Hackaton Management System
Documento de Arquitetura de Software

Versão 1.0

Histórico da Revisão

Data	Versão	Descrição	Autor
23/09/2019	1.0	Versão para Entrega	Mathias, Ramon, Carlos e Rodrigo

1. Introdução

O presente documento tem o propósito de descrever a arquitetura do projeto Hackaton Management System, desenvolvido na disciplina de Projeto e Arquitetura de Software do curso de Engenharia de Software da PUCRS. Espera-se que este documento possa ser útil àqueles que necessitem compreender a arquitetura do sistema em questão.

1.1 Finalidade

O presente documento tem o propósito de descrever a arquitetura do projeto Hackaton Management System, desenvolvido na disciplina de Projeto e Arquitetura de Software do curso de Engenharia de Software da PUCRS. Espera-se que este documento possa ser útil àqueles que necessitem compreender a arquitetura do sistema em questão.

1.2 Escopo

Este documento descreve o sistema em um nível arquitetural, elencando principais componentes e suas interações, cobrindo visões relacionadas aos objetivos da aplicação, seus casos de uso, visão lógica, visão de processo, visão lógica e visão de implementação.

1.3 Visão Geral

O documento de arquitetura de software contém uma visão mais técnica e detalhada de como o software está organizado e como os componentes do software se comunicam.

2. Representação Arquitetural

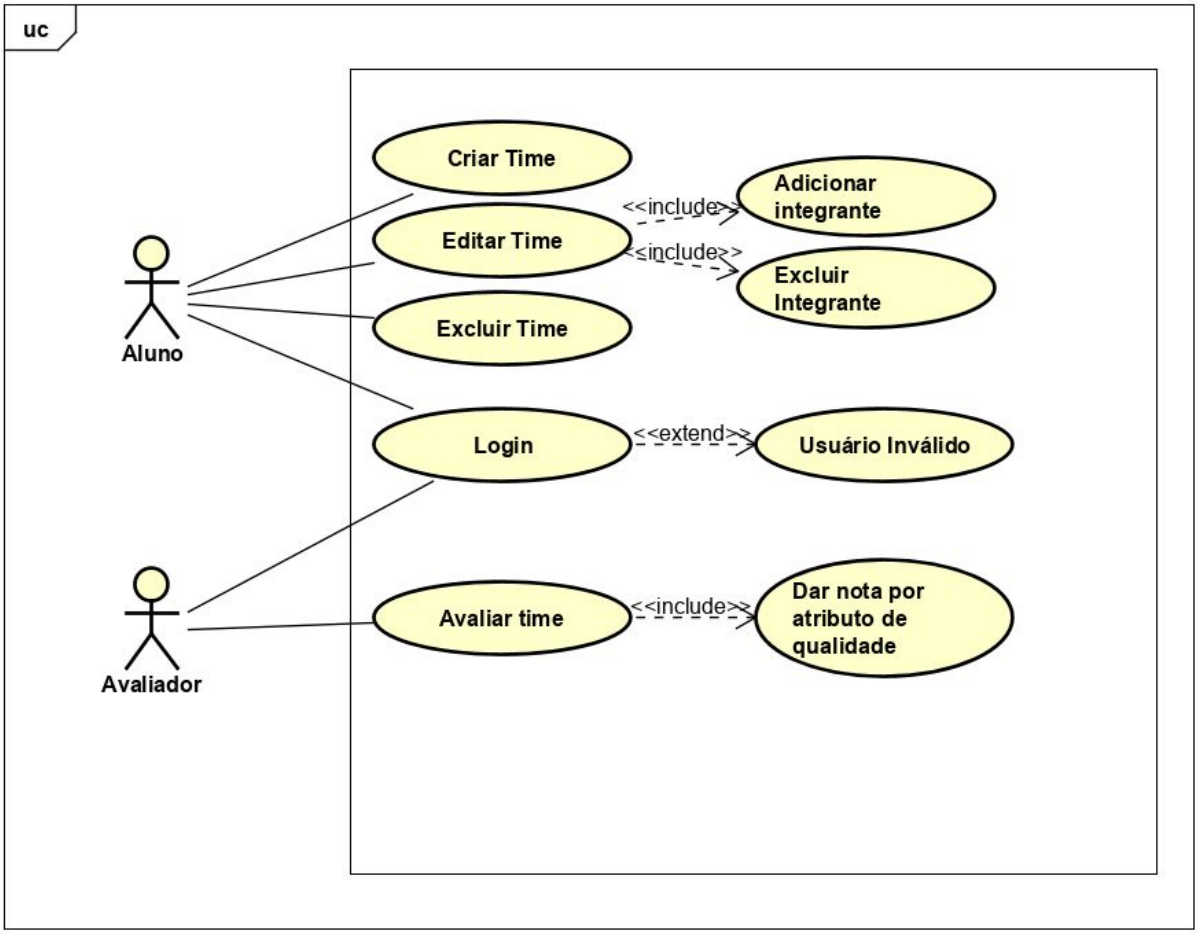
O estilo arquitetural utilizado segue o modelo Cliente Servidor, que aplicando-se ao contexto do projeto, consiste em uma aplicação web no lado do cliente, consumindo uma API no lado servidor. Devido à essa modularização, o lado servidor possui características arquiteturais próprias, que engloba tecnologias que se apropriam de conceitos do modelo MVC, como o SpringMVC, do padrão DAO para o fluxo de dados.

3. Metas e Restrições da Arquitetura

Nesta seção serão apresentadas as metas arquiteturais elencadas, bem como as restrições observadas. Uma das metas críticas para o sucesso do projeto seria uma aplicação que possibilitasse dois perfis de acesso diferentes, sendo o aluno e o professor, impactando na maneira com que a arquitetura foi projetada.

Uma das restrições de arquitetura é que o sistema necessitava de algoritmos que realizam o processamento necessário sobre dados em memória ao invés de dados persistidos externamente.

4. Visão de Casos de Uso



5. Visão Lógica

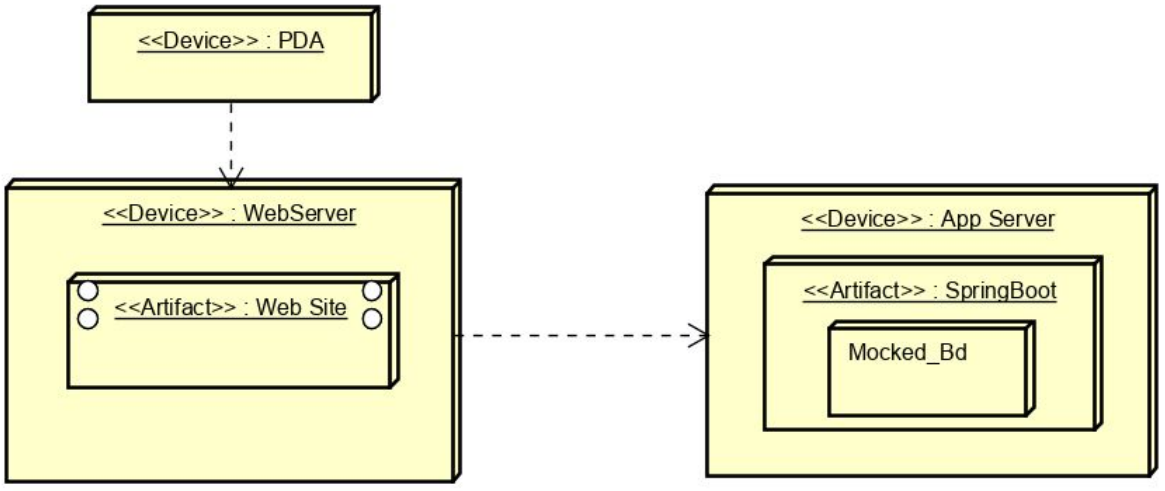
Existiam regras que foram encapsuladas em componentes

5.1 Visão Geral

O projeto engloba dois principais conceitos, o gerenciamento de pessoas englobando professor e aluno e o gerenciamento do time, possuindo uma lógica própria de aceite e lógica. a estrutura de pacote

1. Pessoas
    a. Aluno
    b. Professor
2. Times
    a. Integrantes
    b. Avaliação

6. Visão de Implantação



7. Visão da Implementação

7.1 Camadas

O sistema foi dividido apenas em frontend e backend, devido a simplicidade da lógica no backend a divisão em camadas aumentaria a complexidade sem melhorar o desempenho ou modularidade do projeto. Foi pensado em uma camada de banco de dados caso o mesmo fosse implementado

9. Tamanho e Desempenho

Devido ao escopo do projeto o desempenho não precisou ser considerado, porém devido às limitações de persistência de dados do algoritmo, as regras de negócio tiveram que ser pensadas para melhor contemplar a facilidade da leitura de código, utilizando streams de java, aumentando performance e confiabilidade do código.

10. Qualidade

A qualidade foi trabalhada principalmente no atributo usabilidade, considerando se tratar de um sistema funcionalmente simples e que não necessitava de grandes empenhos relacionados a desempenho, a usabilidade se mostrou ser o atributo mais crítico para o sucesso da aplicação. Dito isso, a arquitetura Front End foi projetada de maneira a garantir simplicidade no uso da aplicação de maneira a corresponder a igual simplicidade funcional da mesma.

Devido a arquitetura escolhida a reusabilidade também se mostrou um requisito que a aplicação contempla. Todos os endpoints estão genéricos e reutilizáveis caso outra aplicação (mobile, por exemplo) queira consumir a aplicação e/ou regras de negócio.