Universidade Federal Fluminense

< IdUni>
Documento de Arquitetura de Software

Versão <1.0>

Histórico da Revisão

Data	Versão	Descrição	Autor
15/05/22	1.0	Levantamento de Requisitos e Casos de Uso	Feito em conjunto pelo grupo
15/05/2022	1.0	Criação do repositório no Github	Lucas Fernandes Ramos
15/05/22	1.0	Commit dos documentos de casos de uso e levantamento de requisitos	Lucas Fernandes Ramos
15/05/2022	1.0	Definição inicial da Arquitetura do Sistema	Caio Saracuza
18/05/2022	1.0	Criação do VCP Manter Turmas	Joel Lopes Cunha de Souza
18/05/2022	1.0	Criação do VCP Ministrar Turmas	Walber Capaci de Araujo
18/05/2022	1.0	Criação do VCP Lançar Notas	Gabriel Vieira
18/05/2022	1.0	Revisão do VCP Ministrar Turmas	Joel Lopes Cunha de Souza
18/05/2022	1.0	Criação do VCP Fechar Período	Marllon Oliveira B. Guida
01/06/2022	1.0	Revisão do VCP Fechar Período	Marllon Oliveira B. Guida
18/05/2022	1.0	Criação VCP Iniciar Período	Daniel Muniz de Carvalho
18/05/2022	1.0	Criação VCP Inscrever Turma	Lucas Natan França Costa
18/05/2022	1.0	Criação VCP Gerar Relatório	Pedro Carvalho Nascimento
01/06/2022	1.0	Mecanismos Arquiteturais: MVC(descrição)	Joel Lopes Cunha de Souza
01/06/2022	1.0	Criação do Diagrama de Classes com base nos VCP's.(Consolidação do diagrama de classes), feito em grupo pelos membros listados.	Joel Lopes Cunha de Souza, Walber Capaci de Araújo, Pedro Carvalho Nascimento, Marllon Oliveira B. Guida, Daniel Muniz de Carvalho.
01/06/2022	1.0	Commit da primeira versão do diagrama de classes no github	Joel Lopes Cunha de Souza
12/06/2022	1.0	Criação do diagrama de sequência para o caso de uso Lançar Notas	Gabriel Vieira
12/06/2022	1.0	Criação do diagrama de sequência para o caso de uso Manter Usuários	Lucas Fernandes Ramos

14/06/2022	1.1	Edição do diagrama de classes para adicionar DAOs e métodos	Caio Saracuza
15/06/2022	1.1	Refinamento da Arquitetura do Sistema	Caio Saracuza
15/06/2022	1.1	Adicionada a análise de projeto (Padrão Controller)	Gabriel Vieira
29/06/2022	1.1	Organizada no Diagrama de Classes	Gabriel Vieira
29/06/2022	1.1	Criação do Padrão de Projeto State	Gabriel Vieira
13/08/2022	1.2	Adicionando Implementação do código do Padrão de Projeto State	Gabriel Vieira

Índice Analítico

1. Introdução 4
   - 1.1 Finalidade
   - 1.2 Escopo
   - 1.3 Definições, acrônimos e abreviações
2. Metas e Restrições da Arquitetura 4
3. Suposições e Dependências 4
4. Requisitos Arquiteturalmente Significantes 4
5. Decisões, Restrições e justificativas 4
6. Mecanismos Arquiteturais 5
7. Camadas da Arquitetura 5
8. Visões da Arquitetura 5
9. Qualidade 5

Documento de Arquitetura de Software

1. Introdução

O idUni é um sistema de apoio à uma universidade durante todo o período letivo, permitindo inscrições de alunos em turmas, manutenção dessas turmas e lançamento de notas no histórico de cada aluno ao fim do período. Os usuários que podem acessar o sistema são os alunos, professores e funcionários administrativos.

1.1 Finalidade

Este documento oferece uma visão geral arquitetural abrangente do sistema, usando diversas visões arquiteturais para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas em relação ao sistema.

1.2 Escopo

O documento se aplica ao sistema acadêmico IdUni e seu desenvolvimento, no intuito de auxiliar os desenvolvedores envolvidos a captar os aspectos arquitetônicos do sistema e se adequar a eles. Além de auxiliar no entendimento do sistema por novos membros da equipe.

1.3 Definições, Acrônimos e Abreviações

MVC: Padrão de arquitetura de software onde M significa modelo sendo responsável pela parte de regras de negócio, V a visualização responsável pela parte de interfaces e C a parte de controle dos dados.

HTTP: Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto)

HTTPs: Hypertext Transfer Protocol Secure (Protocolo de Transferência de Hipertexto Seguro)

2. Metas e Restrições da Arquitetura

A meta principal da arquitetura é ser capaz de ser elástica o suficiente para comportar o grande volume de acessos no início dos períodos, onde existe uma quantidade incomum de acessos devido às inscrições em disciplinas.

Algumas restrições relevantes para a arquitetura se aplicam, sendo elas:

- Banco de dados Postgresql;
- Estrutura MVC;
- Deve ser uma aplicação Web;
- Uso do paradigma orientado a objetos no desenvolvimento.
- Linguagem de programação back-end: Java;
- Linguagem de programação front-end: Javascript.
- Tecnologias utilizadas no front-end: HTML5 e CSS3.
- Frameworks: SpringMVC(Java) e bootstrap(CSS3).

3. Suposições e Dependências

- Interface com banco de dados de disciplinas
- Equipe com conhecimentos em desenvolvimento web

4. Requisitos Arquiteturalmente Significantes

https://drive.google.com/file/d/1Fpc65npE5wSZdkYNHPqTPR_PtQmf1f0t/view

5. Decisões, Restrições e justificativas

- Desenvolvimento do sistema como uma aplicação Web, por permitir maior acessibilidade aos usuários-alvo.
- Utilização do padrão de microsserviços no ecossistema, onde cada microsserviço terá um domínio do negócio, combinado com o padrão MVC, que será utilizado no nível de código, de forma com que cada microsserviço seja capaz de ser expansível para mudanças no decorrer da vida útil do produto.
- A arquitetura fará uso de containers docker para garantir a homogeneidade e a replicabilidade do ecossistema. Além de facilitar a escalabilidade pontual de algum serviço.
- Documentação ampla e detalhada para melhor manutenibilidade do sistema.

6. Mecanismos Arquiteturais

Mecanismo Arquitetural 1

Microsserviços

Essa arquitetura foi escolhida para que cada parte do sistema possa usar a tecnologia que seja conveniente ao escopo. Além de facilitar a escalabilidade horizontal do sistema de forma pontual, de forma a atender picos esperados ou não em algum fluxo do sistema. Ela também ajuda a garantir uma grande disponibilidade do sistema, já que o sistema não vai ter um ponto único de falha e quando alguma microsserviço tiver downtime, é muito mais rápido/fácil isolar e tratar o problema, além de ser mais rápido o deploy.

Mecanismo Arquitetural 2

MVC

A finalidade é separar o código fonte de cada micro serviço, aumentando o encapsulamento e tornando mais fácil a manutenção do sistema. Como estamos lidando com uma aplicação Web, o MVC irá dividir o micro serviço entre a interface(HTML/CSS/JavaScript), controladores(Classes que irão executar os casos de uso) e modelo(Classes representando os conceitos do negócio).

Mecanismo Arquitetural 3

Arquitetura em três Camadas.

Essa arquitetura foi escolhida devido a separação do projeto em três partes principais, A Interface, O Servidor e o Banco de Dados. Pois cada uma das partes principais é dividida em uma dessas funções dessa arquitetura, a Interface são os métodos que interagem com o usuário, o Servidor se trata do MVC e o Banco de Dados é onde serão armazenados os dados.

7. Camadas da Arquitetura

- **Visão/Interface com Usuário**: Classes responsáveis pela apresentação da interface gráfica do sistema, geralmente irá incluir arquivos HTML5, CSS3, JavaScript e JSON.
- **Controle**: Classes responsáveis por receber e tratar os eventos gerados na interface, designando

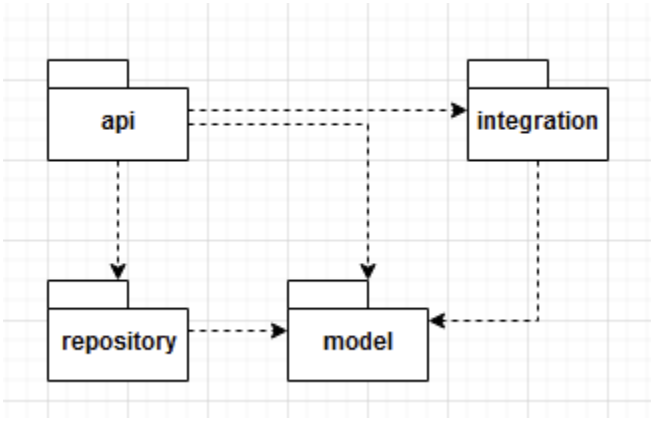
tarefas às classes de modelo e de visão conforme os dados são recebidos e exibidos.

- **Modelo:** Classes que armazenam os dados manipulados pela aplicação e que têm a ver com o domínio do sistema em construção.
- **Microsserviços**: Containers contendo serviços com responsabilidades bem definidas e pequenas, eles devem se comunicar através de métodos HTTP/HTTPS

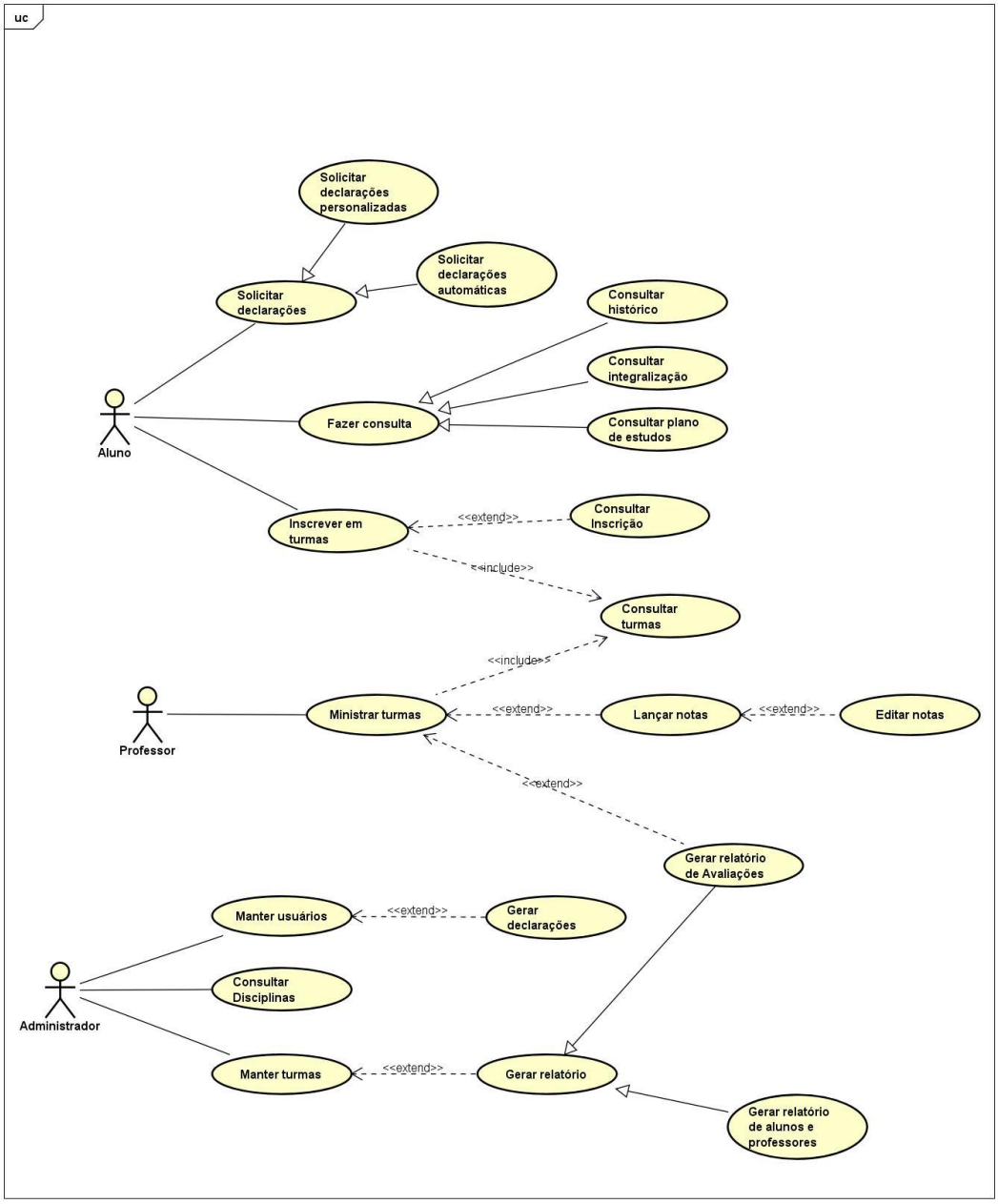
8. Visões da Arquitetura

Visões Recomendadas:

- **Lógica:** Descreva a estrutura e comportamento de porções arquiteturalmente significantes do sistema. Isto deve incluir a estrutura de pacotes, interfaces críticas, importantes classes e subsistemas e as relações entre estes elementos. Isto também inclui visões físicas e lógicas dos dados persistentes.
- **Operacional:** Descreva os nós físicos do sistema e os processos, threads e componentes que rodam em cada um desses nós. Esta visão não é necessária se o sistema roda num único processo e num único thread.
- **Pacote**



- **Visão de Casos de Uso:**



9. Qualidade

O padrão de arquitetura adotado no projeto tem como finalidade garantir uma melhor organização do código-fonte, o que auxilia na manutenção do software, bem como a portabilidade do mesmo.