Academicci

Documento de Arquitetura de Software

Versão 0.003

Histórico de Revisão

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| 19/10/17 | 0.000 | Criação do documento | Thaís Tavares |
| 09/11/17 | 0.001 | Inserção diagrama de caso de uso e modelo MVC | Thaís Tavares |
| 03/03/18 | 0.002 | Adequação de template | Raphael Guedes |
| 24/04/2018 | 0.003 | Refinamento do documento:  - Visão de Casos de Uso;  - Visão de Lógica;  - Visão de Implementação;  - Visão de Dados | Raphael Guedes |

Sumário

1. Introdução 4

1.1 Finalidade 4

1.2 Escopo 4

1.3 Referências 4

1.4 Visão Geral 4

2. Representação da Arquitetura 4

3. Metas e Restrições da Arquitetura 5

4. Visão de Casos de Uso 6

4.1 Realizações dos Casos de Uso 6

5. Visão Lógica 7

5.1 Visão Geral 7

5.2 Pacotes de Design Significativos do Ponto de Vista da Arquitetura 8

6. Visão de Implementação 8

6.1 Visão Geral 8

6.2 Camadas 8

6.2.1 View 8

6.2.2 Controller 8

6.2.3 Model 8

7. Visão de Dados 9

7.1 Diagrama Entidade Relacionamento – DER 9

7.2 Modelo Entidade Relacionamento – MER 9

8. Tamanho e Desempenho 9

9. Qualidade 9

Documento de Arquitetura de Software

# Introdução

## Finalidade

Este documento fornece uma visão arquitetural abrangente do sistema, usando diversas visões de arquitetura para representar diferentes aspectos do sistema. Ele pretende capturar e transmitir as decisões arquiteturas significativas que foram tomadas em relação ao sistema.

## Escopo

Este documento é destinando a toda equipe de desenvolvimento e serve como guia para a visão arquitetural de todo o sistema, assim como dos requisitos de implementação do sistema.

## Referências

* Disciplina de Arquitetura de Software OpenUP v 1.0. Disponível em: <http://www.hectordufau.com.br/openup/index.htm>
* Documento de Glossário. Disponível em: <https://github.com/Fabrica-de-Software/Academicci_II/wiki/Documento-de-Gloss%C3%A1rio>
* Documento de Plano de Projeto

## Visão Geral

São apresentados neste documento diferentes visões arquiteturais de como o sistema deve se comportar em diferentes processos, como deve ser implantado e implementado e restrições de desempenho e qualidade.

# Representação da Arquitetura

[Esta seção descreve qual é a arquitetura de software do sistema atual e como ela é representada. Dos **Casos de Uso**, **Implementação**, **Processo** , **Lógica**e **Visualizações de Implementação**, ela enumera as visualizações necessárias e, para cada uma, explica que tipos de elementos de modelos a mesma contém.]

O projeto Academicci usará a arquitetura MVC, que divide o sistema em três módulos: Visão, responsável pela interação com o usuário. Controller, responsável pela comunicação entre a camada de persistência e as requisições do usuário. Model, a camada de persistência de dados, que interage diretamente com o bando de dados.

* Visão de Caso de Uso

Apresenta as funcionalidades arquiteturais importantes e os usuários do sistema.

* Visão Lógica

Descreve classes e sua organização e apresenta o padrão de arquitetura que deverá ser utilizado para desenvolvimento do sistema.

* Visão de Processos

Mostra o padrão de comportamento do sistema diante de diferentes ações do usuário.

* Visão de Implantação
  + Descreve a estrutura do ambiente onde o software será instalado.
* Visão de Implementação
  + Ilustra a distribuição do processo em um conjunto de nós do sistema, incluindo a distribuição física de processos e threads.

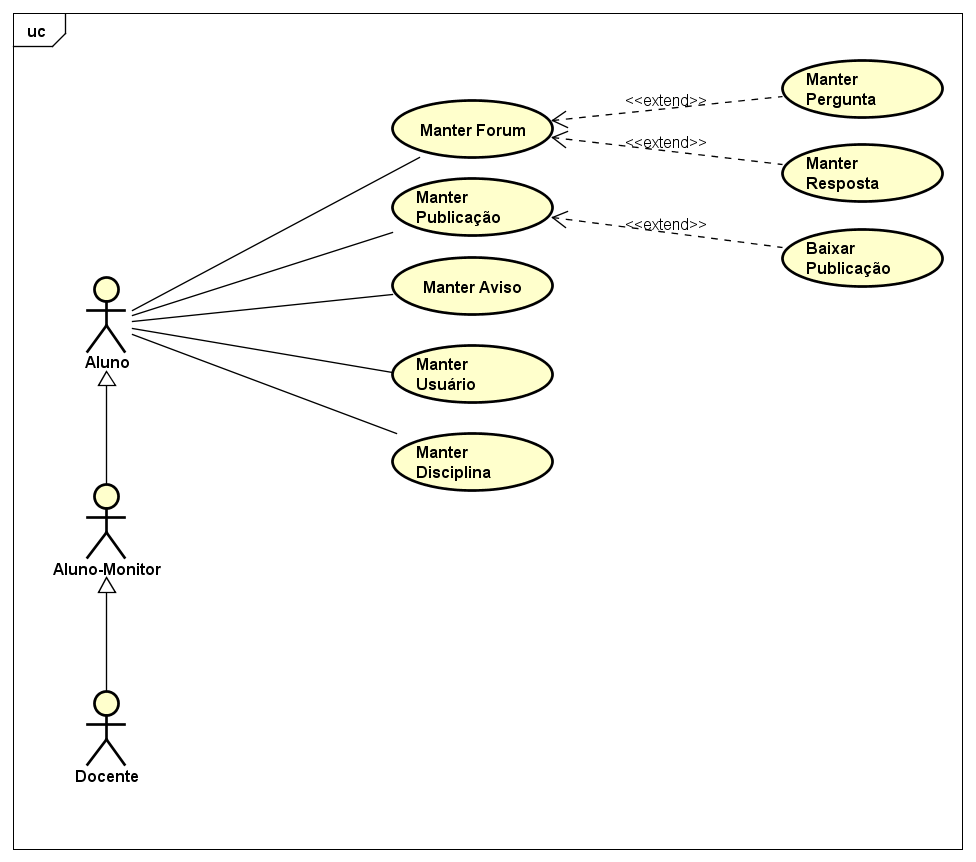
# Metas e Restrições da Arquitetura

O sistema será desenvolvido em plataforma *web* de modo responsivo, para que seja possível sua utilização em dispositivos mobile através de um navegador web.

O projeto será implementado usando a linguagem de programação Java EE. Contará com frameworks de desenvolvimento back-end, front-end e de banco de dados. O framework back-end será o Java Server Faces (JSF), versão 2.2.12. O framework front-end, será o Primefaces, versão 6.2. E o framework de mapeamento objeto relacional será o Hibernate 4.3.11.

O software deverá manter as características de segurança, desempenho, portabilidade e confiabilidade para que atenda aos requisitos de qualidade do usuário.

# Visão de Casos de Uso

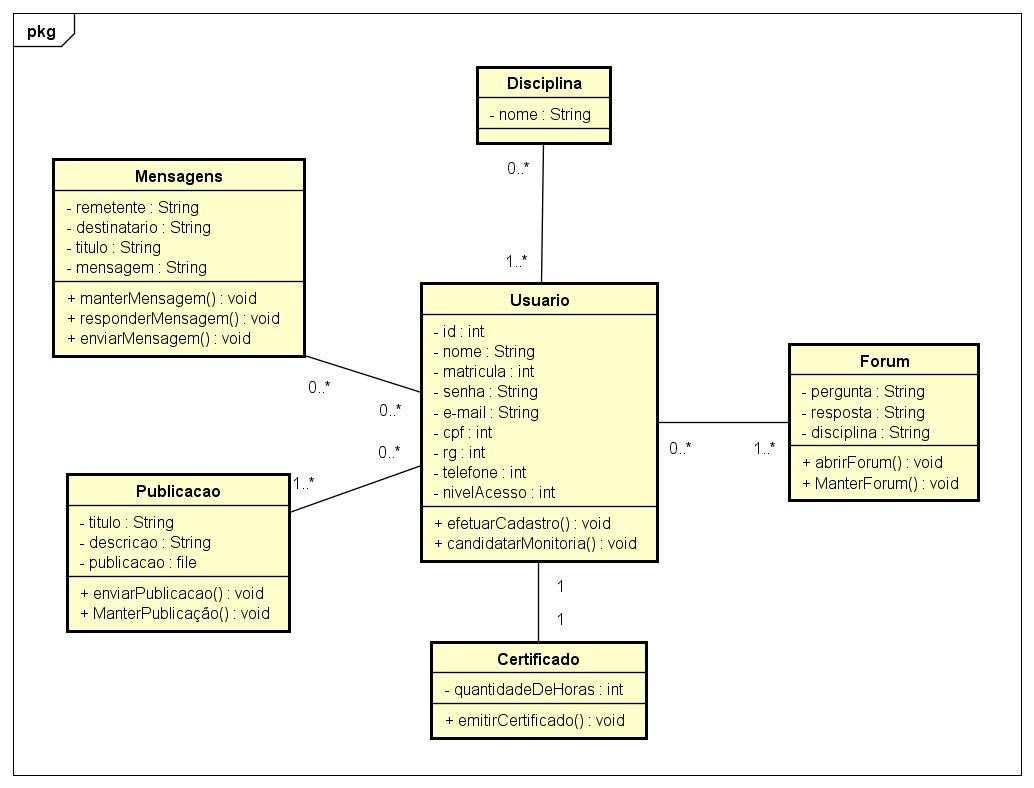


## Realizações dos Casos de Uso

|  |  |
| --- | --- |
| **Caso de Uso** | **Descrição** |
| Autenticar Usuário | O um usuário pode se autenticar no sistema |
| Manter Usuário | Criar, alterar, ativar ou desativar um usuário |
| Manter Disciplina | Criar, alterar, listar, ativar ou desativar uma disciplina |
| Manter Aviso | Criar, alterar, excluir, listar, expirar um aviso |
| Manter Publicação | Criar, alterar, excluir, buscar uma publicação |
| Manter Fórum | Iniciar, alterar, ativar ou desativar um fórum |
| Manter Pergunta | Criar, alterar, excluir, buscar ou listar uma pergunta |
| Manter Resposta | Criar, alterar, excluir uma resposta |
| Baixar Publicação | Realizar download de uma publicação |

# Visão Lógica

Descreve as classes mais importantes, sua organização em pacotes e subsistemas de serviço, e a organização desses subsistemas em camadas.



## Visão Geral

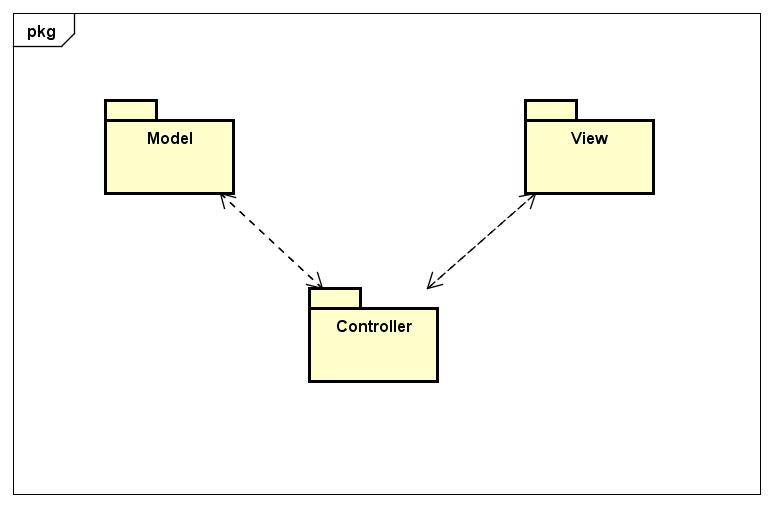
*[Esta subseção descreve toda a decomposição do modelo de design em termos de camadas e de hierarquia de pacotes.]*

## Pacotes de Design Significativos do Ponto de Vista da Arquitetura

[Para cada pacote significativo, inclua uma subseção com o respectivo nome, uma breve descrição e um diagrama com todos os pacotes e classes significativos nele contidos.

Para cada classe significativa no pacote, inclua o respectivo nome, uma breve descrição e, opcionalmente, uma descrição de algumas das suas principais responsabilidades, operações e atributos.]

# Visão de Implementação



## Visão Geral

Seguindo o padrão arquitetural MVC, o sistema está dividido em 3 camadas: Model, View e Controller.

## Camadas

### View

A camada view é responsável pela interação do usuário com o sistema.

### Controller

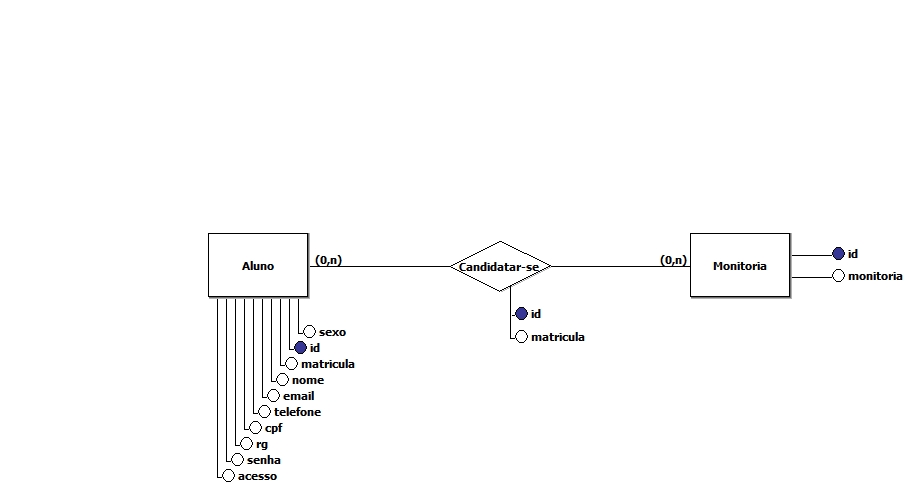
A camada controller é responsável por comunicar as requisições do usuário ao sistema.

### Model

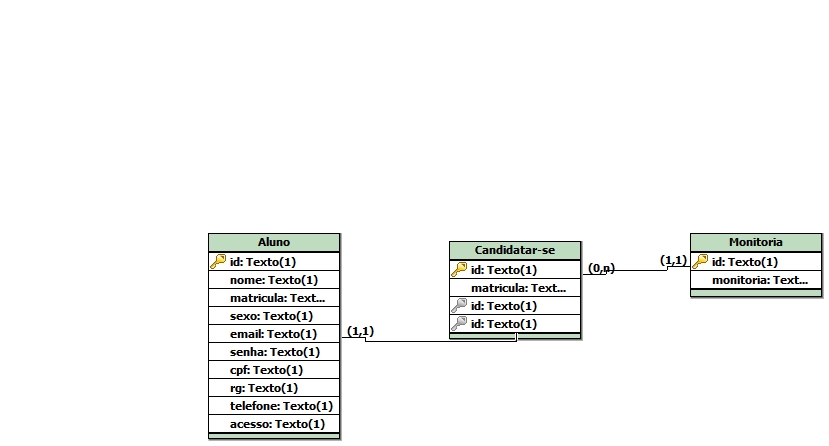
A camada model é responsável pelo armazenamento das classes no banco de dados.

# Visão de Dados

## Diagrama Entidade Relacionamento – DER



## Modelo Entidade Relacionamento – MER



# Tamanho e Desempenho

[Uma descrição das principais características de dimensionamento do software que têm um impacto na arquitetura, bem como as restrições do desempenho desejado.]

# Qualidade

[Uma descrição de como a arquitetura do software contribui para todos os recursos (exceto a funcionalidade) do sistema: capacidade de extensão, credibilidade, portabilidade e assim por diante. Se essas características possuírem significado especial, como implicações de segurança, garantia ou privacidade, elas deverão ser delineadas claramente.]