Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Ciência da Computação
Laboratório de Engenharia de Software
Geraldo Braz Junior

Eduardo Silva Vieira

Coordenação Fácil
Especificação Arquitetural

Este trabalho tem como objetivo a complementação da segunda nota de Laboratório de Engenharia de Software.

São Luís - MA
2018

1. Introdução

Este trabalho tem como objetivo especificar e descrever a arquitetura usada para desenvolver a Coordenação Fácil, um sistema de gerenciamento de processos acadêmicos baseado em uma arquitetura orientada a serviços (REST).

2. Arquitetura

Este trabalho usou a arquitetura orientada a serviços REST para modelar os três módulos pretendidos (Plano de Estudos, Aproveitamento de Cadeiras e Monitoria).

2.1 REST

Web Service é um conceito que continua crescendo nos últimos anos, sobretudo por causa de novas abordagens como o modelo REST. Usando uma linguagem intermediária universal como o XML ou JSON, as aplicações modeladas podem ser programadas em diferentes linguagens de programação e tecnologias, aumentando a interoperabilidade entre sistemas e a comunicação entre aplicações diferentes. Dessa forma, é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis entre si [Deitel H. and Deitel P. 2010].

Esta abordagem permite que os recursos de um sistema estejam disponíveis via rede de maneira uniforme. Isso significa que uma aplicação pode chamar outra para realizar tarefas mesmo que as duas aplicações estejam em diferentes sistemas e escritas em linguagens diferentes. Em outras palavras, os Web Services fazem com que os seus recursos estejam disponíveis para que qualquer aplicação cliente possa operar e extrair os recursos fornecidos pelo serviço.

O REST ou *Representational State Transfer*, em português Transferência de Estado Representacional, é um estilo de arquitetura criado por Roy Fielding, um dos criados do protocolo HTTP, que define um conjunto de restrições e propriedades que orientam os desenvolvedores para uma maneira correta de se modelar aplicações na rede. Nesta arquitetura, cada recurso é identificado por um URI (*Uniform Resource Identifier*), ou Identificador de Recurso Uniforme, o qual será utilizado para identificar unicamente o serviço oferecido.

Portanto, aplicações que obedecem ao estilo arquitetural REST fornecem interoperabilidade entre sistemas de computadores na rede. Os web services compatíveis com REST permitem que os sistemas solicitantes acessem e manipulem os recursos da aplicação por meio dos métodos padrões do HTTP, como GET, POST, PUT e DELETE. Por exemplo, o serviço /app/students/ acessado pelo método GET retorna para o solicitante uma lista de todos os estudantes, por outro lado, o serviço /app/students/20 acessado pelo método DELETE irá deletar o registro do estudante cuja a matrícula é igual a 20.

Como dito anteriormente, para as aplicações se comunicarem, necessita-se de uma linguagem de comunicação intermediária universal, como XML ou JSON. O REST faz uso do JSON, um acrônimo de JavaScript Object Notation, especificado por Douglas Crockford em 2000, é um formato compacto, de padrão aberto, independente, para troca de dados simples e rápida entre sistemas. Ademais utiliza texto legível por humanos, no formato atributo-valor, substituindo o uso do XML.

2.2 Componentes

Como mencionado o sistema de gerenciamento de processos acadêmicos é formada por três módulos principais, a saber, plano de estudos, aproveitamento de cadeiras e monitoria. Ademais para fornecer uma modelagem eficaz e eficiente do sistema evitando redundância de funcionalidades foram criados o módulo de autenticação e administração.

Na figura 1, visualizamos a Coordenação Fácil e seus módulos, juntamente com suas principais funções. Nesta arquitetura, o cliente se conectará a aplicação por meio de um navegador, o qual enviará requisições HTTP para consumir o serviço pretendido.

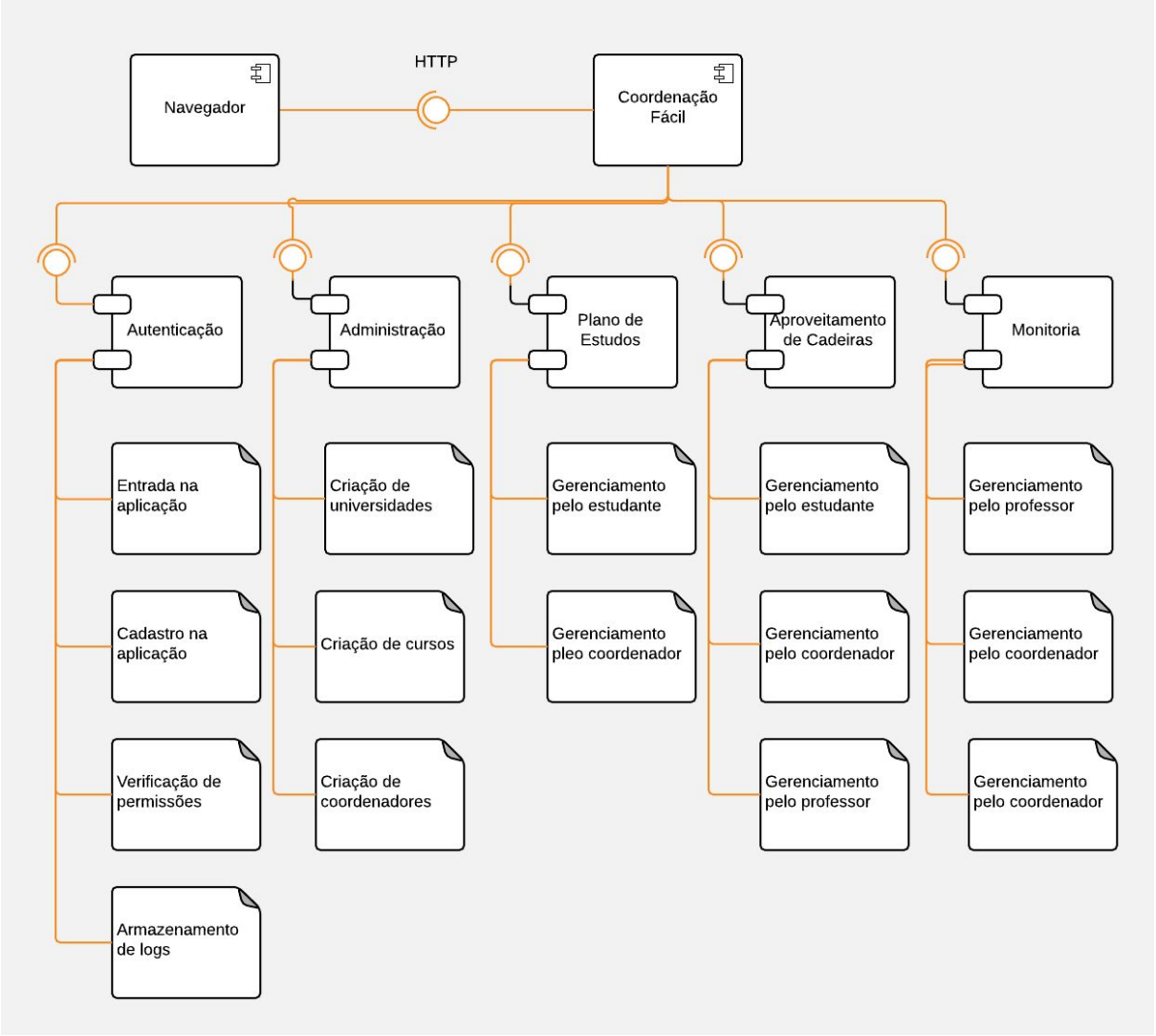

Figura 1 - Arquitetura de Componentes

Na figura 2, vemos as dependências da aplicação e como cada serviço está conectado.

O Amazon Web Services, serviço de computação na nuvem da Amazon, será usado para hospedar a aplicação e suas dependências. Usamos o *Elastic Computing Cloud* (EC2) para hospedar o Flask, microframework python, e MongoDB, banco de dados orientado a documentos. Também usamos o *Simple Storage Service* (S3) para arquivar e armazenar dados pouco acessados evitando sobrecarga de dados desnecessária na aplicação principal.

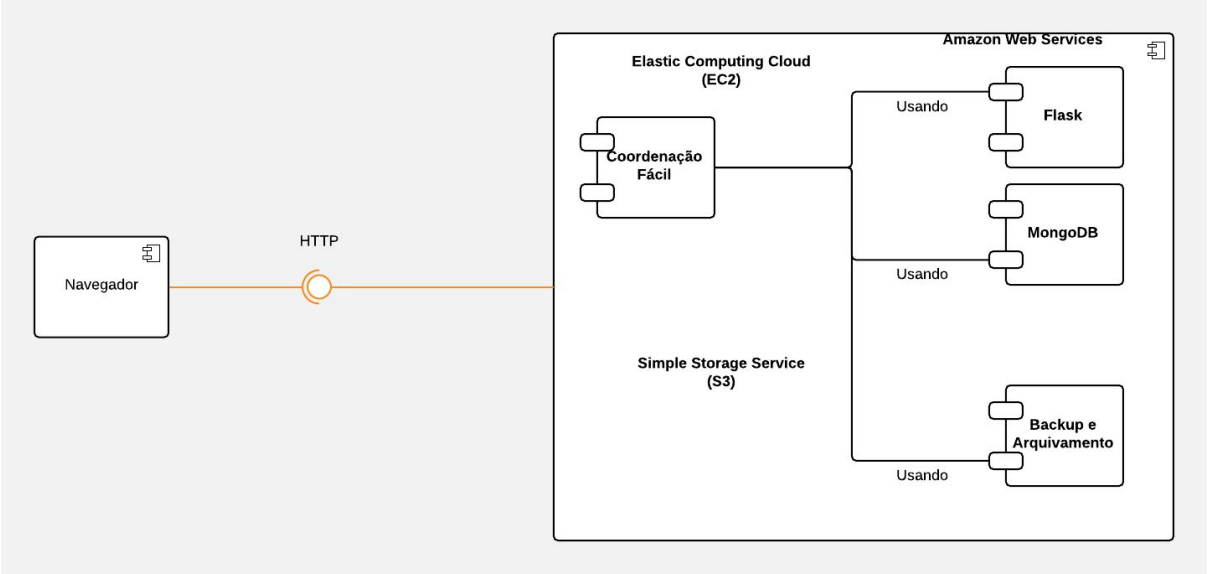

Figura 2 - Arquitetura de Dependências

4. Requisitos Funcionais e Não Funcionais Organizados por Módulo

4.1 Módulo de Autenticação

RF001 - O sistema deve permitir que o estudante cadastre-se na aplicação;
RF002 - O sistema deve permitir que o estudante entre na aplicação;
RF004 - O sistema deve permitir que o estudante sai da aplicação;
RF015 - O sistema deve permitir que o coordenador entre na aplicação;
RF016 - O sistema deve permitir que o coordenador saia da aplicação;
RF027 - O sistema deve permitir que o administrador entre na aplicação;
RF028 - O sistema deve permitir que o administrador saia da aplicação;
RF040 - O sistema deve permitir que o professor entre na aplicação;
RF041 - O sistema deve permitir que o administrador saia da aplicação;

4.2 Módulo de Administrador

RF028 - O sistema deve permitir que administrador cadastre uma nova universidade;
RF029 - O sistema deve permitir que administrador edite uma universidade criada anteriormente;
RF030 - O sistema deve permitir que administrador apague uma universidade criada anteriormente;
RF031 - O sistema deve permitir que administrador visualize todas as universidades criadas;
RF032 - O sistema deve permitir que administrador cadastre um novo curso;
RF033 - O sistema deve permitir que administrador edite um curso criado anteriormente;
RF034 - O sistema deve permitir que administrador apague um curso criado anteriormente;
RF035 - O sistema deve permitir que administrador visualize todos os cursos;
RF036 - O sistema deve permitir que administrador cadastre um novo coordenador;
RF037 - O sistema deve permitir que administrador edite um coordenador criado anteriormente;
RF038 - O sistema deve permitir que administrador apague um coordenador criado anteriormente;

RF039 - O sistema deve permitir que administrador visualize todos os coordenadores criados;

4.1 Módulo de Plano de Estudos

RF008 - O sistema deve permitir que o estudante crie um pedido de plano de estudos de maneira automática com as cadeiras e períodos restantes;
RF009 - O sistema deve permitir que o estudante edite um plano de estudos;
RF010 - O sistema deve permitir que o estudante apague um pedido de plano de estudos submetido;
RF011 - O sistema deve permitir que o estudante visualize todos os pedidos de planos de estudos submetidos;
RF017 - O sistema deve permitir que o coordenador visualize todos os pedidos de planos de estudos do curso;
RF019 - O sistema deve permitir que o coordenador aprove/recuse um pedido de plano de estudos;
RF021 - O sistema deve permitir que o coordenador cadastre um novo professor;
RF022 - O sistema deve permitir que o coordenador edite informações pessoais sobre um professor;
RF023 - O sistema deve permitir que o coordenador apague um professor criado anteriormente;
RF024 - O sistema deve permitir que o coordenador cadastre uma nova cadeira;
RF025 - O sistema deve permitir que o coordenador edite informações sobre uma cadeira criada anteriormente;
RF026 - O sistema deve permitir que o coordenador apague uma cadeira criada anteriormente;

4.2 Módulo de Aproveitamento de Cadeiras

RF005 - O sistema deve permitir que o estudante submeta um pedido de aproveitamento de cadeiras;
RF006 - O sistema deve permitir que o estudante edite um pedido de aproveitamento de cadeiras submetido;
RF007 - O sistema deve permitir que o estudante visualize todos os pedidos de aproveitamento de cadeiras submetidos;
RF016 - O sistema deve permitir que o coordenador visualize todos os pedidos de aproveitamento de cadeiras do curso;
RF020 - O sistema deve permitir que o coordenador aprove/recuse um pedido de aproveitamento de cadeiras;
RF041 - O sistema deve permitir que o professor visualize os pedidos de aproveitamento de cadeiras;
RF042 - O sistema deve permitir que o professor aprove/recuse um pedido de aproveitamento de cadeiras;

4.3 Módulo de Monitoria

RF012 - O sistema deve permitir que o estudante submeta um pedido de monitoria;
RF013 - O sistema deve permitir que o estudante apague um pedido de monitoria submetido;

RF014 - O sistema deve permitir que o estudante visualize todos os pedidos de monitorias submetidos;
RF018 - O sistema deve permitir que o coordenador visualize todos os pedidos de monitorias do curso;
RF043 - O sistema deve permitir que o professor aprove/recuse um pedido de monitoria;

References

Ribeiro, M. F. and R. E.. Web Services REST Conceitos, Análise e Implementação. Educação, Tecnologia e Cultura - E.T.C., [S.l.], n. 14, jun. 2016.

DEITEL, Paul; DEITEL, Harvey. Java: Como Programar. 8. ed. São Paulo: Pearson, 2010. 1144 p. Tradução de: Edson Furmankiewicz.