UNIVERSIDADE FEDERAL DO PARANÁ

ALINE DAISY MARTINS ORZECHOWSKI
CRISTOPHER ZANCHETTA

SISTEMA PARA GERENCIAMENTO DE PROJETOS DE SEQUENCIAMENTO

CURITIBA
2016

ALINE DAISY MARTINS ORZECHOWSKI
CRISTOPHER ZANCHETTA

SISTEMA PARA GERENCIAMENTO DE PROJETOS DE SEQUENCIAMENTO

Trabalho de conclusão de curso apresentado como requisito à conclusão do curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Setor de Educação Profissional e Tecnologia, Universidade Federal do Paraná.

Orientadora: Jeroniza Nunes Marchaukoski

CURITIBA
2016

AGRADECIMENTO

Agradecemos primeiramente àquelas que estiveram ao nosso lado durante todos esses meses, as quais tornaram suportáveis quaisquer dificuldades enfrentadas em nossas horas de trabalho. Aos professores, agradecemos imensamente não apenas o apoio, mas também a oportunidade de perpetuarmos nosso conhecimento em um pequeno espaço da instituição e contribuir singelamente para a melhoria de seus trabalhos. À professora Maria Berenice, especialmente, e à equipe de pesquisadores e técnicos do projeto de processamento de dados de sequenciamento, pela colaboração, disposição, receptividade e confiança depositada em nosso projeto. Agradecemos também ao professor Dieval que prontamente nos apoiou na reta final deste projeto, e indubitavelmente à professora Jeroniza, que nos acompanhou desde o princípio desta jornada, como orientadora, mediadora e cuja presença foi indispensável para a realização deste trabalho. Aos nossos familiares e amigos, agradecemos a paciência, compreensão e o carinho mesmo durante nossa ausência.

RESUMO

O Sistema de Gerenciamento de Projetos de Sequenciamento foi desenvolvido especificamente para uso do Núcleo de Fixação Biológica de Nitrogênio do Departamento de Bioquímica do Setor de Biológicas da Universidade Federal do Paraná (UFPR), cujos laboratórios realizam importantes pesquisas no ramo de biologia celular, entre elas o sequenciamento genético de DNA e RNA. Para controlar seus processos, os pesquisadores mantém o registro de amostras recebidas para análise, informações sobre os requentes (universidades ou institutos parceiros e seus devidos responsáveis) e também os resultados obtidos através do sequenciamento.
Durante o processo de registro de dados porém, a informatização é quase não existente, sendo a documentação em planilhas digitais o único recurso utilizado, ou em certos casos até mesmo cadernos.
Este trabalho tem o propósito de tornar possível o gerenciamento de amostras biológicas de DNA e RNA e os dados de seus sequenciamentos genéticos através de um sistema, facilitando a manutenção e permitindo também análises estatísticas dos resultados.
Ambientado em uma plataforma WEB, o sistema permite que universidades parceiras tenham cadastros próprios, através de servidores responsáveis, podendo realizar cadastros para envios de amostras à UFPR e verificar o andamento dos procedimentos que estão sendo realizados. A atualização das amostras no sistema é feita pelos próprios profissionais da UFPR que efetuam as análises.
A utilização do sistema reduz o tempo gasto com a gerência das informações e organização dos dados, de forma que os pesquisadores possam otimizar o tempo em laboratório, além de visualizarem as massas de resultado de maneira simples e eficiente.

Palavras-chave: Sequenciamento de DNA. Sistema de informação biológica. Sequenciamento genético. Gerenciamento de dados. Single-page-application.

ABSTRACT

The Sequencing Projects Management System was built specifically for the Biological Nitrogen Fixation Center of the Biochemical Department of the UFPR, which laboratories conduct meaningful researches in the field of cell biology, including DNA and RNA genetic sequencing. To manage this process, researches keep track of received samples for analysis, information about its suitors (universities or partner institutions and its responsibles) and also the resultant data from sequencing.
Although, during this data input process, informatization is nearly non-existant, beign digital spreadsheets the only resource used, or even notebooks in some cases.

This work has the purpose of making it possible to manage biological samples os DNA and RNA and also its genetic sequencing data, making it easier to maintain and allowing statistic analysis over the results.
Built under a web platform, the system allows partner universities to have their own access through project managers, making it possible to register samples before sending for analysis at UFPR and also keep track of its progress. The data is updated by the same UFPR lab workers who run the analysis and administrate the system.
Through using this system the time once spent manually managing the data is significantly reduced, so the researchers are able to focus in lab work and analysis having a reliable tool to view huge pieces of results in a simple and optimized way.

Key words: DNA sequencing. Biological information system. Genetic sequencing. Data management. Single-page-application.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

API -*Application Programming Interface*

CGI -*Common Gateway Interface*

CSS - Cascading Style Sheets

DRF - *Django REST Framework*

DNA - Ácido desoxirribonucleico

DOM - *Document Object Model*

EAP - Estrutura Analítica de Trabalho

HTML - *Hypertext Markup Language*

HTTP - *Hypertext Transfer Protocol*

JS – JavaScript

MVC - *Model-View-Controller*

MVVM - *Model-View-ViewModel*

NFN - Núcleo de Fixação Biológica de Nitrogênio

ORM - Mapeamento Objeto-Relacional

REST - *Representational State Transfer*

RUP - *Rational Unified Process*

SQL - Structured Query Language

SPA - *Single Page Application*

UFPR – Universidade Federal do Paraná

UML - *Unified Modeling Language*

URI - *Uniform Resource Identifier*

WWW - *World Wide Web*

SUMÁRIO

1 INTRODUÇÃO

Em seus primórdios, a internet consistia apenas de web sites. Eram basicamente repositórios de informações que continham documentos estáticos. O fluxo das informações era apenas de via única, do servidor para o navegador. Hoje, a rede mundial de computadores é irreconhecível se comparada à sua forma original, a maioria das páginas são na verdade aplicações web, altamente funcionais e dependem da comunicação em duas vias entre o servidor e o navegador. (STUTTARD, PINTO – 2011)

As diferenças entre aplicações direcionadas à internet e aplicações nativas para os sistemas operacionais (popularizadas consideravelmente mais cedo) ainda são grandes e podem causar estranheza aos seus utilizadores. Para suprimir essa lacuna surgiram então os SPAs (*Single Page Applications*) que se diferenciam pela habilidade de redesenharem qualquer parte da interface sem a necessidade de requisitar novamente o HTML (*Hypertext Markup Language*) do servidor, o que é possível devido a separação de camadas de modelo, que se responsabiliza pelos dados, e uma camada de apresentação dos dados que é apenas lê os dados desses modelos. (TAKADA - 2013)

Através de uma ampla gama de tecnologias e frameworks disponíveis gratuitamente, é possível informatizar variados segmentos de atuação e automatizar processos que demandam esforço e tempo de profissionais especializados. Dentro da própria Universidade Federal do Paraná (UFPR) é possível encontrar diversas atividades que ainda não puderam atingir um maior grau de informatização, mas que a partir do mapeamento de seus processos podem ser integradas em um sistema. É o caso do Núcleo de Fixação Biológica de Nitrogênio da UFPR, local de aplicação deste projeto.

Situado no setor de Ciências Biológicas da UFPR, o Núcleo de Fixação Biológica de Nitrogênio (NFN) do Departamento de Bioquímica e Biologia Celular é referência nacional no ramo de pesquisas de biologia molecular, sendo um de seus focos o sequenciamento genômico e análises de sequências de DNA. Através do trabalho realizado nos últimos anos e os investimentos de órgãos como a Secretaria Estadual de Ciência, Tecnologia e Ensino Superior e o Conselho Nacional de Desenvolvimento Científico e Tecnológico. O NFN hoje possui recursos e equipamentos de sequenciamento genético usados extensivamente pela UFPR. Além

disso, o NFN também compartilha sua expertise, infraestrutura e ferramentas com diversos pesquisadores de projetos parceiros em outras instituições de ensino por todo o território nacional e também países vizinhos.

Devido à utilização em larga escala de tais recursos, a quantidade de informações e dados digitais gerados é proporcionalmente grande, impossibilitando seu gerenciamento através de ferramentas convencionais convenientemente adaptadas para o uso dos laboratórios. Além desta, outra necessidade do NFN era a digitalização de requisições e troca de informações entre os pesquisadores locais e seus contatos fora da instituição, facilitando a atualização a respeito de resultados, estágios de análise e dados inerentes às amostras e organismos.

Para sanar as necessidades do NFN, desenvolvemos uma solução web completa para gerenciamento de todos os dados organizacionais que dizem respeito às amostras e suas análises, para ser utilizada pelos profissionais do setor e também disponibilizada aos requerentes dos serviços do NFN, para que acompanhem o andamento dos processos demandados.

1.1 OBJETIVOS

Desenvolver um sistema para internet com a finalidade de gerenciar os projetos de sequenciamento genético do NFN, registrar resultados de análises de sequenciamentos e permitir a avaliação estatística dos dados armazenados.

1.2 OBJETIVOS ESPECÍFICOS

- Desenvolver uma plataforma web capaz de armazenar e disponibilizar informações para que sejam consumidas pela aplicação de acesso, também a ser desenvolvida como aplicação web;

- Permitir cadastro e manutenção de instituições, projetos e colaboradores. Os colaboradores poderão identificar digitalmente cada amostra que será enviada ao NFN, através de formulários específicos;

- Permitir cadastro de amostras e corridas por um administrador, de forma que cada estágio do processo seja identificado no sistema

- Permitir o registro dos resultados finais obtidos após o sequenciamento, que deverão estar disponíveis aos solicitantes do procedimento;

- Garantir através da aplicação a segurança das informações e suas devidas restrições de visualização, edição e acessos controladas por um administrador.

2 FUNDAMENTAÇÃO TEÓRICA

O objetivo desse capítulo é apresentar conceitos e definições relevantes para o trabalho que foi desenvolvido.

2.1 INTRODUÇÃO AO DESENVOLVIMENTO PARA WEB

Desde o início da utilização da internet como meio de comunicação e interação, no início da década de 1980, a demanda pela inovação de seus métodos de utilização foi sempre crescente e teve diversos referenciais como pontos chave em sua história. Em seu princípio, as páginas de internet eram escritas completamente a mão, utilizando o HTML (*Hypertext Markup Language*) e qualquer retrabalho significava reescrever partes inteiras do código para que refletissem no visual da página. (HOLOVATY e KAPLAN-MOSS, 2013)

Além de trabalhosas as páginas eram simples e pouco atrativas, o que levou às pesquisas por uma nova forma de organizar e exibir o código fonte. Batizado de CGI (*Common Gateway Interface*), o protocolo permitia que programas externos gerassem código HTML dinamicamente para que fosse carregado pelo navegador durante o uso, e foi então que as páginas de conteúdo dinâmico tomaram conta da internet. (HOLOVATY e KAPLAN-MOSS, 2013)

Seguindo a inovação, novas linguagens de programação surgiram para suprir as necessidades criadas e também para solucionar os novos problemas como a demanda por manutenção constante para abrigar novos conteúdos e também problemas intrínsecos do código como a sua desnecessária repetição em diversos trechos e páginas, uma vez que nenhum tipo de funcionalidade era aplicada além da linguagem de formatação. A evolução e as frustrações guiaram o desenvolvimento de linguagens de programação com foco no ambiente web e vivemos hoje o que pode ser considerada a terceira geração de frameworks web. (HOLOVATY e KAPLAN-MOSS, 2013)

2.2 SINGLE PAGE APPLICATIONS

A atual geração de tecnologias de desenvolvimento web nos proporciona diversas experiências distintas nos últimos anos e apesar das diferentes temáticas,

ao comparamos sites informativos aos de compras online ou redes sociais, é possível notar certos padrões que nos parecem muito comuns na navegação entre páginas. Esse comportamento se difere, em grande parte, do comportamento que esperamos de aplicações nativas no nosso sistema operacional, portanto, para trazer uma experiência mais habitual para o usuário, foram criadas as aplicações de página única.

Uma definição objetiva dos SPAs é a de que “[...] se distinguem pela sua habilidade de redesenhar qualquer parte da interface sem a necessidade de fazer uma requisição ao servidor para buscar HTML.” (TAKADA, 2013 Tradução pelos autores). Através da utilização de modelos MVC (*Model-View-Controller*) é possível manipular os dados recebidos do servidor através de um modelo, e utilizar camadas diferentes para a sua representação na tela. Assim, podemos imaginar três pilares básicos na concepção de um SPA: A arquitetura, que consiste dos conceitos definidos, módulos e dependências que se comunicarão entre si; A organização dos recursos, lidando com a forma em que os arquivos serão organizados, carregados no browser ou consumidos por módulos lógicos; E o estado de execução, que irá refletir tudo que está carregado no navegador, como serão intercalados os estados de visualização e como esses dados serão transmitidos de volta aos modelos.

Aplicações construídas no modelo de página única são capazes de manipular seus próprios estados e não serem dependentes de uma fonte externa de dados, contudo, não implementariam persistência de dados. Assim como em aplicações nativas para sistemas operacionais, é possível unir tecnologias e utilizarmos outra aplicação ou fonte como servidor de dados, apenas trocando informações com a aplicação web. Seguindo tais conceitos, utilizamos a arquitetura REST e bancos de dados relacionais na construção deste projeto.

2.3 ARQUITETURA REST

Definida no ano 2000, o padrão de arquitetura foi defendido na tese de doutorado de Roy Thomas Fieldling. O nome REST (*Representational State Transfer*), em português Transferência de Estado Representacional “[...]é um estilo de arquitetura para sistemas distribuídos, onde as requisições e as respostas contém representações do estado atual dos recursos do sistema” (STUTTARD e PINTO, 2011).

Baseado no protocolo HTTP (*Hypertext Transfer Protocol)* já existente (o qual Fieldling também é um dos principais autores), o pesquisador reuniu uma série de conceitos básicos que tiveram seus estudos iniciados durante o processo de padronização do HTTP, e visando a performance, confiabilidade e escalabilidade, a arquitetura REST foi definida sobre alguns pilares fundamentais. (FIELDING, 2000)

2.3.1 Requisições são feitas através de recursos

A REST se difere ao oferecer o acesso a um recurso e não a se dispor a receber uma série de comandos imperativos. Uma requisição ao servidor utilizará ações do protocolo HTTP (GET e POST por exemplo) e a especificação de uma URI (*Uniform Resource Identifier*) que identificará a representação do recurso, para que esse então apenas retorne uma representação do recurso requerido através da funcionalidade a qual a URI se refere.

2.3.2 Os recursos são representados por “representações”

Recursos são conjuntos ou subconjuntos de informações e continuarão a manter suas propriedades, independentes do uso que terão após serem requisitados pela aplicação. Para que essa premissa continue sendo verdadeira a REST fornece uma representação desse recurso de acordo com a necessidade da aplicação, sendo sempre acessado através da mesma URI. A representação pode ser por exemplo no formato JSON, para diferentes processamentos na aplicação, ou pode ser em formato XML para que seja exportada, ou mesmo em HTML para que seja lida e compreendida por um usuário. O recurso é o mesmo, e sua representação será definida como parte da negociação de conteúdo do HTTP.

2.3.3 A troca de mensagens deve ser auto descritiva

Nenhuma requisição feita à REST deve depender de um estado prévio ou requisição prévia para que seja compreendida pela servidor. Os recursos são oferecidos de forma que uma única URI seja capaz de identificar exatamente a resposta desejada no momento, conforme as especificações da arquitetura.

2.3.4 Os recursos são interligados a partir de links

As respostas das requisições devem estar padronizadas de acordo com as especificações, para que todo recurso que apresenta ligação com um outro recurso diferente dele sempre o identifique através de um link. Dessa forma a aplicação terá sempre uma forma de oferecer uma navegação intuitiva ao usuário, sem que esse precise necessariamente conhecer a estrutura do sistema.

2.4 BANCOS DE DADOS RELACIONAIS

Bancos de dados são sistemas capazes de armazenar e recuperar informações, independente do estado de funcionamento atual das aplicações ou de qualquer instância em funcionamento. Desde os anos 80, o sistema de banco de dados mais popular é o sistema relacional, cujas relações podem ser tratadas matematicamente, como relações lógicas e de conjuntos. Os dados propriamente ditos são representados como tabelas, contendo linhas e colunas. Entre as tabelas serão definidos relacionamentos, através das relações entre as coleções de objetos nelas contidos (representados por cada linha na tabela). (POSTGRESQL c, 2003)

A linguagem utilizada para manipular os bancos de dados relacionais é o SQL (*Structured Query Language*) e suas definições seguem diversas regras básicas que garantem a integridade dos dados armazenados. Além das definições intrínsecas do sistema, através da definição do banco de dados a ser manipulado é possível definir diversos outros conjuntos de regras que irão garantir que as aplicações que farão uso das informações contidas no mesmo possam fazê-lo sem colocar o banco em risco, seja por problemas na transação, conexão, ou mesmo defeitos na implementação da aplicação. (POSTGRESQL b, 2016)

2.4.1 PostgreSQL

O PostgreSQL é um banco de dados relacional de código aberto, sob licença liberal, permitindo seu uso para fins pessoais, comerciais ou governamentais. É compatível com diversos sistemas operacionais Windows, Linux ou UNIX. Implementa todas as funcionalidades padrões do SQL além de funções avançadas como expressões de tabela comum e funções de janela. Além dos tipos de dados padrão, o

PostgreSQL é capaz de implementar arranjos multidimensionais e tipos especiais definidos pelo usuário. (POSTGRESQL a, 2016)

2.5 JAVASCRIPT

Podemos dizer que hoje em dia as aplicações web não teriam existido se não fosse pela criação do JavaScript. Através da busca por uma forma de modificar os elementos de uma página da internet em tempo real, o Javascript foi introduzido em 1995, como parte do navegador Netscape e posteriormente foi adotado por todos os browsers de relevância. (HAVERBEKE, 2016)

A linguagem é multi-plataforma, baseada em scripts seguindo o paradigma de orientação a objetos, e entre suas vantagens estruturais estão o seu rápido processamento e o pouco espaço que requere. Suas funcionalidades podem ir além da aplicação em execução em navegadores (que funcionam no “cliente”, ou “client-side”) e se estenderem também à servidores (server-side). (MOZILLA, 2016)

Aos servidores, a linguagem pode ser utilizada para tratar requisições do cliente no manejo de arquivos, comunicações com banco de dados e diversas outras utilidades no repasse de informações.

No ambiente do cliente porém é onde o Javascript apresenta suas maiores funcionalidades, relacionadas a manipulação do DOM (Document Object Model, que de forma resumida é a interface de manipulação dos objetos contidos no HTML da página). A partir da linguagem, diversos frameworks foram construídos para estender suas funcionalidades e prover ambientes mais complexos e capazes de tratar toda a construção do *front-end* (módulos que agem no lado do cliente). Um dos frameworks mais populares é o Angular.js, (FIGURA 1) escolhido como plataforma para o desenvolvimento desse trabalho.

2.6 ANGULAR JS

O HTML puro, na finalidade que foi criado, é capaz de suprir sem problemas o seu propósito de criação e edição de páginas estáticas, mas de forma alguma cumpre algum requisito do dinamismo necessário aos aplicativos web. O Angular é o que o HTML deveria ter sido, caso fosse elaborado para aplicações. (ANGULAR a, 2010)

Se nativamente o HTML não detém todas as funções necessárias, o Angular permite que a sua sintaxe seja estendida e usada como base para que os componentes da aplicação possam se comunicar de maneira clara. A aplicação de técnicas de ligação de dados e injeção de dependências por exemplo, permitem que diferentes controles e funções sejam aplicados sobre os elementos da página e até mesmo estabelecendo comunicação direta com o servidor para a atualização da mesma.

Sendo categorizado como um framework "*client-side*" e tendo como uma de suas premissas a manipulação do DOM, o Angular utiliza a arquitetura MVC (*Model-View-Controller*) em conjunto com MVVM (*Model-View-ViewModel*, chamada também de "MV*"), pois não necessariamente depende de um controller para acessar os elementos no DOM. Dentre as diversas características do frameworks, destacam-se as seguintes:

a) ligação de dados em duas vias (*Two-Way Data Binding*);
b) *Model-View-ViewModel*;
c) injeção de dependências;
d) diretivas.

2.6.1 Ligação de Dados em Duas Vias (*Two-Way Data Binding*)

Normalmente as aplicações web precisam de vários controles e "listeners" na interface para identificar ações do usuário, alterar o estado de um elemento, ou enviar um dado e atualizar após a resposta. Para melhorar esse processo e evitar muitas linhas de código antes necessárias, o Angular lida com as atualizações de objetos através dos seus modelos. Os modelos são únicos, definidos na aplicação independente do estado da visualização, e permitem que uma representação sua seja refletida no DOM. Desta forma, o usuário pode interagir com esse modelo, ou apenas perceber uma mudança no seu estado, sem a necessidade de recarregar o elemento, ou recarregar a página. (ANGULAR b, 2016)

Essas atualizações podem ter sido feitas pelo servidor e talvez sido disparadas pelo usuário, ou mesmo por operações no próprio lado do cliente, mas o relevante é que e a simples alteração no modelo irá alterar seu estado de exibição sem que nenhum outro tipo de evento seja necessário. Como definido pelo próprio Angular, "O modelo é uma 'origem única de verdades' para o estado da aplicação.

Podemos pensar na view simplesmente como uma projeção instantânea do seu modelo” (ANGULAR b, 2016, Tradução pelos autores).

2.6.2 Model-View-Controller e Model-View-ViewModel

O Angular não se propõe a implementar exatamente o modelo *Model-View-Controller*, mas sim uma modificação de seus conceitos chamada de *Model-View-ViewModel* (ou, “MV*”). (ANGULAR b, 2016)

O “*Model*” é representado por objetos do JavaScript puro, são o modelo de dados da aplicação, sem necessidade de interferência de qualquer outra especificação do framework. Já o “*ViewModel*” é um objeto diferente, específico para prover os dados do modelo, de diferentes formas, para diferentes “*Views*”, agindo como uma variável de escopo. (MICROSOFT, 2012)

O “*Controller*” é responsável por ditar o estado inicial e então atualizar o escopo com métodos necessários para o seu comportamento. Apesar da sua característica de controle de ações e atualização de modelos, um *controller* não armazena estados da aplicação e também não interage com nenhum serviço externo

Por fim, a “*View*” é a camada de visualização da aplicação, onde o conteúdo será renderizado no HTML, refletindo os estados e os comportamentos definidos. Desta forma então, a interação realizada na *view* é redirecionada para tratamento no *controller*, que pode acessar as referências dos dados presentes no escopo e então novamente atualizar a *view* de acordo com o comportamento esperado.

2.6.3 Injeção de Dependências

O Angular nativamente oferece uma série de funcionalidades e pacotes que irão facilitar o desenvolvimento de cada parte da aplicação. Para tornar esse trabalho um pouco mais fácil de lidar e testar, o Angular contém um sistema de injeção de dependências, para que a cada módulo desejado sejam embutidos e carregados apenas as dependências necessárias para o seu funcionamento. As dependências podem ser outros módulos definidos pela própria linguagem, ou também módulos desenvolvidos pelo próprio usuário ou pela comunidade, e é necessário apenas referenciá-las e então o Angular será incumbido de buscar, carregar e criar o acesso à dependência. (ANGULAR b, 2016)

2.6.4 Diretivas

Uma das criações mais interessantes do Angular: A capacidade de inserir novas capacidades ao código HTML. As diretivas permitem a criação de tags HTML customizadas, que serão inseridas no código padrão, podendo então aplicar diferentes padrões de exibição aos elementos nela contidos ou inferindo novas regras de comportamento, por exemplo. (ANGULAR b, 2016)

Além do conjunto de funcionalidades citadas acima, a escolha pelo Angular teve outra influência: sua participação no mercado é alta e sua primeira versão ainda é um dos maiores frameworks de Javascript utilizados para grandes aplicações comerciais. Sua popularidade por ser mensurada através das suas buscas no Google, como mostra um indicativo elaborado no Google Trends entre 2010 e 2016 (FIGURA 1), comparando-o com outros grandes frameworks que competem com o Angular pelo mercado.

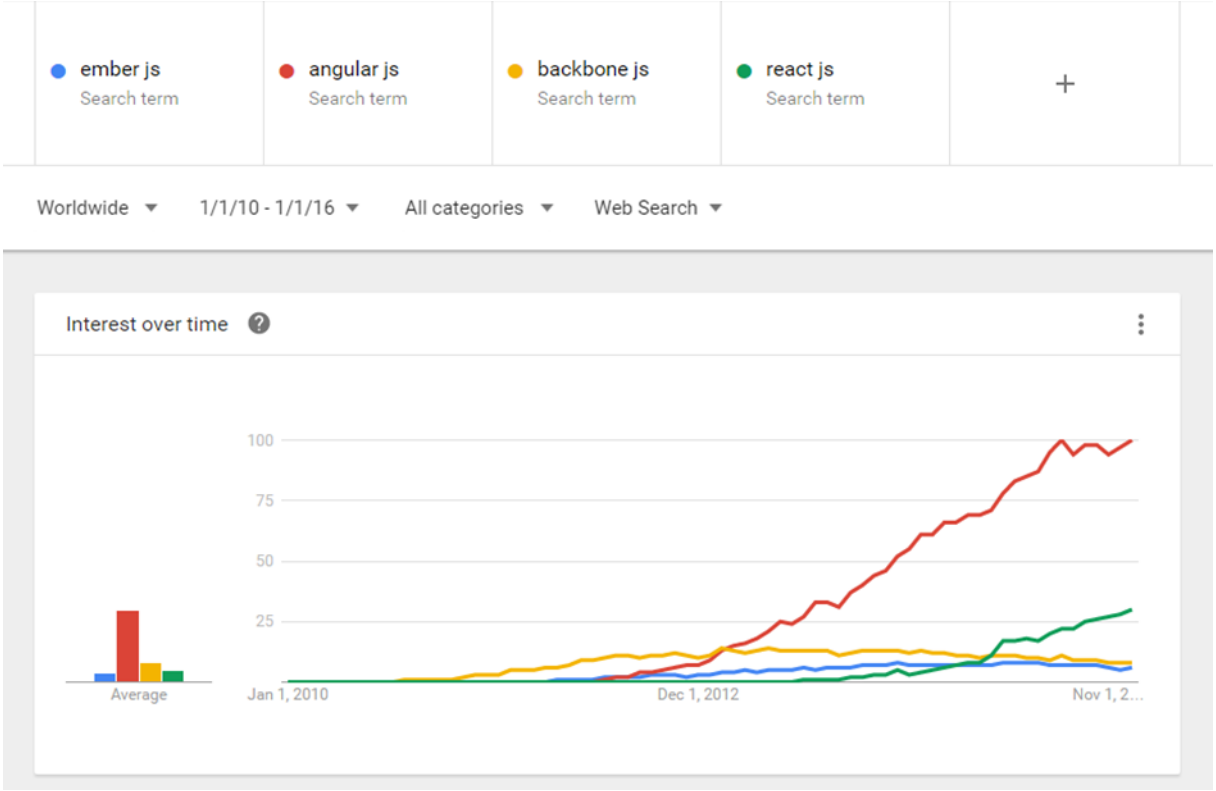


FIGURA 1 – GOOGLE TRENDS FRAMEWORKS JS
FONTE: Google (2016)

2.7 BOOTSTRAP

Definido pelos próprios criadores, “Bootstrap é o framework de HTML, CSS e JS mais popular para o desenvolvimento de projetos web com foco adaptativo à mobile.” (BOOTSTRAP, 2016)

Aliado ao Angular, e também fazendo uso do framework jQuery, o Bootstrap reúne uma grande variedade de componentes e comportamentos para serem aplicados no *front-end* da aplicação, ou seja, na sua visualização e interação com o usuário. Uma de suas grandes vantagens é o suporte nativo dos componentes à adaptação da visualização mobile, onde através de alteração automática do comportamento de algumas diretivas da visualização dos componentes é possível obter em *tablets* e *smartphones* uma experiência de uso tão fluida quanto o da aplicação original na web.

Entre os componentes de layout do Bootstrap encontram-se botões, ícones, componentes de menus, navegação e paginação, todos construídos sobre a plataforma de grids responsivos que adaptam visualização dos elementos em cada tipo de dispositivo utilizado.

2.8 DJANGO

Django é um *framework* para desenvolvimento web, baseado na linguagem Python. O framework foca na construção de páginas de conteúdo dinâmico, de maneira rápida, prática e reutilizável, subtraindo o máximo possível de repetição de código durante o desenvolvimento. (HOLOVATY e KAPLAN-MOSS, 2008)

O projeto do *framework* teve início em 2003 quando dois desenvolvedores passaram a utilizar Python para construir aplicações para uma rede de notícias. Os conteúdos precisam ser modificados com muita frequência e apresentando alguns recursos dinâmicos, e por conta da alta quantidade de trabalho ser refeito, aos poucos os desenvolvedores começaram a moldar o seu próprio *framework*. Devido a esse histórico, o Django é considerado um dos solucionadores de problemas com linguagem mais próxima à dos profissionais no mercado, pois foi construída a partir dos problemas vividos no meio, e não a partir de estudos teóricos em universidades.

Ao prover a visão prática e os moldes descritos para o desenvolvimento, o Django provê as abstrações habituais de padrões de desenvolvimento web, atalhos

para tarefas repetitivas durante a codificação, e claras convenções sobre como tratar problemas encontrados no desenvolvimento. Ao mesmo tempo, o framework é aberto o suficiente para deixar o programador trabalhar fora do seu escopo de funções quando necessário.

Apesar de sua capacidade de controlar todo o ambiente de uma aplicação web, optamos por utilizar o Django para manter uma aplicação de gerenciamento do servidor, como plataforma de suporte de dados para o *front-end*. Para este fim, utilizamos um novo *framework*, o Django REST Framework (DRF).

2.9 DJANGO REST FRAMEWORK

Como o nome deixa evidente, este é um *framework* baseado em Django para utilização da arquitetura REST. Através da sua utilização é possível desenvolver uma API web que nos fornecerá uma forma de comunicação entre o *front-end* e o banco de dados, servindo então como interface para praticamente todas as operações com dados na nossa aplicação, desde a autenticação e criação de usuários, até busca e filtragem de dados e informações relevantes à seções específicas do sistema.

Algumas das vantagens do DRF são:

a) disponibilização de uma API Web para consultas e testes da API em desenvolvimento na aplicação. É possível utilizar as mesmas URLs definidas no sistema durante a sua construção e verificar seu acesso e suas respostas, assim como a navegação pelos resultados retornados, simulando o *feedback* que a nossa aplicação no *front-end* terá como resposta às suas requisições;
b) sistema próprio de gerenciamento de autenticação, deixando o desenvolvedor livre para seguir as políticas já definidas por padrão e construir sua aplicação seguindo as diretrizes deste modelo, ou mesmo para utiliza-las apenas como base para a construção do seu próprio módulo de autenticação;
c) serialização dos dados pode ser feita tanto através de dados recebidos por um sistema ORM (Mapeamento Objeto-Relacional) ou non-ORM, provendo então suporte para uma extensa gama de fontes e bancos de dados que serão construídos e servidos com base nos modelos desenvolvidos no DRF.

Com base nesses preceitos, o desenvolvimento *back-end* da aplicação construída nesse trabalho é completamente baseada no Django e seu Rest Framework. Trabalhando em conjunto, os modelos de dados são construídos e armazenados num banco de dados PostgreSQL controlado unicamente pelo Django, que fica também responsável por prover os dados a partir das requisições do *front-end*, utilizando a serialização dos mesmos para a transmissão pela rede.

2.10 UNIFIED MODELING LANGUAGE

A linguagem UML (Unified Modeling Language) é utilizada como um recurso para modelagem de um sistema e como um meio para levantamentos dos requisitos. (GUEDES, 2011) Através de diagramas é possível representar vários aspectos do sistema, independentemente da linguagem utilizada para o seu desenvolvimento. Neste trabalho foi utilizado o diagrama de caso de uso, o diagrama de classes, o diagrama de atividades e o diagrama de componentes. O diagrama de casos de uso é voltado a apresentação das funcionalidades do sistema e suas relações com os usuários. O diagrama de classes é utilizado para a visualização do sistema do ponto de vista de Orientação a Objetos. O diagrama de atividades apresenta as tarefas necessárias para realização de uma funcionalidade (atividade) do sistema. Já o diagrama de componentes se refere aos componentes que fazem parte do sistema e suas dependências.

3 METODOLOGIA

Este capítulo apresenta as metodologias, linguagens e ferramentas utilizadas para o desenvolvimento do projeto.

3.1 MODELO DE PROCESSO DE ENGENHARIA DE SOFTWARE

O modelo escolhido para este trabalho foi o modelo RUP (Rational Unified Process). Esse modelo é formado por 4 fases distintas, cada uma finalizada por um milestone, um ponto definido e decisivo da evolução do projeto. As fazes do modelo RUP são:

a) Concepção: Fase inicial do RUP, onde é definido o escopo do projeto através do levantamento dos requisitos e casos de usos iniciais para a aceitação dos stackholders;
b) Elaboração: Fase voltada para a definição da arquitetura do sistema e do plano do projeto;
c) Construção: Durante essa fase, os componentes do sistema são desenvolvidos e testados;
d) Transição: Fase de implantação do sistema, onde o objetivo é a entrega final do sistema ao cliente para uso. (IBM, 2011)

3.2 PLANO DE ATIVIDADES

O plano de atividades apresenta as ferramentas utilizadas para o gerenciamento e acompanhamento do desenvolvimento do projeto. Foram utilizados o diagrama WBS e o diagrama de Gantt.

3.2.1 Diagrama WBS

WBS (*Work Breakdown Structure*), também conhecida como EAP (Estrutura Analítica de Trabalho), é uma ferramenta de divisão das entregas e do trabalho do projeto em componentes menores e de fácil gerenciamento. O objetivo de um WBS é facilitar o entendimento de tudo que deve ser feito, organizando e definindo cada parte do escopo do projeto e diminuindo a complexidade de cada atividade ao máximo.

Segundo o PMI (2008), o WBS é "uma decomposição hierárquica orientada às entregas do trabalho a ser executado pela equipa de projeto, para atingir os objetivos do projeto e criar as entregas requisitadas".

Para o projeto desenvolvido, segue o WBS conforme Figura 2.

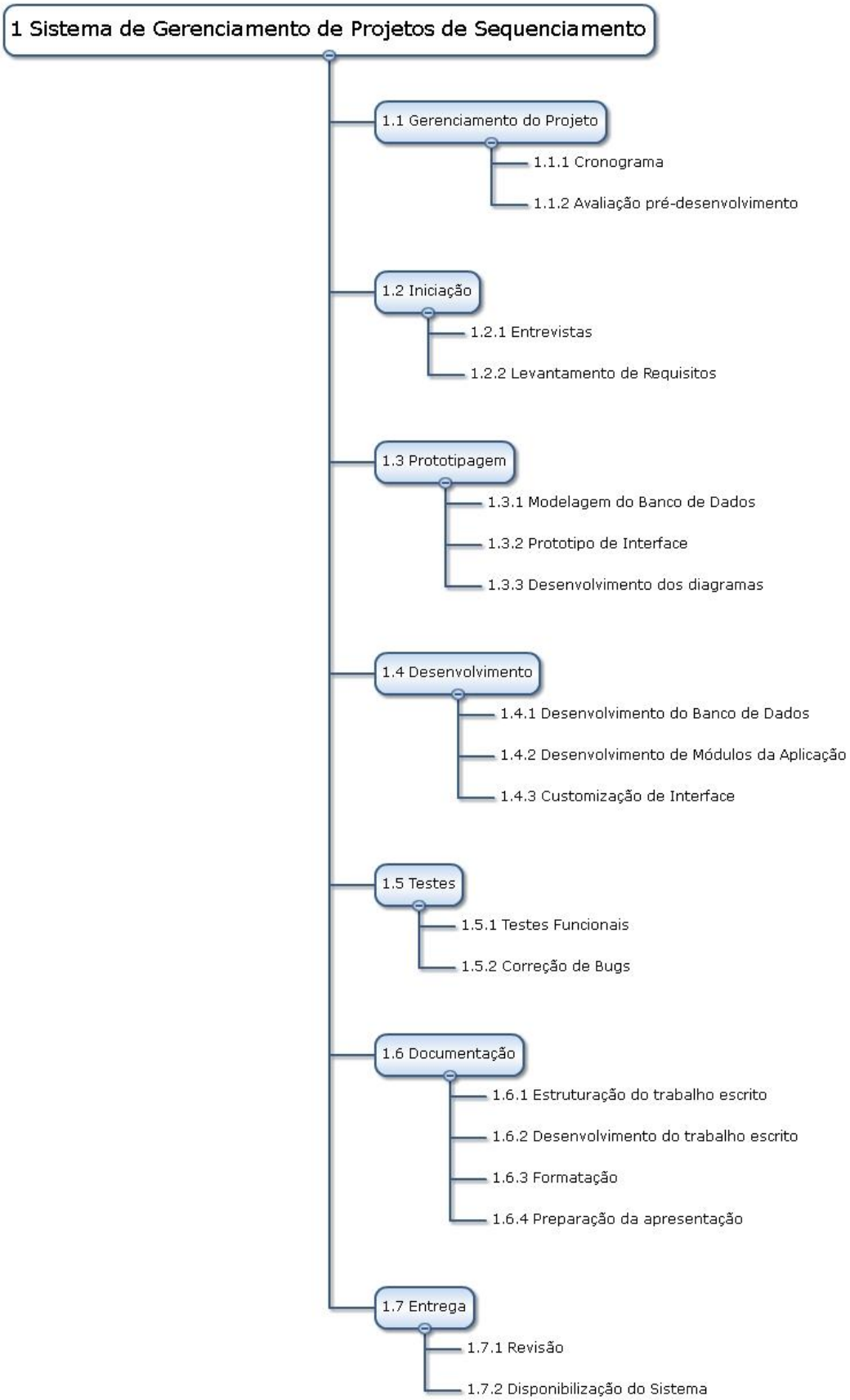
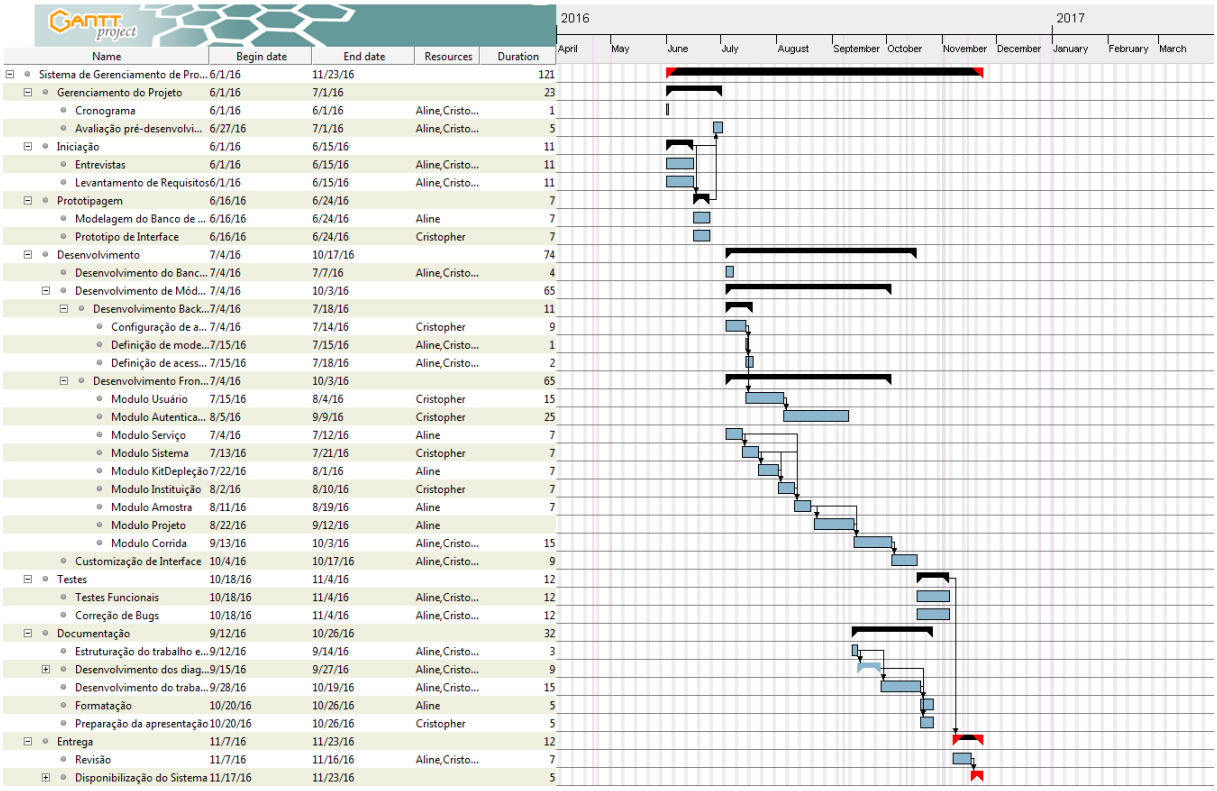


FIGURA 2 – DIAGRAMA WBS
FONTE: Os autores (2016)

3.2.2 Diagrama de Gantt

Outra ferramenta utilizada para o gerenciamento de um projeto é o diagrama de Gantt. O diagrama de Gantt (Figura 3) permite a visualização do cronograma de realização e do responsável por cada tarefa. Ele é construído tendo como base o WBS.



Name	Begin date	End date	Resources	Duration
Sistema de Gerenciamento de Pro...	6/1/16	11/23/16		121
Gerenciamento do Projeto	6/1/16	7/1/16		23
Cronograma	6/1/16	6/1/16	Aline,Cristo...	1
Avaliação pré-desenvolvi...	6/27/16	7/1/16	Aline,Cristo...	5
Iniciação	6/1/16	6/15/16		11
Entrevistas	6/1/16	6/15/16	Aline,Cristo...	11
Levantamento de Requisitos	6/1/16	6/15/16	Aline,Cristo...	11
Prototipagem	6/16/16	6/24/16		7
Modelagem do Banco de ...	6/16/16	6/24/16	Aline	7
Prototipo de Interface	6/16/16	6/24/16	Cristopher	7
Desenvolvimento	7/4/16	10/17/16		74
Desenvolvimento do Banc...	7/4/16	7/7/16	Aline,Cristo...	4
Desenvolvimento de Mód...	7/4/16	10/3/16		65
Desenvolvimento Back...	7/4/16	7/18/16		11
Configuração de a...	7/4/16	7/14/16	Cristopher	9
Definição de mode...	7/15/16	7/15/16	Aline,Cristo...	1
Definição de acess...	7/15/16	7/18/16	Aline,Cristo...	2
Desenvolvimento Fron...	7/4/16	10/3/16		65
Modulo Usuário	7/15/16	8/4/16	Cristopher	15
Modulo Autentica...	8/5/16	9/9/16	Cristopher	25
Modulo Serviço	7/4/16	7/12/16	Aline	7
Modulo Sistema	7/13/16	7/21/16	Cristopher	7
Modulo KitDepleção	7/22/16	8/1/16	Aline	7
Modulo Instituição	8/2/16	8/10/16	Cristopher	7
Modulo Amostra	8/11/16	8/19/16	Aline	7
Modulo Projeto	8/22/16	9/12/16	Aline	
Modulo Corrida	9/13/16	10/3/16	Aline,Cristo...	15
Customização de Interface	10/4/16	10/17/16	Aline,Cristo...	9
Testes	10/18/16	11/4/16		12
Testes Funcionais	10/18/16	11/4/16	Aline,Cristo...	12
Correção de Bugs	10/18/16	11/4/16	Aline,Cristo...	12
Documentação	9/12/16	10/26/16		32
Estruturação do trabalho e...	9/12/16	9/14/16	Aline,Cristo...	3
Desenvolvimento dos diag...	9/15/16	9/27/16	Aline,Cristo...	9
Desenvolvimento do traba...	9/28/16	10/19/16	Aline,Cristo...	15
Formatação	10/20/16	10/26/16	Aline	5
Preparação da apresentação	10/20/16	10/26/16	Cristopher	5
Entrega	11/7/16	11/23/16		12
Revisão	11/7/16	11/16/16	Aline,Cristo...	7
Disponibilização do Sistema	11/17/16	11/23/16		5

FIGURA 3 – DIAGRAMA DE GANTT
FONTE: Os autores (2016)

3.3 MATERIAIS

Os materiais e ferramentas utilizados para o desenvolvimento do trabalho serão apresentados nessa sessão.

3.3.1 Ambiente de Desenvolvimento

Para o desenvolvimento foram utilizados os seguintes computadores:

a) Notebook DELL INSPIRON 5680, processador Intel i5-3337U 1.8GHz, 8GB de memória RAM, sistema operacional Windows 10 Home 64bits;
b) Máquina Virtual Intel i5-3337U 1.8GHz, 2GB de memória RAM, sistema operacional Linux Mint 18;
c) Desktop Intel i5-750 2.7GHz, 6GB de memória RAM, sistema operacional Windows 10 Pro 64bits;
d) Notebook HP tm2-1070us, processador Intel Core 2 Duo U7300 1.30GHz, 4GB de memória RAM, sistema operacional Windows 7 Home Premium 64bits.

3.3.2 IDE

Para a construção e desenvolvimento do código fonte, fazemos uso de ferramentas específicas de edição que estendem as características de um editor de textos comum, são estes as chamadas IDEs (Integrated Development Environment), ou “Ambiente de Desenvolvimento Integrado”. Segundo PUREWAL(2014), “esses produtos normalmente possuem características que facilitam a manipulação do código, mas também diversas outras características importantes para o desenvolvimento de código a nível empresarial.” As seguintes IDEs foram utilizadas durante o desenvolvimento da aplicação:

a) ATOM 1.12.5 Editor by GitHub: Editor de texto e código, *opensource*, construído em HTML, CSS e Javascript, e baseado no *framework* multiplataforma Electron, permite alto nível de customização através de funções nativas e também pacotes desenvolvidos pela comunidade;

b) Notepad ++ v6.9.2:Editor de texto simples, feito para substituir o bloco de notas padrão utilizando um sistema semelhante, leve e capaz de reconhecer diversas linguagens de programação, servindo também então como editor de código;
c) Sublime Text version 2.0.2: Editor de texto com enfoque na edição de códigos, multiplataforma, altamente customizável e com suporte a dezenas de linguagens de programação. Entre suas principais capacidades estão o fácil gerenciamento de projetos e navegação de arquivos, paleta de comandos e uma API de suporte a plugins Python para melhorias na aplicação.

3.3.3 Ferramentas de Suporte *Front-End*

Para o desenvolvimento do *Front-End* da aplicação, foram utilizadas algumas ferramentas de automação e gerenciamento.

3.3.3.1 Grunt

Ferramenta de automatização de tarefas para pacotes JavaScript. O Grunt é capaz de gerenciar dependências, atualizações, *deploy* e concatenação de arquivos de configuração do servidor. No nosso projeto, utilizamos essa ferramenta para gerenciar o *template* sb-admin, carregar nossas dependências customizadas, e gerenciar os pacotes do AngularJS que trabalharão com o framework do servidor.

Além do gerenciamento de inicialização, o Grunt reinicializa o servidor e recarrega qualquer dependência ou arquivo fonte alterado no ambiente de desenvolvimento, facilitando os testes e diminuindo o tempo de espera e carregamento manual que precisaria ser feito a cada refatoração ou adição no código. (GRUNT, 2016)

Grunt e suas dependências são executadas a partir do NPM, o gerenciador de pacotes do Node.js.

3.3.3.2 ocLazyload

Ferramenta de gestão de dependências desenvolvida para o AngularJS. Utilizamos os métodos do ocLazyLoad em conjunto às declarações de rotas do Angular, e este fica responsável por buscar os controladores e serviços requisitados,

carregando automaticamente os arquivos necessários para o modulo a ser ativado. Com essa ferramenta podemos carregar dinamicamente múltiplos arquivos HTML, CSS, JavaScript, e utilizar recursos como limitação e cache e chamados assíncronos se necessário. (OCLAZYLOAD, 2016)

3.3.3.3 NPM e Node.js

O Node.js é uma plataforma construída nos moldes do JavaScript com foco no desenvolvimento do lado servidor da aplicação, para servir como base para a construção de aplicativos ou servidores JS. Seu gerenciador de pacotes e extensões é o NPM, e ambos funcionam como suporte de instalação e execução do Grunt, nosso automatizador de tarefas e servidor HTTP. (NPM ENTERPRISE, 2016)

3.3.3.4 Bower

Componente para gerenciamento de pacotes de aplicações web, capaz de instalar e manter componentes de acordo com a configuração descrita, sejam estes pacotes CSS, JavaScript ou até mesmo fontes e imagens (BOWER, 2016). A instalação dos pacotes deste trabalho depende diretamente da configuração do Bower, cujas diretivas podem ser encontradas no arquivo LEIA-ME.

3.3.4 Ferramentas de Construção de Diagramas e Modelagem

Para a construção dos diagramas foram utilizadas as seguintes ferramentas:

a) WBS Tool v0.9 beta: Software online gratuito que permite a criação de WBSs, organogramas e outros diagramas;
b) GanttProject 2.8.1 Pilsen (build 2024): Software gratuito para elaboração de diagramas de Gantt;
c) Draw.iov6.0.1.7: Software online gratuito que permite a elaboração de diagramas, desenhos.

4 DESENVOLVIMENTO DO PROJETO

Neste capítulo será apresentado o processo de desenvolvimento do projeto, desde as primeiras definições até a finalização do mesmo.

4.1 LEVANTAMENTO DE REQUISITOS

O início do projeto se deu ainda no final do ano de 2015, quando após apresentação da ideia tivemos uma reunião inicial com os principais profissionais que viriam a ser os utilizadores locais do sistema. Nesta ocasião pudemos ter conhecimento das nossas fontes de dados (equipamentos e serviços) e que tipo de informações eram gerenciadas atualmente e necessitariam ser digitalizadas e controladas pela aplicação. A partir desses dados, pudemos construir o primeiro modelo de Diagrama Relacional que usaríamos como base para a concepção inicial do sistema.

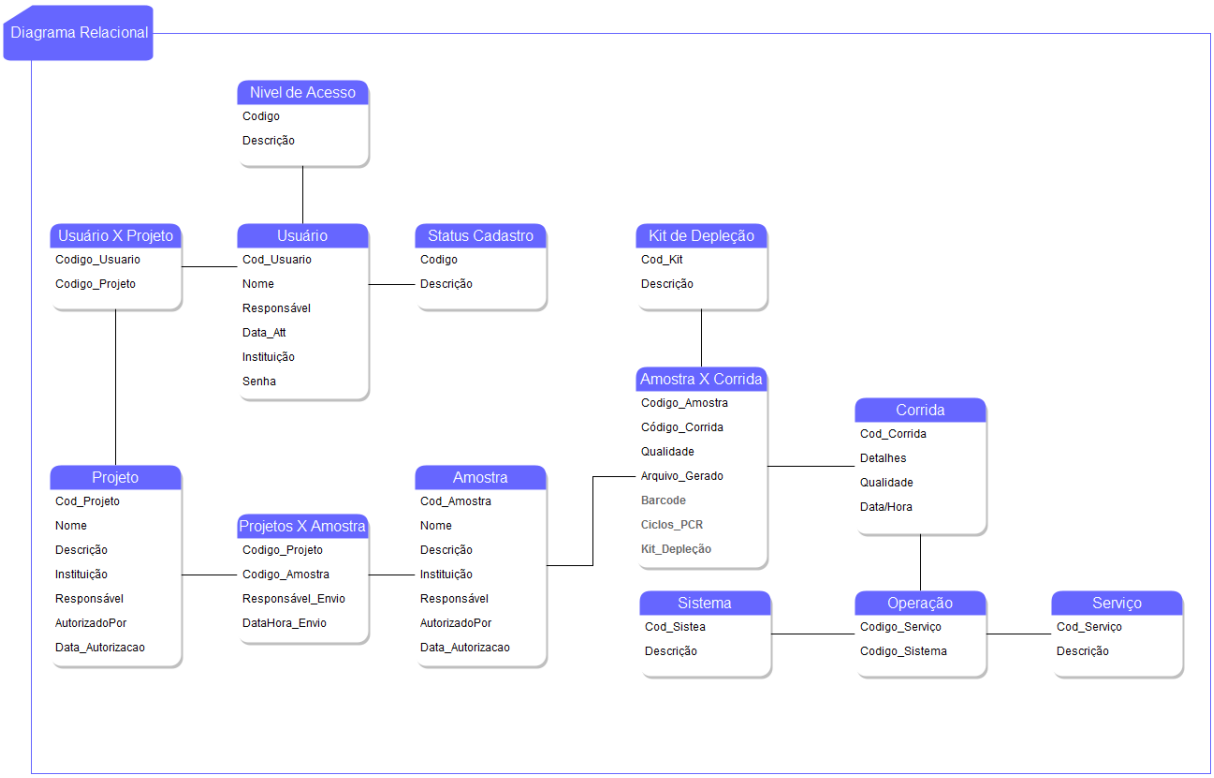


FIGURA 4 – ESBOÇO INICIAL DIAGRAMA ENTIDADE-RELACIONAMENTO
FONTE: Os autores (2016)

Com esse modelo em mãos e diversas dúvidas sobre a especificação dos dados que julgávamos necessários pudemos realizar uma série de questionamentos à coordenadora do núcleo, cujas respostas possibilitaram a elaboração do conceito de módulos da aplicação: Amostras, Projetos, Usuários e Corrida.

4.2 DEFINIÇÃO DOS MÓDULOS

Os módulos são os conceitos chave sobre os quais decidimos estruturar nossa aplicação. A partir das interações desses módulos é que as operações ocorrem e é do relacionamento de seus dados intrínsecos que poderemos avaliar estatisticamente o funcionamento do sistema.

4.2.1 Usuários

Os usuários são os utilizadores que irão nutrir o sistema com informações. Os usuários podem ser os parceiros e pesquisadores, que irão cadastrar amostras para envio e identificar seus projetos para controle das mesmas, e podem também ser os profissionais administradores do sistema, os responsáveis pela realização das análises e divulgação de seus resultados. São os administradores que adicionam as informações sobre as corridas de sequenciamento, atualizam os status das amostras e também gerenciam as informações cadastrais de outros usuários e também de seus projetos.

4.2.2 Projeto

Um projeto é a pesquisa ou estudo que une as amostras e os utilizadores. É através do projeto que uma amostra é inicialmente atrelada e assim identificamos quais são os outros participantes que irão também acompanhar o desenvolvimento desta no sistema. Os projetos são vinculados a uma universidade e contém uma lista de colaboradores que podem visualizar seu conteúdo e realizarem novos envios de amostrar para a UFPR.

4.2.3 Amostras

Objeto principal de estudo, tendo como objetivo final os fatos obtidos de sua análise. A amostra é uma parte de material biológico enviado pelos parceiros/pesquisadores para análise no Núcleo de Fixação Biológica de Nitrogênio. Seu desenvolvimento é acompanhado no sistema desde o seu cadastro e envio até o estado definido pelo final de sua análise. Todas as amostras são vinculadas a um projeto, assim facilmente definimos quais são as pessoas que poderão acompanhar o seu progresso e ter acesso aos resultados das corridas de sequenciamento.

4.2.4 Corridas

As corridas são as detentoras das informações finais do sistema, no caso, os resultados das análises. As corridas são realizadas utilizando um “sistema” (equipamento de análise do laboratório) e um “serviço” (tipo de análise realizada), e ambos aspectos são definidos pelos administradores na aplicação para que sejam atrelados a toda análise realizada.

4.2.5 Outros módulos

Além desses conceitos principais, algumas outras notas foram importantes para a definição dos modelos:

a) Kit de Depleção: Material de análise utilizado em cada corrida. Podem haver diversos tipos diferentes, portanto, os administradores podem gerenciar o cadastro desses tipos;
b) Sistema: Maquinas/Equipamentos que realizam o processo de análise. Recebem uma série de amostras e kits de depleção e retornam uma série de informações sobre cada amostra inserida;
c) Serviço: Tipo de análise realizada na corrida de sequenciamento;
d) Níveis de Acesso: Para garantir que nenhum usuário possa visualizar informações que não lhe competem, tampouco alterar definições do sistema, um administrador do sistema define no cadastro do usuário qual é o tipo de restrição de acesso, para garantir que nenhuma regra definida na aplicação seja quebrada.

4.3 PROTOTIPAGEM

Com os dados em mãos e modelos definidos, iniciamos a prototipagem do sistema. Através da utilização do software Axure (AXURE, 2016) efetuamos a diagramação dos componentes, definição de estilos e paleta de cores. Procuramos seguir uma orientação base no design, se tratando de uma aplicação de página única, portanto os elementos mutáveis são apenas os que fazem parte do conteúdo central da página, enquanto todos os menus e cabeçalhos se mantém os mesmos, apenas refletindo mudanças nos estados de seus objetos.

4.3.1 Tela Principal

A tela principal tem como intuito mostrar algumas informações resumidas e atualizadas sobre os status dos principais elementos do sistema: Amostras e Projetos. Nossa ideia conceitual foi reunir em painéis essas informações para que estivessem facilmente visíveis no momento que o usuário efetuar o *login* no sistema. A tela principal já exibe a definição do menu lateral, cabeçalho, e informação do usuário atual no canto superior direito.

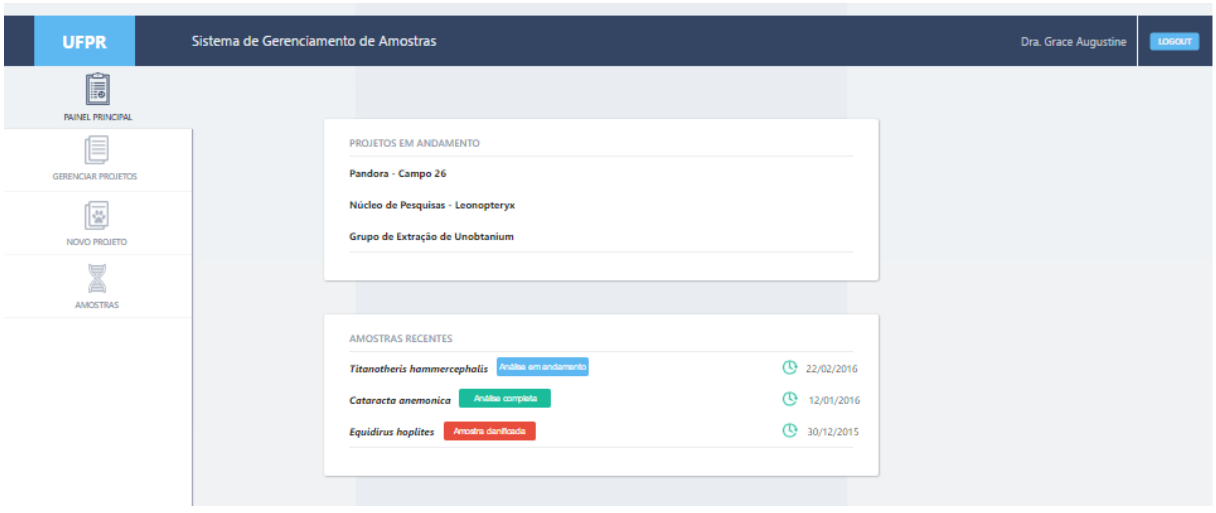


FIGURA 5 – PROTÓTIPO - TELA PRINCIPAL
FONTE: Os autores (2016)

4.3.2 Cadastro de Usuário

Ponto de partida para novos usuários, nesse modelo definimos o formato de apresentação dos dados de formulários. É possível observar na parte inferior do protótipo a presença da paleta de cores e fontes, e definição de elementos de espaçamento.

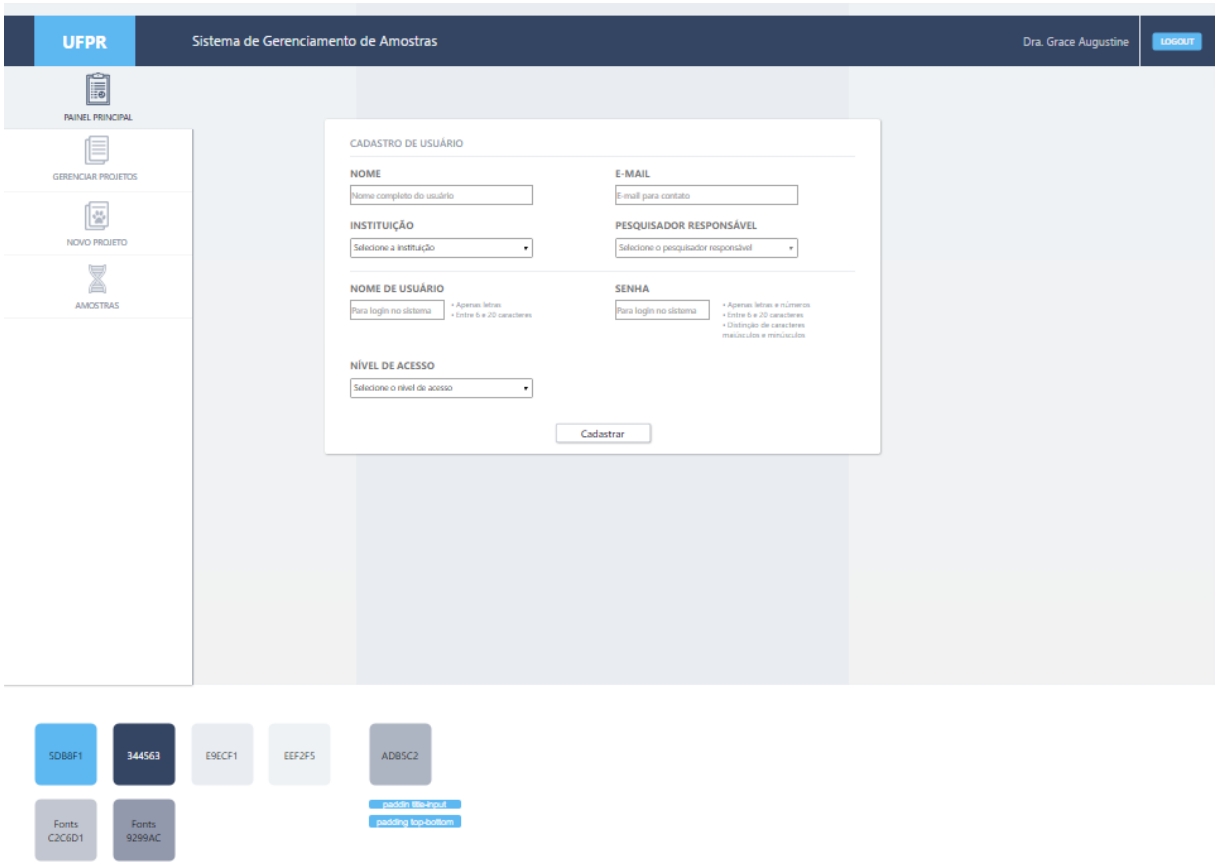


FIGURA 6 – PROTÓTIPO – CADASTRO DE USUÁRIO
FONTE: Os autores (2016)

4.3.3 Cadastro de Amostra

Utilizando o modelo de formulário já apresentado, utilizamos essa tela para introduzir nosso diagrama para formulário de detalhes pop-up, nesse caso sendo utilizado para inserir uma nova amostra na lista que estava sendo gerenciada na tela principal do módulo.

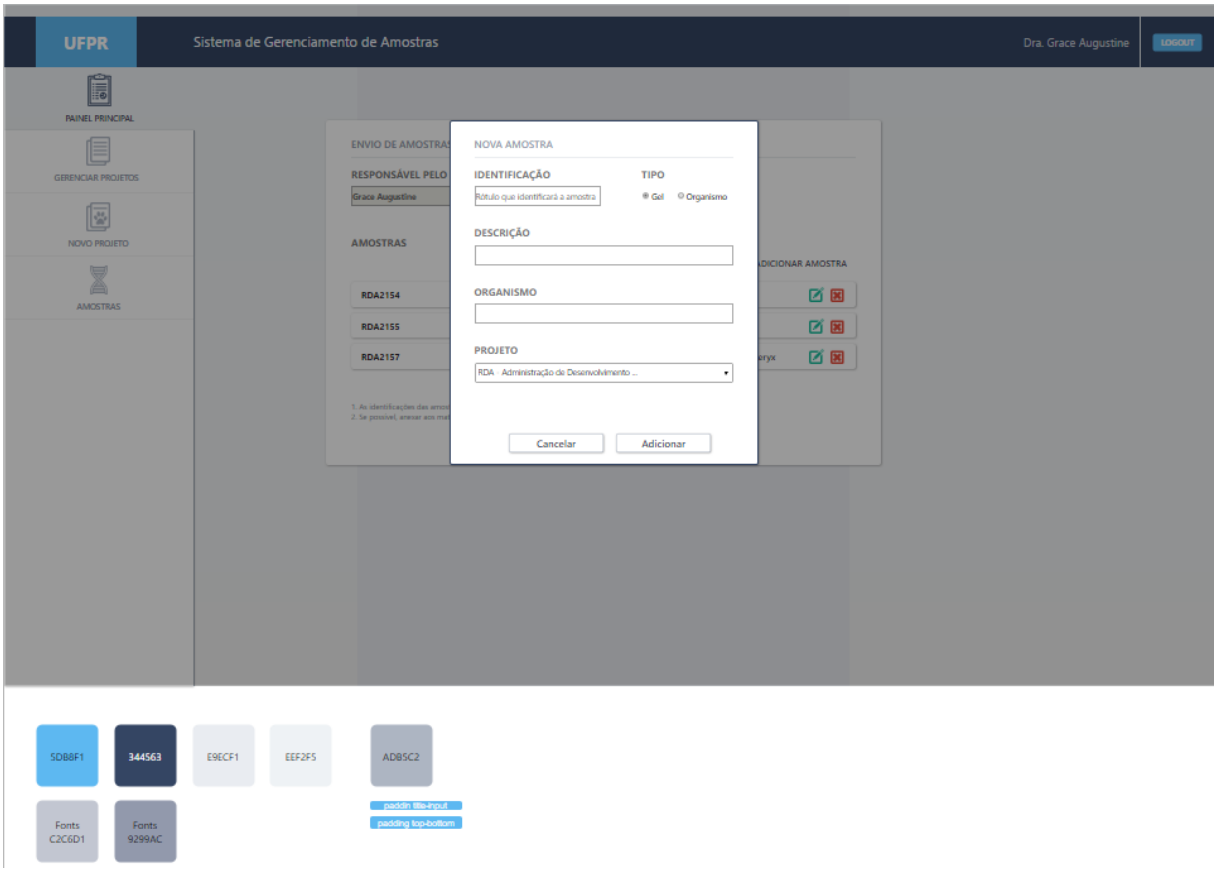


FIGURA 7 – PROTÓTIPO – CADASTRO DE AMOSTRA
FONTE: Os autores (2016)

4.3.4 Cadastro de Projeto

Protótipo inicial da tela de cadastro de um novo projeto.

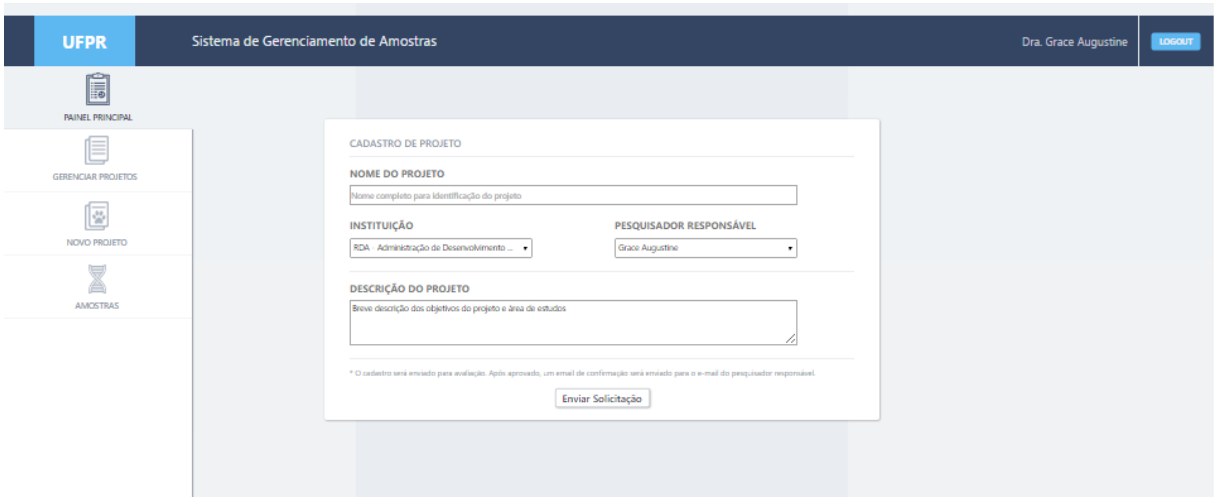


FIGURA 8 – PROTÓTIPO – CADASTRO DE PROJETO
FONTE: Os autores (2016)

4.3.5 Cadastro de Corrida

Protótipo inicial da tela de cadastro de uma nova corrida.

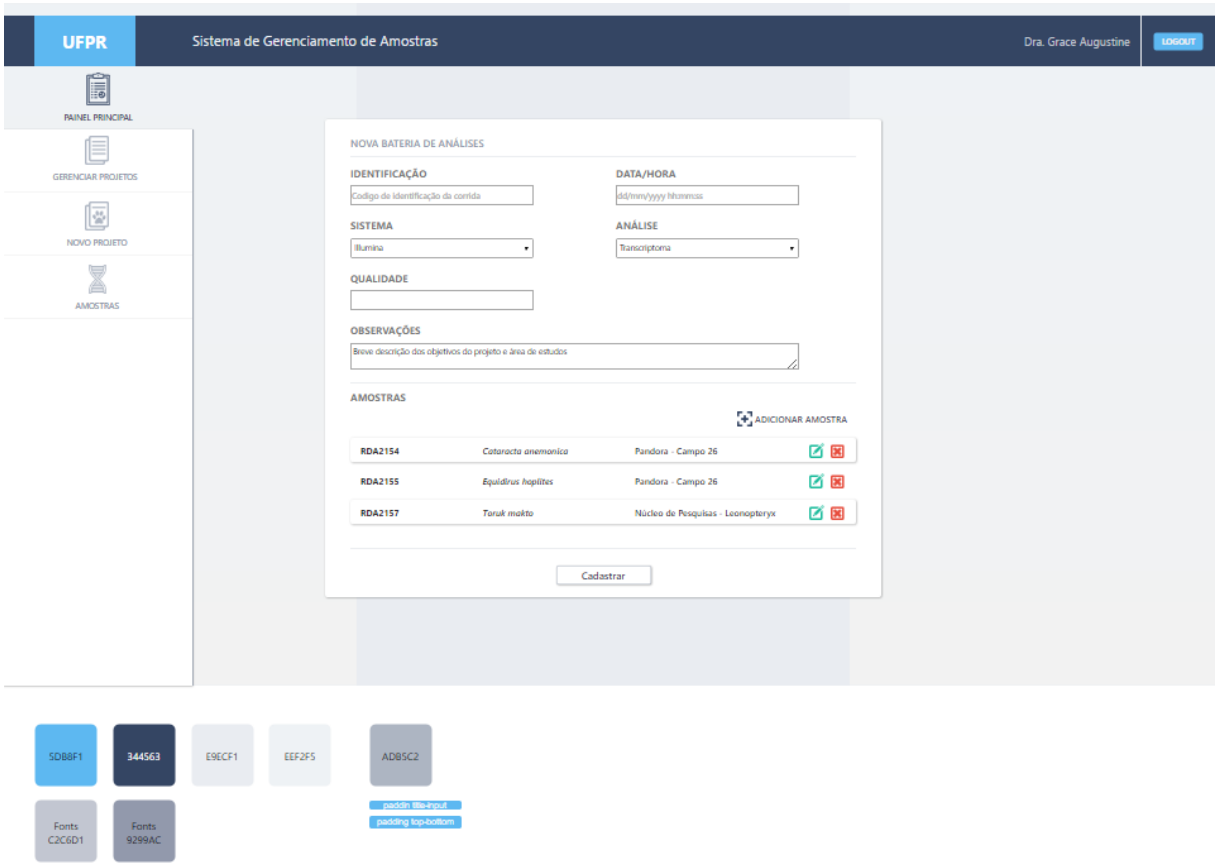


FIGURA 9 – PROTÓTIPO – CADASTRO DE CORRIDA
FONTE: Os autores (2016)

Com a prototipagem dos elementos básicos definida, apresentamos para a equipe responsável e recebemos sua aprovação. A partir desse momento iniciávamos então o desenvolvimento do código da aplicação.

4.4 DEFINIÇÃO DA API

O primeiro passo no desenvolvimento foi definir os modelos que seriam utilizados no banco de dados e suas interações no mesmo. Nesse momento já havíamos optado pela utilização do PostreSQL como banco de dados, devido às nossas experiências anteriores com o mesmo, e também a utilização do Django (framework web Python), escolhido devido as conversas iniciais sobre o projeto e o

destaque que a linguagem Python teve nos diálogos com a nossa orientadora, que ressaltou os esforços dessa linguagem dentro do meio acadêmico da biologia.

Os modelos foram então definidos no Django, codificamos sua interação com o banco de dados (gerenciado inteiramente pelos modelos da linguagem), e declaramos também os pontos de acesso à esses dados, realizados no formato de rotas, como uma API. A partir desta definição temos os seguintes pontos de acesso ao servidor:

a) URI Base da API:
   - x.x.x.x:8000/genseq_api/

b) URIs de acesso aos modelos:
   - x.x.x.x:8000/genseq_api/usuarios
   - x.x.x.x:8000/genseq_api/projeto
   - x.x.x.x:8000/genseq_api/usuarioprojeto
   - x.x.x.x:8000/genseq_api/instituição
   - x.x.x.x:8000/genseq_api/amostra
   - x.x.x.x:8000/genseq_api/projetoamostra
   - x.x.x.x:8000/genseq_api/corrida
   - x.x.x.x:8000/genseq_api/amostracorrida
   - x.x.x.x:8000/genseq_api/servicos
   - x.x.x.x:8000/genseq_api/sistemas
   - x.x.x.x:8000/genseq_api/kit_deplecao

c) URIs especiais utilizadas na autenticação de usuários:
   - x.x.x.x:8000/genseq_api/login
   - x.x.x.x:8000/genseq_api/logout

(Utilizamos neste exemplo a definição “x.x.x.x” para simbolizar o endereço do servidor local)

Essa API será posteriormente consumida pelos serviços definidos no AngularJS, nosso *framework front-end* para esse projeto.

4.5 DESENVOLVIMENTO *FRONT-END*

Para a apresentação dos dados e gerenciamento da nossa aplicação no navegador, escolhemos o *framework* javascript AngularJS, devido a sua popularidade e robustez, verificados através das definições tratadas anteriormente neste trabalho.

Após um período de estudos, começamos nosso trabalho utilizando um *template* de desenvolvimento fornecido pelos profissionais do StartAngular (GEEKYANTS, 2016). Analisamos o *template* “sb-admin” (disponível gratuitamente em https://startbootstrap.com/template-overviews/sb-admin-2/) e baseamos nossa escolha devido aos componentes utilizados no seu desenvolvimento: Bower, NPM/Node.js e Bootstrap. Uma definição breve destes pode ser encontrada na seção de metodologias deste projeto.

A partir dos protótipos definidos e dos diagramas de telas, diagramas de atividades e os demais que compõem este trabalho, iniciamos o desenvolvimento dos módulos de acesso, compostos por basicamente três elementos: *Template* de visualização, *controller* e um serviço para acessos à API e gerenciamento de requisições. As rotas definidas para o acesso desses módulos são as seguintes:

MÓDULO	**ROTAS**
Login	#/login
Home	#/dashboard/home
Projetos	#/dashboard/projetos
Amostras	#/dashboard/amostras
Usuários	#/dashboard/novo_usuario
	#/dashboard/listar_usuarios
	#/dashboard/editar_cadastro
Corridas	#/dashboard/corridas
Instituição	#/dashboard/instituicao
Sistemas	#/dashboard/sistemas
Serviços	#/dashboard/servicos
Kit de Depleção	#/dashboard/kit_deplecao

QUADRO 1 – ROTAS DE ACESSO
FONTE: Os autores (2016)

Além destes, dois módulos compartilhados entre os demais não fazem uso de rotas específicas, são esses: O módulo “Autenticação” e o módulo “Navegação”

4.5.1 Componentes Adicionais

Adicionalmente, fizemos uso de alguns complementos extras durante o desenvolvimento para que pudéssemos atingir os resultados desejados:

a) ngStorage: Modulo de AngularJS desenvolvido pela comunidade para a manipulação de armazenamento de sessão e armazenamento local, ambos recursos definidos pelo navegador utilizado pelo usuário. O modulo está disponível gratuitamente no endereço <https://github.com/gsklee/ngStorage>;
b) jQueryMask Plugin: Complemento desenvolvido pelo membro da comunidade jQuery, Igor Escobar, sob divulgação livre de seu código fonte. O plugin oferece tratamento customizado para elaboração de mascaras para campos de textos e suas manipulações. O plugin está disponível gratuitamente no endereço <https://igorescobar.github.io/jQuery-Mask-Plugin/>;
c) Snackbar: Componente que trata visualmente da exibição de mensagens na tela. É capaz de ser evocado com configurações simples e seu comportamento é padronizado, facilitando a manipulação dos demais eventos que ocorrem na página. O plugin está disponível sob licença de distribuição gratuita do MIT através do endereço <https://github.com/eu81273/angular.snackbar>.

4.6 ACESSO À APLICAÇÃO

A aplicação tem dois tipos de acesso, cada um deles com suas definições e finalidades próprias: Um servidor *back-end* para acesso à API, utilizado apenas pela aplicação do AngularJS e um segundo servidor, este HTTP, que gerencia a aplicação *front-end* e suas requisições tanto para o servidor Django quanto para cada usuário conectado através de seu navegador.

Ambos os servidores estarão ativos em uma máquina servidora localizada no Departamento de Bioquímica da UFPR, e através da configuração local de seus acessos, todos os usuários externos se conectarão ao sistema através de um link definido posteriormente pelos administradores técnicos do setor.

5 APRESENTAÇÃO DO SOFTWARE

Para maior compreensão do trabalho desenvolvido, este capítulo apresentará mais detalhadamente cada tela do sistema. No diagrama de telas (FIGURA 10) é possível identificar todas as telas, possibilitando assim uma visão geral do sistema.

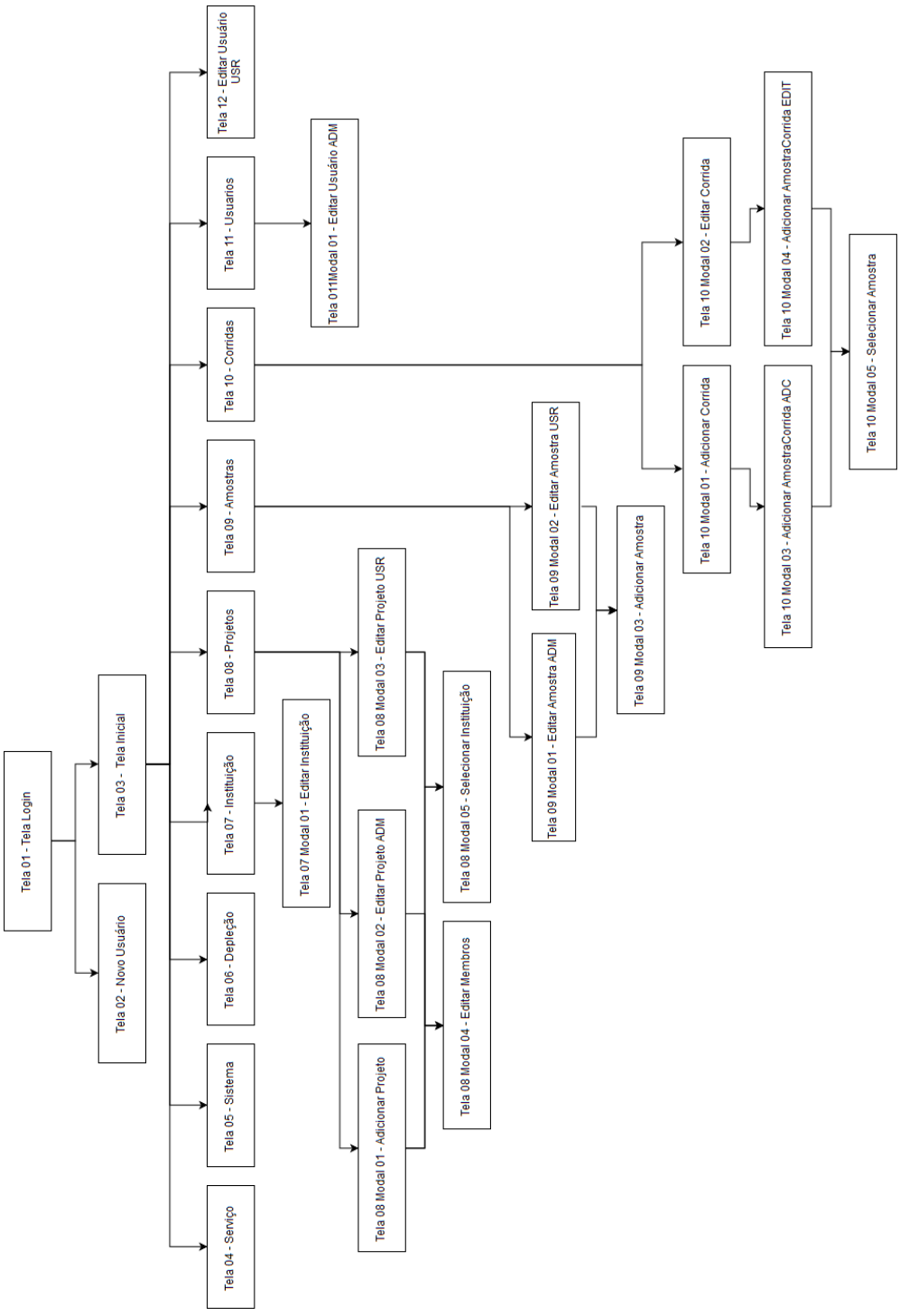


FIGURA 10 – DIAGRAMA DE TELAS
FONTE: Os autores (2016)

5.1 LOGIN

Ao acessar o sistema, a primeira tela que será carregada é a tela de *login* (FIGURA 11). Se o visitante já possuir um cadastro no sistema, pode efetuar o *login* e será redirecionado para a página inicial (FIGURA 12). Caso o visitante deseje realizar o cadastro, poderá fazê-lo utilizando a tela novo usuário (FIGURA 13).

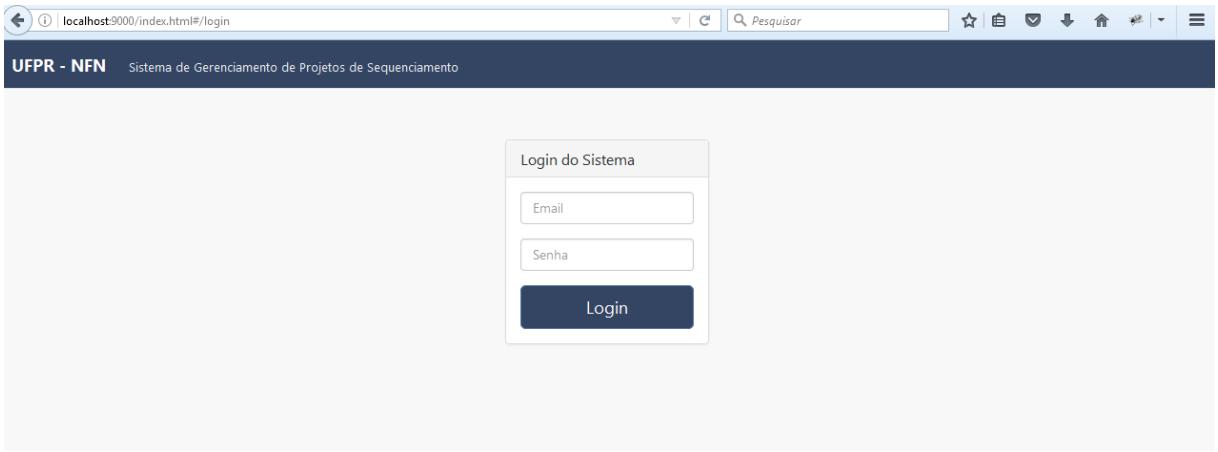


FIGURA 11 – TELA DE LOGIN
FONTE: Os autores (2016)

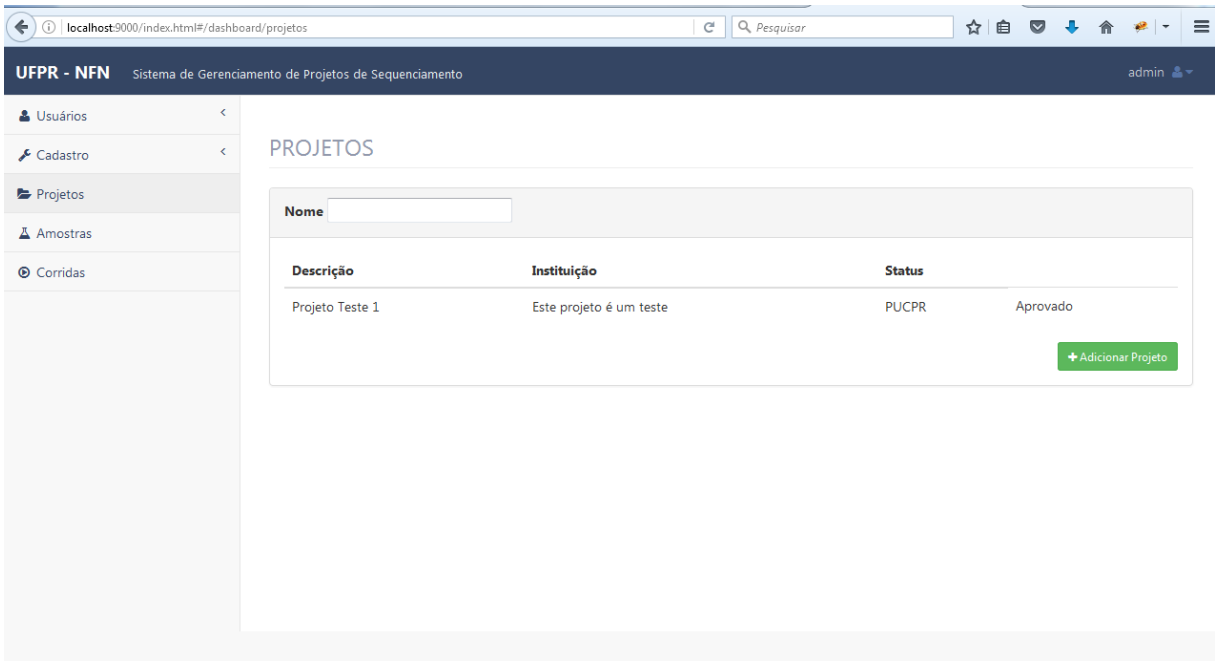


FIGURA 12 – TELA INICIAL DO SISTEMA
FONTE: Os autores (2016)

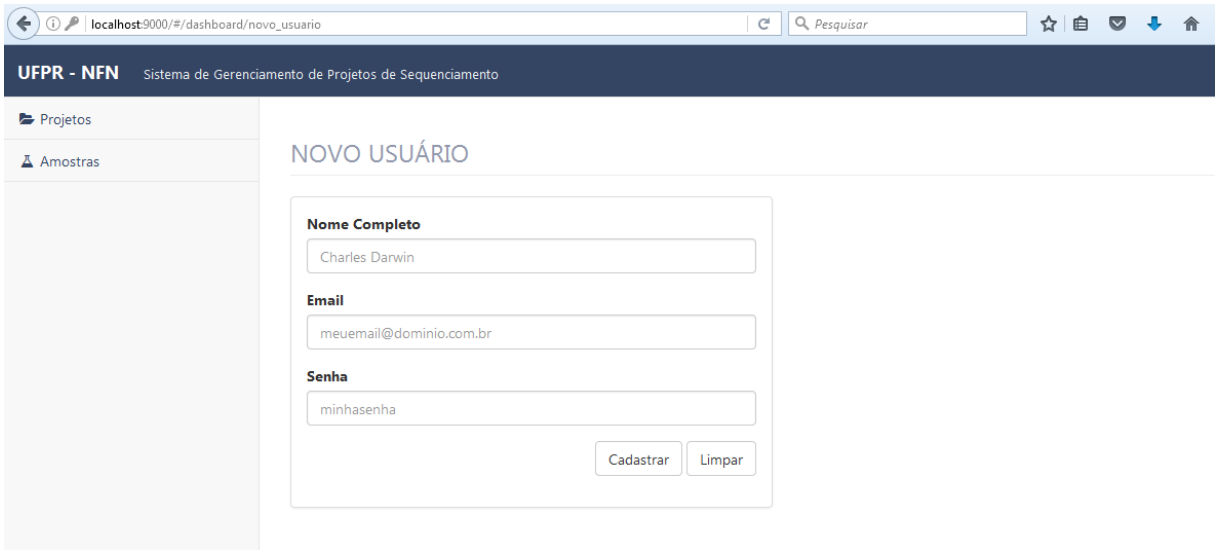


FIGURA 13 – TELA NOVO USUÁRIO
FONTE: Os autores (2016)

5.2 SERVIÇOS, SISTEMAS E KITS DEPLEÇÃO

Os serviços, sistemas e kits depleção devem ser cadastrados para poderem ser utilizados nas outras funcionalidades do sistema. A exclusão dos mesmos é feita na mesma tela onde é feito o cadastro. O acesso a essas telas é restrito a administradores do sistema.

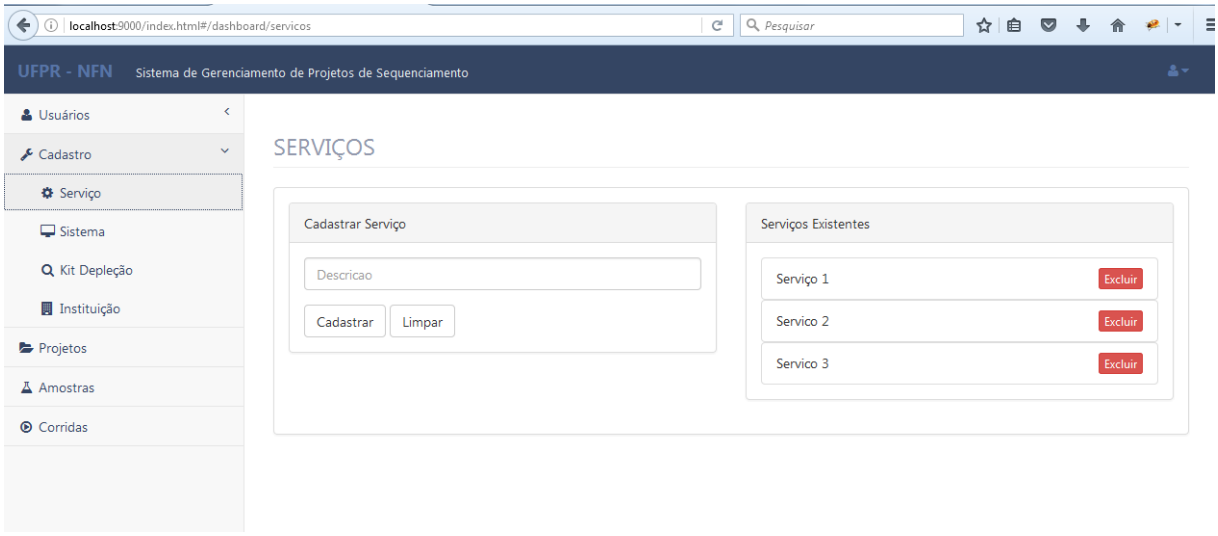


FIGURA 14 – TELA SERVIÇO
FONTE: Os autores (2016)

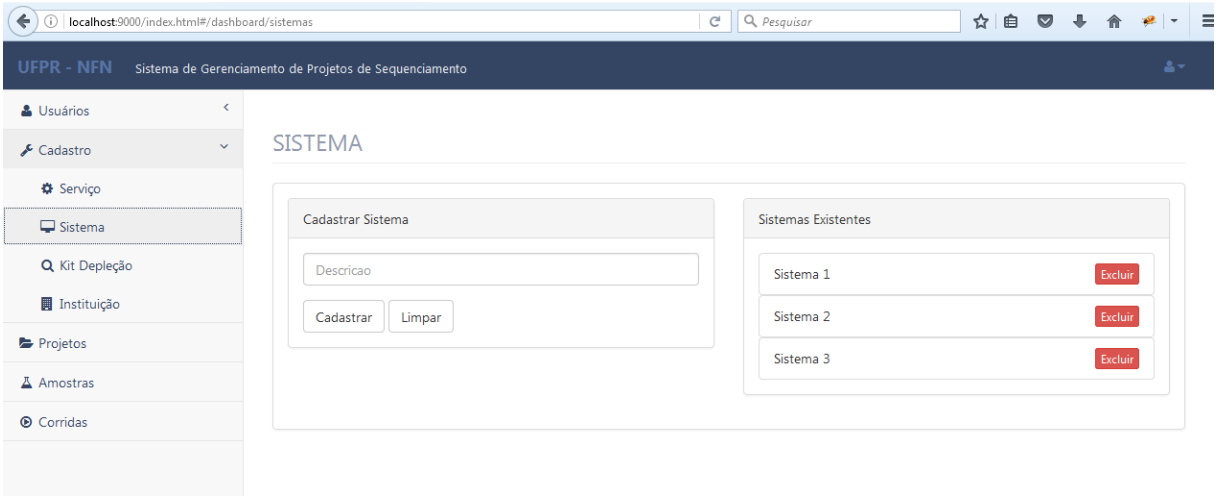


FIGURA 15 – TELA SISTEMA
FONTE: Os autores (2016)

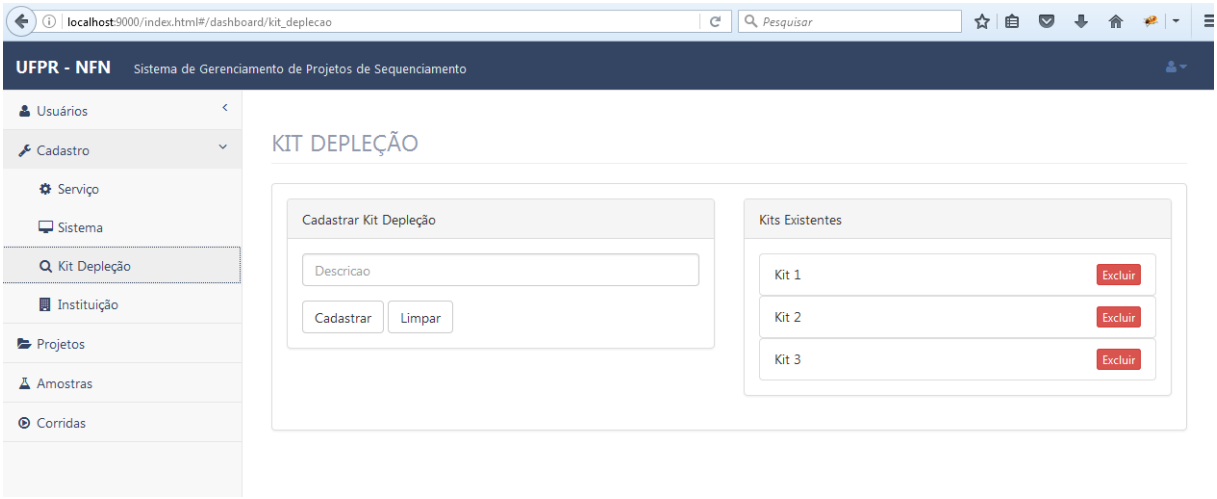


FIGURA 16 – TELA KIT DEPLECAO
FONTE: Os autores (2016)

5.3 INSTITUIÇÃO

Para manutenção das instituições cadastradas, os administradores do sistema podem utilizar a tela instituição (FIGURA 17), que permite o cadastro, edição e exclusão de instituições no sistema.

FIGURA 17 – TELA INSTITUIÇÃO
FONTE: Os autores (2016)

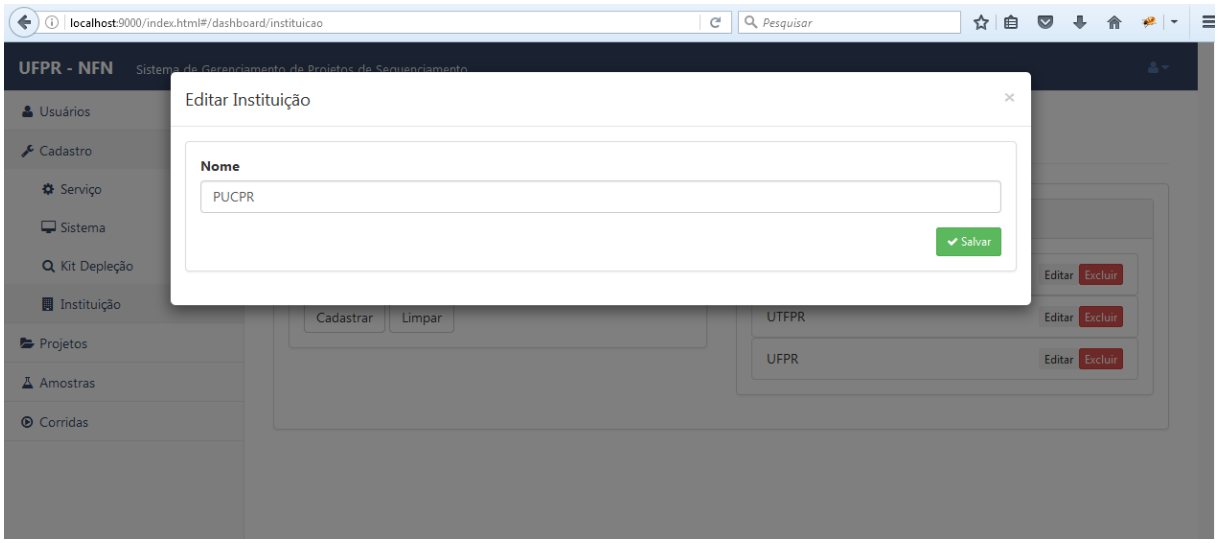


FIGURA 18 – MODAL EDITAR INSTITUIÇÃO
FONTE: Os autores (2016)

5.4 PROJETOS

Os projetos cadastrados podem ser vistos através da tela projetos (FIGURA 19). Nela é apresentada uma lista diferenciada de acordo com o nível de acesso que o usuário possui:

a) Administrador: São listados todos os projetos já cadastrados, tendo os projetos ainda não aprovados no início da lista;
b) Usuário: São listados apenas os projetos em que o usuário logado é membro.

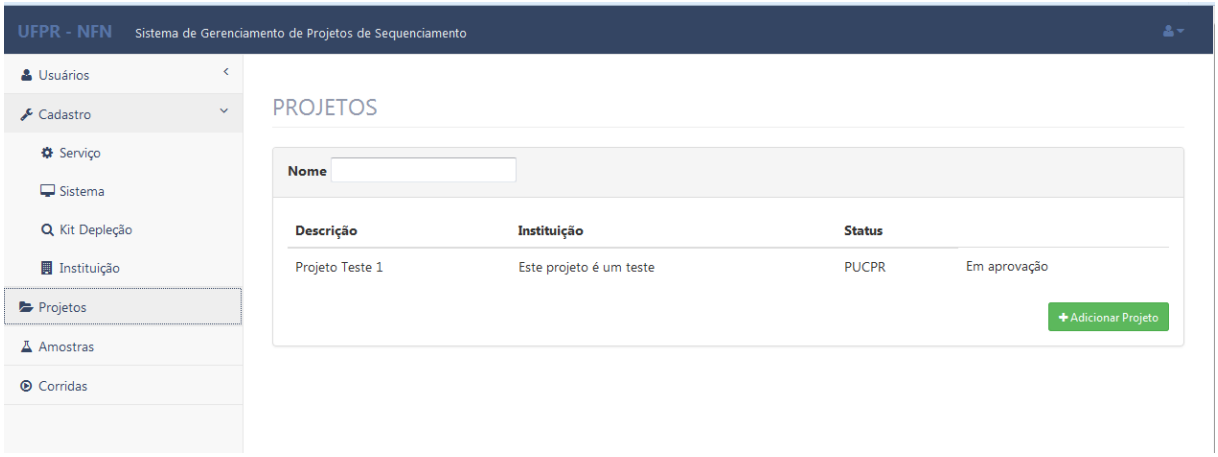


FIGURA 19 – TELA PROJETOS
FONTE: Os autores (2016)

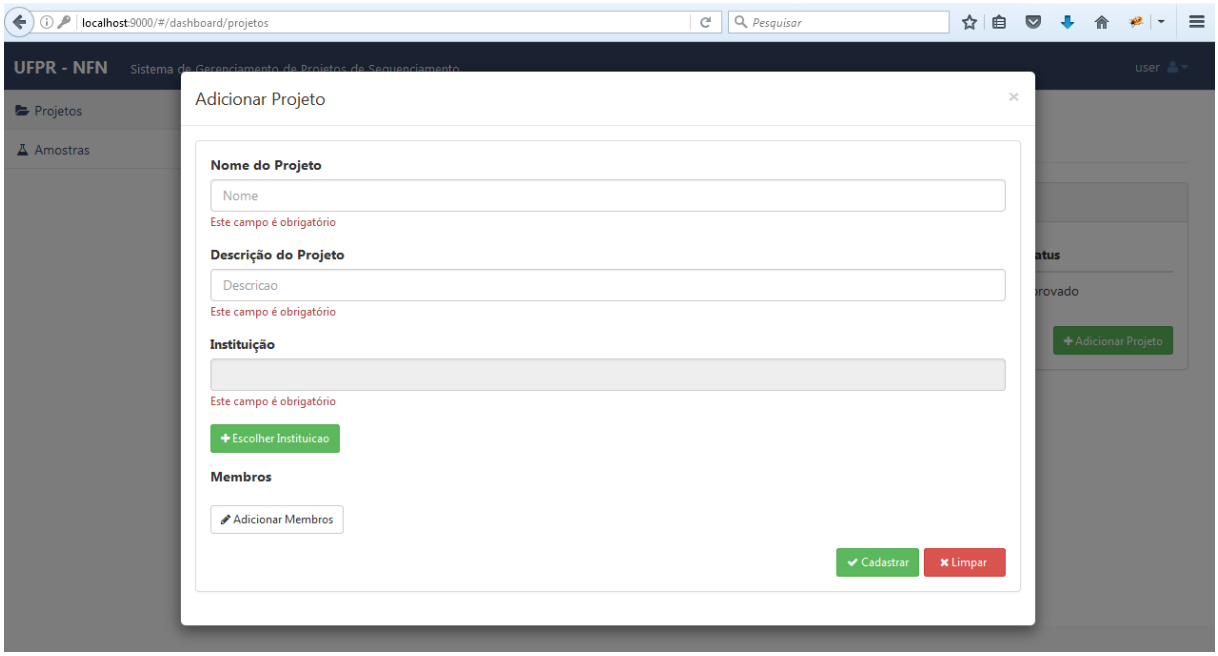


FIGURA 20 – MODAL ADICIONAR PROJETO
FONTE: Os autores (2016)

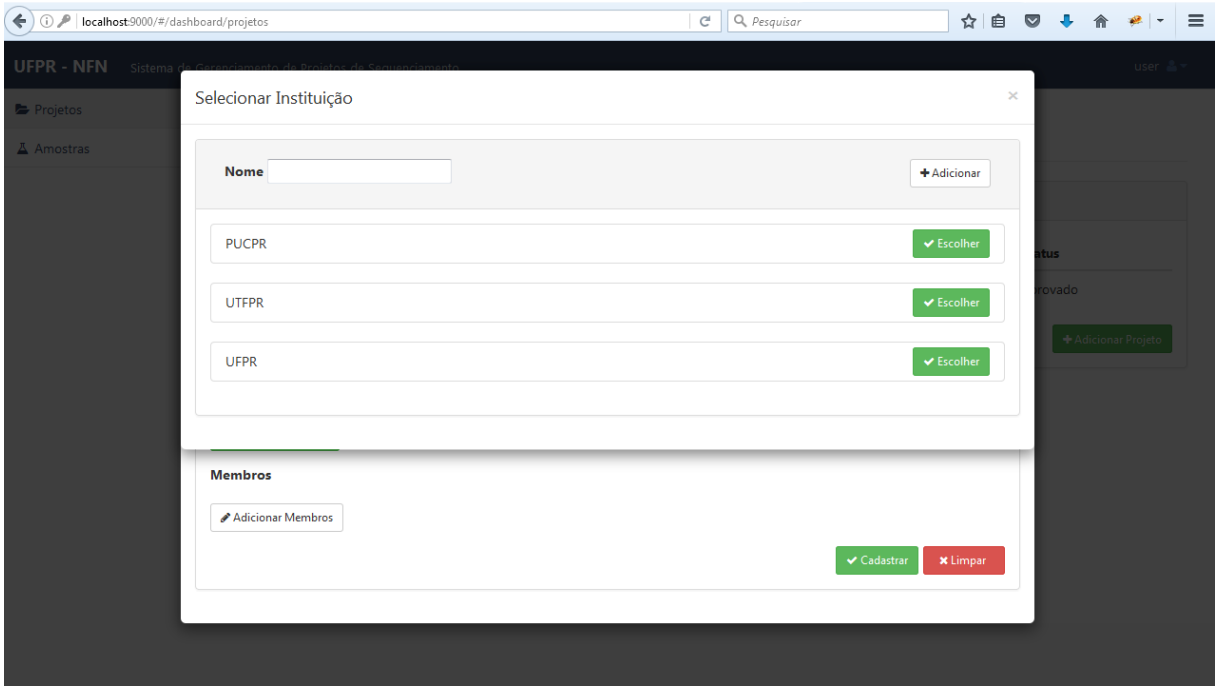


FIGURA 21 – MODAL SELECIONAR INSTITUIÇÃO
FONTE: Os autores (2016)

Os administradores do sistema podem editar, aprovar e rejeitar os projetos cadastrados (FIGURA 22), já os usuários podem apenas editar os dados do projeto enquanto o mesmo não estiver aprovado (FIGURA 23). Apenas os dados referentes aos membros do projeto podem ser editados independentemente do status do projeto (FIGURA 24).

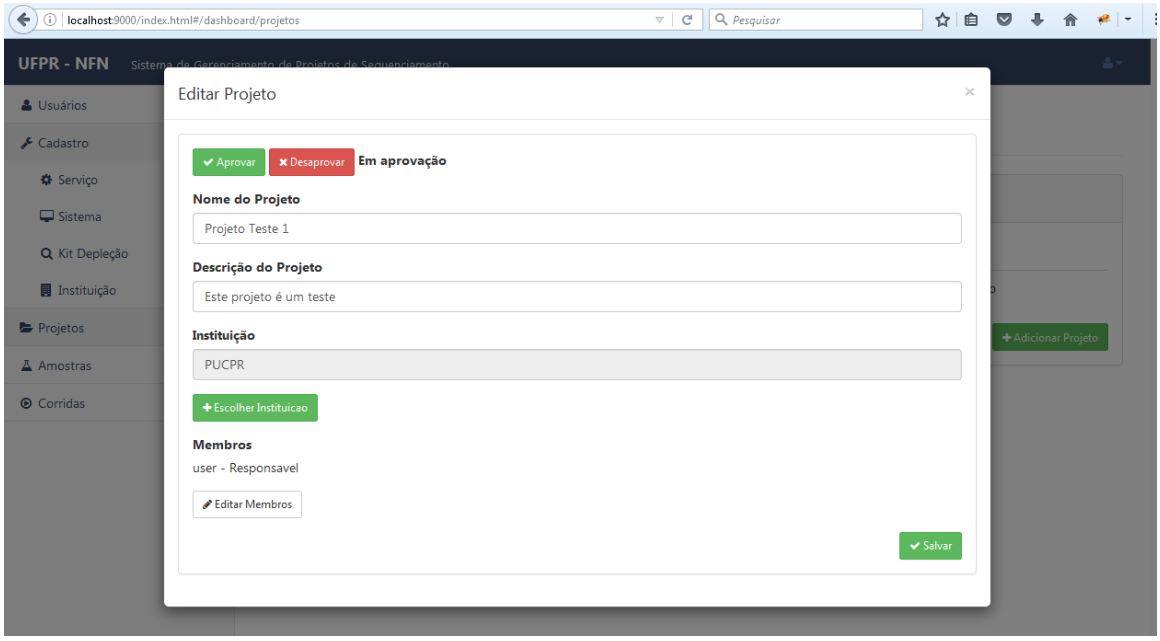


FIGURA 22 – MODAL EDITAR PROJETOS - ADMINISTRADOR
FONTE: Os autores (2016)

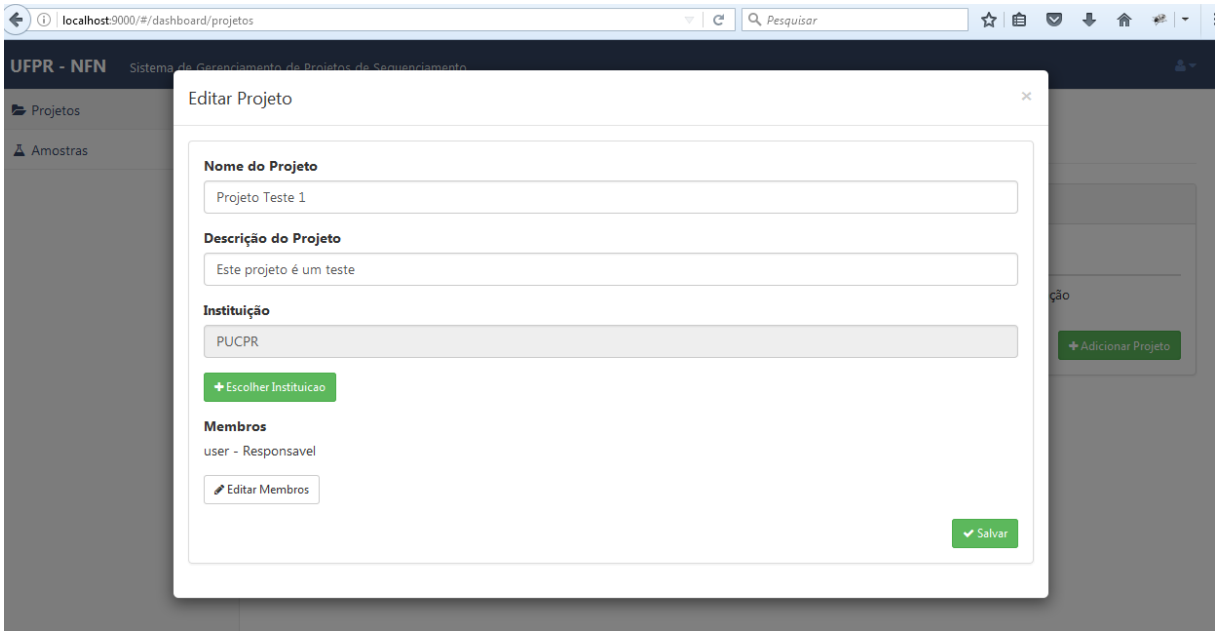


FIGURA 23 – MODAL EDITAR PROJETOS - USUÁRIO
FONTE: Os autores (2016)

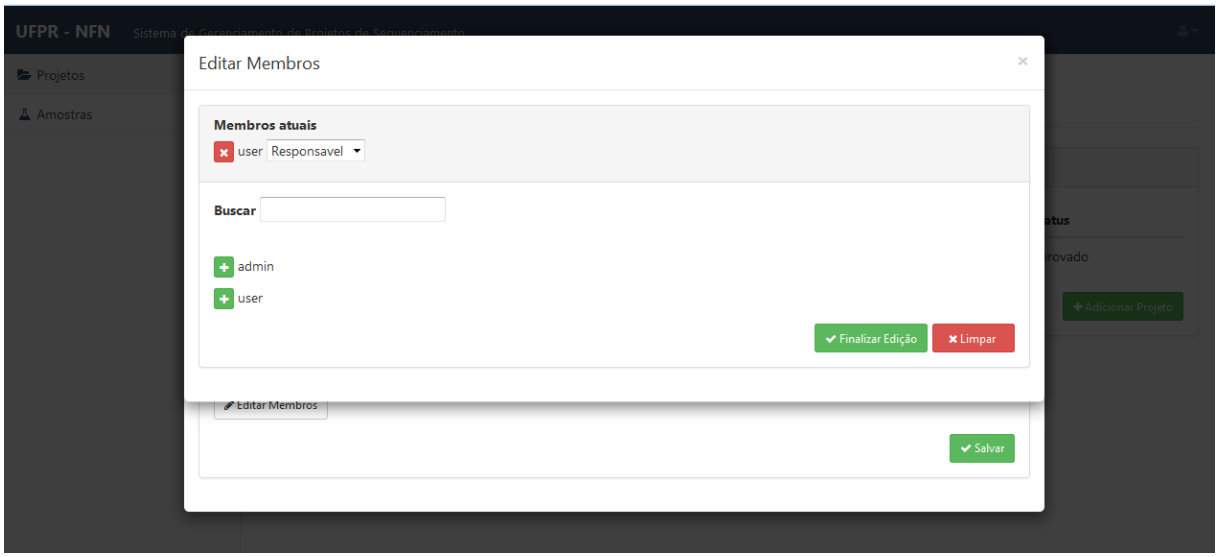


FIGURA 24 – MODAL EDITAR MEMBROS
FONTE: Os autores (2016)

5.5 AMOSTRAS

A tela inicial do módulo amostras (FIGURA 25) apresenta uma lista de projetos que difere de acordo com o nível de acesso do usuário logado da mesma maneira que a tela projeto (FIGURA 19). A tela editar amostras mostra uma lista com todas as amostras cadastradas para o projeto selecionado. Nessa tela, os administradores do sistema podem receber ou rejeitar uma amostra (FIGURA 26) e os usuários podem excluir uma amostra (somente se ela estiver com o status enviada, ou seja, ainda não

foi recebida e nem rejeitada) (FIGURA 27). A partir da mesma tela, também podem ser adicionadas novas amostras ao projeto, acessando a modal Adicionar Amostra (FIGURA 28).

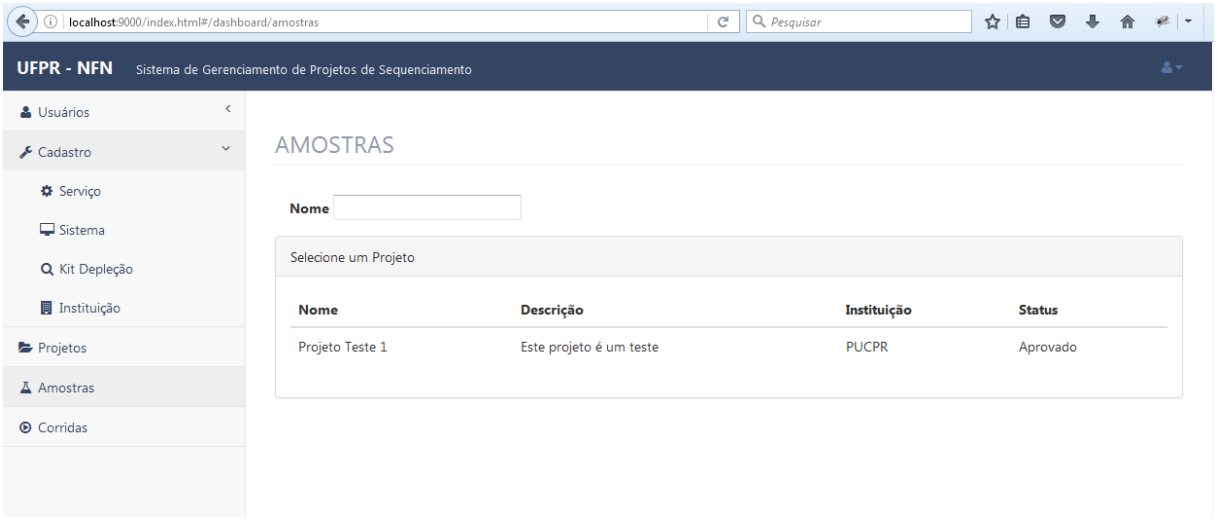


FIGURA 25 – TELA AMOSTRAS
FONTE: Os autores (2016)

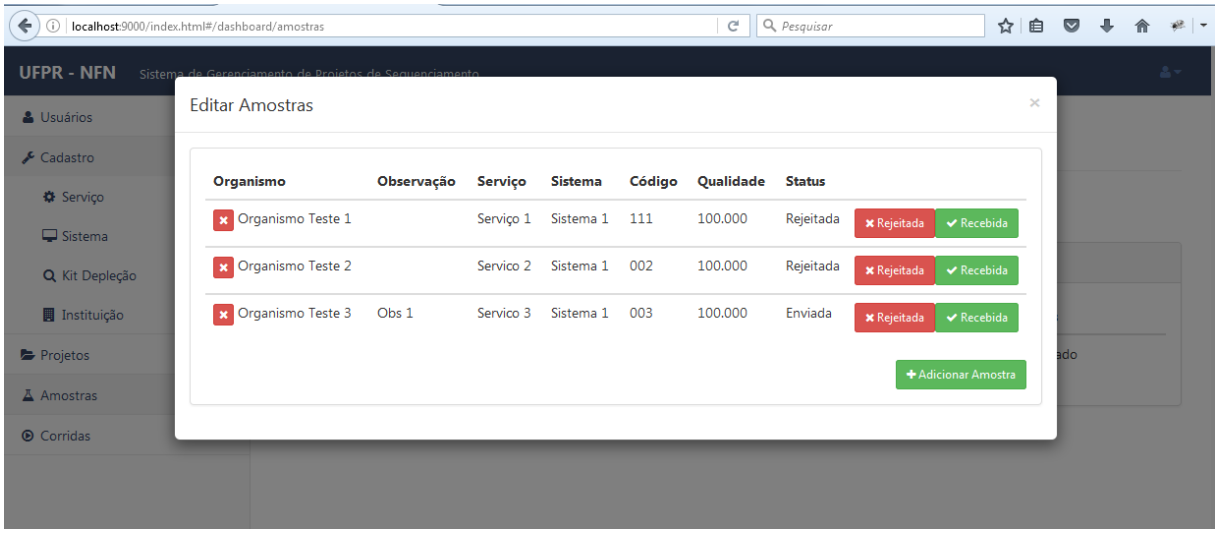


FIGURA 26 – MODAL EDITAR AMOSTRAS - ADMINISTRADOR
FONTE: Os autores (2016)

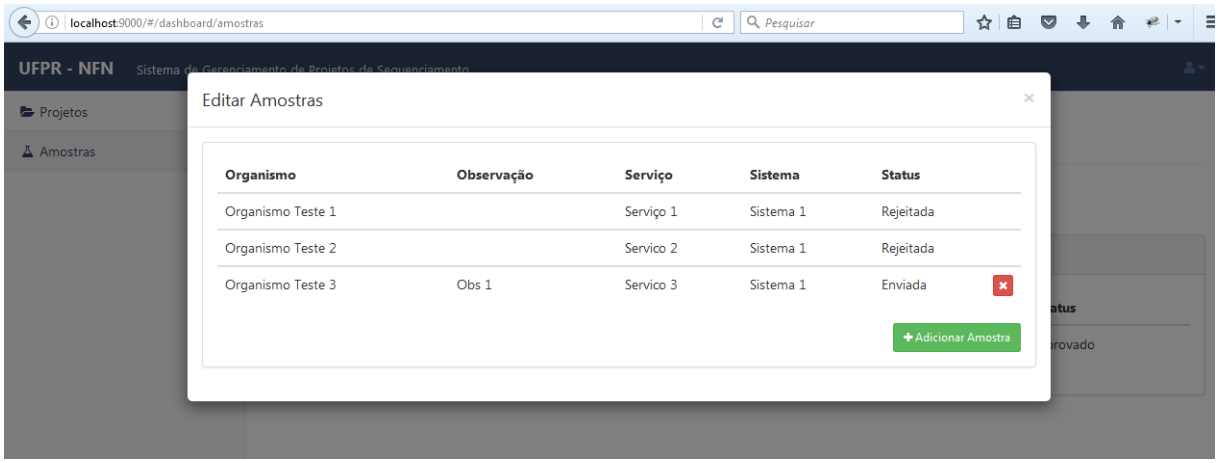


FIGURA 27 – MODAL EDITAR AMOSTRAS - USUÁRIO
FONTE: Os autores (2016)

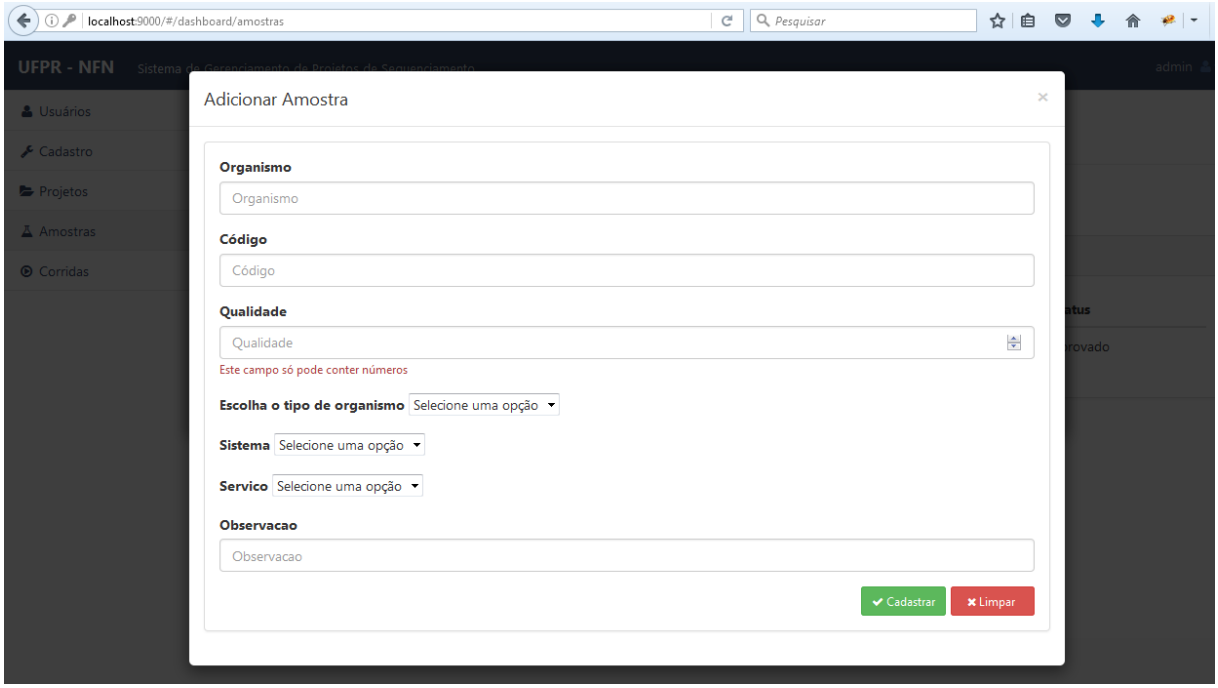


FIGURA 28 – MODAL ADICIONAR AMOSTRA
FONTE: Os autores (2016)

5.6 CORRIDA

O módulo corrida é exclusivo para os administradores do sistema. A tela Corrida apresenta todos as corridas cadastradas no sistema (FIGURA 29). Ao cadastrar uma corrida (FIGURA 30), também devem ser cadastrados as amostras que foram examinadas. A tela de cadastro de amostras na corrida (FIGURA 31) permite o cadastro de dados referentes à a amostra na corrida. Para selecionar a amostra, é

utilizada a modal Selecionar Amostra (FIGURA 32), que apresenta uma lista com todas as amostras cadastradas no sistema, sendo possível filtrar por organismo.

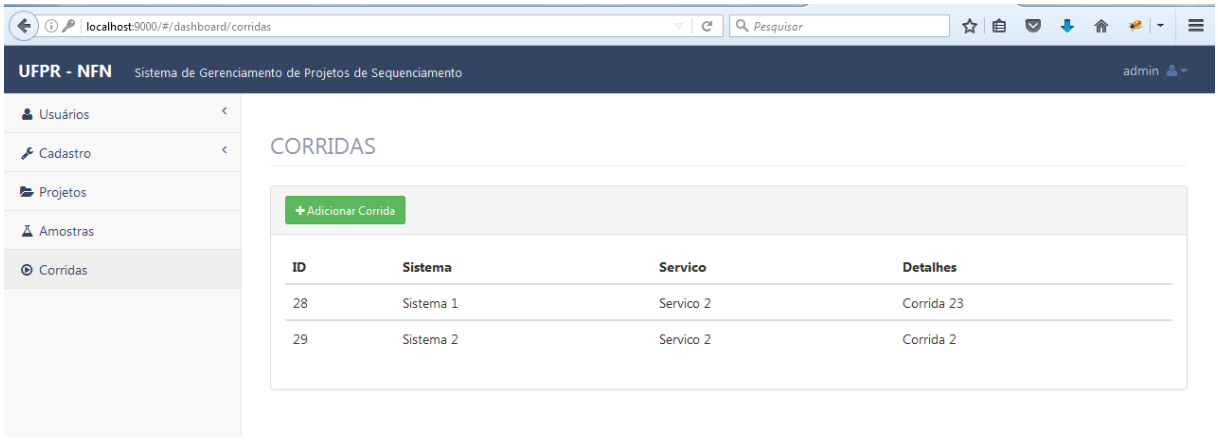


FIGURA 29 – TELA CORRIDAS
FONTE: Os autores (2016)

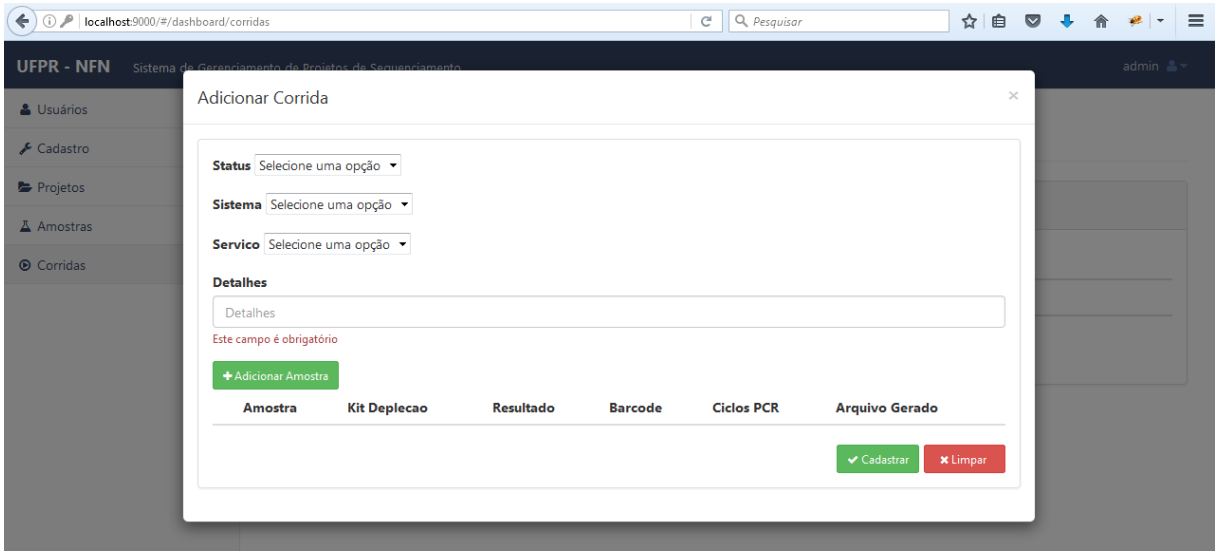


FIGURA 30 – MODAL ADICIONAR CORRIDA
FONTE: Os autores (2016)

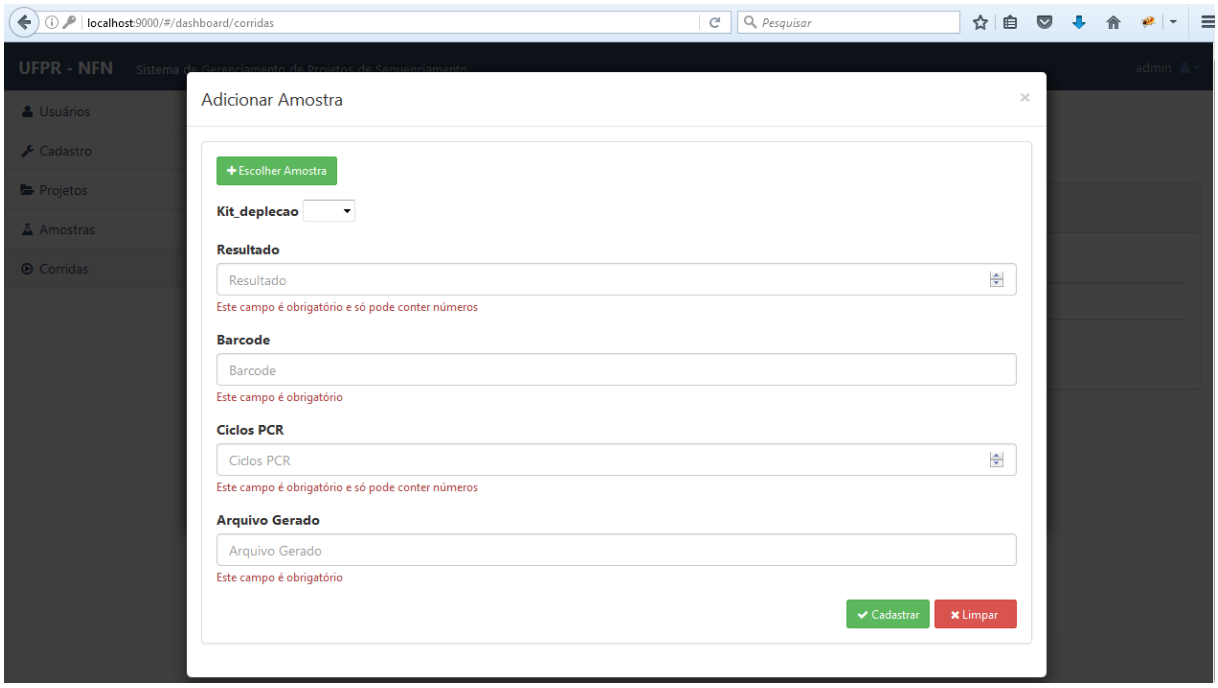


FIGURA 31 – MODAL ADICIONAR AMOSTRACORRIDA
FONTE: Os autores (2016)

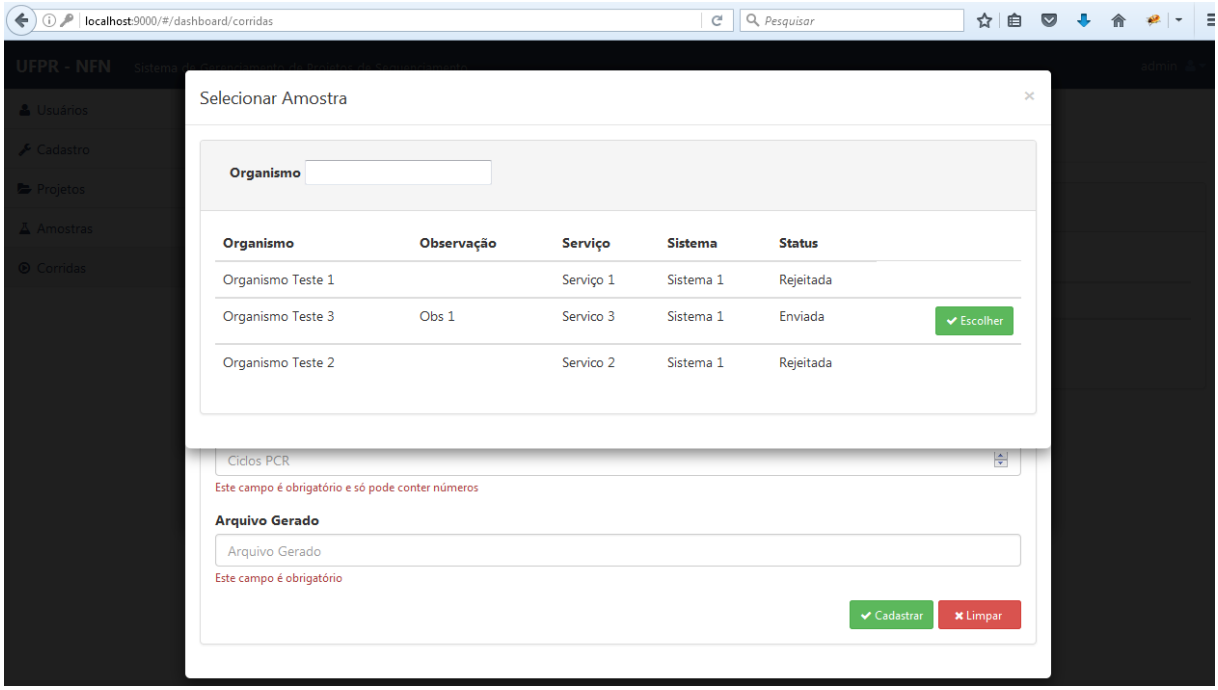


FIGURA 32 – MODAL SELECIONAR AMOSTRA
FONTE: Os autores (2016)

5.7 USUARIO

A edição de um usuário pode ser feita de duas maneiras. O próprio usuário pode editar o seu cadastro ou o administrador pode edita-lo. Caso o administrador

necessite editar o cadastro de um usuário, deve seleciona-lo na tela usuários (FIGURA 33) e uma modal será apresentada para edição dos dados (FIGURA 34).

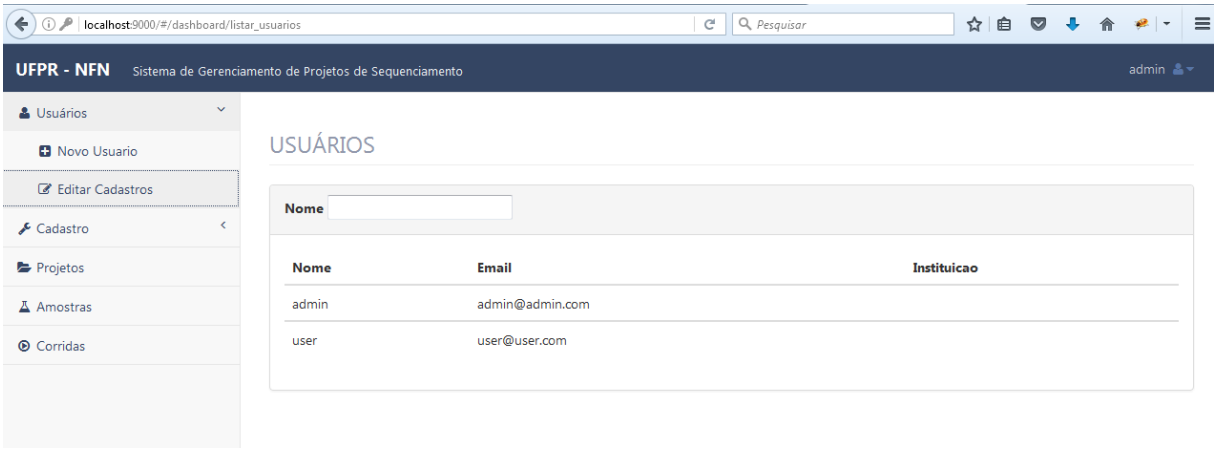


FIGURA 33 – TELA USUÁRIOS
FONTE: Os autores (2016)

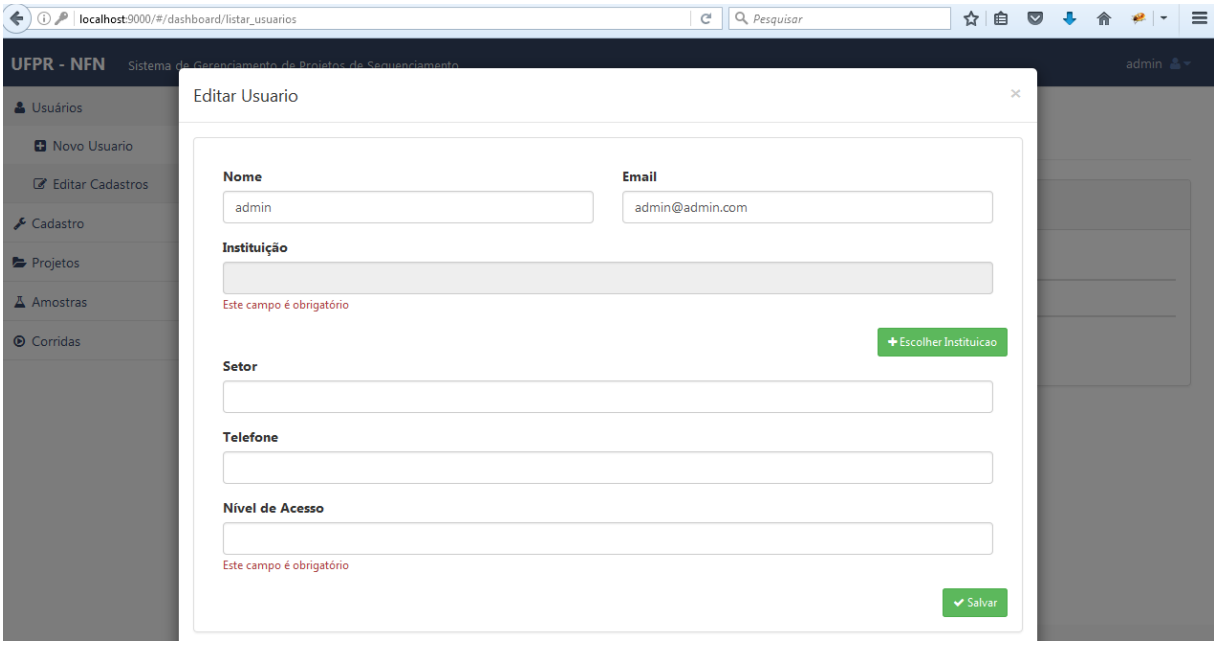


FIGURA 34 – MODAL EDITAR USUÁRIO
FONTE: Os autores (2016)

6 CONSIDERAÇÕES FINAIS

Neste trabalho apresentamos uma solução para um problema real vivenciado pelos profissionais do Núcleo de Fixação Biológica de Nitrogênio da UFPR. Através da compreensão das suas dificuldades e análise do caso de gerenciamento de projetos de corridas de sequenciamento genético, através da elaboração de diagramas, modelos de dados e protótipos, fomos capazes de propor e implementar uma solução viável em plataforma web, gerenciando um servidor *back-end* utilizando API REST e um servidor *front-end* para uma aplicação AngularJS, pronto para ser colocado em prática.

O sistema final disponibilizado permite o gerenciamento dos dados relativos a todo o processo de gerenciamento de projetos que competem às corridas de sequenciamento genético, tendo entre suas capacidades: Gerenciamento de usuários e dos projetos em que trabalham; gerenciamento de amostras que serão enviadas para análise na UFPR, desde de seu cadastro por usuários externos até a atualização do último estágio de observação desta amostra dentro do laboratório; Documentação dos dados relativos às corridas de sequenciamento genético, seus resultados e sua relação com amostras externas e pesquisadores afora.

Através deste estudo buscamos atender uma necessidade de nossa própria instituição, mostrando que é possível colocarmos em prática os ensinamentos nela absorvidos e trabalhar em prol da nossa própria comunidade, através da oportunidade oferecida por coordenadores e professores. Esperamos que este trabalho possa ser utilizado como base para outras melhorias a serem realizadas no sistema ou mesmo em outras aplicações por vir.

6.1 TRABALHOS FUTUROS

A partir dos conhecimentos teóricos aqui desenvolvidos e sua aplicação prática demonstrada através do sistema, acreditamos que algumas melhorias são possíveis através do aproveitamento da plataforma *front-end* já construída e também do consumo da API REST em funcionamento:

a) Módulo de Análise Estatística: Disponibilizar gráficos comparativos, tabelas de agrupamento de dados e outros recursos semelhantes para

facilitar a compreensão e permitir a análise dos dados armazenados no banco de dados da aplicação;

b) Extensão dos modelos atuais para tornar possível o armazenamento de resultados mais detalhados sobre as corridas de sequenciamento, incluindo dados técnicos acerca do funcionamento e manutenção dos equipamentos de análise, fornecidos por seus próprios sistemas, e utilizar esses dados para obtenção de estatísticas mais complexas;
c) Extensão das capacidades de hierarquia de projetos e papel/projetos, permitindo que novas funcionalidades sejam delegadas para diferentes tipos de usuários e expandindo o nível de controle e independência dos módulos;
d) Utilização da API para integração com outros sistemas do setor, como por exemplo a capacidade de armazenar e recuperar arquivos gerados pelo processo de análise.

REFERÊNCIAS

ANGULAR a. **AngularJS: Developer Guide - Introduction. “What is Angular?”,** 2010. Disponível em: <https://docs.angularjs.org/guide/introduction>. Último acesso: 19/10/2016.

ANGULAR b. **AngularJS: Developer Guide–Databinding**, 2016. Disponível em: <https://docs.angularjs.org/guide/databinding>. Último acesso: 01/12/2016.

AXURE. **Prototypes, Specifications and Diagrams in One Tool**. Disponível em: <https://www.axure.com/>. Último acesso: 01/12/2016.

BOOTSTRAP. **About**. Disponível em: <http://getbootstrap.com/about/>. Último acesso: 02/12/2016.

BOWER. **A Packet Manager For the Web**. Disponível em: <http://bower.io>. Último acesso: 01/12/2016.

FIELDING, Roy Thomas. **Architectural Styles and the Design of Network-based Software Architectures**. Doctoral dissertation, University of California, Irvine, 2000.

GEEKYANTS. **Start Angular.** Disponível em: <http://startangular.com>. Último acesso: 01/12/2016.

GOOGLE. **Trends - EmberJS, AngularJS, BackboneJS e ReactJS**. 2016. Disponível em: <https://www.google.com/trends/explore?date=2010-01-01%202016-01-01&q=ember%20js,angular%20js,backbone%20js,react%20js&hl=en-US>. Último acesso: 02/11/2016.

GRUNT. **Grunt: The JavaScript Task Runner**. Disponível em: <http://gruntjs.com/>. Último Acesso: 01/12/2026.

GUEDES, Gilleanes T. A. **UML 2: uma abordagem prática - 2ª Ed.** São Paulo: Novatec, 2011.

HOLOVATY, Adrian H and KAPLAN-MOSS, Jacob. **Mastering Django: Core.Version 1.3**, 3 November 2008.

HAVERBEKE, Marijn - **Eloquent JavaScript: A Modern Introduction to Programming** - No Starch Press, 2nd Edition, 2014.

IBM. **Rational Unified Process: Best Practices for Software Development Teams.** Disponível em: <https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf>. Último acesso: 30/11/2016.

MICROSOFT. **MSDN – The MVVM Pattern**, 2012. Disponível em: <https://msdn.microsoft.com/en-us/library/hh848246.aspx>. Último acesso: 01/12/2016.

MOZILLA. **JavaScript**, 2016. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction>. Último acesso: 05/11/2016.

NPM ENTERPRISE. **Nifty Pun Master**. Disponível em: <https://www.npmjs.com/>. Último acesso: 01/12/2016

OCLAZYLOAD. **ocLazyLoad Documentation**. Disponível em: <https://oclazyload.readme.io/>. Último acesso 01/12/2026.

POSTGRESQL a. **About**, 2016. Disponível em: <https://www.postgresql.org/about/>. Último acesso: 05/11/2016.

POSTGRESQL b. **The SQL Language** – 2016.Disponível em: <https://www.postgresql.org/docs/9.6/static/sql.html>. Último acesso: 30/11/2016.

POSTGRESQL c. **What is Postgres?** – 2003. Disponível em: <https://www.postgresql.org/docs/6.3/static/c0101.htm>. Último acesso: 30/11/2016.

PROJECT MANAGEMENT INSTITUTE – **PMI. A Guide to the Project Management Body of Knowledge (PMBoK)**. 4. ed. PMI Standard – ANSI, 2008.

PUREWAL, Semmy–**Learning Web App Development** –O'Reily Media Inc., 2014

STUTTARD, Dafydd and PINTO, Marcus - **The Web Application Hacker's Handbook - Second Edition**, 2011 - Wyley Publishing Inc.

TAKADA, Mikito. - **Single Page App Book**– 2013. Disponível em:<http://singlepageappbook.com/index.html>. Último acesso: 27/10/2016.

APÊNDICES

APÊNDICE A – DIAGRAMA DE CASOS DE USO

FIGURA 35 – DIAGRAMA DE CASOS DE USO
FONTE: Os autores (2016)

APENDICE B – ESPECIFICAÇÃO DOS CASOS DE USO

Caso de uso 001 - Cadastrar Usuário

Descrição

Este caso de uso descreve como cadastrar um usuário no sistema.

Pré-condições

Não há.

Pós-condições

O sistema deve ter salvado o cadastro do usuário

Ator Primário

Visitante

Fluxo Principal de Eventos

1. O sistema apresenta a tela de cadastro de usuários;
2. O visitante preenche o formulário e clica no botão cadastrar;
3. O sistema valida as informações recebidas;
4. O sistema salva as informações do cadastro de usuário;
5. O sistema apresenta mensagem de cadastro realizado;
6. O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 002 - Cadastrar Amostra

Descrição

Este caso de uso descreve como cadastrar uma amostra no sistema.

Pré-condições

O usuário deve estar logado no sistema

Pós-condições

O sistema deve ter salvado o cadastro da amostra

Ator Primário

Usuário

Fluxo Principal de Eventos

1) O usuário clica no botão Amostras do menu;

2) O sistema apresenta tela com lista de projetos que o usuário é membro ou todos os projetos se o usuário for administrador;
3) O usuário seleciona o projeto que deseja adicionar uma amostra;
4) O sistema apresenta tela com lista de amostras cadastradas no projeto selecionado;
5) O usuário clica no botão Adicionar Amostra;
6) O sistema apresenta o formulário de cadastro de amostras;
7) Ousuário preenche o formulário e clica no botão Cadastrar;
8) O sistema valida as informações recebidas;
9) O sistema salva as informações do cadastro de amostra
10) O sistema apresenta mensagem de cadastro realizado com sucesso
11) O caso de uso é finalizado

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 003 - Cadastrar Instituição

Descrição

Este caso de uso descreve como cadastrar uma Instituição no sistema.

Pré-condições

O usuário deve estar logado no sistema como administrador

Pós-condições

O sistema deve ter salvado o cadastro da Instituição

Ator Primário

Administrador

Fluxo Principal de Eventos

1) O administrador clica no menu Cadastro > Instituição
2) O sistema apresenta tela com a lista de instituições cadastradas e o formulário de cadastro de instituições;
3) O usuário preenche o formulário e clica no botão Cadastrar;
4) O sistema valida as informações recebidas;
5) O sistema salva a informação de cadastro de instituição;
6) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 004 - Cadastrar Projeto

Descrição

Este caso de uso descreve como cadastrar um projeto no sistema.

Pré-condições

O usuário deve estar logado no sistema

Pós-condições

O sistema deve ter salvado o cadastro do projeto

Ator Primário

Usuário

Fluxo Principal de Eventos

1) O usuário clica no menu Projetos
2) O sistema apresenta tela com a lista de projetos que o usuário é membro ou todos os projetos se for administrador;
3) O usuário clica no botão Adicionar Projeto;
4) O sistema apresenta o formulário de cadastro de projetos;
5) O usuário preenche o formulário;
6) O usuário clica no botão Escolher Instituição;
7) O sistema apresenta tela de seleção de instituição;
8) O usuário clica no botão Escolher da instituição desejada;
9) O usuário clica no botão Adicionar Membros da tela de cadastro de projetos;
10) O sistema chama o Caso de Uso 006 - Cadastrar UsuárioProjeto;
11) O usuário clica no botão Cadastrar;
12) O sistema valida os dados recebidos;
13) O sistema salva as informações do cadastro do projeto e dos membros do projeto;
14) O sistema apresenta mensagem de cadastro realizado com sucesso;
15) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 005 - Manter Projeto

Descrição

Este caso de uso descreve como editar um projeto no sistema.

Pré-condições

O usuário deve estar logado no sistema

O projeto não pode estar aprovado

Pós-condições

O sistema deve ter salvado a edição do projeto

Ator Primário

Usuário/ Administrador

Fluxo Principal de Eventos

1) O usuário clica no menu Projetos;
2) O sistema apresenta tela com a lista de projetos que o usuário é membro ou todos os projetos se for administrador;
3) O usuário clica no projeto que deseja editar;
4) O sistema apresenta tela com os dados do projeto escolhido;
5) O usuário edita o dados do projeto;
6) O usuário clica no botão Escolher Instituição;
7) O sistema apresenta a tela de seleção de instituição;
8) O usuário clica no botão Escolher da instituição desejada;
9) O usuário clica no botão Editar Membros da tela de edição de projetos;
10) O sistema chama o Caso de Uso 007 –Manter UsuárioProjeto;
11) O usuário clica no botão Salvar;
12) O sistema valida os dados recebidos
13) O sistema salva as informações da edição do projeto;
14) O sistema apresenta mensagem de projeto editado com sucesso;
15) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 006 - Cadastrar UsuárioProjeto

Descrição

Este caso de uso descreve como cadastrar um UsuárioProjeto no sistema.

Pré-condições

O usuário deve estar logado no sistema

Pós-condições

O sistema deve ter salvado o cadastro do UsuárioProjeto

Ator Primário

Usuário/ Administrador

Fluxo Principal de Eventos

1) O sistema apresenta tela com a lista de usuários cadastrados no sistema;
2) O usuário clica no botão “+” para adicionar um usuário como membro do projeto;
3) O usuário seleciona o papel do membro no projeto;
4) O usuário clica no botão finalizar adição para finalizar a adição de membros;
5) O sistema retorna ao Caso de Uso 004 - Adicionar Projetos;
6) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 007 - Manter UsuárioProjeto

Descrição

Este caso de uso descreve como editar um UsuárioProjeto no sistema.

Pré-condições

O usuário deve estar logado no sistema

Pós-condições

O sistema deve ter salvado a edição do UsuárioProjeto

Ator Primário

Usuário/ Administrador

Fluxo Principal de Eventos

1) O sistema apresenta a tela com a lista de usuário membros do projeto e lista com os usuários cadastrados no sistema;

2) O usuário clica no botão “+” para adicionar um usuário como membro do projeto;
3) O usuário seleciona o papel do membro no projeto;
4) O usuário clica no botão “x” para excluir um usuário membro do projeto;
5) O usuário clica no botão finalizar edição para finalizar a edição de membros;
6) O sistema salva as informações da edição de membros do projeto;
7) O sistema retorna ao Caso de Uso 005 - Editar Projeto;
8) O caso de uso é finalizado;

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 008 - Manter Usuário

Descrição

Este caso de uso descreve como editar um Usuário no sistema.

Pré-condições

O usuário deve estar logado no sistema.

Pós-condições

O sistema deve ter salvado a edição do Usuário

Ator Primário

Usuário/ Administrador

Fluxo Principal de Eventos

1) O Sistema apresenta a tela para editar o usuário;
2) O usuário edita o formulário;
3) O usuário clica no botão salvar;
4) O sistema salva as informações da edição de usuário;
5) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 009 - Cadastrar AmostraCorrida

Descrição

Este caso de uso descreve como cadastrar uma AmostraCorrida no sistema.

Pré-condições

O usuário deve estar logado no sistema com um perfil administrador

Pós-condições

O sistema deve ter salvado o cadastro da AmostraCorrida

Ator Primário

Administrador

Fluxo Principal de Eventos

1) O sistema apresenta tela com formulário para adição de uma amostra à corrida;
2) O usuário preenche o formulário;
3) O usuário clica no botão Escolher Amostra para selecionar uma amostra;
4) O sistema apresenta tela com as amostras cadastradas;
5) O usuário clica no botão Escolher da amostra desejada (Apenas amostras não rejeitadas podem ser escolhidas);
6) O usuário clica no botão Cadastrar da tela de adição de amostras;
7) O sistema retorna ao Caso de Uso 011 - Adicionar Corrida;
8) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 010 - ManterAmostraCorrida

Descrição

Este caso de uso descreve como excluir uma AmostraCorrida no sistema.

Pré-condições

O usuário deve estar logado no sistema com um perfil administrador

Pós-condições

O sistema deve ter salvado a exclusão da AmostraCorrida

Ator Primário

Administrador

Fluxo Principal de Eventos

1) O administrador clica no menu Corridas;
2) O sistema apresenta tela com a lista de corridas cadastradas;
3) O administrador clica na corrida que deseja editar;
4) O sistema apresenta a tela de edição de corridas com a lista de amostras cadastradas;
5) O usuário clica no botão com “x” para deletar uma AmostraCorrida;
6) O sistema salva as informações da exclusão da AmostraCorrida;
7) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 011 - Cadastrar Corrida

Descrição

Este caso de uso descreve como cadastrar uma Corrida no sistema.

Pré-condições

O usuário deve estar logado no sistema com um perfil administrador

Pós-condições

O sistema deve ter salvado o cadastro da Corrida

Ator Primário

Administrador

Fluxo Principal de Eventos

1) O administrador clica no menu Corridas;
2) O sistema apresenta tela com a lista de corridas cadastradas;
3) O administrador clica no botão Adicionar Corrida;
4) O sistema apresenta tela com o formulário de adição de corrida;
5) O usuário preenche o formulário;
6) O usuário clica no botão Adicionar Amostra;
7) O sistema chama o Caso de Uso 008 - Cadastrar AmostraCorrida;
8) O usuário clica no botão Cadastrar;
9) O sistema valida os dados recebidos;

10) O sistema salva as informações do cadastro da corrida e das amostras da corrida;
11) O sistema apresenta mensagem de cadastro realizado com sucesso;
12) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 012 - ManterCorrida

Descrição

Este caso de uso descreve como editar uma Corrida no sistema.

Pré-condições

O usuário deve estar logado no sistema com um perfil administrador

Pós-condições

O sistema deve ter salvado a edição da Corrida

Ator Primário

Administrador

Fluxo Principal de Eventos

1) O administrador clica no menu Corridas;
2) O sistema apresenta tela com a lista de corridas cadastradas;
3) O administrador clica na corrida que deseja editar;
4) O usuário edita o formulário;
5) O usuário clica no botão Salvar;
6) O sistema salva as informações de edição da corrida;
7) O sistema apresenta mensagem de edição realizada com sucesso;
8) O caso de uso é finalizado;

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 013 - Cadastrar Serviço

Descrição

Este caso de uso descreve como cadastrar um Serviço no sistema.

Pré-condições

O usuário deve estar logado no sistema com um perfil administrador

Pós-condições

O sistema deve ter salvado o cadastro do Serviço

Ator Primário

Administrador

Fluxo Principal de Eventos

1) O administrador clica no menu Cadastro > Serviço;
2) O sistema apresenta tela com a lista de serviços cadastrados e formulário de cadastro de serviços;
3) O administrador preenche o formulário e clica no botão Cadastrar;
4) O sistema salva as informações do cadastro de Serviço;
5) O sistema apresenta mensagem de cadastro realizado com sucesso;
6) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 014 - Manter Serviço

Descrição

Este caso de uso descreve como excluir um Serviço no sistema.

Pré-condições

O usuário deve estar logado no sistema com um perfil administrador

Pós-condições

O sistema deve ter salvado a exclusão do Serviço

Ator Primário

Administrador

Fluxo Principal de Eventos

1) O administrador clica no menu Cadastro > Serviço

2) O sistema apresenta tela com lista de serviços cadastrados e formulário de cadastro de serviços;
3) O administrador clica no botão Excluir do serviço que deseja excluir;
4) O sistema salva a exclusão do serviço;
5) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 015 - Cadastrar Sistema

Descrição

Este caso de uso descreve como cadastrar um Sistema no sistema.

Pré-condições

O usuário deve estar logado no sistema com um perfil administrador

Pós-condições

O sistema deve ter salvado o cadastro do sistema

Ator Primário

Administrador

Fluxo Principal de Eventos

1) O administrador clica no menu Cadastro > Sistema
2) O sistema apresenta tela com a lista de sistemas cadastrados e formulário para cadastro de sistemas;
3) O administrador preenche o formulário e clica no botão Cadastrar;
4) O sistema salva as informações do cadastro de sistema;
5) O sistema apresenta mensagem de cadastro realizado com sucesso;
6) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 016 - Manter Sistema

Descrição

Este caso de uso descreve como excluir um Sistema no sistema.

Pré-condições

O usuário deve estar logado no sistema com um perfil administrador

Pós-condições

O sistema deve ter salvado a exclusão do Sistema

Ator Primário

Administrador

Fluxo Principal de Eventos

1) O administrador clica no menu Cadastro > Sistemas
2) O sistema apresenta tela com a lista de sistemas cadastrados e formulário para cadastro de sistemas;
3) O administrador clica no botão Excluir do sistema que deseja excluir;
4) O sistema salva a exclusão do sistema;
5) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 017 - Cadastrar Kit Depleção

Descrição

Este caso de uso descreve como cadastrar um Kit Depleção no sistema.

Pré-condições

O usuário deve estar logado no sistema com um perfil administrador

Pós-condições

O sistema deve ter salvado o cadastro do Kit Depleção

Ator Primário

Administrador

Fluxo Principal de Eventos

1) O administrador clica no menu Cadastro > Kit Depleção;
2) O sistema apresenta tela com a lista de kits depleção cadastrados e formulário de cadastro de kits depleção;

3) O administrador preenche o formulário e clica no botão Cadastrar;
4) O sistema salva as informações do cadastro de kit depleção;
5) O sistema apresenta mensagem de cadastro realizado com sucesso;
6) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 018 - Manter Kit Depleção

Descrição

Este caso de uso descreve como excluir um Kit Depleção no sistema.

Pré-condições

O usuário deve estar logado no sistema com um perfil administrador

Pós-condições

O sistema deve ter salvado a exclusão do Kit Depleção

Ator Primário

Administrador

Fluxo Principal de Eventos

1) O administrador clica no menu Cadastro > Kit Depleção;
2) O sistema apresenta tela com a lista de kits depleção cadastrados e formulário de cadastro de kits depleção;
3) O administrador clica no botão Excluir do kit depleção que deseja excluir;
4) O sistema salva a exclusão do kit depleção;
5) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 019 - Receber Amostra

Descrição

Este caso de uso descreve como receber uma amostra no sistema.

Pré-condições

O usuário deve estar logado no sistema como administrador

Pós-condições

O sistema deve ter salvado o recebimento da amostra

Ator Primário

Administrador

Fluxo Principal de Eventos

1) O administrador clica no menu Amostras
2) O sistema apresenta tela com a lista de projetos cadastrados;
3) O administrador clica no projeto que deseja realizar o recebimento da amostra;
4) O sistema apresenta tela com as amostras cadastradas para o projeto selecionado;
5) O administrador clica no botão Recebida da amostra que deseja receber;
6) O sistema salva a informação de recebimento da amostra;
7) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 020 - Rejeitar Amostra

Descrição

Este caso de uso descreve como rejeitar uma amostra no sistema.

Pré-condições

O usuário deve estar logado no sistema como administrador

Pós-condições

O sistema deve ter salvado a rejeição da amostra

Ator Primário

Administrador

Fluxo Principal de Eventos

1) O administrador clica no menu Amostras;
2) O sistema apresenta tela com a lista de projetos cadastrados;
3) O administrador clica no projeto que deseja rejeitar uma amostra;
4) O sistema apresenta tela com a lista de amostras cadastradas do projeto selecionado;

5) O administrador clica no botão Rejeitada da amostra que deseja rejeitar;
6) O sistema salva a informação de rejeição da amostra;
7) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 021 - Aprovar Projeto

Descrição

Este caso de uso descreve como aprovar um projeto no sistema.

Pré-condições

O usuário deve estar logado no sistema como administrador

Pós-condições

O sistema deve ter salvado a aprovação do projeto

Ator Primário

Administrador

Fluxo Principal de Eventos

1) O administrador clica no menu Projetos
2) O sistema apresenta tela com a lista de projetos cadastrados;
3) O administrador clica no projeto que deseja aprovar
4) O administrador clica no botão Aprovar;
5) O sistema salva a informação de aprovação do Projeto;
6) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 022 - Desaprovar Projeto

Descrição

Este caso de uso descreve como desaprovar um projeto no sistema.

Pré-condições

O usuário deve estar logado no sistema como administrador

Pós-condições

O sistema deve ter salvado a desaprovação do projeto

Ator Primário

Administrador

Fluxo Principal de Eventos

1) O administrador clica no menu Projetos;
2) O sistema apresenta tela com a lista de projetos cadastrados;
3) O administrador clica no projeto que deseja desaprovar;
4) O administrador clica no botão Desaprovar;
5) O sistema salva a informação de desaprovação do Projeto;
6) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 023–Manter Instituição

Descrição

Este caso de uso descreve como realizar edição de uma instituição no sistema.

Pré-condições

O usuário deve estar logado no sistema como administrador

Pós-condições

O sistema deve ter salvado a edição da instituição

Ator Primário

Administrador

Fluxo Principal de Eventos

1) O administrador clica no menu Cadastrar > Instituição
2) O sistema apresenta a tela com a lista de instituições cadastradas e formulário de cadastro de instituições;
3) O usuário clica no botão Editar da instituição que deseja editar (A1)
4) O sistema apresenta tela com os dados da instituição escolhida;
5) O usuário modifica o formulário e clica no botão Salvar;
6) O sistema salva os dados de edição da instituição

7) O caso de uso é finalizado.

Fluxo Alternativo

A1- Exclusão de instituição

1) O usuário clica no botão Excluir da instituição que deseja excluir;
2) O sistema salva a exclusão da instituição;
3) O caso de uso é finalizado.

Fluxo de Exceção

Não há.

Regras de Negócio

Não há.

Caso de uso 024 - Login

Descrição

Este caso de uso descreve como realizar *login* no sistema.

Pré-condições

Não há

Pós-condições

O sistema deve ter logado o usuário no sistema

Ator Primário

Visitante

Fluxo Principal de Eventos

1) O sistema apresenta a tela de *login* no sistema;
2) O usuário preenche o formulário de *login*;
3) O usuário clica no botão logar;
4) O sistema valida as informações de *login*;
5) O sistema loga o usuário;
6) O caso de uso é finalizado.

Fluxo de Exceção

E1- Combinação usuário/senha incorretos ou em branco

7) O sistema apresenta mensagem de alerta de informações de *login* invalidas;
8) O caso de uso é finalizado.

Regras de Negócio

Não há.

APÊNDICE C – DIAGRAMA RELACIONAL

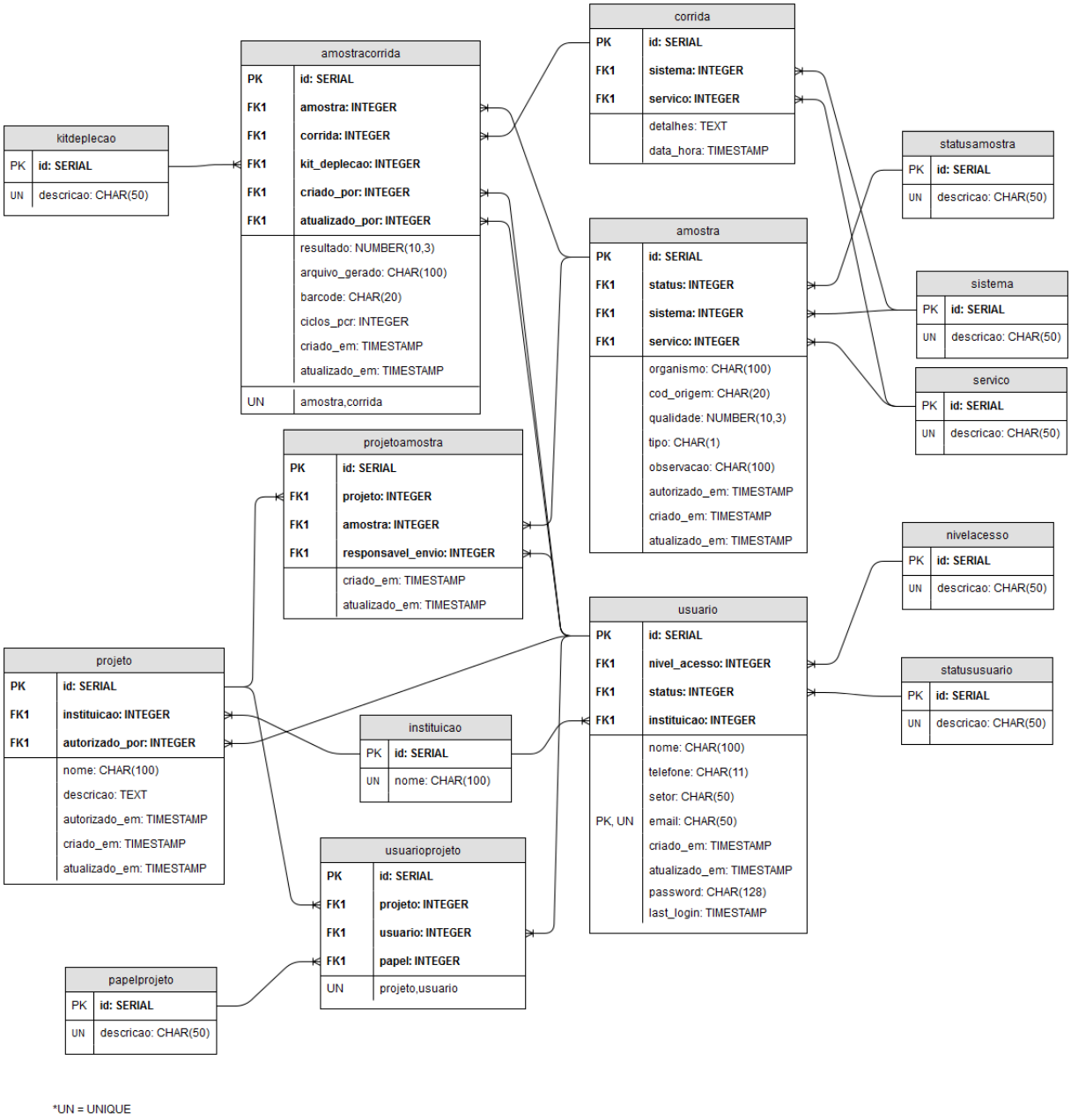


FIGURA 36 – DIAGRAMA ENTIDADE RELACIONAMENTO
FONTE: Os autores (2016)

APÊNDICE D – DICIONÁRIO DE DADOS

genseq_amostra					
Tabela que guarda os dados das amostras cadastradas no sistema. Amostras são materiais biológicos que são enviados do parceiro para análise.					
Campo	**Tipo**	**FK/PK**	**Referência**	**Observação**	**Descrição**
id	serial	PK		not null	ID da amostra
organismo	char(100)			not null	nome do organismo
cod_origem	char(20)				código da amostra cadastrado pelo usuário
qualidade	number(10,3)				qualidade da amostra
tipo	char(1)			not null	tipo da amostra (O = Organismo, G= Gel)
observacao	char(100)				observação
autorizado_em	timestamp				data de autorização
criado_em	timestamp			not null	data de criação
atualizado_em	timestamp			not null	data da atualização
servico_id	integer	FK	genseq_servico	not null	ID do serviço a ser realizado na amostra
sistema_id	integer	FK	genseq_sistema	not null	ID do sistema utilizado para o serviço na amostra
status_id	integer	FK	genseq_statusamostra	not null	ID do status da amostra

genseq_amostracorrida					
Tabela que guarda os dados referentes a uma amostra numa corrida específica. Uma amostra pode ser avaliada em várias corridas assim como uma corrida pode ter várias amostras.					
Campo	**Tipo**	**FK/PK**	**Referência**	**Observação**	**Descrição**
id	serial	PK		not null	ID da amostracorrida
resultado	number(10,3)			not null	resultado da corrida da amostra
arquivo_gerado	char(100)			not null	caminho do arquivo gerado
barcode	char(20)			not null	barcode da amostra na corrida
ciclos_pcr	integer			not null	ciclos PCR da amostra na corrida
criado_em	timestamp				data de criação
atualizado_em	timestamp				data de atualização
amostra_id	integer	FK	genseq_amostra	not null	ID da amostra.
atualizado_por	integer	FK	genseq_usuario	not null	ID do usuário que realizou a atualização
corrida_id	integer	FK	genseq_corrida	not null	ID da corrida realizada
criado_por_id	integer	FK	genseq_usuario	not null	ID do usuário que realizou a criação
kit_deplecao_id	integer	FK	genseq_kitdeplecao	not null	ID do kit_deplecao utilizado na amostra

genseq_corrida					
Tabela que guarda os dados das corridas cadastradas no sistema. As corridas representam a análise que está sendo feita sobre um determinado lote de amostras, e nela ocorre o processo do sequenciamento genético.					
Campo	**Tipo**	**FK/PK**	**Referência**	**Observação**	**Descrição**
id	serial	PK		not null	ID da corrida
detalhes	text			not null	detalhes da corrida
data_hora	timestamp			not null	data da criação
servico_id	integer	FK	genseq_servico	not null	ID do serviço realizado na corrida
sistema_id	integer	FK	genseq_sistema	not null	ID do sistema no qual foi realizado a corrida

genseq_instituicao					
Tabela que guarda as instituições cadastradas no sistema, podendo ser estas, universidades, institutos ou outras entidades parceiras.					
Campo	**Tipo**	**FK/PK**	**Referência**	**Observação**	**Descrição**
id	serial	PK		not null	ID da instituição
nome	char(100)			not null	nome da instituição

genseq_kitdeplecao					
Tabela que guarda os kit depleção cadastradas no sistema. Os kits são materiais utilizados pela análise e podem ser diferentes para cada amostra dentro da corrida.					
Campo	**Tipo**	**FK/PK**	**Referência**	**Observação**	**Descrição**
id	serial	PK		not null	ID do kit depleção
descricao	char(50)			not null	Descrição do kit depleção

genseq_nivelacesso					
Tabela que guarda os níveis de acesso que um usuário pode ter no sistema. São fixos, e o banco de dados se encarregará de identicar o nível de acesso do usuário enquanto a aplicação irá reger as restrições utilizando esse nível de acesso como base.					
Campo	**Tipo**	**FK/PK**	**Referência**	**Observação**	**Descrição**
id	serial	PK		not null	ID do nível acesso
descricao	char(50)			not null	Descrição do nível acesso

genseq_papelprojeto					
Tabela que guarda os papéis que um usuário pode ter num projeto					
Campo	**Tipo**	**FK/PK**	**Referência**	**Observação**	**Descrição**
id	serial	PK		not null	ID do papelprojeto
descricao	char(50)			not null	Descrição do papelprojeto

genseq_projeto					
Tabela que guarda os dados dos projetos cadastrados no sistema. Projetos podem ser cadastrados pelos próprios usuários e dependem da aprovação de um administrador.					
Campo	**Tipo**	**FK/PK**	**Referência**	**Observação**	**Descrição**
id	serial	PK		not null	ID do projeto
autorizado_em	timestamp				data de autorização do projeto
nome	char(100)			not null	nome do projeto
descricao	text			not null	descrição do projeto
criado_em	timestamp			not null	data de criação
atualizado_em	timestamp			not null	data de atualização
atualizado_por	integer	FK	genseq_usuario	not null	ID do usuário que realizou a atualização
instituicao_id	integer	FK	genseq_instituicao	not null	ID da instituição do projeto

genseq_projetoamostra					
Tabela que guarda os dados das relações entre projetos e amostras. Um projeto pode conter várias amostras.					
Campo	**Tipo**	**FK/PK**	**Referência**	**Observação**	**Descrição**
id	serial	PK		not null	ID do a amostra no projeto
criado_em	timestamp			not null	data de criação da amostra no projeto
atualizado_em	timestamp			not null	data de atualização da amostra no projeto
amostra_id	integer	FK	genseq_amostra	not null	ID da amostra
projeto_id	integer	FK	genseq_projeto	not null	ID do projeto
responsavel_envio_id	integer	FK	genseq_usuario	not null	ID do usuário que realizou o cadastro da amostra

genseq_servico					
Tabela que guarda os dados dos serviços que podem ser realizados. Os serviços são referentes aos tipos de análises que podem ser realizados pelos equipamentos como por exemplo os processos de “transcriptoma” ou “genoma”.					
Campo	**Tipo**	**FK/PK**	**Referência**	**Observação**	**Descrição**
id	serial	PK		not null	ID do serviço
descricao	char(50)			not null	Descrição do serviço

genseq_sistema					
Tabela que guarda os dados dos sistemas que podem ser utilizados. Os sistemas são os equipamentos propriamente ditos, como por exemplo as maquinas “Proton” e “Illumina”.					
Campo	**Tipo**	**FK/PK**	**Referência**	**Observação**	**Descrição**
id	serial	PK		not null	ID do sistema
descricao	char(50)			not null	Descrição do sistema

genseq_statusamostra					
Tabela que guarda as descrições dos status que as amostras podem ter no sistema					
Campo	**Tipo**	**FK/PK**	**Referência**	**Observação**	**Descrição**
Id	serial	PK		not null	ID do statusamostra
Descrição	char(50)			not null	Descrição do statusamostra

genseq_statususuario					
Tabela que guarda as descrições dos status que os usuários podem ter no sistema					
Campo	**Tipo**	**FK/PK**	**Referência**	**Observação**	**Descrição**
id	serial	PK		not null	ID do statususuario
descricao	char(50)			not null	Descrição do statususuario

genseq_usuario					
Tabela que guarda os dados dos usuários cadastrados no sistema					
Campo	**Tipo**	**FK/PK**	**Referência**	**Observação**	**Descrição**
id	serial	PK		not null	ID do usuário
password	char(128)			not null	Senha do usuário criptografada através de algoritmo PBKDF2 utilizando hash SHA256
last_login	timestamp				Data do último login do usuário
nome	char(100)			not null	nome do usuário
telefone	char(11)			not null	telefone do usuário
setor	char(50)			not null	setor do usuário
email	char(50)	PK		unique	email do usuário
criado_em	timestamp			not null	data de criação
atualizado_em	timestamp			not null	data de atualização
instituicao_id	integer	FK	genseq_instituicao		ID da instituição do usuário
nivel_acesso_id	integer	FK	genseq_nivelacesso		ID do nível de acesso do usuário
status_id	integer	FK	genseq_statususuario		ID do status do usuário

genseq_usuarioprojeto					
Tabela que guarda os dados das relações entre projetos e usuários. Um usuário pode fazer parte de vários projetos assim como um projeto pode conter vários usuários.					
Campo	**Tipo**	**FK/PK**	**Referência**	**Observação**	**Descrição**
id	serial	PK		not null	ID do usuário no projeto
papel_id	integer	FK	genseq_papelprojeto	not null	ID do papel do usuário no projeto
projeto_id	integer	FK	genseq_projeto	not null	ID do projeto
usuario_id	integer	FK	genseq_usuario	not null	ID do usuário

QUADRO 2 – DICIONÁRIO DE DADOS
FONTE: Os autores (2016)

APÊNDICE E – DIAGRAMA DE COMPONENTES

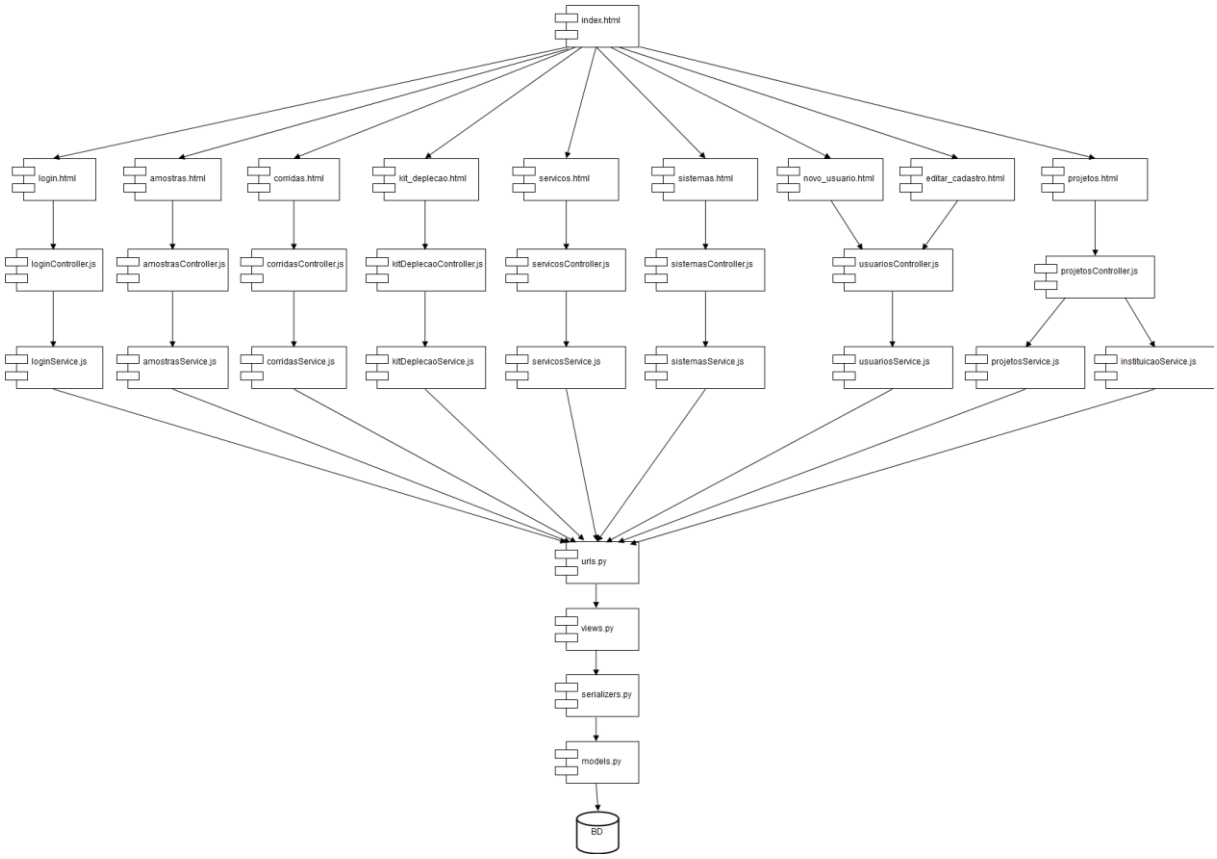


FIGURA 37 – DIAGRAMA DE COMPONENTES
FONTE: Os autores (2016)

APÊNDICE F – DIAGRAMA DE CLASSES

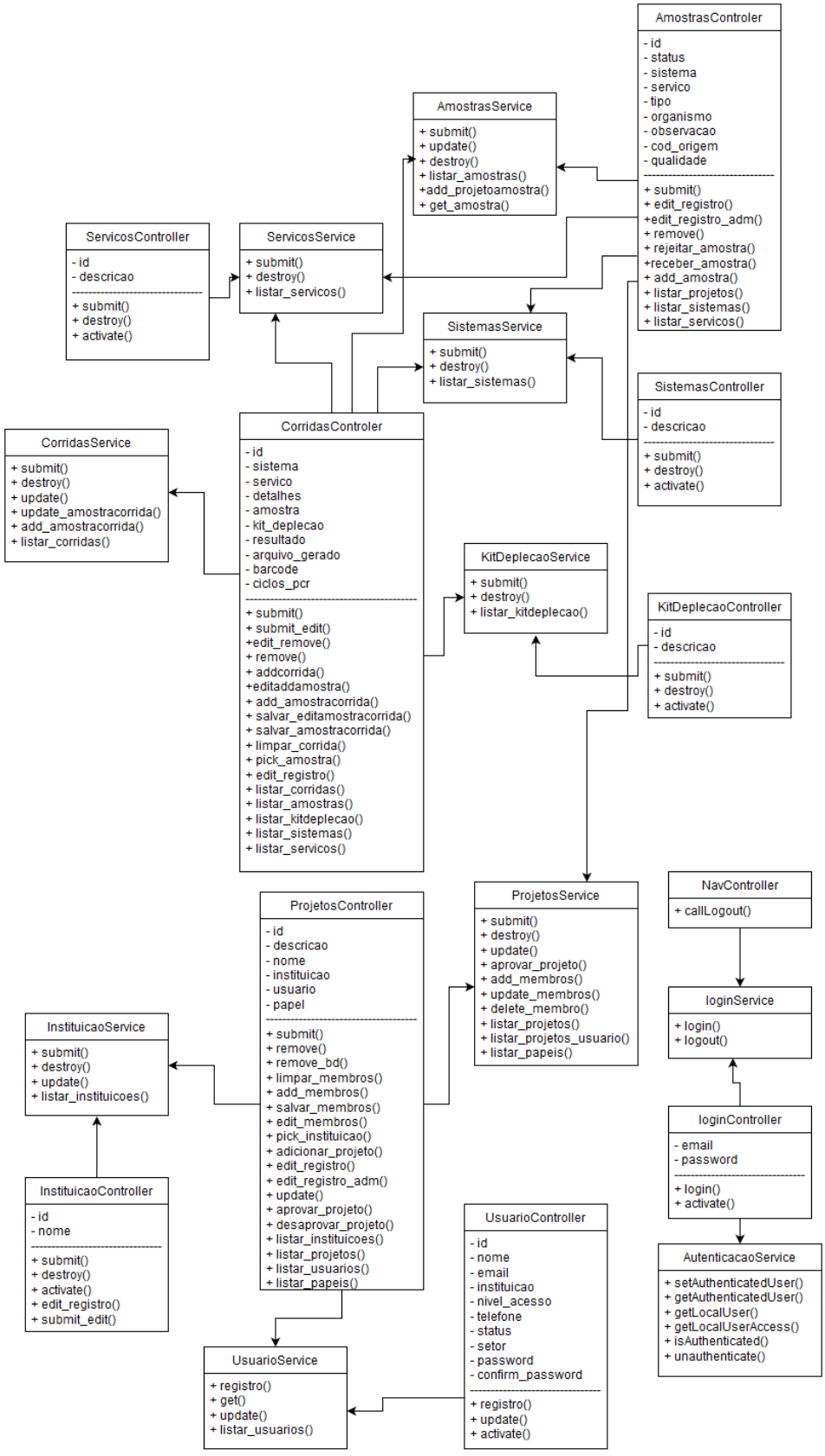
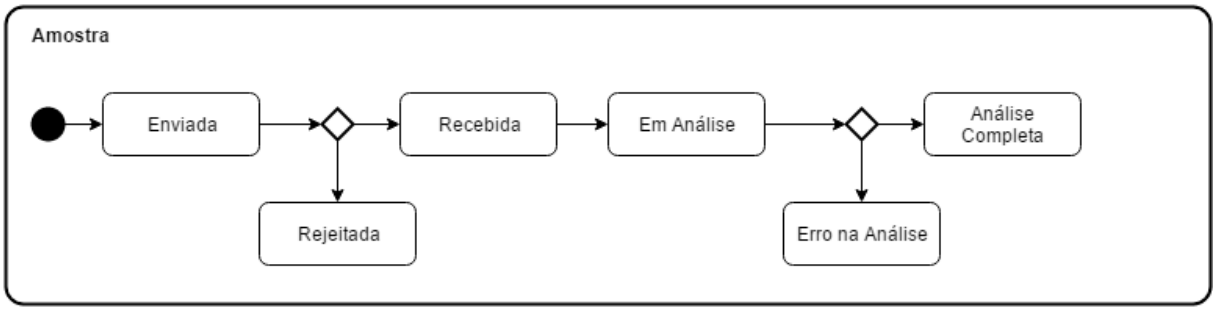


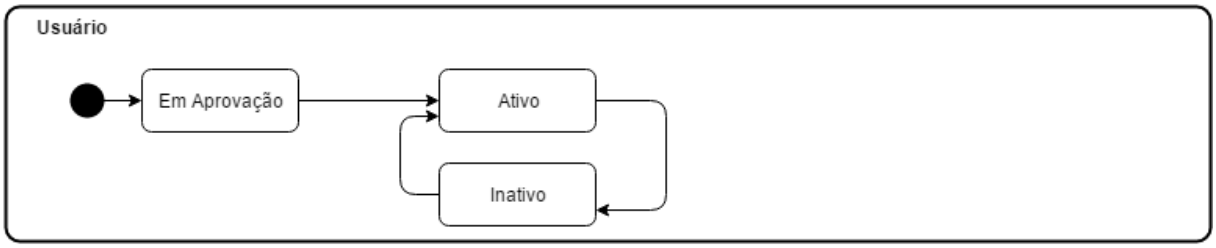
FIGURA 38 – DIAGRAMA DE CLASSES
FONTE: Os autores (2016)

APÊNDICE G – DIAGRAMA DE ESTADOS



Enviada: A amostra foi cadastrada no sistema;
Rejeitada: A amostra foi recebida no NFN mas não poderá ser analisada por não estar em condições para análise;
Recebida: A amostra foi recebida no NFN e será encaminhada para análise;
Em Análise: A amostra está sendo analisada;
Erro na Análise: Ocorreu um erro na análise da amostra;
Análise Completa: A análise da amostra foi completada.

Em aprovação: O projeto foi cadastrado mas ainda não foi aprovado por um administrador;
Aprovado: O projeto foi aprovado por um administrador;



Em aprovação: O usuário foi cadastrado mas ainda não foi ativado por um administrador;
Ativo: O usuário foi ativado por um administrador;
Inativo: O usuário foi inativado por um administrador;

FIGURA 39 – DIAGRAMA DE ESTADOS
FONTE: Os autores (2016)

APÊNDICE H – DIAGRAMA DE ATIVIDADES

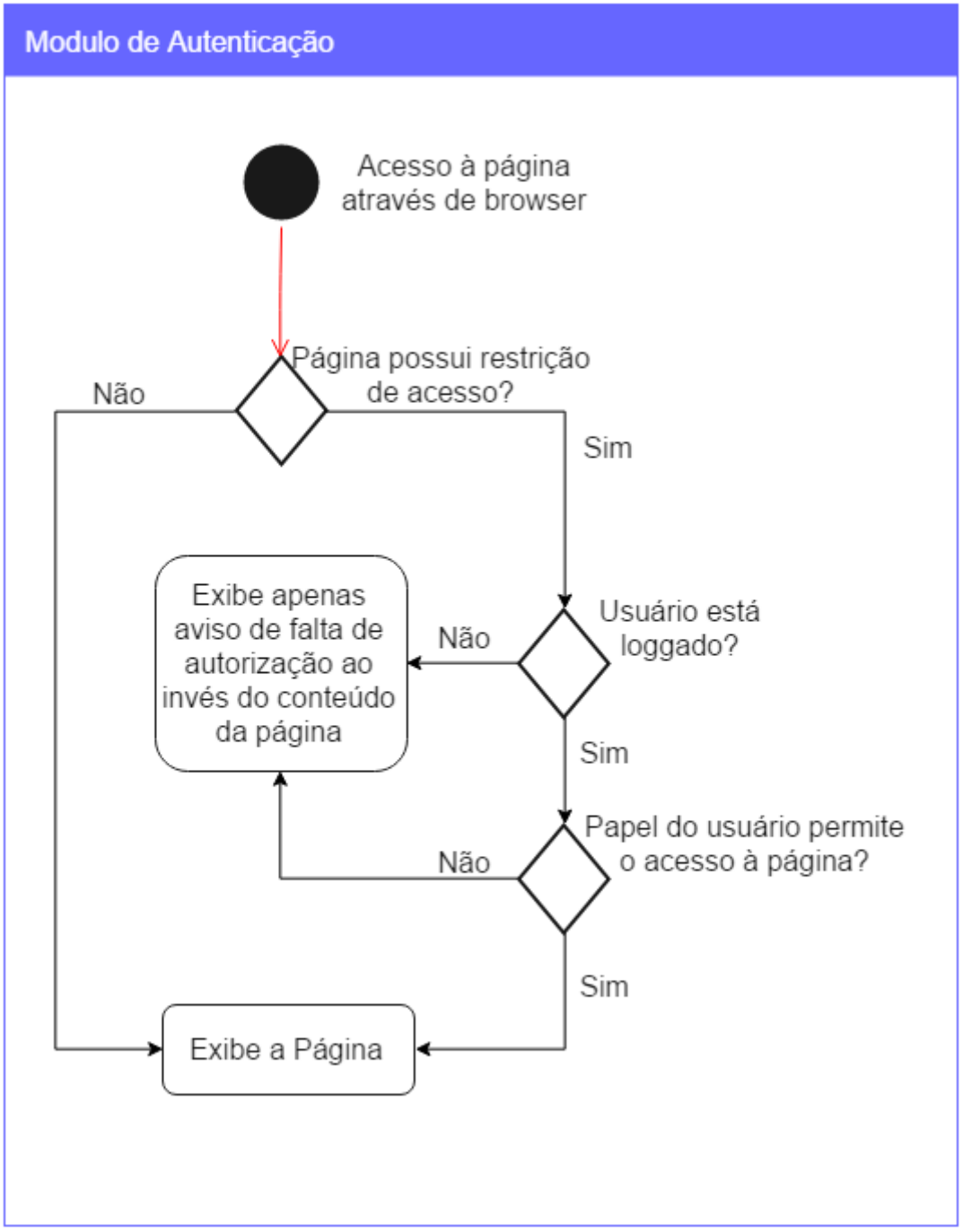


FIGURA 40 – DIAGRAMA DE ATIVIDADES – MÓDULO DE AUTENTICAÇÃO
FONTE: Os autores (2016)

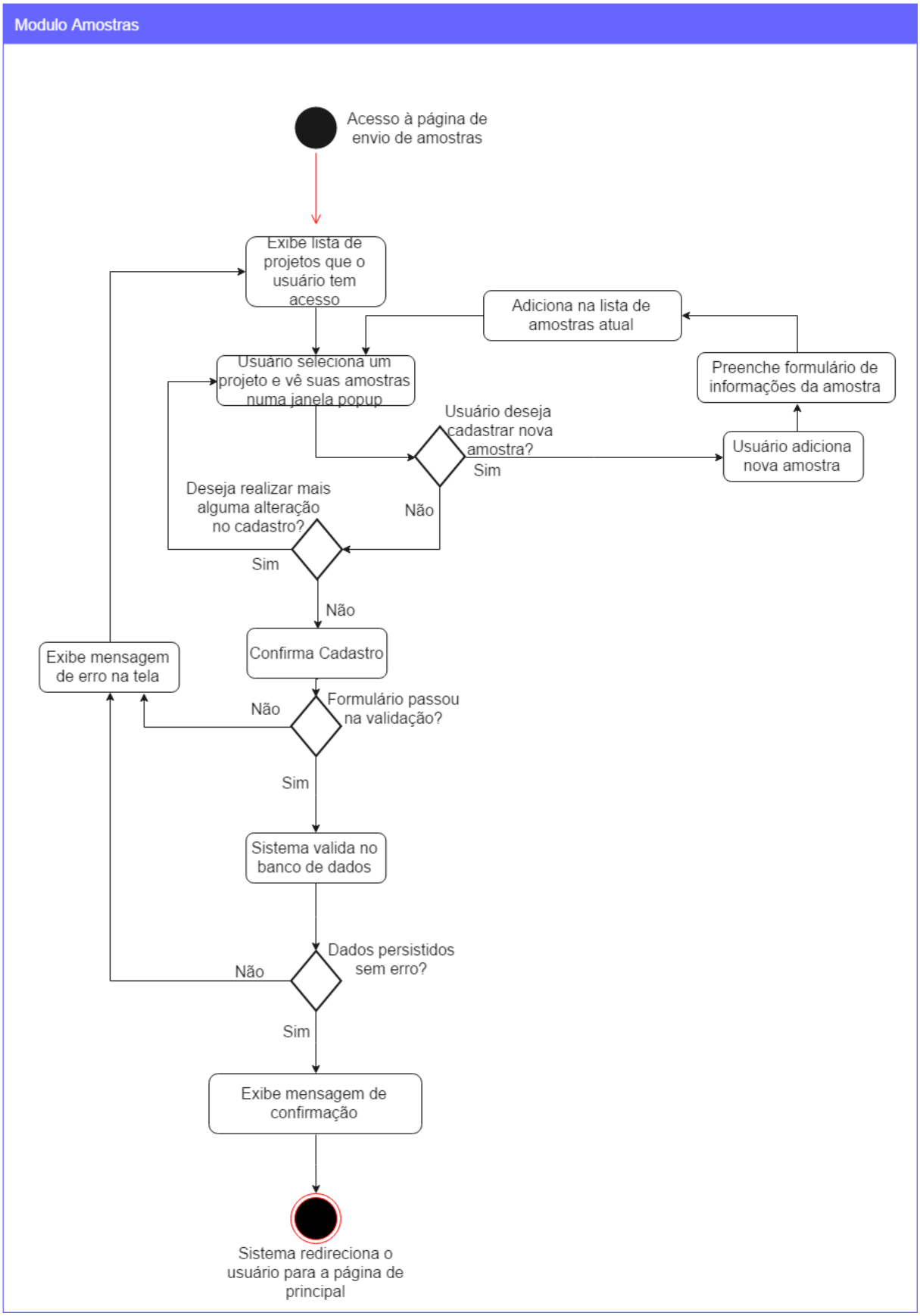


FIGURA 41 – DIAGRAMA DE ATIVIDADES – MÓDULO AMOSTRAS
FONTE: Os autores (2016)

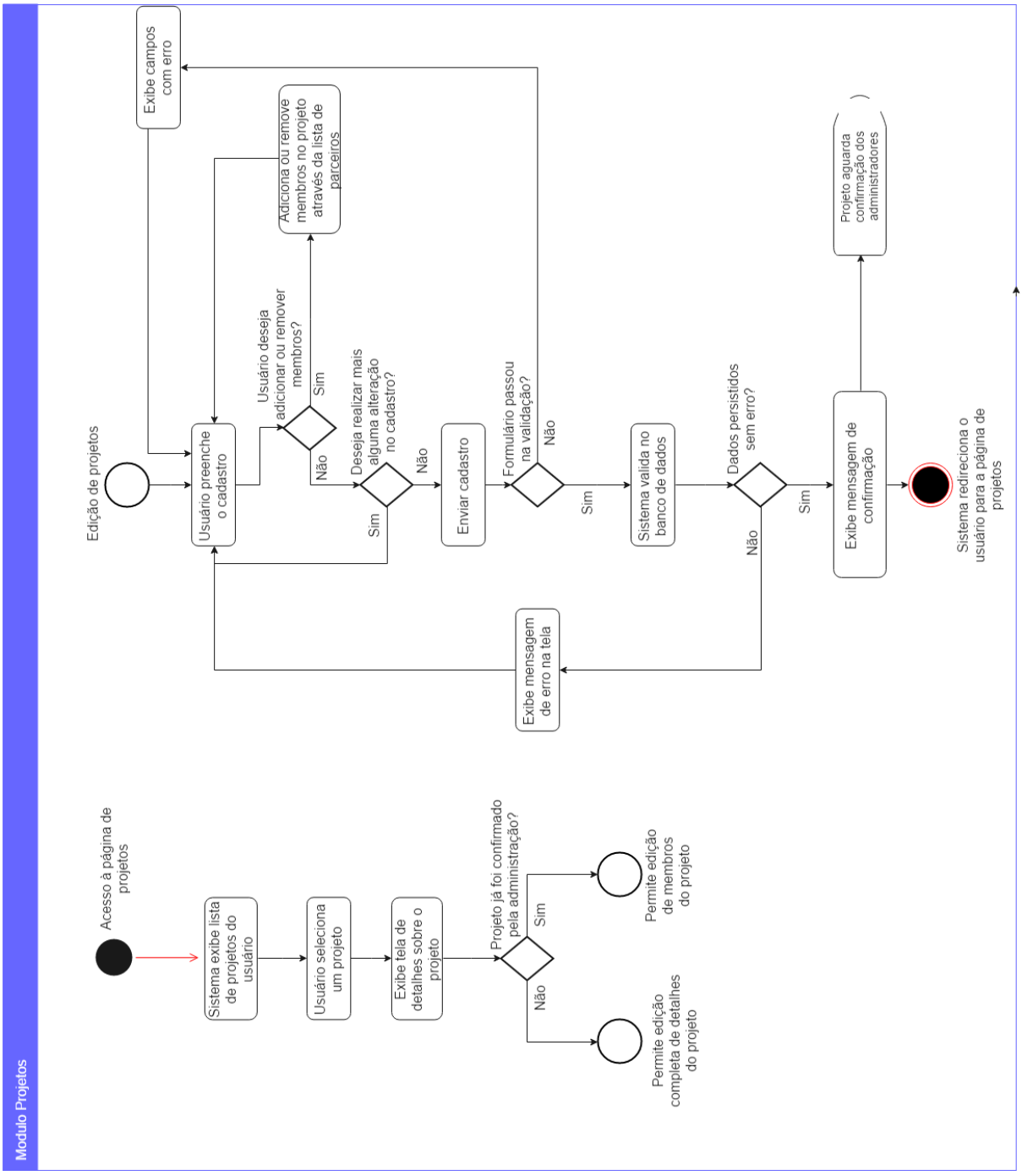


FIGURA 42 – DIAGRAMA DE ATIVIDADES – MÓDULO PROJETOS
FONTE: Os autores (2016)

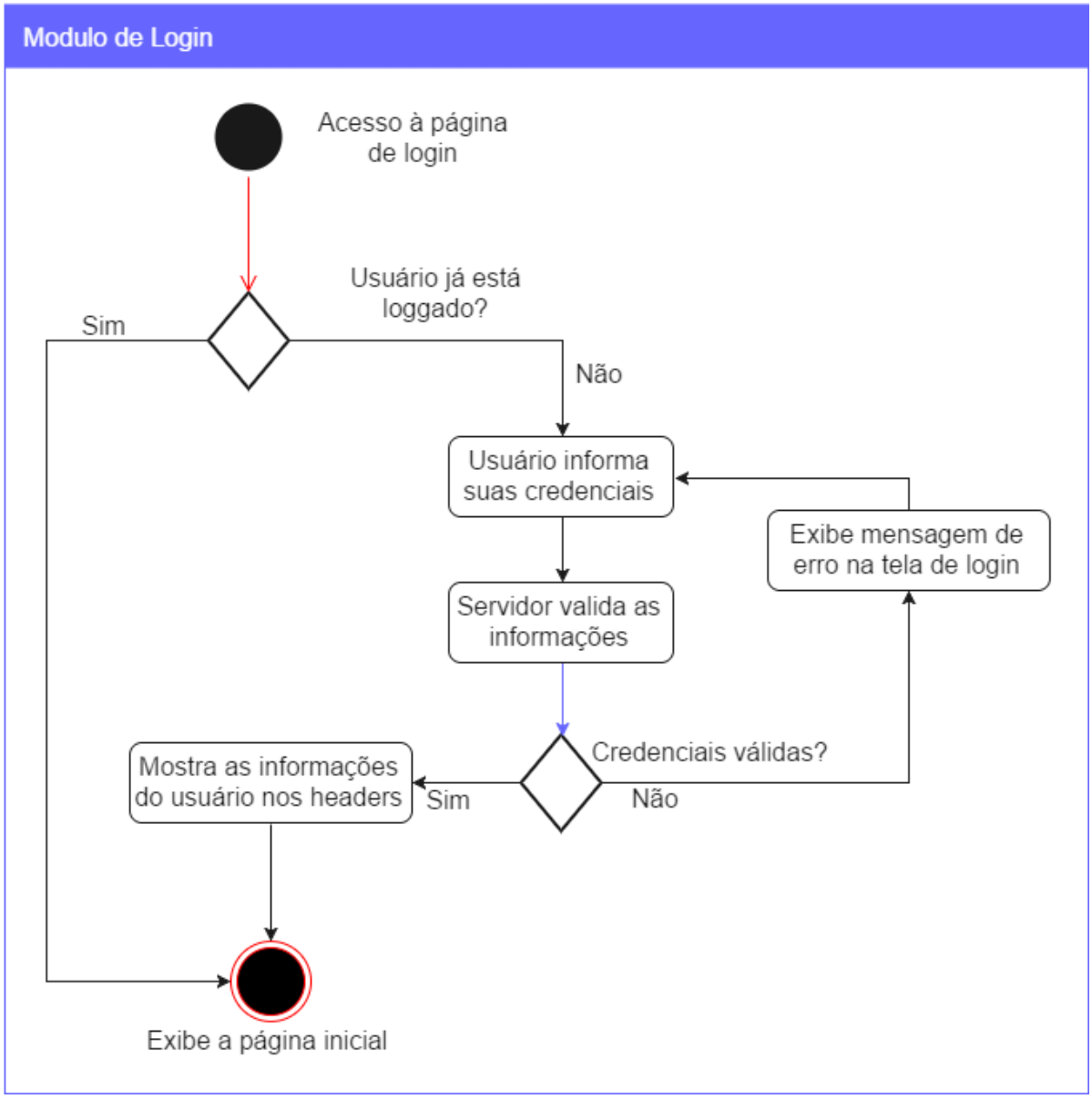


FIGURA 43 – DIAGRAMA DE ATIVIDADES – MÓDULO LOGIN
FONTE: Os autores (2016)

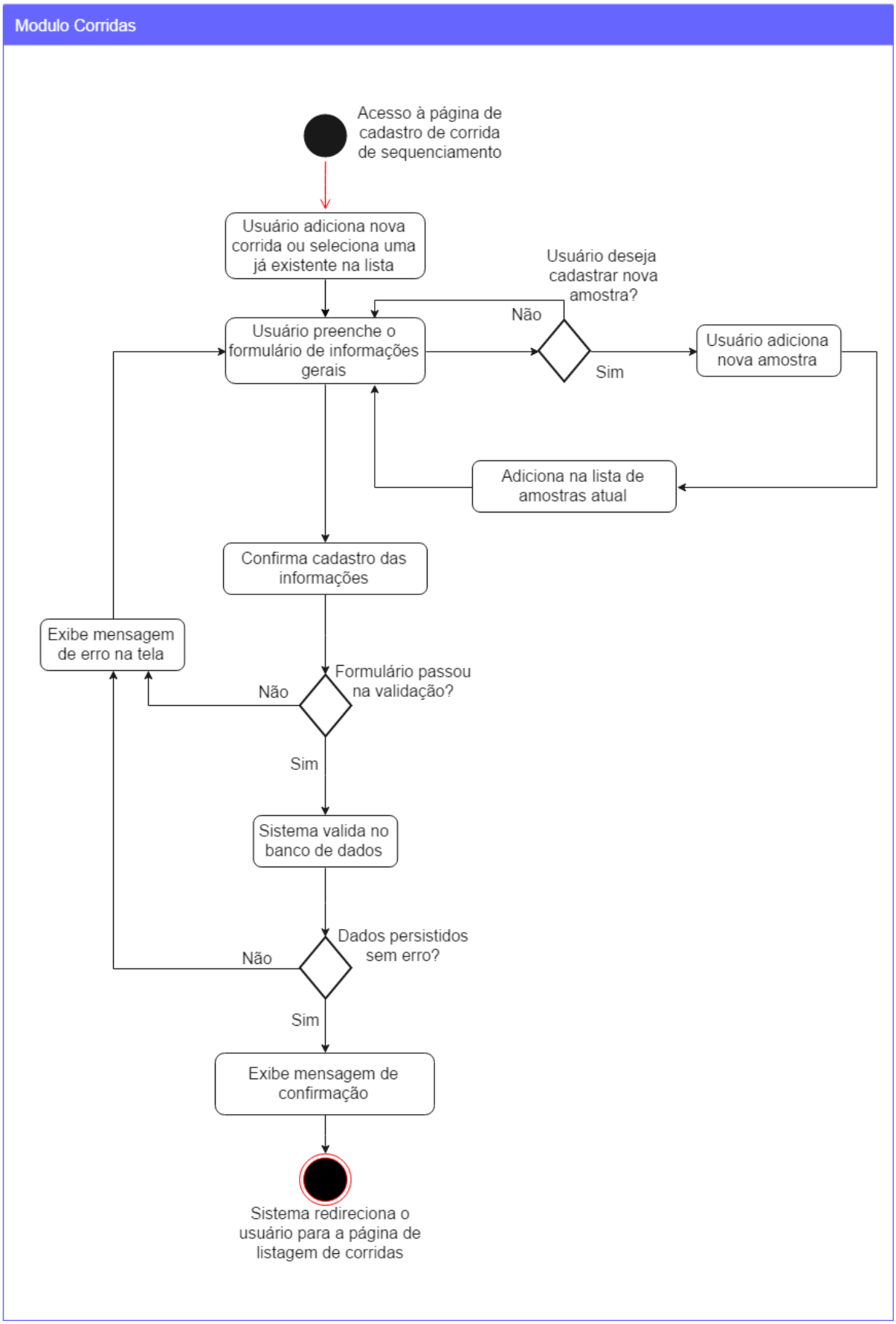


FIGURA 44 – DIAGRAMA DE ATIVIDADES – MÓDULO CORRIDA
FONTE: Os autores (2016)

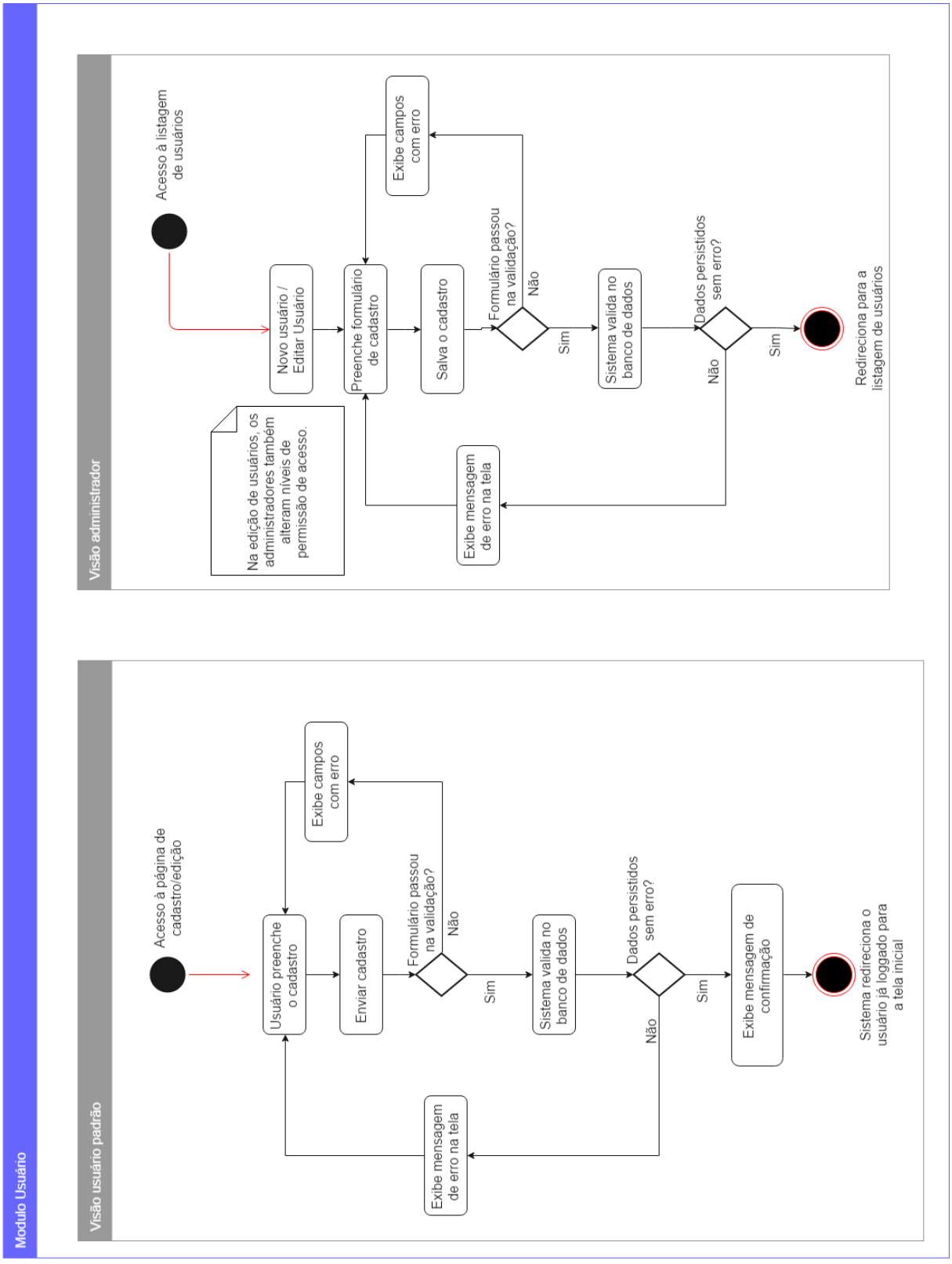


FIGURA 45 – DIAGRAMA DE ATIVIDADES – MÓDULO USUÁRIO
FONTE: Os autores (2016)