Documentação de Projeto Para o Sistema LearnUp

Versão 1.0

Projeto de sistema elaborado pelo(s) aluno(s): Islayder Jackson

Disciplina: Projeto de Software

Sumário

1. Introdução

Este documento agrega:

1. A elaboração e revisão de modelos de domínio

2. Modelos de projeto para o sistema **Sistema LearnUp**. 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

Nesta subseção é apresentado descrição de cada um dos atores que interagem com o sistema.

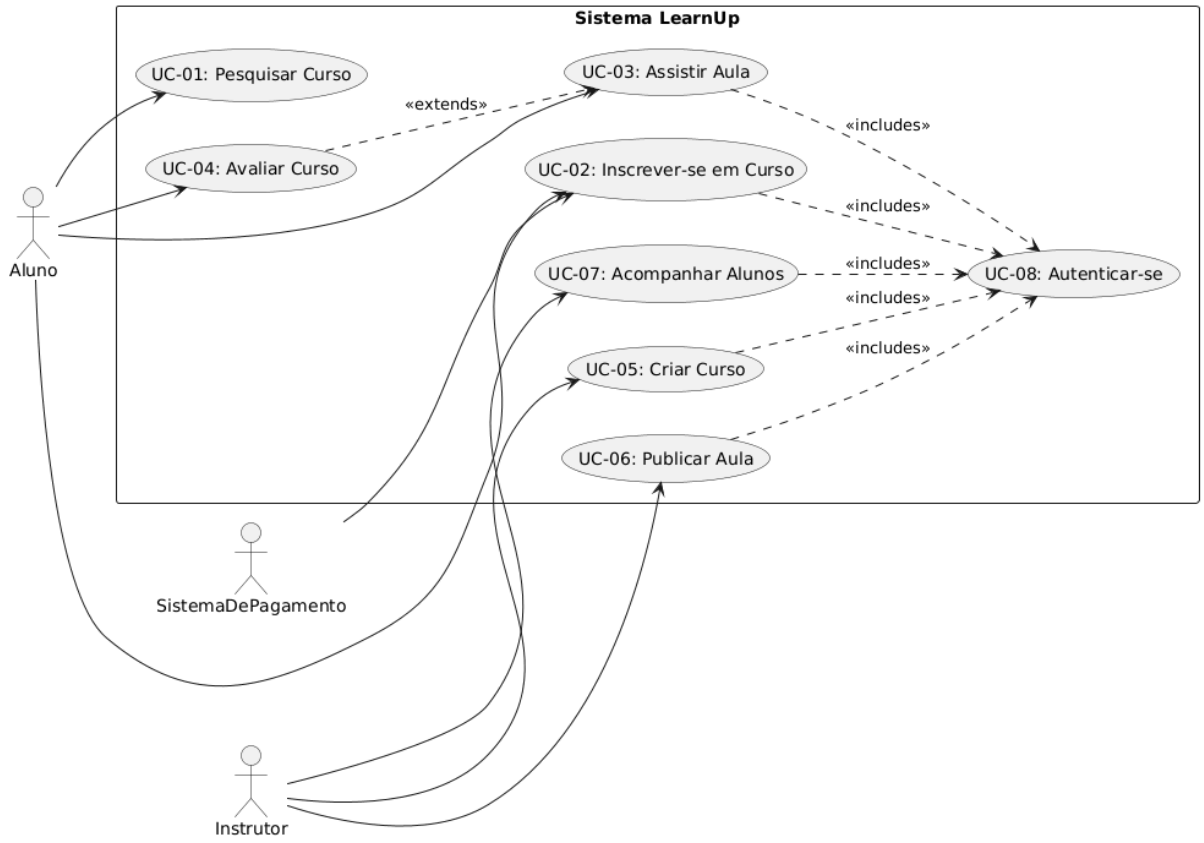
Ator: Aluno O Aluno é o consumidor do conteúdo. Seu objetivo principal é pesquisar cursos de seu interesse, realizar a inscrição e consumir os materiais (vídeos, textos) para seu aprendizado.

Ator: Instrutor O Instrutor é o criador do conteúdo. Seu objetivo principal é criar e gerenciar seus cursos, enviar os materiais das aulas e acompanhar o progresso de seus alunos.

Ator: SistemaDePagamento Ator secundário que representa um serviço externo (gateway de pagamento). É responsável por processar e validar as transações financeiras das inscrições.

2.2 Modelo de Casos de Uso

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



Código:

```
@startuml
' Define a direção do layout para ficar similar à imagem
left to right direction

' Definição dos Atores
actor Aluno
actor Instrutor
actor SistemaDePagamento

' Delimitação do Sistema
rectangle "Sistema LearnUp" {
  ' Definição dos Casos de Uso com aliases (nomes curtos)
  usecase "UC-01: Pesquisar Curso" as UC01
```

```
  usecase "UC-02: Inscrever-se em Curso" as UC02
  usecase "UC-03: Assistir Aula" as UC03
  usecase "UC-04: Avaliar Curso" as UC04
  usecase "UC-05: Criar Curso" as UC05
  usecase "UC-06: Publicar Aula" as UC06
  usecase "UC-07: Acompanhar Alunos" as UC07
  usecase "UC-08: Autenticar-se" as UC08
}

' Relacionamentos Ator -> Caso de Uso (Associações)
Aluno --> UC01
Aluno --> UC02
Aluno --> UC03
Aluno --> UC04

Instrutor --> UC05
Instrutor --> UC06
Instrutor --> UC07

' Relacionamento do sistema externo com o caso de uso
SistemaDePagamento --> UC02

' Relacionamentos entre Casos de Uso (Includes e Extends)

' Relação <<extends>>
' UC-04 (Avaliar Curso) estende UC-03 (Assistir Aula)
UC04 ..> UC03 : <<extends>>
```

' Relações <<includes>>

' Vários casos de uso incluem a autenticação

UC02 ..> UC08 : <<includes>>

UC03 ..> UC08 : <<includes>>

UC05 ..> UC08 : <<includes>>

UC06 ..> UC08 : <<includes>>

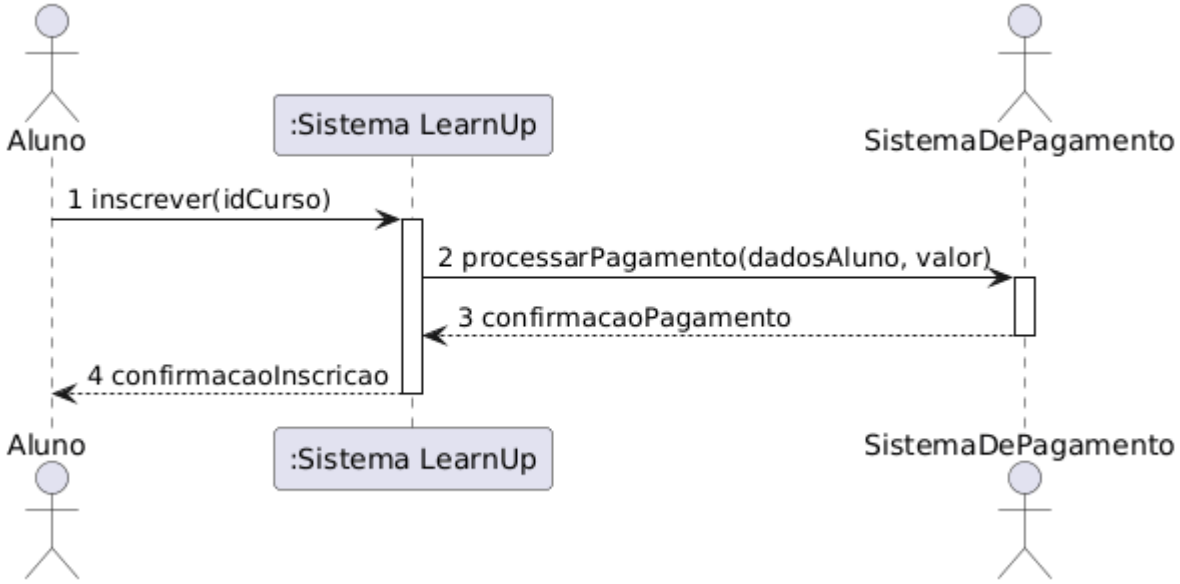
UC07 ..> UC08 : <<includes>>

@enduml

2.3 Diagrama de Sequência do Sistema

Nesta subseção é apresentado o diagrama de sequência do sistema.

Artefato 1: UC-02: Inscrever-se em Curso



Código:

@startuml

' Define os participantes

actor Aluno

participant ":Sistema LearnUp" as Sistema

actor SistemaDePagamento

' Início da sequência de mensagens

Aluno -> Sistema: 1 inscrever(idCurso)

activate Sistema

Sistema -> SistemaDePagamento: 2 processarPagamento(dadosAluno, valor)

activate SistemaDePagamento

' Mensagem de retorno do pagamento

SistemaDePagamento --> Sistema: 3 confirmacaoPagamento

deactivate SistemaDePagamento

' Mensagem de retorno para o aluno

Sistema --> Aluno: 4 confirmacaoInscricao

deactivate Sistema

@enduml

Formato para cada contrato de operação

Contrato Operação: inscrever

Referências cruzadas UC-02: Inscrever-se em Curso

Pré-condições

1. O Aluno deve estar autenticado (logado).
2. O Aluno não deve estar previamente inscrito no curso referido por idCurso.

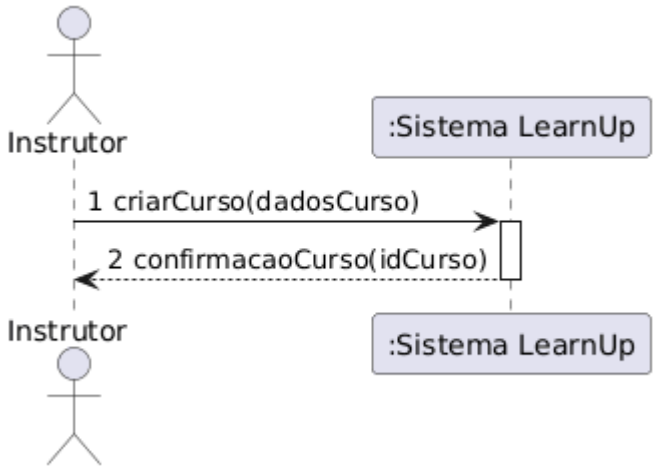
Pós-condições

1. Uma nova Inscrição foi criada, associando o Aluno ao Curso.

2. O pagamento foi processado e registrado pelo SistemaDePagamento.

3. O Aluno agora está na lista de alunos do Curso.

Artefato 2: UC-05: Criar Curso



Código:

```
@startuml

' Define os participantes

actor Instrutor

participant ":Sistema LearnUp" as Sistema



' Início da sequência de mensagens

Instrutor -> Sistema: 1 criarCurso(dadosCurso)

activate Sistema



' Mensagem de retorno

Sistema --> Instrutor: 2 confirmacaoCurso(idCurso)
```

deactivate Sistema

@enduml

Contrato Operação: criarCurso

Referências cruzadas UC-05: Criar Curso

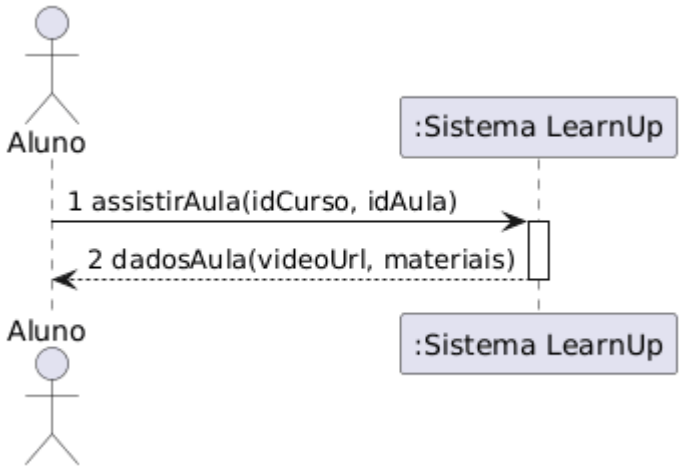
Pré-condições

1. O usuário deve estar autenticado (logado).
2. O usuário deve ter permissões de Instrutor.

Pós-condições

1. Uma nova instância de Curso foi criada no sistema.
2. O Instrutor que fez a chamada foi associado como o proprietário do Curso.
3. O Curso é criado em um estado "Rascunho".

Artefato 3: UC-03: Assistir Aula



Código:

@startuml

' Define os participantes

```
actor Aluno

participant ":Sistema LearnUp" as Sistema


' Início da sequência de mensagens

Aluno -> Sistema: 1 assistirAula(idCurso, idAula)

activate Sistema


' Mensagem de retorno

Sistema --> Aluno: 2 dadosAula(videoUrl, materiais)

deactivate Sistema


@enduml
```

Contrato Operação: assistirAula

Referências cruzadas UC-03: Assistir Aula

Pré-condições

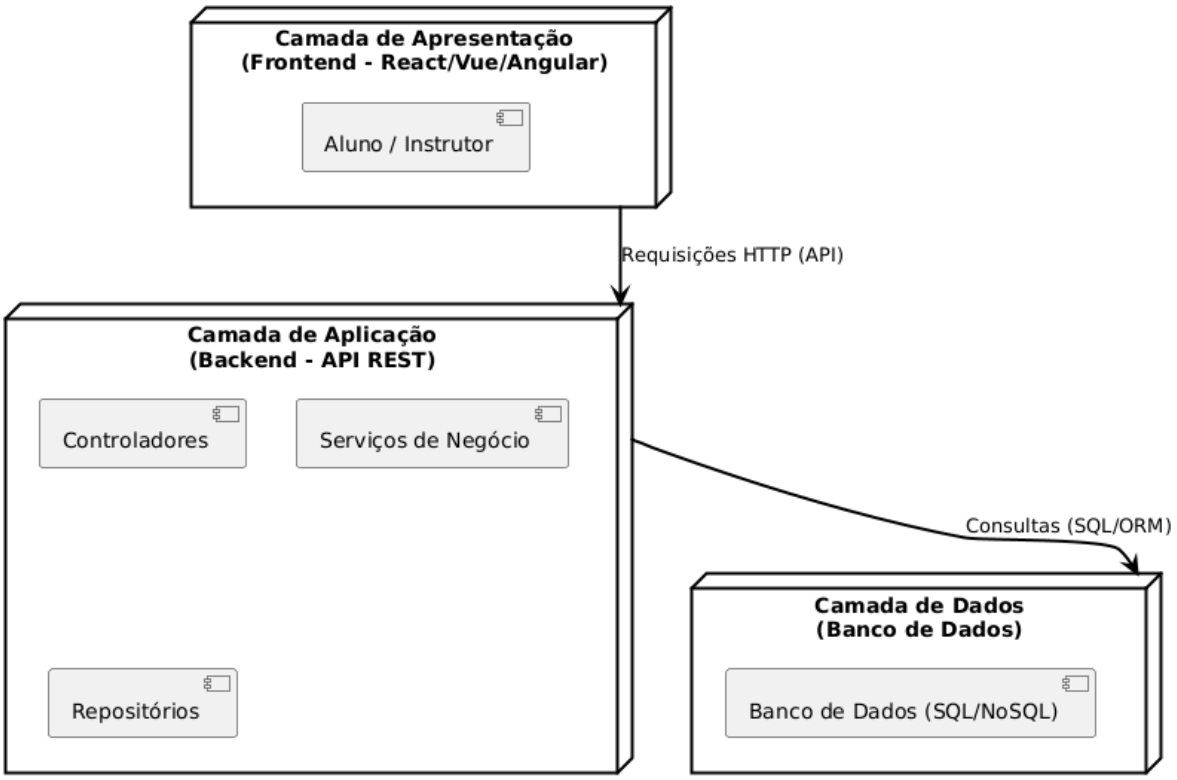
1. O Aluno deve estar autenticado (logado).
2. O Aluno deve estar inscrito no curso (idCurso).

Pós-condições

1. O conteúdo da aula é retornado para o Aluno.
2. O sistema registra o progresso do Aluno.

3. Modelos de Projeto

3.1 Arquitetura



Código:

@startuml

' Define a direção do layout

top to bottom direction

' Remove o ícone padrão de "pacote" e "componente" para ficar mais limpo

skinparam package {

  StereotypeFontColor transparent

}

skinparam component {

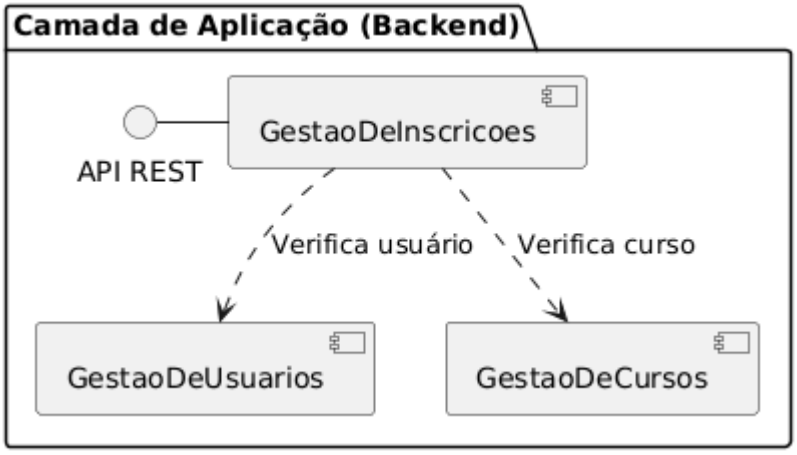
```
   StereotypeFontColor transparent
}

' Camada de Apresentação (Frontend)
package "Camada de Apresentação\n(Frontend - React/Vue/Angular)" as Frontend {
  component [Aluno / Instrutor]
}

' Camada de Aplicação (Backend)
package "Camada de Aplicação\n(Backend - API REST)" as Backend {
  component [Controladores]
  component [Serviços de Negócio]
  component [Repositórios]
}

' Camada de Dados (Banco de Dados)
package "Camada de Dados\n(Banco de Dados)" as DB {
  component [Banco de Dados (SQL/NoSQL)]
}

' Relacionamentos entre as camadas
Frontend --> Backend : "Requisições HTTP (API)"
Backend --> DB : "Consultas (SQL/ORM)"
```

@enduml

3.2 Diagrama de Componentes e Implantação.

Diagramas de componentes do sistema. Diagrama de implantação mostrando onde os componentes estarão alocados para a execução.

Diagrama de Componentes:



Código:

@startuml

' Versão alternativa mais provável

skinparam component {

StereotypeFontColor transparent

}

package "Camada de Aplicação (Backend)" {

interface "API REST" as API

component [GestaoDeInscricoes]

component [GestaoDeUsuarios]

component [GestaoDeCursos]

' GestaoDeInscricoes expõe/implementa a API

API - [GestaoDeInscricoes]

' Dependências entre componentes

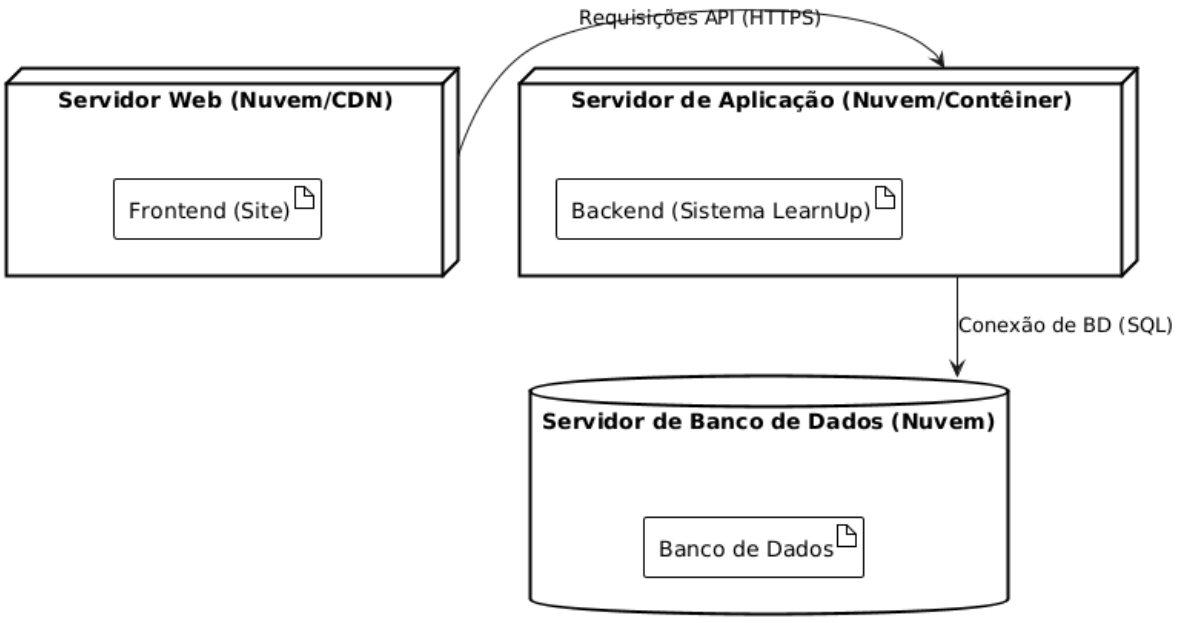
[GestaoDeInscricoes] ..> [GestaoDeUsuarios] : "Verifica usuário"

[GestaoDeInscricoes] ..> [GestaoDeCursos] : "Verifica curso"

}

@enduml

Diagrama de Implantação:



Código:

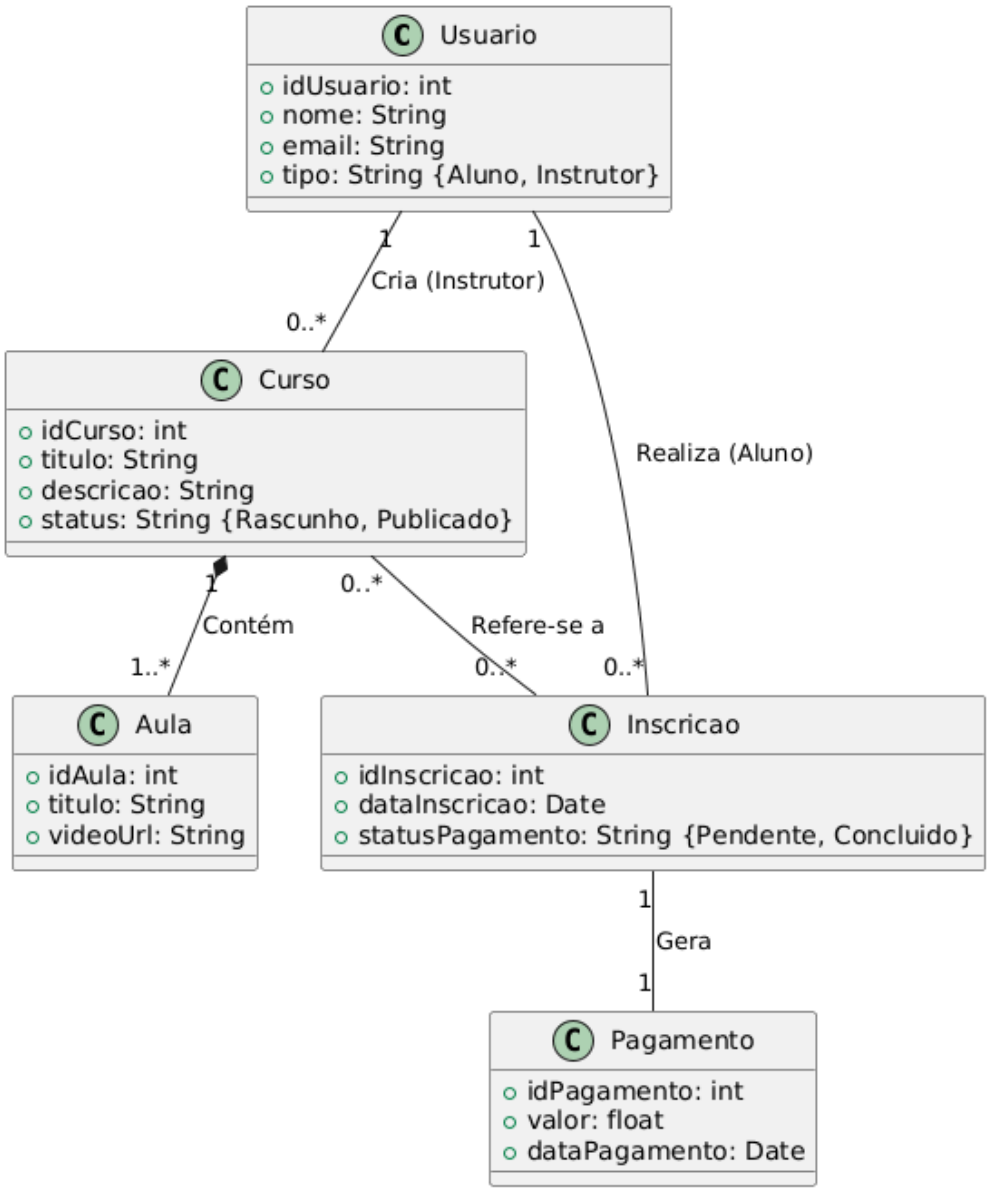
@startuml

' Define a direção do layout (embora o PlantUML organize nós automaticamente)

```
' Define os nós (servidores)
node "Servidor Web (Nuvem/CDN)" as ServidorWeb {
  artifact "Frontend (Site)" as Frontend
}

node "Servidor de Aplicação (Nuvem/Contêiner)" as ServidorApp {
  artifact "Backend (Sistema LearnUp)" as Backend
}

database "Servidor de Banco de Dados (Nuvem)" as ServidorBD {
  artifact "Banco de Dados"
}

' Define os relacionamentos (links) entre os nós
ServidorWeb --> ServidorApp : "Requisições API (HTTPS)"
ServidorApp --> ServidorBD : "Conexão de BD (SQL)"

@enduml
```

3.3 Diagrama de Classes

Diagrama de classes do sistema.



Código:

@startuml ' Configuração para remover o ícone 'C' e combinar com o estilo da imagem skinparam class { StereotypeFontColor transparent StereotypeCBackgroundColor transparent }

' Definição das Classes class Usuario { + idUsuario: int + nome: String + email: String + tipo: String {Aluno, Instrutor} }

class Curso { + idCurso: int + titulo: String + descricao: String + status: String {Rascunho, Publicado} }

class Aula { + idAula: int + titulo: String + videoUrl: String }

class Inscricao { + idInscricao: int + dataInscricao: Date + statusPagamento: String {Pendente, Concluido} }

class Pagamento { + idPagamento: int + valor: float + dataPagamento: Date }

' DefiniSção dos Relacionamentos ' A 'o' no diagrama é representada como 'public' (+) no PlantUML

' Relação Instrutor -> Curso Usuario "1" -- "0..*" Curso : "Cria (Instrutor)"

' Relação Aluno -> Inscrição Usuario "1" -- "0..*" Inscricao : "Realiza (Aluno)"

' Relação Curso -> Aula (Composição) Curso "1" -- *"1..*" Aula : "Contém"

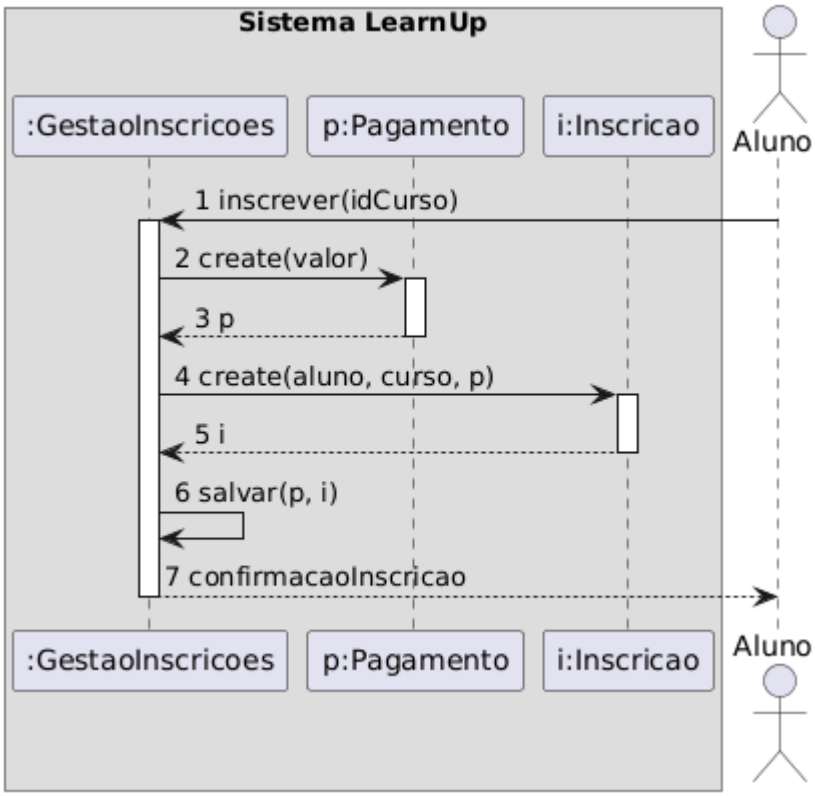
' Relação Curso <-> Inscrição ' A imagem mostra uma relação M-N (0..* para 0..*) Curso "0..*" -- "0..*" Inscricao : "Refere-se a"

' Relação Inscrição -> Pagamento Inscricao "1" -- "1" Pagamento : "Gera"

@enduml

3.4 Diagramas de Sequência

Diagramas de sequência para realização de casos de uso. (Detalhando UC-02)



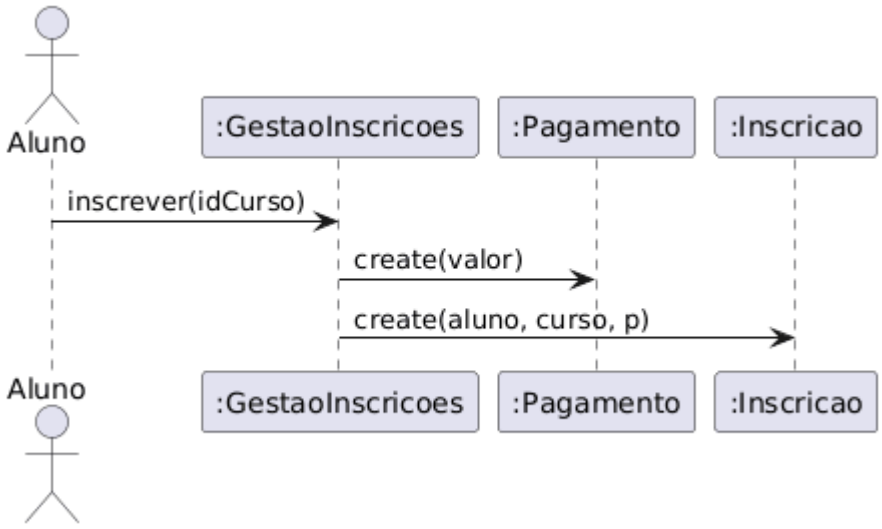
Código:

```
@startuml

' Delimita o sistema

box "Sistema LearnUp"

   participant ":GestaoInscricoes" as Gestao

   participant "p:Pagamento" as Pagamento

   participant "i:Inscricao" as Inscricao

end box


' Define o ator externo

actor Aluno


' Início da sequência de mensagens
```

```
Aluno -> Gestao: 1 inscrever(idCurso)

activate Gestao


' Criação do Pagamento

Gestao -> Pagamento: 2 create(valor)

activate Pagamento

Pagamento --> Gestao: 3 p

deactivate Pagamento


' Criação da Inscrição

Gestao -> Inscricao: 4 create(aluno, curso, p)

activate Inscricao

Inscricao --> Gestao: 5 i

deactivate Inscricao


' Salvar

Gestao -> Gestao: 6 salvar(p, i)


' Confirmação final

Gestao --> Aluno: 7 confirmacaoInscricao

deactivate Gestao


@enduml
```

3.5 Diagramas de Comunicação

Diagramas de comunicação para realização de casos de uso. (Detalhando UC-02)



Código:

@startuml

' Define os participantes

actor Aluno

participant ":GestaoInscricoes" as Gestao

participant ":Pagamento" as Pagamento

participant ":Inscricao" as Inscricao

' Sequência de mensagens

Aluno -> Gestao: inscrever(idCurso)

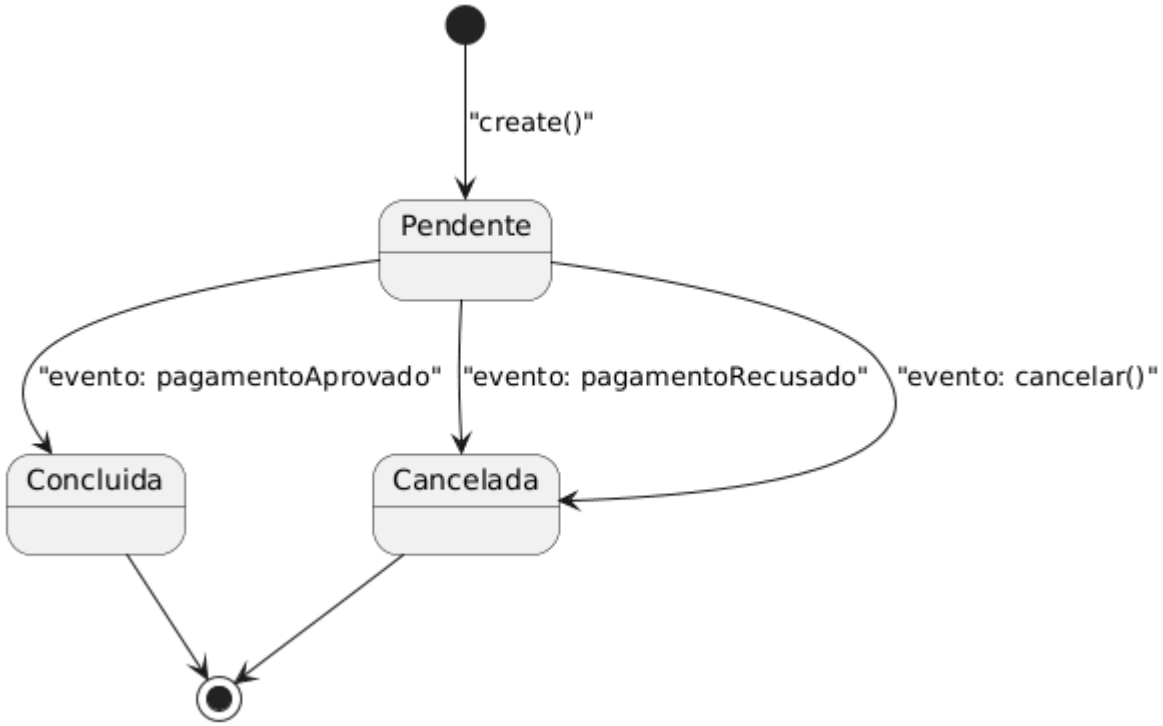
Gestao -> Pagamento: create(valor)

Gestao -> Inscricao: create(aluno, curso, p)

@enduml

3.6 Diagramas de Estados

Diagramas de estados do sistema. (Ciclo de Vida do objeto Inscricao)



Código:

@startuml

' Define os estados

state Pendente

state Concluida

state Cancelada

' Define as transições

[*] --> Pendente : "create()"

Pendente --> Concluida : "evento: pagamentoAprovado"

Pendente --> Cancelada : "evento: pagamentoRecusado"

Pendente --> Cancelada : "evento: cancelar()"

Concluida --> [*]

Cancelada --> [*]

' Nota: A imagem original continha um estado "Concluída" duplicado

' (sem conexões ou com uma seta dividida de "Pendente"),

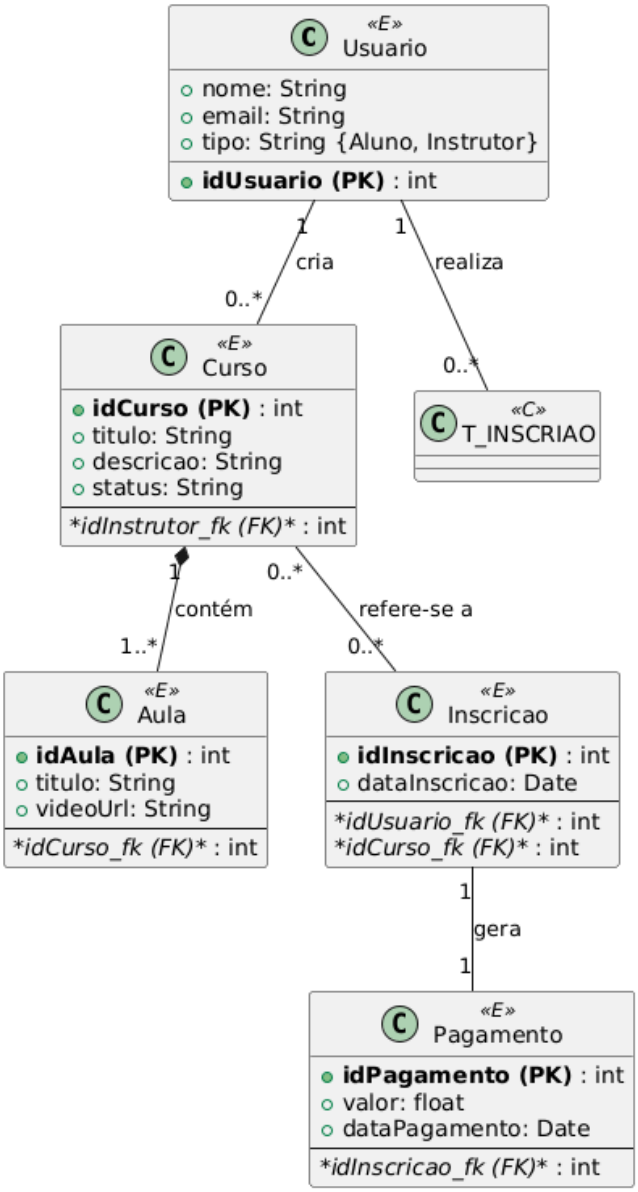
' o que parece ser um erro de diagramação.

' Este código representa a lógica funcional com um único estado "Concluida".

@enduml

4. Modelos de Dados

Esquema de banco de dados e as estratégias de mapeamento entre as representações de objetos e não-objetos.



Código:

@startuml ' Configuração para estereótipos (E, C) skinparam class { StereotypeCBackgroundColor transparent StereotypeEBackgroundColor transparent }

' Definição das Classes class Usuario <> { + **idUsuario (PK)** : int + nome: String + email: String + tipo: String {Aluno, Instrutor} }

class Curso <> { + **idCurso (PK)** : int + titulo: String + descricao: String + status: String --- *idInstrutor_fk (FK)* : int }

class Aula <> { + **idAula (PK)** : int + titulo: String + videoUrl: String --- *idCurso_fk (FK)* : int }

class Inscricao <> { + **idInscricao (PK)** : int + dataInscricao: Date --- *idUsuario_fk (FK)* : int *idCurso_fk (FK)* : int }

class Pagamento <> { + **idPagamento (PK)** : int + valor: float + dataPagamento: Date --- *idInscricao_fk (FK)* : int }

' A classe T_INSCRIAO (com estereótipo C) class T_INSCRIAO <> { ' Sem atributos visíveis na imagem }

' Definição dos Relacionamentos Usuario "1" -- "0.." *Curso : "cria" Usuario "1" -- "0.."* T_INSCRIAO : "realiza"

Curso "1" -- *"1.."* Aula : "contém" Curso "0.." -- *"0.."* Inscricao : "refere-se a"

Inscricao "1" -- "1" Pagamento : "gera"

@enduml