concedida, exibe dados. Usuário modifica. Controller chama Service para atualizar, que valida e persiste (RepositoryPersonagem).**

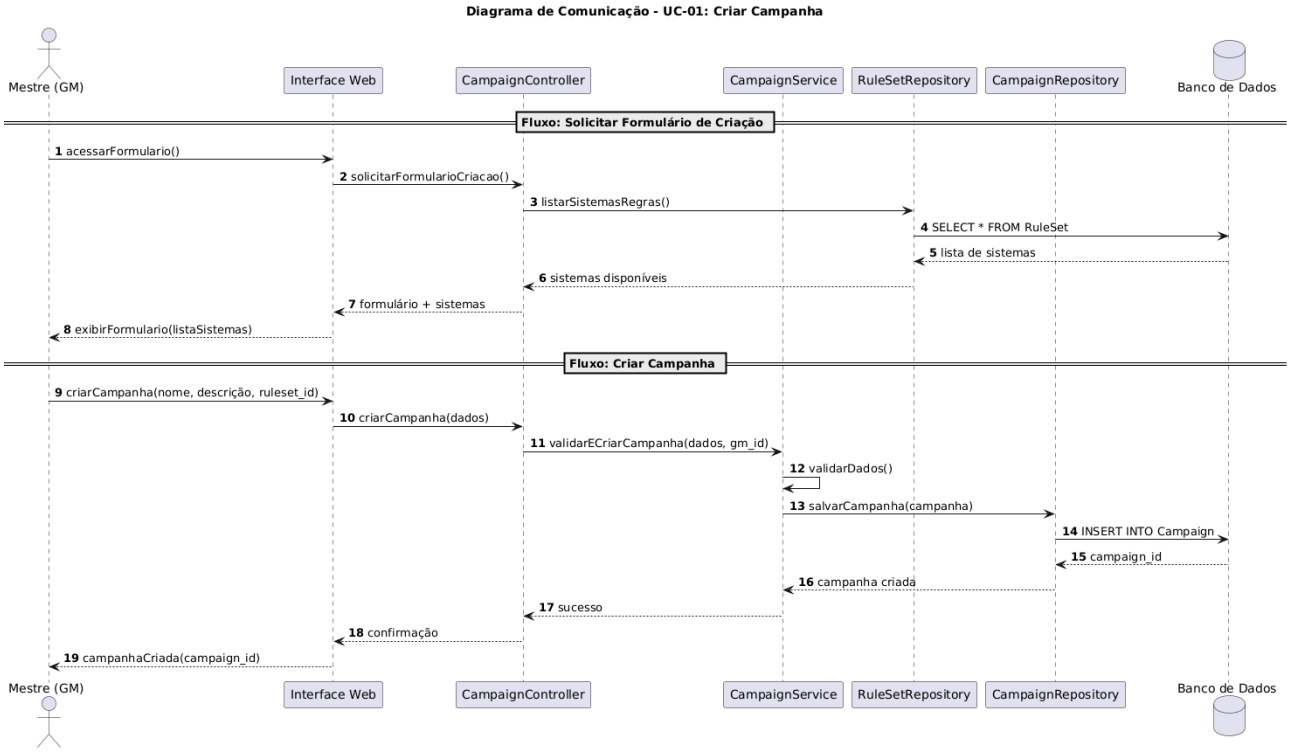
# UC-05:




**Fluxo: GM acessa registro. Controller lista campanhas ativas. GM seleciona e registra. Controller solicita validação de permissão (GM deve ser o dono da campanha). Se autorizado, Service valida dados, salva a Session (RepositorySession) e atualiza a data da última sessão na Campaign (RepositoryCampanha).**

## 3.5 Diagramas de Comunicação

# UC-01:



# UC-02:

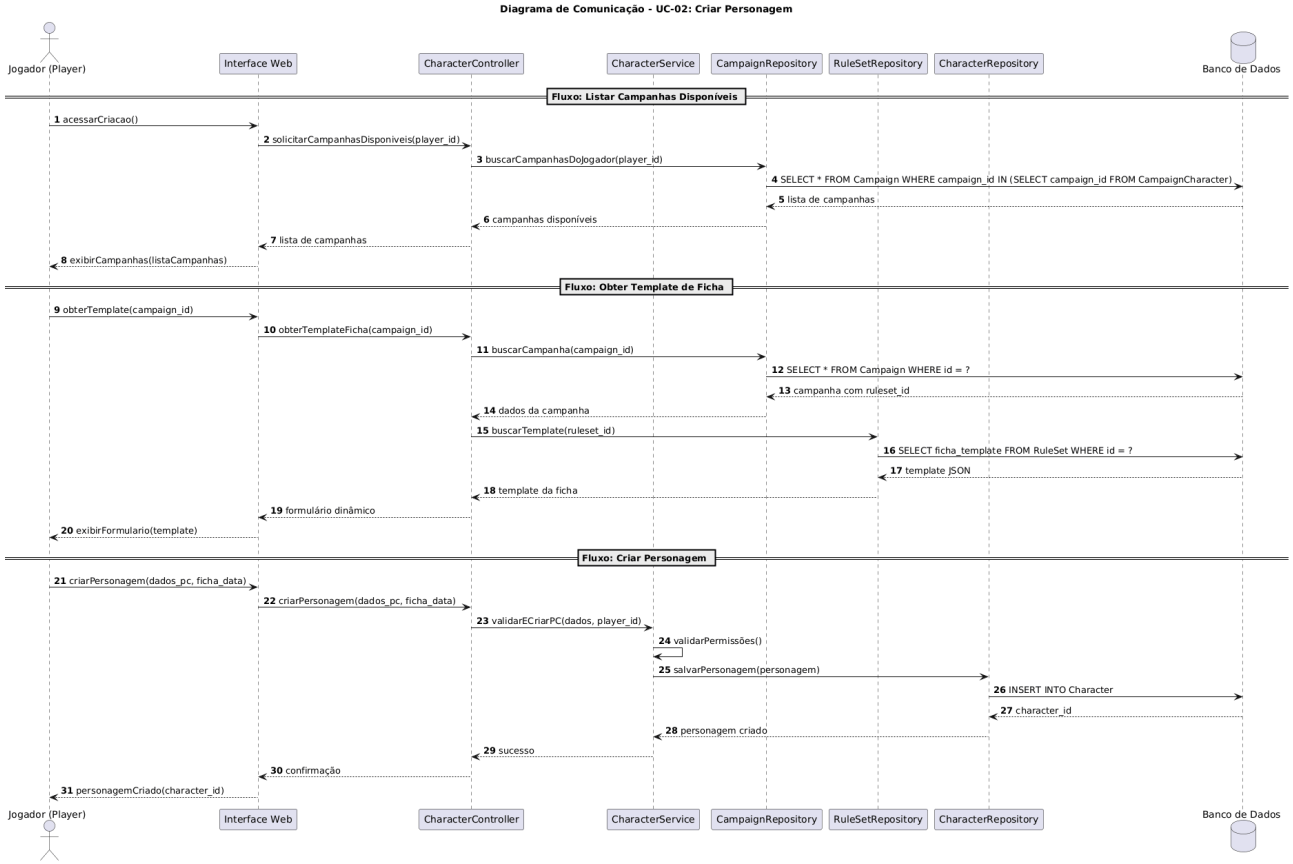


Diagrama de Comunicação - UC-02: Criar Personagem
Jogador (Player)
Interface Web
CharacterController
CharacterService
CampaignRepository
RuleSetRepository
CharacterRepository
Banco de Dados
Fluxo: Listar Campanhas Disponíveis
1 acessarCriacao()
2 solicitarCampanhasDisponiveis(player_id)
3 buscarCampanhasDoJogador(player_id)
4 SELECT * FROM Campaign WHERE campaign_id IN (SELECT campaign_id FROM CampaignCharacter)
5 lista de campanhas
6 campanhas disponíveis
7 lista de campanhas
8 exibirCampanhas(listaCampanhas)
Fluxo: Obter Template de Ficha
9 obterTemplate(campaign_id)
10 obterTemplateFicha(campaign_id)
11 buscarCampanha(campaign_id)
12 SELECT * FROM Campaign WHERE id = ?
13 campanha com ruleset_id
14 dados da campanha
15 buscarTemplate(ruleset_id)
16 SELECT ficha_template FROM RuleSet WHERE id = ?
17 template JSON
18 template da ficha
19 formulário dinâmico
20 exibirFormulario(template)
Fluxo: Criar Personagem
21 criarPersonagem(dados_pc, ficha_data)
22 criarPersonagem(dados_pc, ficha_data)
23 validarECriarPC(dados, player_id)
24 validarPermissões()
25 salvarPersonagem(personagem)
26 INSERT INTO Character
27 character_id
28 personagem criado
29 sucesso
30 confirmação
31 personagemCriado(character_id)

# UC-03:

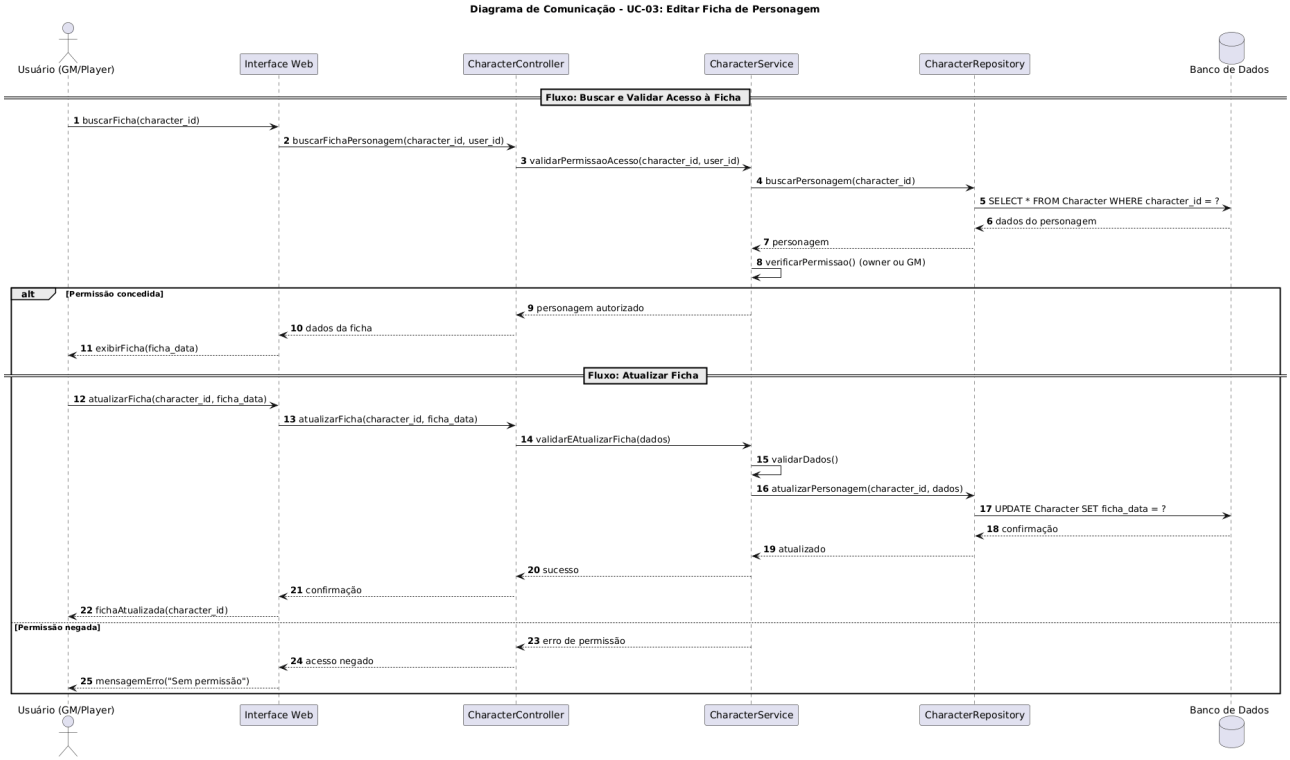


Diagrama de Comunicação - UC-03: Editar Ficha de Personagem
Usuário (GM/Player)
Interface Web
CharacterController
CharacterService
CharacterRepository
Banco de Dados
Fluxo: Buscar e Validar Acesso à Ficha
1 buscarFicha(character_id)
2 buscarFichaPersonagem(character_id, user_id)
3 validarPermissaoAcesso(character_id, user_id)
4 buscarPersonagem(character_id)
5 SELECT * FROM Character WHERE character_id = ?
6 dados do personagem
7 personagem
8 verificarPermissao() (owner ou GM)
alt
[Permissão concedida]
9 personagem autorizado
10 dados da ficha
11 exibirFicha(ficha_data)
Fluxo: Atualizar Ficha
12 atualizarFicha(character_id, ficha_data)
13 atualizarFicha(character_id, ficha_data)
14 validarEAtualizarFicha(dados)
15 validarDados()
16 atualizarPersonagem(character_id, dados)
17 UPDATE Character SET ficha_data = ?
18 confirmação
19 atualizado
20 sucesso
21 confirmação
22 fichaAtualizada(character_id)
[Permissão negada]
23 erro de permissão
24 acesso negado
25 mensagemErro("Sem permissão")

# UC-05:

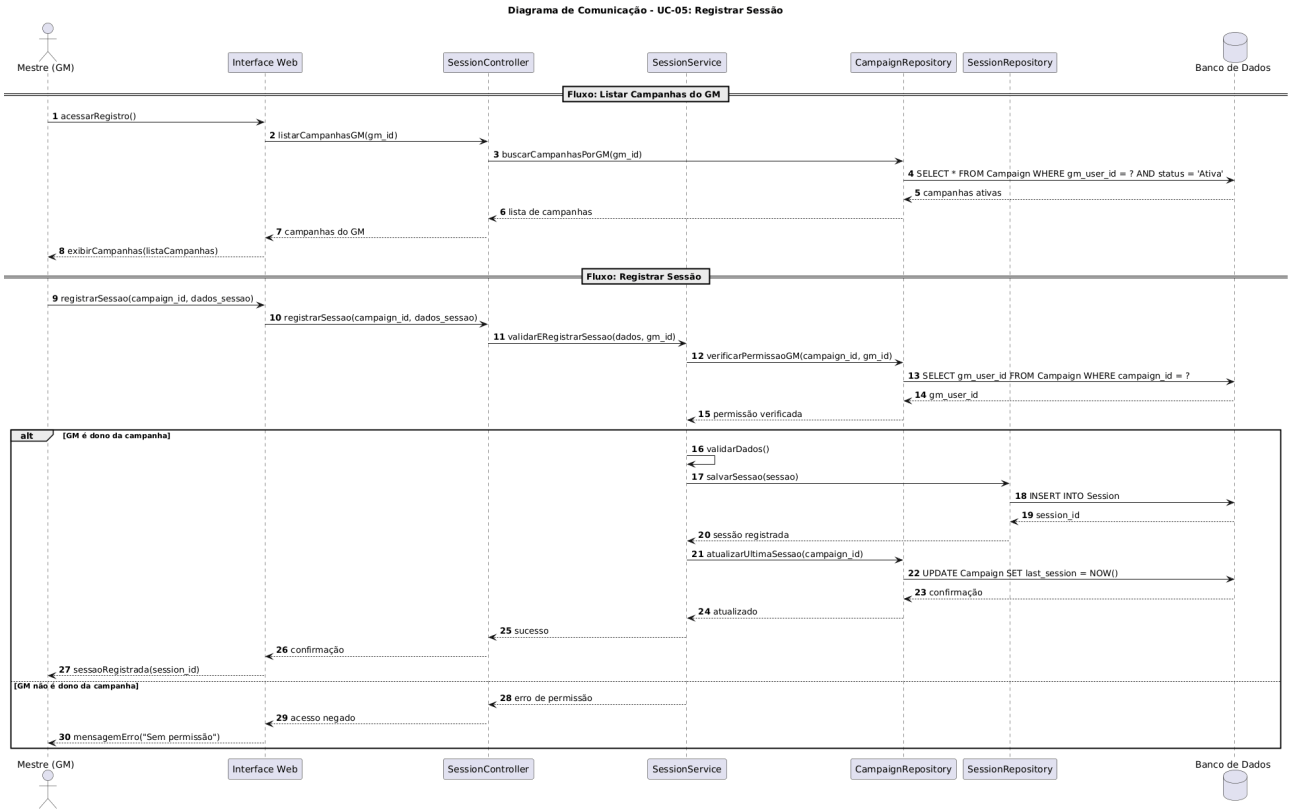
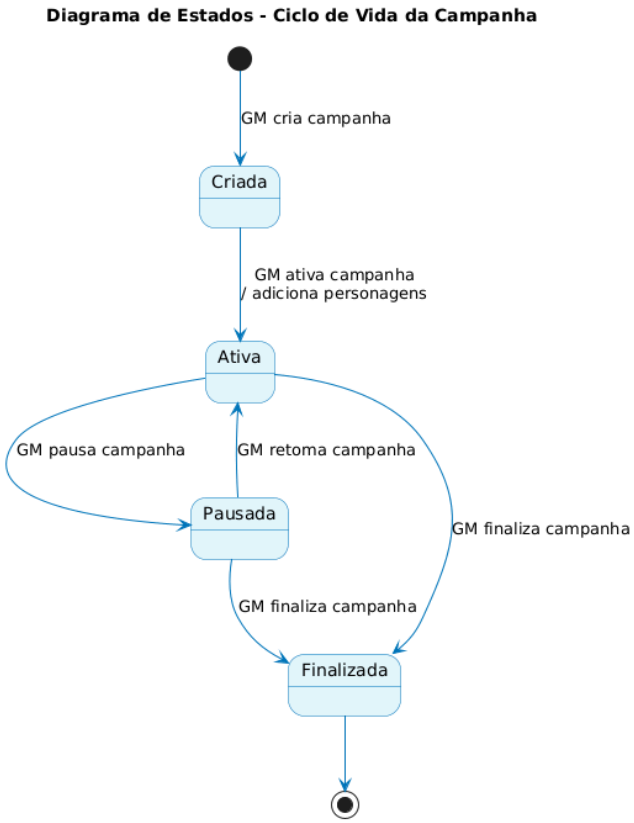


Diagrama de Comunicação - UC-05: Registrar Sessão
Mestre (GM)
Interface Web
SessionController
SessionService
CampaignRepository
SessionRepository
Banco de Dados
Fluxo: Listar Campanhas do GM
1 acessarRegistro()
2 listarCampanhasGM(gm_id)
3 buscarCampanhasPorGM(gm_id)
4 SELECT * FROM Campaign WHERE gm_user_id = ? AND status = 'Ativa'
5 campanhas ativas
6 lista de campanhas
7 campanhas do GM
8 exibirCampanhas(listaCampanhas)
Fluxo: Registrar Sessão
9 registrarSessao(campaign_id, dados_sessao)
10 registrarSessao(campaign_id, dados_sessao)
11 validarERegistrarSessao(dados, gm_id)
12 verificarPermissaoGM(campaign_id, gm_id)
13 SELECT gm_user_id FROM Campaign WHERE campaign_id = ?
14 gm_user_id
15 permissão verificada
alt
[GM é dono da campanha]
16 validarDados()
17 salvarSessao(sessao)
18 INSERT INTO Session
19 session_id
20 sessão registrada
21 atualizarUltimaSessao(campaign_id)
22 UPDATE Campaign SET last_session = NOW()
23 confirmação
24 atualizado
25 sucesso
26 confirmação
27 sessaoRegistrada(session_id)
[GM não é dono da campanha]
28 erro de permissão
29 acesso negado
30 mensagemErro("Sem permissão")

## 3.6 Diagramas de Estados



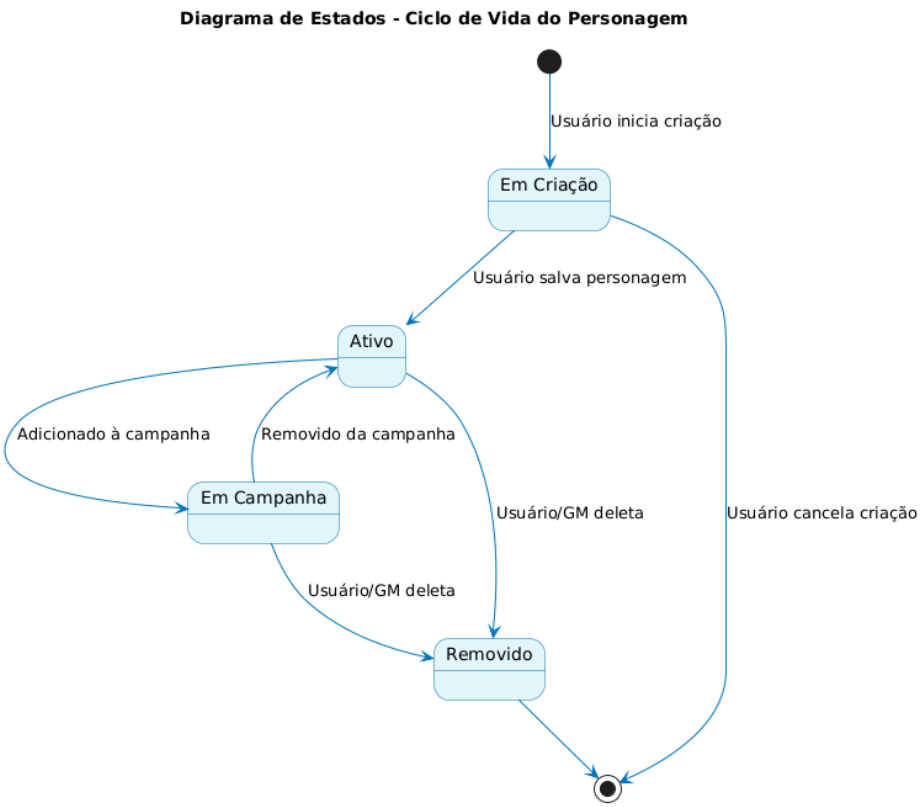
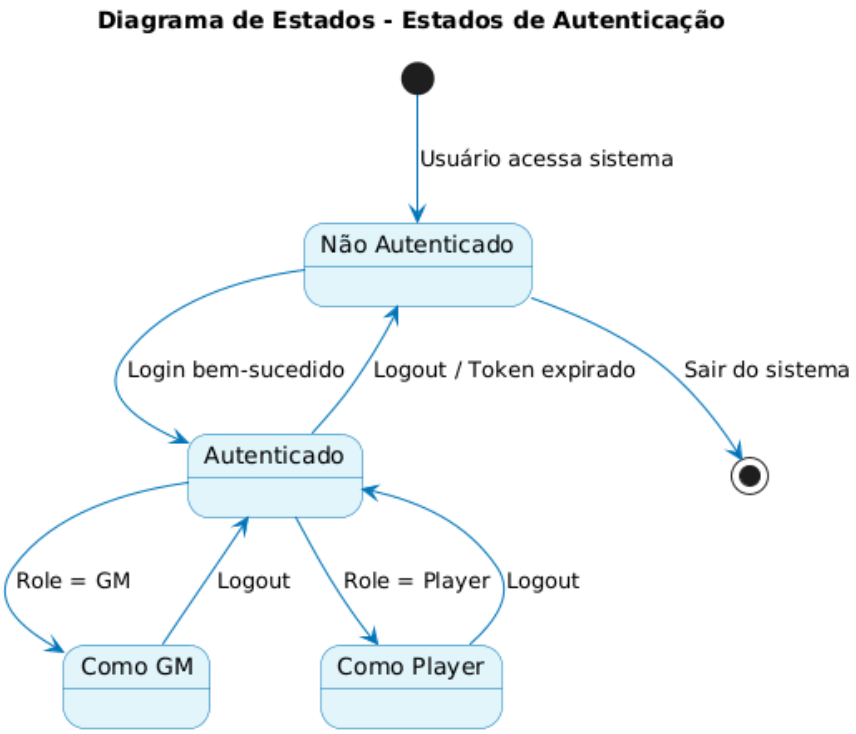



**Diagrama de Estados - Ciclo de Vida da Sessão**




# 4. Modelos de Dados