Documento de Projeto de Software e Decisões

Projeto: Scribe
Função primária: Gerenciamento de arquivos e pastas
Versão do documento: 1.0
Última modificação: 08/08/2016

1. Introdução

Este documento apresenta o documento de projeto (design) do sistema de gerenciamento de arquivos Scribe. Essa atividade foi conduzida em refinamentos sucessivos, começando pelo projeto da arquitetura do sistema, passando ao detalhamento dos componentes da arquitetura, até chegar ao projeto detalhado das classes. Este documento está organizado da seguinte forma: seção 2 explica a metodologia de desenvolvimento utilizada, a seção 3 exibe decisões de marca e design; seção 4 caracteriza-se pela apresentação das plataformas a serem utilizadas na implementação do sistema; a seção 5 discute aspectos do projeto da arquitetura do sistema; as seções 4 e 5 apresentam os modelos relativos aos subsistemas identificados.

2. Metodologia

A seção apresenta características de metodologia de desenvolvimento associada e módulos adicionais relacionados:

- O desenvolvimento em conjunto foi focado utilizando o Modelo Cascata. A equipe realiza reuniões presenciais semanais e acompanhamento por meio da plataforma Slack para desenvolvimento contínuo;
- Em reuniões semanais, a equipe de desenvolvimento colige-se e programam com participação conjunta, de modo que os integrantes realizam as alterações e implementações em um mesmo computador;
- O controle de versões foi feito pela plataforma Git;
- Os envios (*commits*) das versões implementadas é vinculado ao participante da equipe cujo usuário está vinculado ao Git no momento da reunião semanal.

3. Marca e identidade

O nome do sistema foi intitulado **Scribe** pela veiculação temática. Como se trata de um sistema de criação, escrita e gerência de arquivos e pastas, o uso de um nome que denota a uma profissão que realiza a mesma atividade (escribas) cria uma identidade própria.

As cores de predominância escolhidas foram tonalidades de azul, branco e cinza; justificadas pela decisão de criar uma interface limpa, amigável e confortável ao usuário. Além disso, corrobora com temática de sistemas concorrentes no setor.

Cores de predominância: #232556 #2a3890 #005ca5 #1a75bb #f2f2f2 .

Figura 1 - Identidade visual

4. Plataformas de Implementação

O sistema em questão trata-se de um Sistema de Informação e apresenta as seguintes características:

- Envolve quantidade moderada de dados e a sua gerência deve ser feita usando um banco de dados;
- Usuários acessam os dados concorrentemente. As funcionalidades são disponibilizadas ao cliente por meio de uma aplicação web;
- Há uma grande quantidade de interfaces gráficas com o usuário;
- As interfaces gráficas utilizam, primariamente, o modelo de *Single Page Application*; permitindo que o usuário não migre de páginas a todo o momento e seja possível uma visualização clara e objetiva das funcionalidades do sistema (foco em praticidade);

- Foi decidida a utilização de Angular Material, justificando-se pelas suas funcionalidades pré-fabricadas, customização e edição abordável, diretivas veiculadas do Angular e UIF (User Friendly Interface);
- AngularJS foi utilizado no projeto pela possibilidade de criação de aplicações dinâmicas;
- Ruby on Rails está anexo ao back-end, uso legitimado pela integração e suporte com várias plataformas, incluindo de hospedagem e gerência, como Heroku;

Contexto-chave: Scribe utiliza Angular Material/AngularJS para front-end e Ruby on Rails em seu back-end. Hospedagem e gerência por meio do Heroku. Foco em interface dinâmica e amigável ao usuário.

5. Arquitetura de Software

Como se pode perceber pela especificação de requisitos para o sistema em questão, não há grandes restrições de desempenho e disponibilidade, ainda que algumas restrições tenham sido explicitamente apontadas. Assim, levando-se em consideração os requisitos para o sistema proposto, foram considerados como os principais atributos de qualidade a serem incorporados ao sistema os seguintes, apresentados juntamente com as táticas a serem aplicadas:

- Usabilidade:
    - Prover ao usuário uma interface limpa e iconográfica, com funcionalidades dispostas de maneira a evitar migração de páginas e conteúdo repetitivo
    - Separar a interface do restante da aplicação.
- Manutenibilidade:
    - Coerência semântica: a organização do sistema deve se dar de modo que as responsabilidades em um módulo trabalhem em conjunto sem depender excessivamente de outros módulos;
    - Uso de interfaces com ocultação de informações específicas sobre a implementação dos módulos;
    - Uso de um intermediário para isolar o mecanismo de persistência de dados.
- Segurança:
    - Autenticação primária por meio de usuário (e-mail) e senha;
    - Limitar a exposição do sistema apenas ao que foi designado ao usuário, sem funcionalidades de gerência interna expostas no front-end;
    - Checagem de arquivos, conteúdo e pastas. Impedindo repetições e perdas.

Ainda que os demais atributos de qualidade não tenham sido considerados como sendo condutores da arquitetura, algumas táticas foram aplicadas visando garantir o nível de atendimento requerido com base nas *User Stories* pré-definidas pelo cliente. A seguir, as táticas consideradas são listadas:

- Disponibilidade: uso de exceções e tratamentos de erro diretamente no front-end para evitar consultas internas;
- Portabilidade: o uso de Angular Material e AngularJS permite que a aplicação seja utilizada tanto a partir de computadores pessoais como dispositivos portáteis;

Tomando por base as características do sistema discutidas na seção 2 e os atributos de qualidade e táticas selecionadas para tratá-los apresentados anteriormente, decidiu-se adotar um estilo combinando camadas e partições de interface. Inicialmente, duas partições principais foram definidas, procurando-se preservar a divisão em subsistemas realizada na fase de análise.

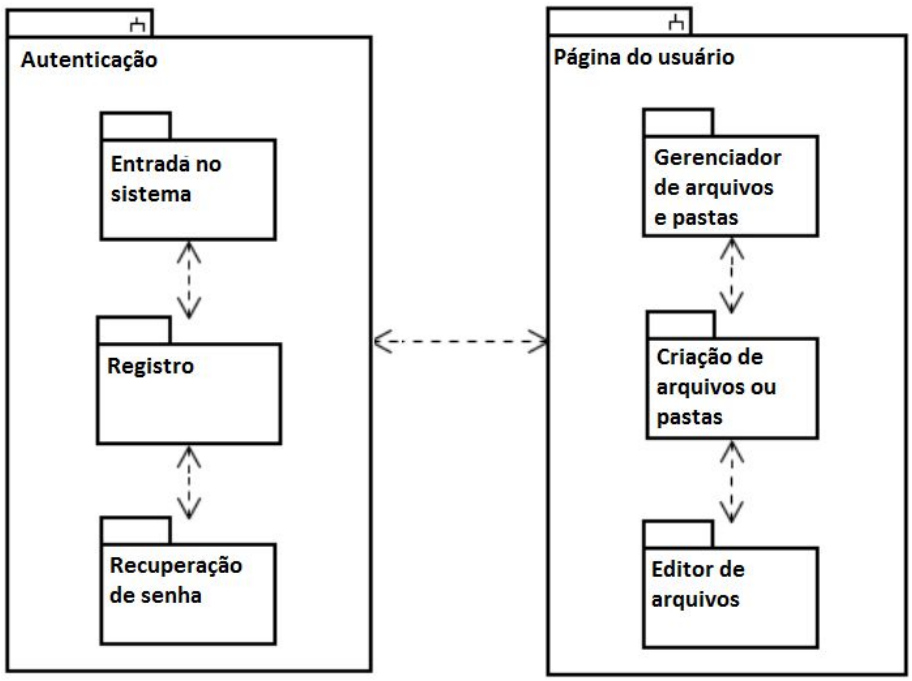

Figura 2 - Arquitetura Inicial

Com o detalhamento do projeto, em função de algumas decisões, a arquitetura originalmente proposta sofreu algumas alterações, a saber:

- No módulo de Autenticação e Página de Usuário, optou-se por usar o padrão Camada de Serviço. Assim, os módulos possuem uma página de visualização inicial que gere outros submódulos[1] conforme descritos na Figura 2.

Vale ressaltar que a dependência entre os Autenticação e Página do usuário existe apenas para instanciar objetos recuperados do banco de dados com base em um ID de identificação. Nenhum outro serviço é utilizado e, portanto, esta é uma dependência fraca.

[1] também existe a presença de submódulos internos que não estão disponíveis neste documento