Open Library

Ramon Willer da Silva Melo
Luciana Rocha Cardoso
Curso: Tecnólogo em Análise e Desenvolvimento de Sistemas
Período: 6º Área de Pesquisa: Ciências Exatas e da Terra

Resumo: O acesso a novos meios tecnológicos está cada vez mais presente em nosso cotidiano, porém o acesso a literatura através de livros impressos, ainda é a principal fonte de acesso à educação de grande parte da população. Visando centralizar e otimizar o acesso a literatura de crianças e jovens de escolas públicas, foi desenvolvido o projeto *Open Library*, que busca centralizar o acervo literário de todas as bibliotecas do município através de um único sistema, permitindo uma melhor gestão das bibliotecas municipais e escolares, aumentando o número de títulos disponíveis à população. Para o desenvolvimento do projeto foi utilizado o *framework* de *Ruby on Rails* juntamente com os melhores padrões de desenvolvimento ágil.

Palavras-chave: Bibliotecas. Desenvolvimento Ágil. *Ruby on Rails*.

1. INTRODUÇÃO

O acesso à educação é fundamental para o desenvolvimento de uma criança ou jovem, e o incentivo a literatura é parte essencial no processo de educação de qualquer pessoa, porem muitas vezes o conteúdo literário disponível, a localização ou o horário de atendimento de uma biblioteca pode ser um fator desestimulante na criação de um hábito tão saudável, o hábito de ler. Visando resolver esse problema, foi desenvolvido um sistema que permita um melhor acesso à população ao acervo literário das bibliotecas do município.

Com base no problema do acesso da população aos livros disponibilizados pela biblioteca municipal, em especial os alunos matriculados em escolas municipais em perímetro urbano ou em distritos mais distantes do município, foi elaborado um modelo de sistema que permita o cadastramento de todos os livros pertencentes a biblioteca municipal e as escolas presentes no município, aumentando significativamente o número de títulos e exemplares aos usuários.

O objeto principal do projeto, é permitir a comunicação entre bibliotecas pertencentes ao município, organizando e otimizando o processo de catalogação, reserva e controle do acervo, expandindo o conteúdo literário disponível a população. O sistema é dividido em três perfis de acesso, administrador, bibliotecário e usuário, sendo função do administrador gerenciar todo o sistema, ao bibliotecário é destinado o cadastro e gerenciamento do acervo literário da biblioteca em que está associado, ao usuário cabe a reserva e consulta ao acervo.

Visando a colaboração e o autogerenciamento do sistema, os dados cadastrais pertencentes aos livros, são compartilhados entre todas as bibliotecas do sistema e a gestão do acervo será realizada pelo uso do código de barras. O cadastro do usuário será realizado pelo próprio usuário, sendo ele o responsável por manter seus dados atualizados.

2. DESENVOLVIMENTO

Este capítulo contém o referencial teórico do trabalho. O foco é a utilização dos conceitos de análise e desenvolvimento de uma aplicação WEB, baseada na programação orientada a objetos utilizando metodologias ágeis.

2.1. Referencial Teórico

A internet tem se tornado cada vez mais presente em nossas vidas, se tornando uma importante ferramenta no acesso a informação, a sua utilização tem permitido a diversas instituições organizarem suas informações e disponibilizarem seus dados de forma rápida e eficiente a seus clientes, utilizando o conceito de sistemas distribuídos, na forma de aplicações do tipo cliente-servidor, possibilitando o acesso aos usuários por diferentes plataformas.

Para Sommerville (2011, p.339),

> "Sistemas distribuídos que são acessados pela internet normalmente são organizados como sistemas cliente-servidor. Em um sistema cliente-servidor, o usuário interage com um programa em execução em seu computador local (por exemplo, um browser de web ou uma aplicação baseada em telefone). Este interage com outro programa em execução em um computador remoto."

Segundo Lisboa (2011),

> O desenvolvimento de sistemas consiste, basicamente, em criar um conjunto de atividades, parcialmente ordenadas, com a finalidade de obter um produto final, porém o maior esforço não está na criação, e sim na manutenção. As aplicações tornam-se cada vez mais complexas, e os requisitos dos clientes alteram-se muitas vezes, antes da conclusão do projeto. É preciso uma estrutura que permita a reutilização de código-fonte e o desenvolvimento simultâneo de partes do sistema, e se possível desvincular a aplicação do banco de dados, de forma que ela possa ser trocada sem causar nenhum (ou o mínimo de) impacto.

Visando os melhores aspectos abordados no desenvolvimento de um *software* foi criada as metodologias ágeis, afim de garantir uma melhor qualidade do produto desenvolvido e reduzir as falhas no processo de desenvolvimento, integrando todas partes envolvidas no projeto.

2.2. Engenharia de *Software* Ágil

A utilização de uma abordagem de *software* ágil nesse projeto, foi viabilizada pela utilização do *framework* de desenvolvimento web *Ruby on Rails*, que implementa mecanismos de abstração do banco de dados e controle de versionamento.

Para Pressam (2006),

> A engenharia de *software* ágil combina uma filosofia e um conjunto de diretrizes de desenvolvimento. A filosofia encoraja a satisfação do cliente e a entrega incremental do software logo de início; equipes de projeto pequenas, altamente motivadas, métodos informais; produtos de trabalho de engenharia de

> software mínimos e simplicidade global do desenvolvimento. As diretrizes de desenvolvimento enfatizam a entrega em contraposição à análise e ao projeto (apesar dessas atividades não serem desencorajadas) e a comunicação ativa e contínua entre desenvolvedores e clientes.

Segundo Sommerville (2011),

> Desenvolvimento incremental de *software*, que é uma parte fundamental das abordagens ágeis, é melhor que uma abordagem em cascata para a maioria dos sistemas de negócios, *e-commerce* e sistemas pessoais. Desenvolvimento incremental reflete a maneira como resolvemos os problemas. Raramente elaboramos uma solução do problema com antecedência.

O processo de desenvolvimento incremental, da engenharia de *software* ágil, permite melhor definição dos requisitos de um sistema, visando resolver problemas de comunicação entre os participantes do projeto, levando em consideração o carácter mutável tomado por uma aplicação, onde os requisitos definidos no planejamento do *software* podem sofrer alterações no decorrer do seu desenvolvimento ou implementação.

2.3. Especificação de *software*

A análise de requisito é fundamental para elaborar um sistema ou *software* que atenda e satisfaça plenamente os desenvolvedores, clientes e usuários. Quando bem definido e relatados evitam muitos problemas futuros e um deles é a alta manutenção de sistemas e *software* (REZENDE, 2005).

Os requisitos de um *software*, também chamados de requerimento de software ou de requisitos funcionais de um sistema, devem ser elaborados no início de um projeto de sistema ou *software*. (REZENDE, 2005).

Segundo Sommerville (2011),

> Especificação de *software* ou engenharia de requisitos é o processo de compreensão e definição dos serviços requisitados do sistema e identificação de restrições relativas à operação e ao desenvolvimento do sistema. A engenharia de requisitos é um estágio particularmente crítico do processo de software, pois erros nessa fase inevitavelmente geram problemas no projeto e na implementação do sistema.

2.4. Ferramentas

Nesta sessão serão abordadas as ferramentas, linguagem de programação e os princípios do *framework* utilizado no desenvolvimento do sistema proposto.

O desenvolvimento utilizando a linguagem *Ruby* com o *framework Ruby on Rails* não necessita de nenhuma IDE (*Integrated Development Environment*), sendo necessário apenas um editor de texto, porém e altamente recomendado a utilização de uma IDE, visto dos recursos disponibilizados e facilidade no entendimento do código e ajuda com possíveis erros de sintaxe da linguagem, e preenchimento automáticos de trechos de códigos.

2.4.1. *Ruby*

Ruby é uma linguagem dinâmica, orientada a objetos e que possui algumas características funcionais. Seu criador, Yukihino Matsumoto, queria uma linguagem que juntasse programação funcional e imperativa, mas acima de tudo que fosse uma linguagem legível. (Souza, 2014)

Esta abordagem facilita a utilização do *Ruby*, uma vez que as regras que se aplicam aos objetos aplicam-se a tudo em *Ruby*.

A escolha pela linguagem Ruby é graças a sua flexibilidade, uma vez que permite aos seus utilizadores alterar partes da linguagem. Partes essenciais do *Ruby* podem ser removidas ou redefinidas à vontade. Partes existentes podem ser acrescentadas. O *Ruby* tenta não restringir o programador.

De forma diferente a muitas linguagens de programação orientadas a objeto, o *Ruby* suporta somente herança simples, propositadamente. Mas em *Ruby* existe o conceito de módulos. Os módulos são coleções de métodos. As classes podem fazer o *mixin* de um modulo e receber todos os métodos do módulo diretamente.

O código desenvolvido é totalmente portável de plataforma, podendo ser executado em qualquer sistema operacional, a gestão das dependências do projeto ocorre através do gerenciador dependências *bundle*, e o controle de das versões do *Ruby* e das *Gems* e do *framework* e de responsabilidade do RVM (*Ruby Version Manager*), todo esse controle garante maior confiabilidade e segurança no momento do *deploy*.

2.4.2. *Ruby on Rails*

Um dos *frameworks* mais conhecidos do desenvolvimento, o *Ruby on Rails* é um *metaframework*, sendo suas partes independentes, isso permite que alguns desenvolvedores optem por soluções diferentes dos padrões, fornecidas pelo *framework*. O *Ruby on Rails* é um projeto *open-source* criado por David Heinemeier Hansson, a partir de um produto de sua empresa, o Basecamp.

Os principais aspectos propostos por David Heinemeier Hansson (DHH), são:

- "*Convention over configuration*", ou convenção à configuração: ao invés de configurar um conjunto de arquivos XML, por exemplo, adota-se a convenção e apenas muda-se o que for necessário;
- "*Dont Repeat Yourself*", ou "não se repita": nunca você deve fazer mais de uma vez o que for necessário (como checar uma regra de negócio);
- Automação de tarefas repetidas: nenhum programador deve perder tempo em tarefas repetidas e sim investir seu tempo em resolver problemas interessantes.

Os pilares acima, fazem parte da doutrina inicial do *framework*, porem ao longo dos anos, a mesma foi evoluindo e acrescentando outros pilares, chegando a nove pilares principais, são eles:

1. *Optimize for programmer happiness*
2. *Convention over Configuration*
3. *The menu is omakase*
4. *No one paradigma*

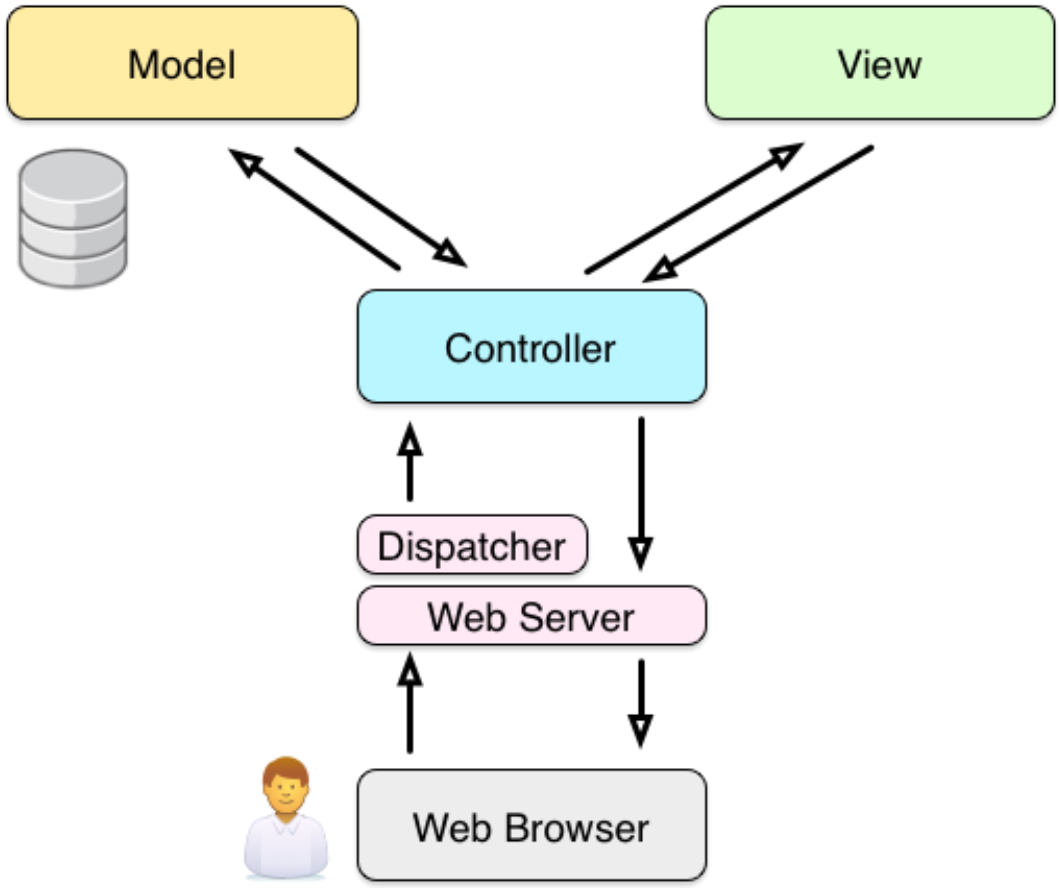
5. *Exalt beautiful code*
6. *Provide sharp knives*
7. *Value integrated systems*
8. *Progress over stability*
9. *Push up a big tent*

Esses pilares, juntamente com a filosofia da linguagem *Ruby*, garantiram ao *framework* sucesso entre as pequenas empresas e *startups*, nos Estados Unidos e no mundo.

O *Ruby on Rails*, segue a estrutura de MVC (*Model View Controller*), bastante conveniente para a construção de aplicações Web, pois permite a separação da lógica envolvida na persistência dos dados, dos aspectos destinados a apresentação, isso permite facilmente a mudança de tecnologia de banco de dados, layout de apresentação da aplicação e facilita nos testes automatizados.

O ciclo de uma requisição no *framework,* é representado pela imagem a seguir, sendo incluído na representação da estrutura MVC, o *Web Server* e a camada interna *Dispatcher,* sendo essa uma camada interna do *framework,* responsável pela analisa do recurso solicitado, e o encaminhamento ao controlador responsável por responder a solicitação.

Figura 1 – Estrutura MVC *Ruby on Rails*

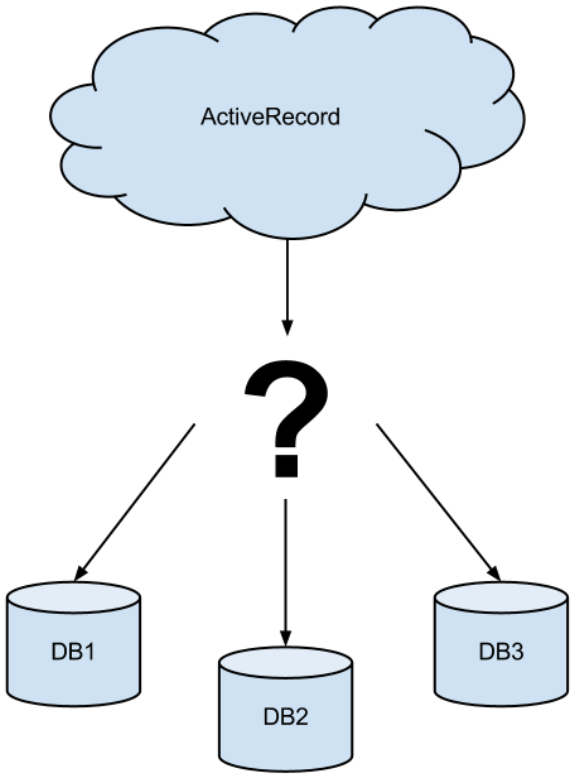


Fonte: https://pemcg.gitbooks.io/introduction-to-cloudformsautomation/content/chapter4/a_little_rails_knowledge.html

Levando em consideração que o *framework* é agnóstico a tecnologia de banco de dados, o sistema desenvolvimento, pode ser facilmente portável para diferentes tecnologias presentes no mercado, vistas as necessidades nos momentos da implantação, com bases propriedade do *framework* não faz sentido incluir nesse artigo, a tecnologia de banco de dados utilizada no projeto, já que a mesma pode ser alterada, bastando incluir a referência ao conector utilizado pela tecnologia escolhida.

A camada do *Model,* responsável pela integração com o banco de dados da aplicação é uma abstração que utiliza o framework ORM (*Object Relational Mapping*), *ActiveRecord*, para a persistência dos dados, a imagem a seguir representa a abstração disponibilizada pelo uso do *ActiveRecord*.

Figura 2 – *ActiveRecord*



Fonte: http://swx.com.br/2014/01/model-em-ruby-on-rails-parte-01/

3. METODOLOGIA

O projeto baseia-se no estudo de caso da biblioteca municipal do município de Manhuaçu. Sendo uma pesquisa qualitativa, visto que todos os dados necessários foram extraídos diretamente do cotidiano de trabalho dos servidores públicos envolvidos na administração da biblioteca.

No desenvolvimento do projeto partiu de uma pesquisa aplicada, pelo fato de demostrar como um *software* pode integrar e auxiliar na inclusão social de jovens a partir do acesso facilitado a literatura.

Quanto aos objetivos trata-se de uma pesquisa descritiva, descrevendo as experiências e processos proveniente do trabalho executado pelos servidores públicos, no cotidiano da biblioteca.

Na pesquisa descritiva realiza-se o estudo, a análise, o registro e a interpretação dos fatos do mundo físico sem a interferência do pesquisador. São exemplos de pesquisa descritiva as pesquisas mercadológicas e de opinião (Barros e Lehfeld, 2007).

3.1. Desenvolvimento do *Software*

A seguir será apresentado as etapas envolvidas no processo de desenvolvimento do sistema Open Library, essas etapas consistem na análise de requisitos, desenvolvimento, teste e implantação. Todas os diagramas utilizados no desenvolvimento do sistema, podem ser visualizados no apêndice A deste trabalho, e as formulários desenvolvidos no apêndice B.

3.2. Modelagem e Requisitos

Os requisitos definidos para a criação do software foram baseados na análise dos processos utilizados na biblioteca, planilhas gerenciais utilizadas, práticas de catalogação de livros já utilizadas pela biblioteca, e reuniões com a bibliotecária.

A etapas para o desenvolvimento do projeto seguiu conforme descrito na Figura 3 a seguir, sendo que o desenvolvimento o e o teste andaram em conjunto.

Figura 3 – Cronograma

Etapa	Janeiro	Fevereiro	Março	Abril	Maio	Junho	Julho
Analise de Requisitos							
Desenvolvimento							
Testes							
Implantação							

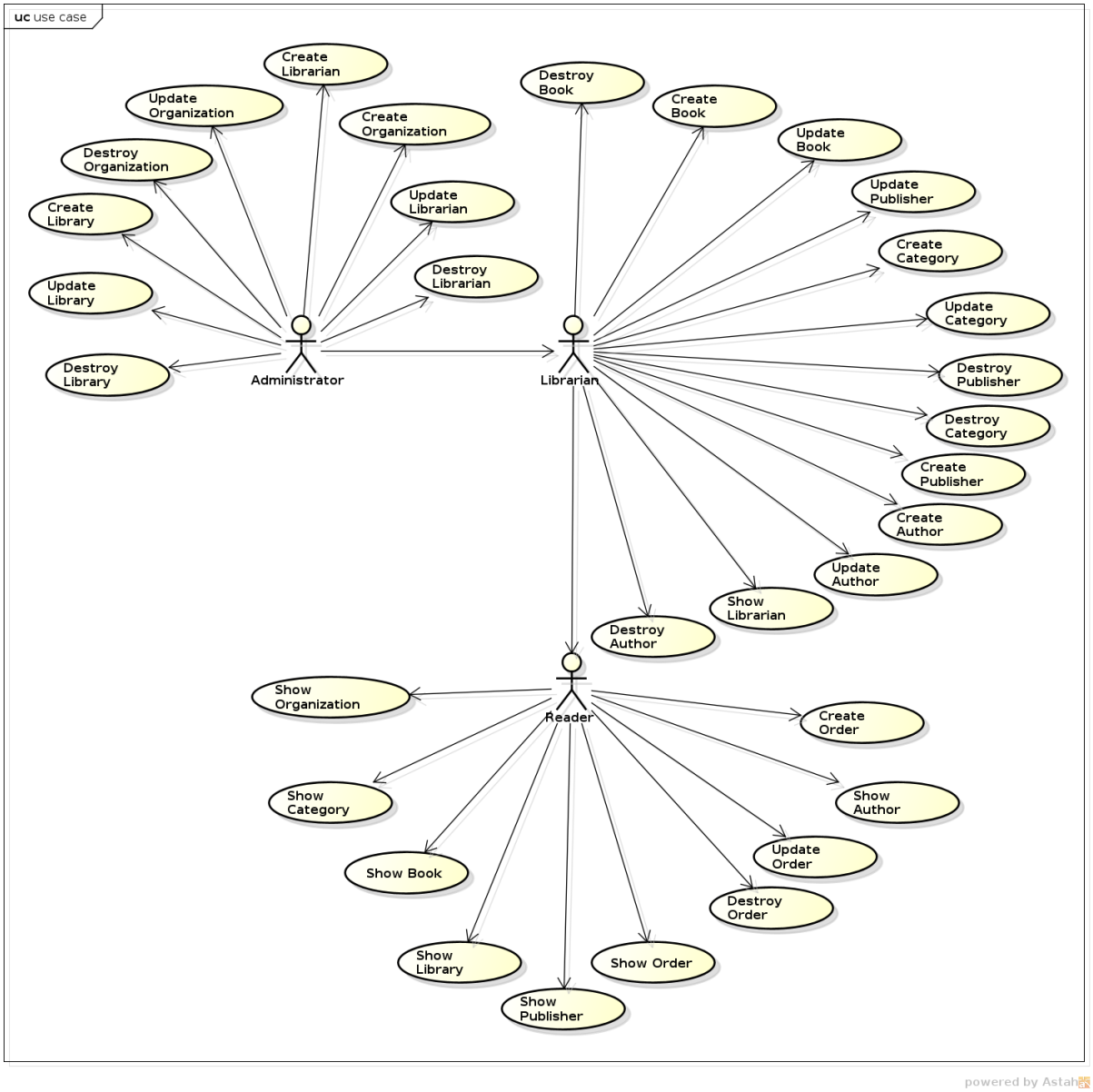
Fonte: Arquivo Pessoal

Dentre os requisitos definidos nas reuniões, o sistema abordara o processo de cadastro de manutenção do acervo literário, presente na biblioteca municipal, sendo expansível às bibliotecas das escolas, pertencentes ao município, o cadastro do livro será realizado pela bibliotecária, e está disponível para consulta e reserva por partes dos leitores cadastrados no sistema. O sistema não permitirá o cadastro duplicado dos livros, porem permite a inclusão de vários exemplares para o mesmo título, sendo esses codificados individualmente pelo sistema.

O leitor poderá realizar a reserva acessando o sistema, porem está será analisada pela bibliotecária, podendo ser indeferida e cancelada pela mesma, caso a mesma realize a reserva dos livros, o pedido ficara aguardando devolução, sendo apresentado uma mensagem de notificação a bibliotecária e ao usuário. A bibliotecária poderá realizar a reserva dos livros diretamente ao usuário, caso o mesmo não a tenha realizado pelo site, previamente.

A partir do diagrama de caso de uso, Figura 4, apresentado a seguir, teremos uma visão dos requisitos apresentados pelo sistema, e uma representação do papel de cada usuário no sistema, e suas atividades.

Figura 4 – Diagrama de Caso de Uso



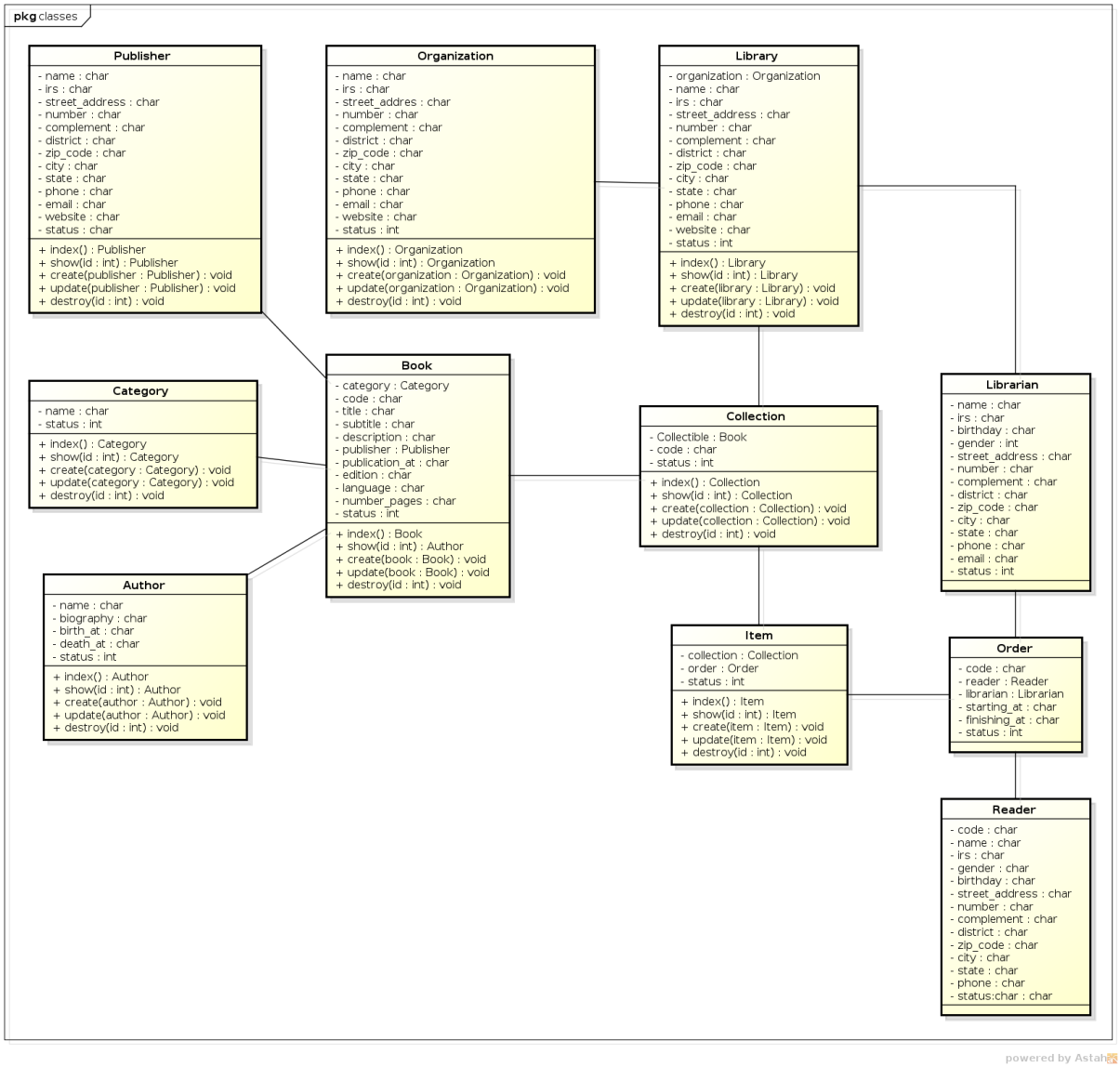
Fonte: Arquivo Pessoal

Com o diagrama apresentado podemos observar a representação de três entidades, assumindo papeis diferentes no sistema, o administrador com as permissões de criação de uma nova organização, biblioteca e cadastro de um novo bibliotecário. Ao perfil associado ao bibliotecário, estão as atividades de cadastro do acervo, e manutenção, dos pedidos de reserva e controle da biblioteca, observando que o bibliotecário herda todas as funções disponíveis ao leitor. O papel do leitor se limita a consulta do acervo, organização, biblioteca, criação e manutenção das reservas associadas ao seu perfil.

O diagrama a seguir, representa com maior detalhe os relacionamentos presentes entre os objetos do sistema, esses relacionamentos são a base para a criação do modelo de dados, lembrando que o *framework* utilizado cria o modelo de persistência a partir do modelo das classes geradas, sendo os relacionamentos

definidos no momento da criação das classes, isso permite que a fonte de dados seja alterada tão facilmente.

Figura 5 – Diagrama de Classe



Fonte: Arquivo Pessoal

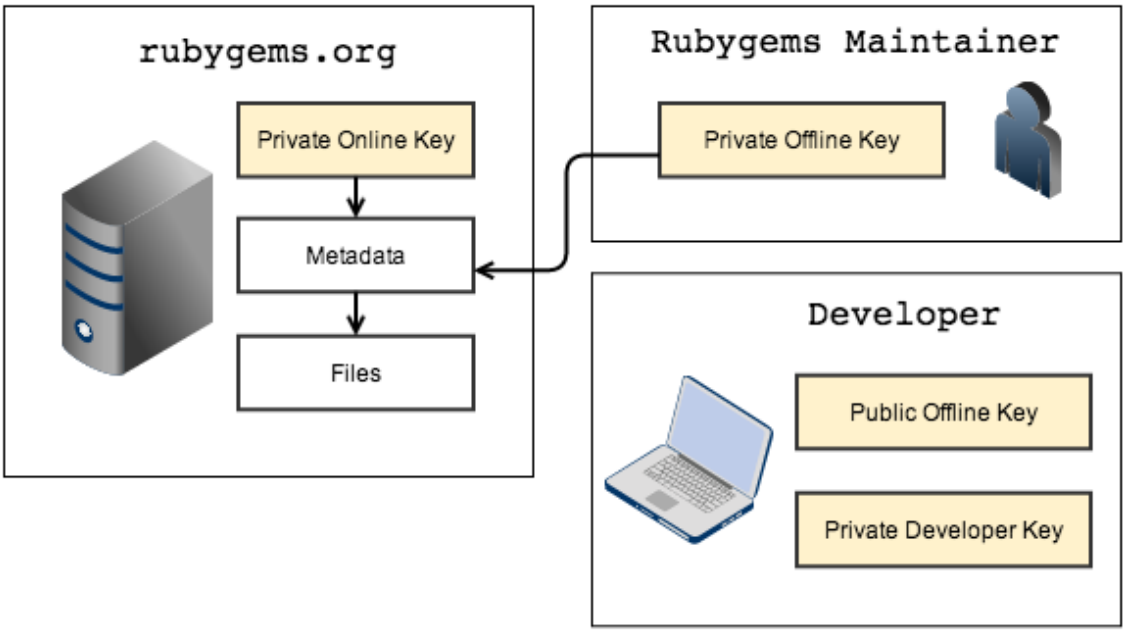
O diagrama de classes, para o desenvolvimento baseado em camadas, MVC, pode ser considerado um dos mais importantes, visto que o mesmo representara toda a estrutura dos dados apresentados pela camada de persistência, o M do modelo MVC, utilizada pelo *framework*. A partir da representação das classes, podemos estruturar os atributos, métodos e estrutura do modelo de dados, de forma que o usuário final tenha uma perspectiva geral do sistema, tornando mais prático o alinhamento dos requisitos com todos os envolvidos no projeto. Após a etapa de análise dos requisitos, a etapa mais importe do processo de desenvolvimento de um *software* segundo IEEE (Apud Quiterio, 2012, pag. 1), seguiremos para a etapa de codificação do *software*.

3.3. Codificação do Sistema.

O processo de desenvolvimento do sistema, utilizando o *framework Ruby on Rails*, seguiu rigorosamente os padrões estabelecidos. Sendo utilizado o modelo de desenvolvimento MVC, adotado como padrão pelo *framework*. Todas as dependências utilizadas no projeto foram centralizadas no arquivo *Gemfile*, sendo este utilizado pelo gerenciador de dependências *bundle* para a manutenção das bibliotecas.

A origem das dependências, baixadas pelo gerenciador de dependências *bundle*, e o repositório *RubyGems*, a manutenção das bibliotecas e o acesso aos arquivos é representado pela imagem abaixo:

Figura 6 – *RubyGems Server*



Fonte: https://medium.com/square-corner-blog/securing-rubygems-with-tuf-part-1-d374fdd05d85

Todo o processo de persistência dos dados vou destinado ao componente Active Record, que realizado o mapeamento dos atributos pertencentes aos modelos, e controla as versões com o banco de dados, para mero efeito de comodidade, foi utilizado o banco de dados MySQL, porem o sistema tem suporte a diferentes tecnologias para o armazenamento dos dados, sendo as principais, PostgreSQL, MySQL, SQLite, Oracle, SQL Server, sendo de responsabilidade do cliente, a escolha pela ferramenta de persistência. O controle de acesso aos modelos baseado no perfil do sistema, foi realizado utilizando o conceito de políticas de segurança, da biblioteca (*gem*) *Pundit*.

A *gem* de controle de acesso*, Devise,* foi utilizada em conjunto com a *gem* Pundit, para o controle de permissões e acesso ao sistema, garantindo a criptografia da sessão do usuário e suas diretrizes de acesso.

As políticas de acesso foram definidas baseado no escopo do usuário e nos controles utilizados no projeto, sendo que cada recurso disponibilizado pelo controle, recebeu uma política de acesso.

Os controles e rotas seguiram o padrão REST (*Representational State Transfer*), Transferência de Estado Representacional, que utiliza dos protocolos do HTTP, como padronização das chamadas a aplicação.

O conceito de recursos apresentado pelo REST juntamente com os métodos do protocolo HTTP, permite ao sistema, uma fácil implementação futura de uma API (*Application programming interface*), permitindo que outros sistemas utilizam dos dados e funções da aplicação, de forma fácil e consistente, aumentando ainda mais os canais de acesso a informação.

A camada de visualização utilizou o *framework Bootstrap*, sendo este utilizado o principal *framework front-end* disponível no mercado. O *framework* permite uma interação com diferentes dispositivos, garantindo uma melhor visualização nos diferentes tamanhos de tela, tornando os componentes presentes na tela ajustáveis a resolução do dispositivo utilizado para o acesso a aplicação.

O *framework* CSS (*Cascading Style Sheets*), possibilitou também uma melhor definição dos elementos utilizados na construção da página, além de organizar facilitando alterações futuras de cores, tamanhos e outros elementos da aplicação, graças aos recursos disponibilizados pela utilização do SASS (*Syntactically Awesome Style Sheets*), na definição da construção do arquivo de *layout* da aplicação, facilitando a manutenção futura e otimizando as páginas de estilo.

Os testes foram realizados durante todo o desenvolvimento da aplicação, sendo o desenvolvimento direcionado ao teste, permitindo a redução das mudanças no escopo do projeto, e garantindo um software realmente funcional ao final do projeto.

O desenvolvimento do projeto, seguiu as convenções definidas pelo framework Ruby on Rails, em especial, os dois conceitos propostos:

- *“Convention over configuration”*
- *“Dont Repeat Yourself”*

Estes conceitos permitem a criação de códigos mais limpos, elegantes e de fácil manutenção, uma vez que o desenvolvedor seria impedido de realizar grandes mudanças no projeto, fugindo do padrão de configuração proposto, além de impedir que trechos de códigos sejam repetidos e espalhados livremente pelo código, gerando uma complexidade desnecessária.

O *framework* possibilita além do controle de versionamento do banco de dados, estruturas de manutenção, criação e recuperação de dados, roteamento dos controles, e separação da camada de apresentação, mecanismos de execução de tarefas e compressão dos arquivos de estilo, disponibilizados para na camada de apresentação.

A principal diferença no processo de criação da aplicação, foi a velocidade de criação dos processos básicos, a partir da análise de requisitos, utilizando as tarefas de automatização disponibilizadas pelo *Ruby on Rails*, através da linha de comando, essas tarefas possibilitaram a criação dos métodos de inclusão, alteração, exclusão e listagem dos modelos utilizados na aplicação, de forma rápida, eficiente, garantindo um alto padrão de qualidade.

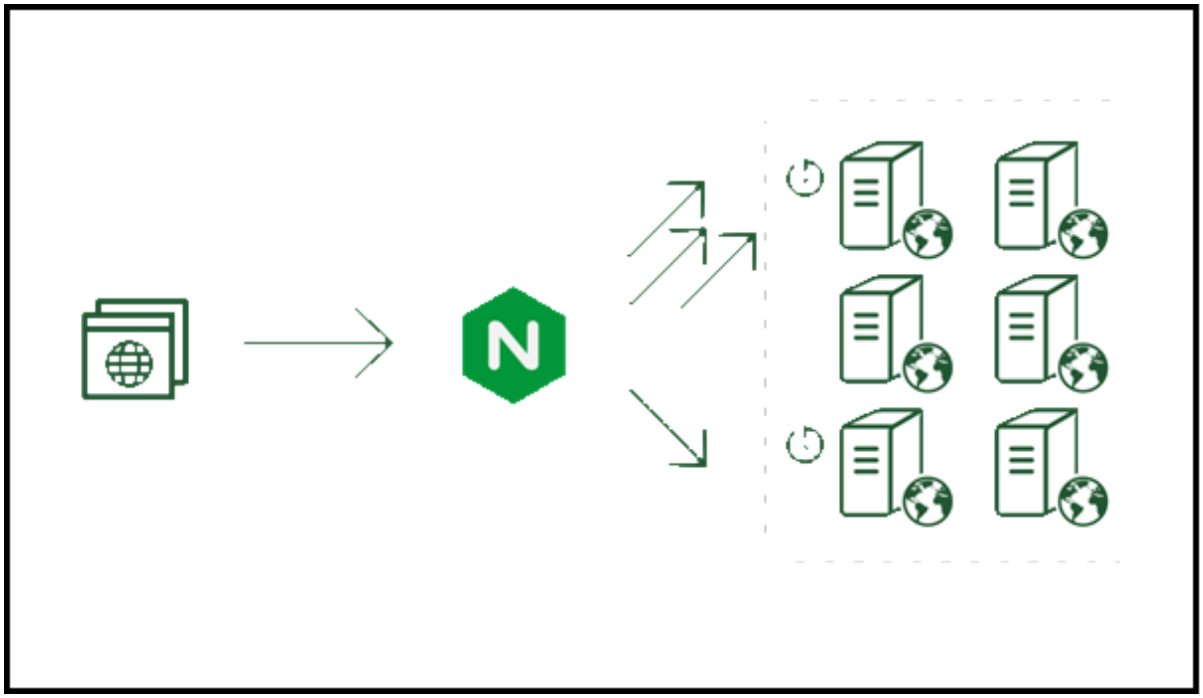
O padrão no desenvolvimento possibilita uma curva de aprendizado mais curta, facilitando a inclusão de novos colaboradores no desenvolvimento além de diminuir o tempo gasto na manutenção do sistema.

3.4. Implantação.

O sistema foi desenvolvido para que o seu processo de implantação e utilização fosse o mais simples possível, sendo necessário apenas que o sistema seja hospedado em servidor web, Nginx ou Apache.

Sendo recomentado fortemente a utilização do servidor web Nginx, pela sua performance superior e disponibilidade de recursos, segue imagem representando um ambiente de hospedagem distribuído e redundante.

Figura 7 – Ambiente de hospedagem



Fonte: https://www.nginx.com/blog/nginx-plus-r6-released/

Após configurado o servidor e a fonte de dados, basta executar o comando *bundle install*, e todas as dependências do projeto serão baixadas e instaladas automaticamente.

O sistema é independente de plataforma, porém para evitar custos com licença, e melhor performance é sugerido a utilização de um sistema baseado no Kernel Linux.

A utilização do sistema por parte dos usuários e realizada através de navegador WEB, permitindo assim que o usuário utilize qualquer sistema operacional com suporte e acesso a navegação WEB, este processo permite uma economia significativa no processo de implantação de um sistema.

Durante o processo de implantação do sistema, será necessária a definição de chaves de segurança, e criação de variáveis de ambiente, essas chaves e variáveis serão utilizadas como mecanismos de segurança pelo sistema, tanto na definição de chaves de acesso, senhas de usuários e acesso ao banco de dados.

A organização das chaves através de variáveis de ambiente, é o padrão mais utilizado pelos desenvolvedores atualmente, evitando a proliferação de arquivos de configuração pelo sistema, além de difundir a senha de acesso por diversos lugares aumentando a vulnerabilidade do sistema.

4. CONCLUSÃO

Ao acesso a literatura, por parte dos estudantes da rede pública de ensino, e um grande problema, criar o habito de ler, sem uma fonte consistente de livros, e um fator dificultador para qualquer educador.

Pensando nesse problema, como criar mecanismos que facilite o acesso do jovem a literatura? E que surgiu o presente trabalho.

Podemos concluir que utilização de um *software* que permita maior controle do acervo literário disponível no município e facilite o acesso de alunos e leitores a informação seja um grande diferencial no processo de inclusão da literatura nas escolas, e na vida de mais pessoas.

Com a implantação do sistema desenvolvido, o município poderá disponibilizar de forma mais fácil e consistente todo o seu acervo literário a população, principalmente aos jovens das escolas municipais presente na zona rural, o sistema disponibilizara aos estudantes e a população um acervo melhor organizado e categorizado, tornando fácil o acesso, e o controle.

Para desenvolvimento futuro, serão realizadas alterações no *software*, de forma a permitir que os usuários realizem sugestões de livros, avaliem o atendimento e os livros disponibilizados pela biblioteca.

Tradando-se da pesquisa realizada, podemos destacar o aprendizado adquirido com o desenvolvimento do *software*, técnicas e padrões de desenvolvimento utilizados pelo framework e pela linguagem de programação adotada, ficando claro que a utilização de padrões no desenvolvimento de um software e um fator decisivo para a sua qualidade e capacidade de manutenção, além de permitir uma melhor interação, em um ambiente colaborativo.

Por fim, podemos concluir que a disponibilidade, controle e organização do acervo literário do município, pode se tornar um fator importante na inclusão de jovens e adolescentes a literatura. O sistema será disponibilizado a outros municípios que demostrarem interesse no projeto, permitindo assim um maior número de usuários.

5. REFERÊNCIAS

Astah. Astah community. Disponível em: < http://astah.changevision.com/en/product/astah-community.html>. Acesso em: 22 de junho. 2017.

BALBÉ, M.; **Como gerenciar a Tecnologia da Informação?** 2012. Disponível em:<http://mariliabalbe.com/como-gerenciar-a-tecnologia-da-informacao-2/>. Acesso em 25 de junho. 2017.

CHOPRA, S.; MEINDL, P. **Supply chain management: strategy, planning and operations**. New York: Prentice Hall, 2003.

DAVID, M. F. **Programação Orientada a Objetos: uma introdução**. Disponível em: < http://www.hardware.com.br/artigos/programacao-orientada-objetos/>Acesso em: 20 de junho de 2017.

GONÇALVES, Gilberto. Implantação de um sistema de informação-*ENTERPRISE RESOURCE PLANNING (ERP)*: Estudo de caso de uma indústria eletrônica. **Revista de Engenharia e Tecnologia,** v. 02, n. 1, p. 57-68, 2010.

GUEDES, T. A.; **UML2: Uma abordagem prática** 2011. Disponível em:< http://img.americanas.com.br/produtos/01/00/manual/7482014.pdf>. Acesso em 22 de junho. 2017.

PRESSMAN, Roger S. **Engenharia de software**. McGraw Hill Brasil, 2011.

QUITERIO, A. P.; **Análise de requisitos** 2012. Disponível em:<http:// infoescola.com/engenharia-de-software/analise-de-requisitos/>. Acesso em 24 de junho. 2017.

ROCHA, Julio Fernande; DIAS, Jaime William.; **Importância do banco de dados nas aplicações 2015**. Disponível em:< http://web.unipar.br/~seinpar/2015/_include/artigos/Julio_Fernandes_Rocha.pdf>. Acesso em 25 de junho. 2017.

Ruby. Linguagem de Programação Ruby em: < https://www.ruby-lang.org/pt/>. Acesso em: 21 de junho. 2017.

Ruby on Rails. Framework de Desenvolvimento Web em: < hhttp://rubyonrails.org/>. Acesso em: 21 de junho. 2017.

SELLTIZ, C.; WRIGHTSMAN, L. S.; COOK, S. W. Métodos de pesquisa das relações sociais. São Paulo: Herder, 1965.

SIMÕES, Natanael Augusto Viana. Modelagem UML Através do Microsoft Visual Studio 2010. **Revista Olhar Cientifico,** v. 01, n. 1, p.195-204, 2010.

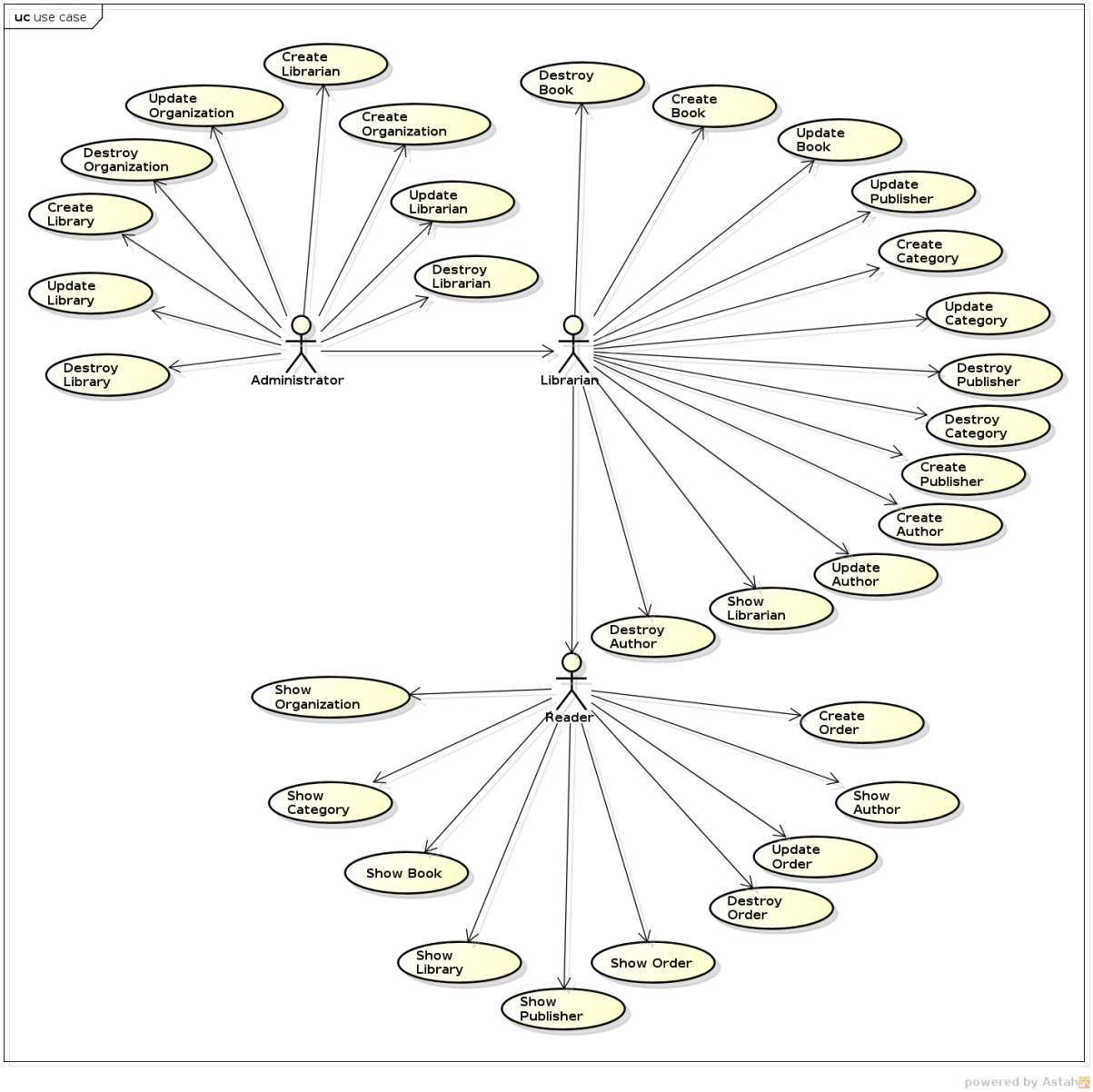
TEIXEIRA, F., HASTENREITER, H. & SOUZA, C. Diferenças entre inovação tecnológica e desempenho: evidências de uma rede de aprendizado. In: **ENCONTRO ANUAL DA ASSOCIAÇÃO NACIONAL DOS PROGRAMAS DE PÓS GRADUAÇÃO EM ADMINISTRAÇÃO**, 25º, 2001, Campinas. Anais ...São Paulo: Anpad, 2001.

TORALDO, Ronaldo Schmitz; SAPORITI, Alexandre Figueiredo; ZANQUETTO FILHO, Hélio. Implementação de um sistema ERP: O caso de uma grande empresa. **Anais do 21 ENEGEP-ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO**, 2001.

APÊNDICE A

Nessa sessão serão apresentados os diagramas utilizados na modelagem de dados do sistema Open Library.

Figura 8 – Diagrama de Caso de Uso



Fonte: Arquivo Pessoal.

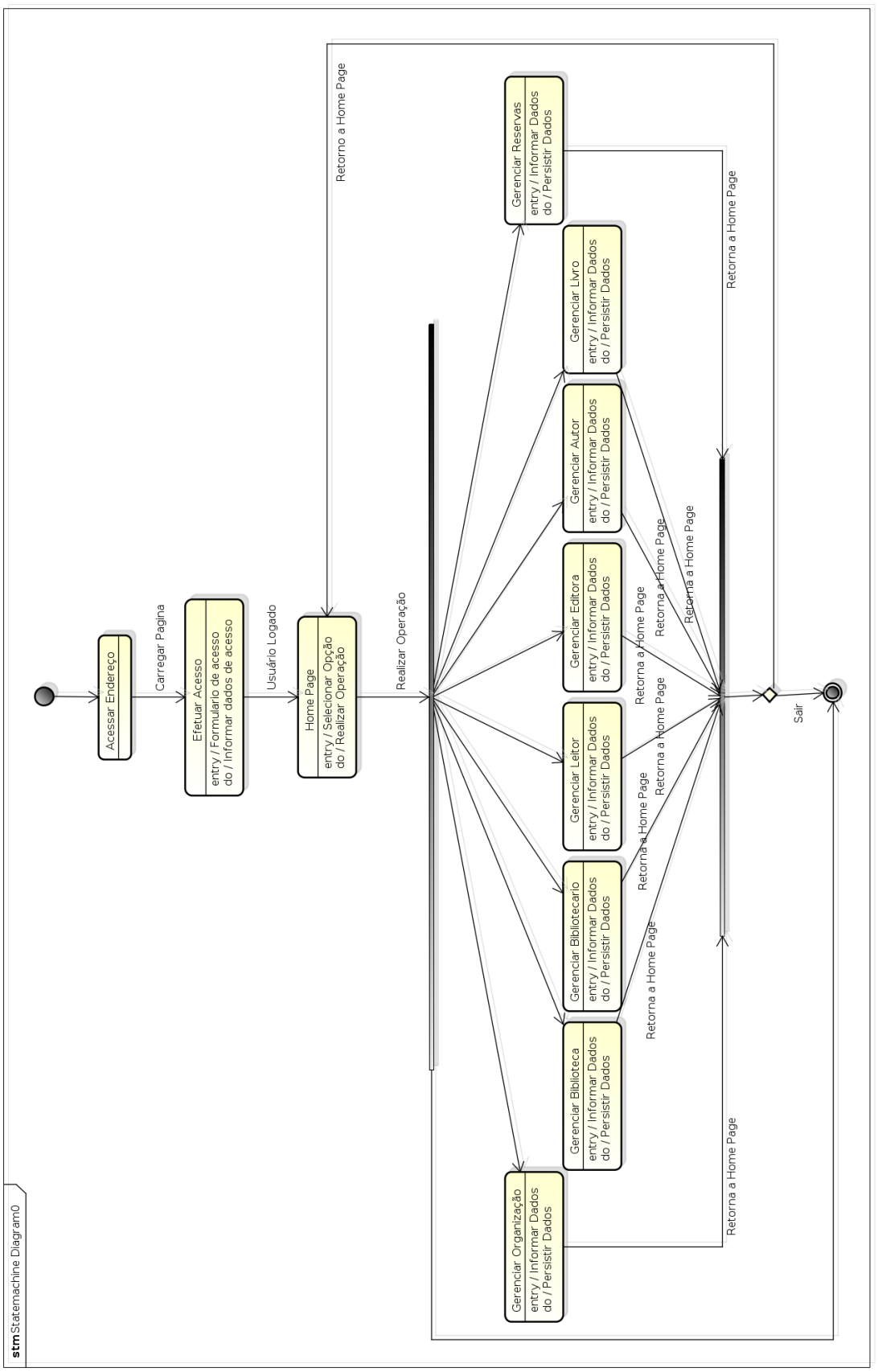
A tabela a seguir apresenta a descrição de cada caso de uso, apresentado no diagrama acima.

Caso de Uso	Usuário	Descrição
Create Library	Administrador	Cadastro de Biblioteca
Update Library	Administrador	Atualização da Biblioteca
Destroy Library	Administrador	Exclusão da Biblioteca
Show Library	Usuário, Bibliotecário, Administrador	Visualização da Biblioteca
Create Organization	Administrador	Cadastro da Prefeitura
Update Organization	Administrador	Atualização da Prefeitura
Destroy Organization	Administrador	Exclusão da Prefeitura
Show Organization	Usuário, Bibliotecário, Administrador	Visualização da Prefeitura
Create Librarian	Administrador	Cadastro do Bibliotecário
Update Librarian	Bibliotecário, Administrador	Atualização do Bibliotecário
Destroy Librarian	Administrador	Exclusão do Bibliotecário
Show Librarian	Bibliotecário, Administrador	Visualização do Bibliotecário
Create Book	Bibliotecário, Administrador	Cadastro do Livro
Update Book	Bibliotecário, Administrador	Atualização do Livro
Destroy Book	Bibliotecário, Administrador	Exclusão do Livro
Show Book	Usuário, Bibliotecário, Administrador	Visualização do Livro
Create Category	Bibliotecário, Administrador	Cadastro da Categoria
Update Category	Bibliotecário, Administrador	Atualização da Categoria
Destroy Category	Bibliotecário, Administrador	Exclusão da Categoria
Show Category	Usuário, Bibliotecário, Administrador	Visualização da Categoria
Create Author	Bibliotecário, Administrador	Cadastro do Autor
Update Author	Bibliotecário, Administrador	Atualização do Autor
Destroy Author	Bibliotecário, Administrador	Exclusão do Autor
Show Author	Usuário, Bibliotecário, Administrador	Visualização do Autor
Create Publisher	Bibliotecário, Administrador	Cadastro da Editora
Update Publisher	Bibliotecário, Administrador	Atualização da Editora
Destroy Publisher	Bibliotecário, Administrador	Exclusão da Editora
Show Publisher	Usuário, Bibliotecário, Administrador	Visualização da Editora
Create Order	Usuário, Bibliotecário, Administrador	Cadastro da Reserva
Update Order	Usuário, Bibliotecário, Administrador	Atualização da Reserva
Destroy Order	Usuário, Bibliotecário, Administrador	Exclusão da Reserva
Show Order	Usuário, Bibliotecário, Administrador	Visualização da Reserva

Fonte: Arquivo pessoal

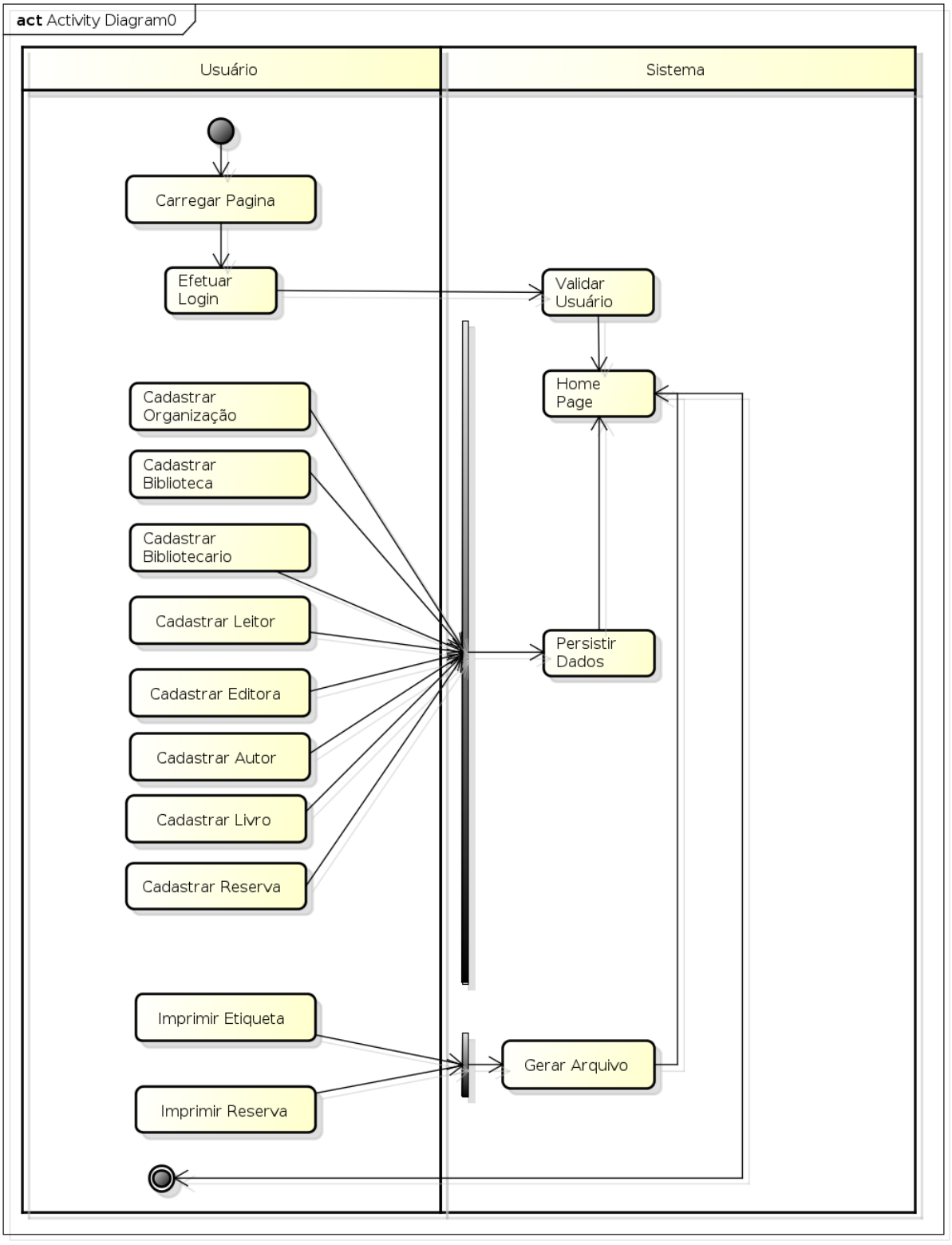
Diagrama de Máquina de Estado, foi utilizado na modelagem de dados para demostrar o comportamento dinâmico do sistema, ao invés de detalhar separadamente os processos. No diagrama é apresentado os processos realizados a partir do acesso ao site.

FIGURA 9 – Diagrama de Máquina de Estado



Fonte: Arquivo pessoal.

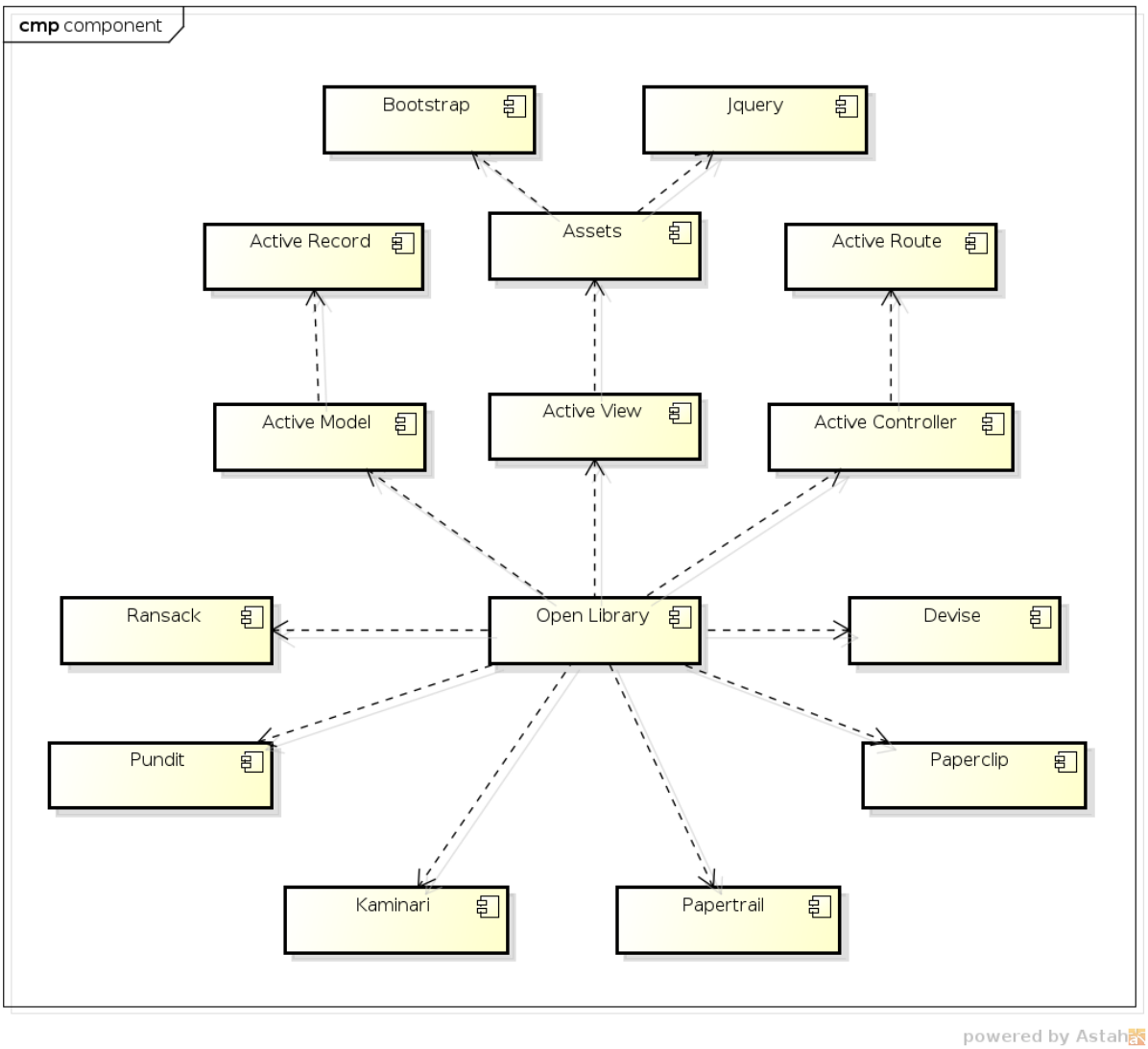
FIGURA 10 – Diagrama de Atividade



Fonte: Arquivo pessoal.

O diagrama de atividade apresenta as funcionalizadas disponibilizadas ao usuário, representando o comportamento do sistema.

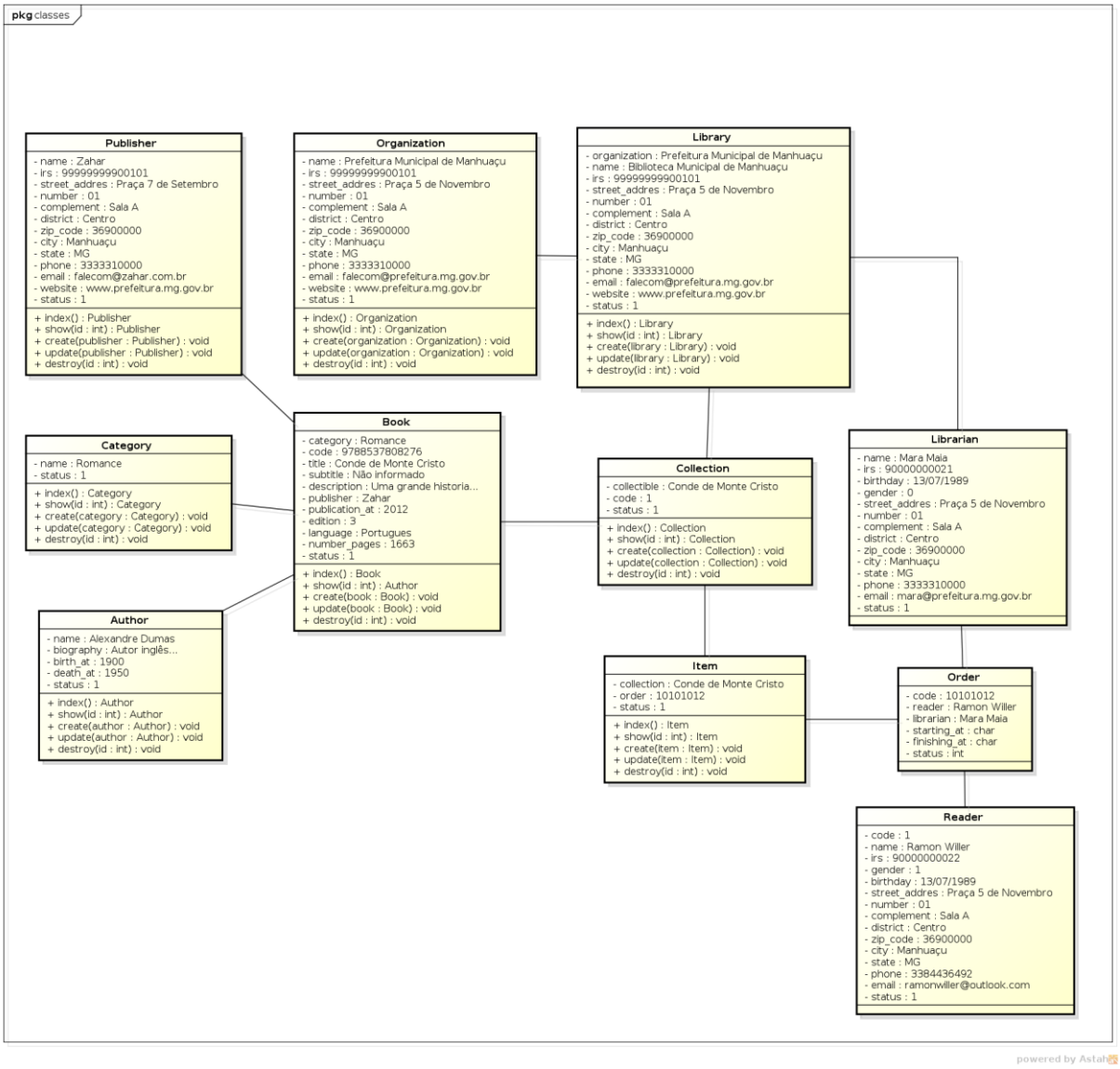
FIGURA 11 – Diagrama de Componentes



Fonte: Arquivo pessoal.

O diagrama de componente, representa as dependências do projeto, possibilitando ao desenvolvedor ter uma visão detalhada do sistema.

FIGURA 12 – Diagrama de Objeto



O diagrama de objeto, possibilita ao usuário final uma visão da estrutura dos dados, de forma mais familiar, apresentando valores utilizados pelo próprio usuário.

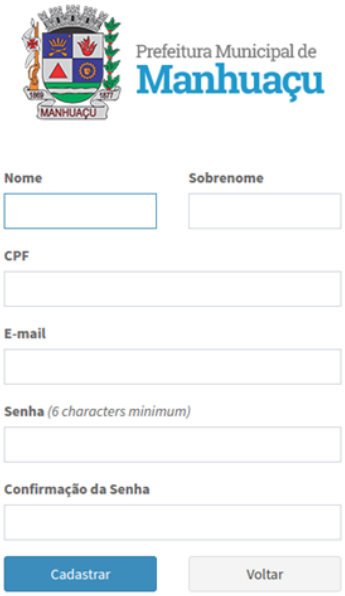
APÊNDICE B – Telas do Sistema

Ao carregar a página do sistema, o usuário será direcionado a tela de *login*, permitindo ao mesmo realizar o acesso ao sistema, a partir da combinação de *e-mail* e senha.

Figura 13 – Tela de *Login*

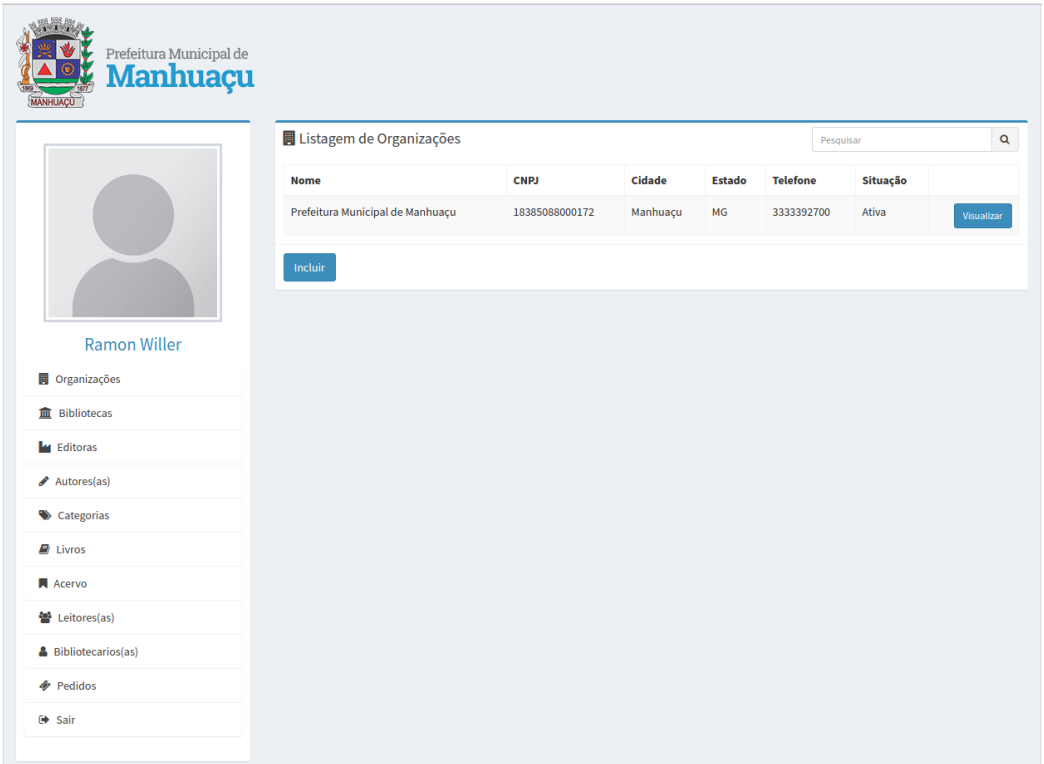
A tela de cadastro permite ao leitor se cadastrar no sistema, essa tela pode ser acessada através do botão *Sign up*.

Figura 14 – Tela de Cadastro



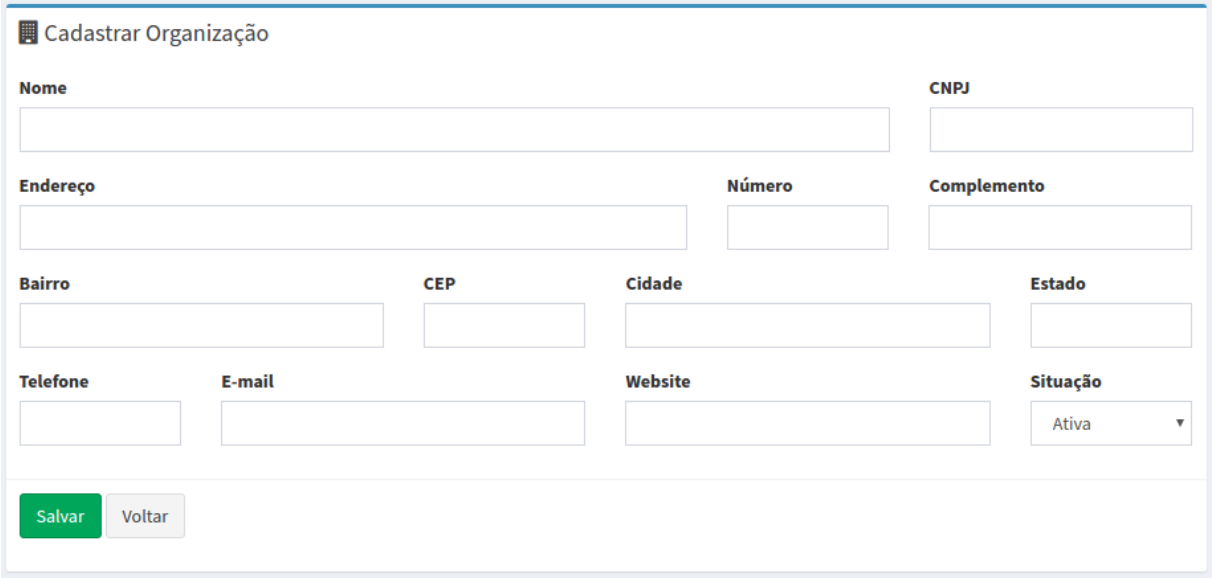
Depois de realizado o acesso ao sistema o usuário terá disponível o menu de opções do sistema, este *menu* sofre variações dependendo do perfil do usuário presente no sistema.

Figura 15 – Página Inicial



Todos os formulários do sistema são carregados à direita do sistema, sendo possibilitado ao usuário percorrer pelas páginas do sistema, criar novos registros e visualizar os registros cadastrados.

Figura 16 – Cadastro de Organização

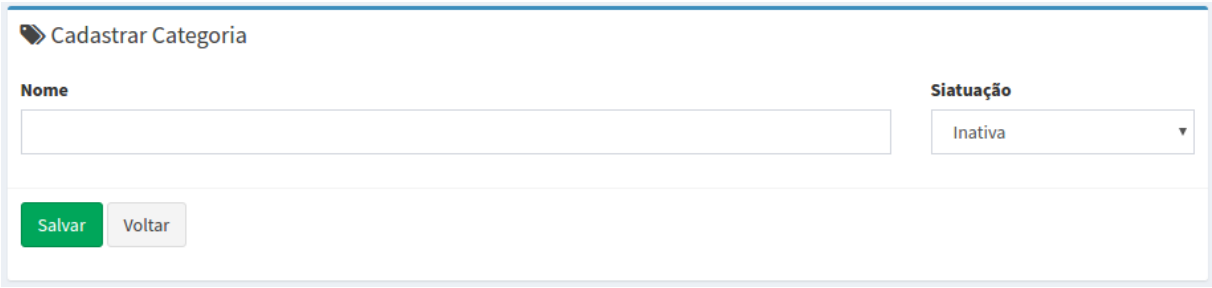


Os formulários de edição são baseados nos formulários de inclusão, portando a partir do formulário a seguir, serão apresentados apenas os formulários de inclusão.

Figura 17 – Edição da Organização

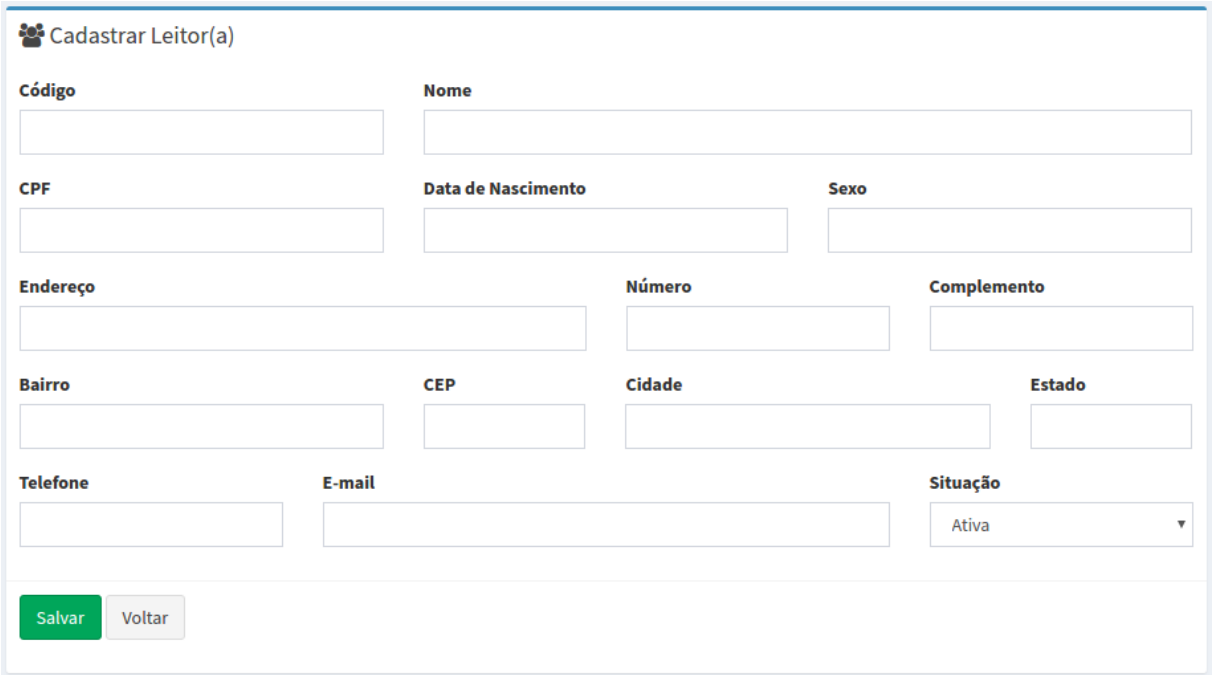
A edição de organização é realizada através do formulário acima, sendo permitido somente ao administrador do sistema.

Figura 18 – Cadastro de Categoria



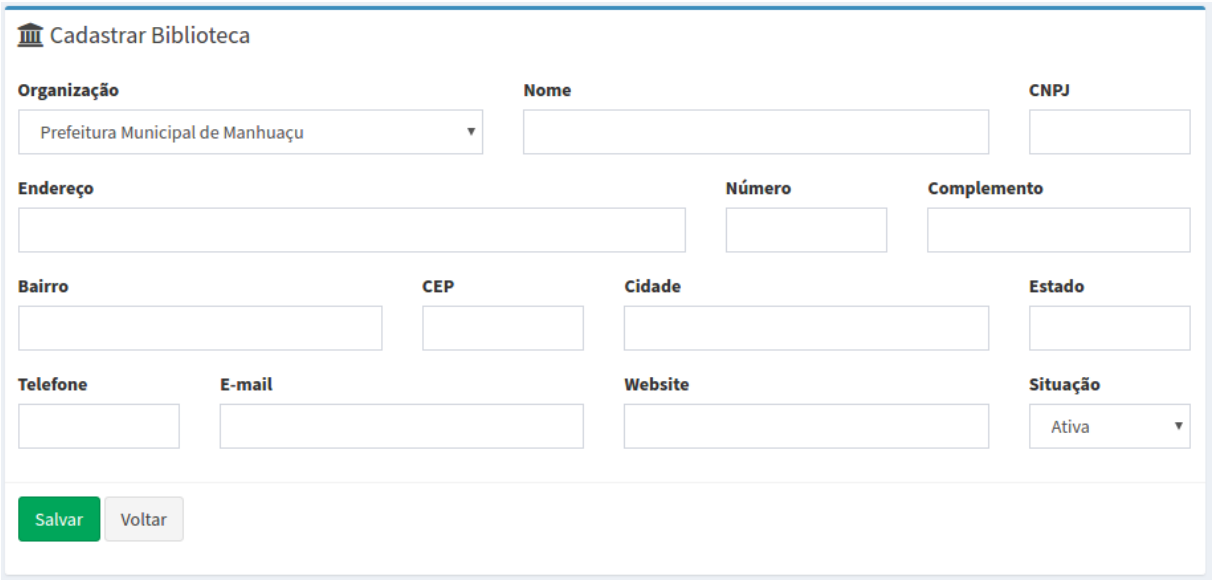
O cadastro de categoria é realizado através do formulário acima, sendo esse formulário disponível apenas para o bibliotecário e administrador do sistema. As categorias são utilizadas na classificação dos livros.

Figura 19 – Cadastro de Leitor



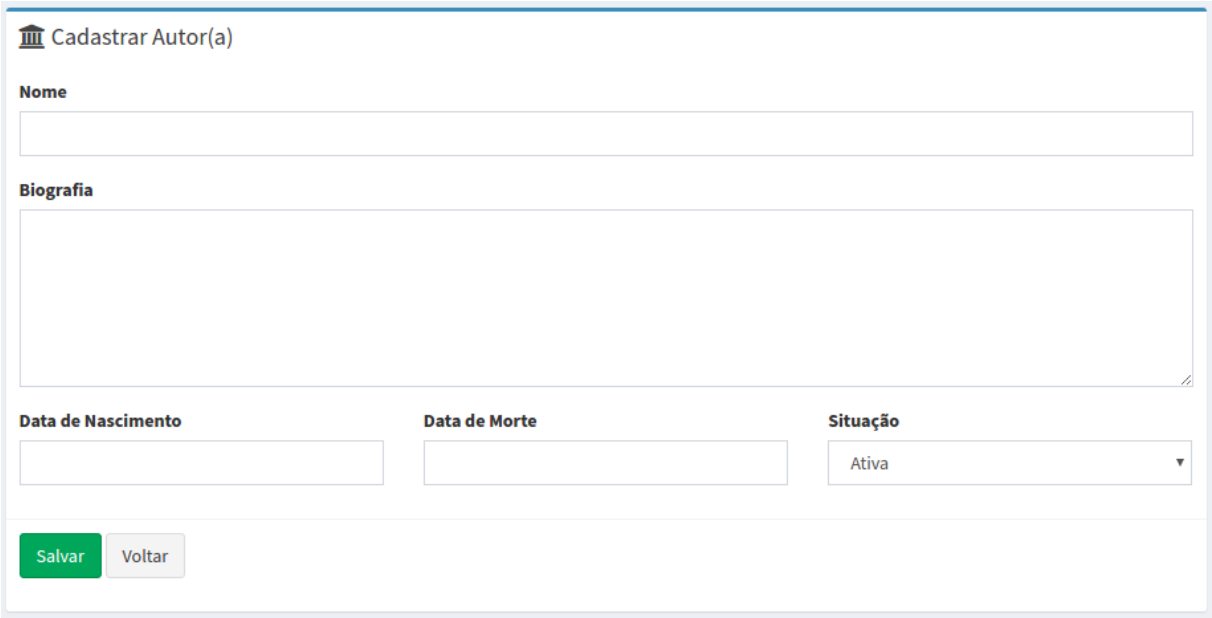
O cadastro do leitor, representado pelo formulário acima, está disponível ao bibliotecário e ao administrador, sendo utilizado como complementação as informações do usuário do sistema.

Figura 20 – Cadastro de Biblioteca



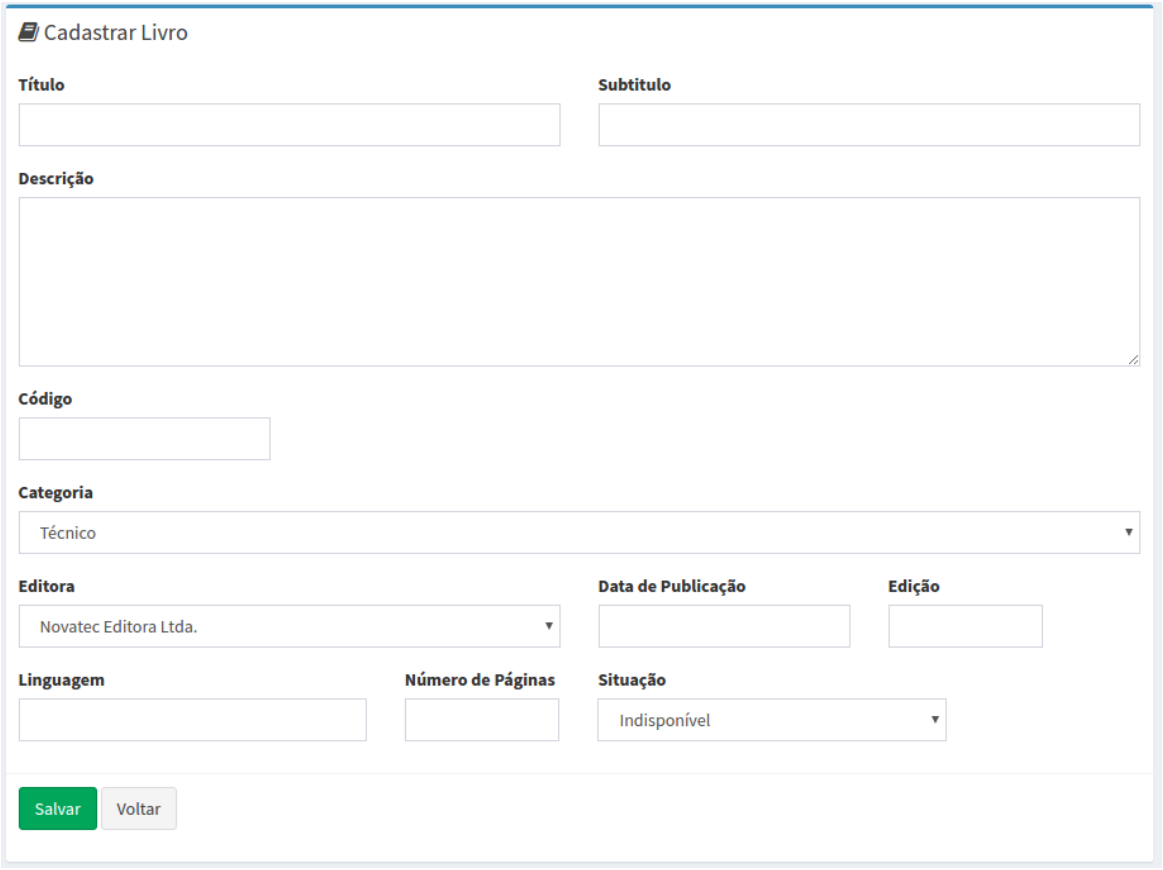
O formulário de cadastro de biblioteca, permite a inclusão de novas bibliotecas vinculadas a uma organização, sendo possível o cadastro de bibliotecas escolares e instituições independentes.

Figura 21 – Cadastro de Autor



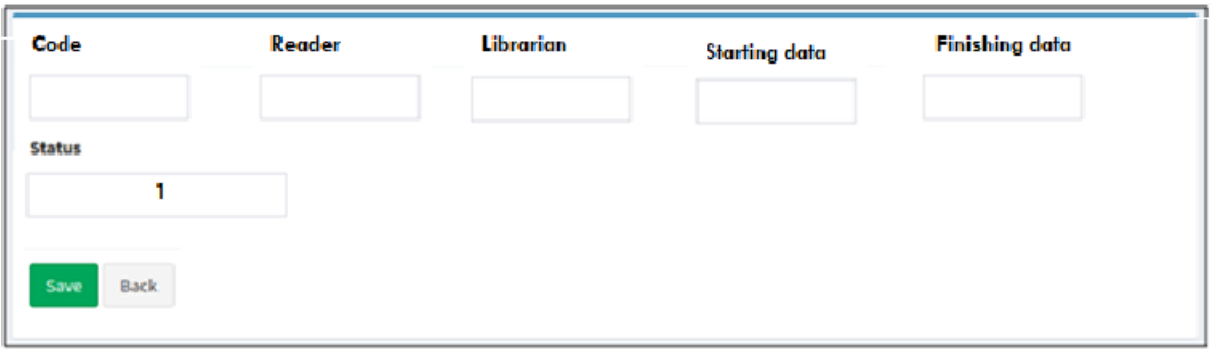
O cadastro de autor, permite a inclusão dos escritores das obras literárias, sendo informados os dados de nascimento e morte, além da sua biografia.

Figura 22 – Cadastro de Livros

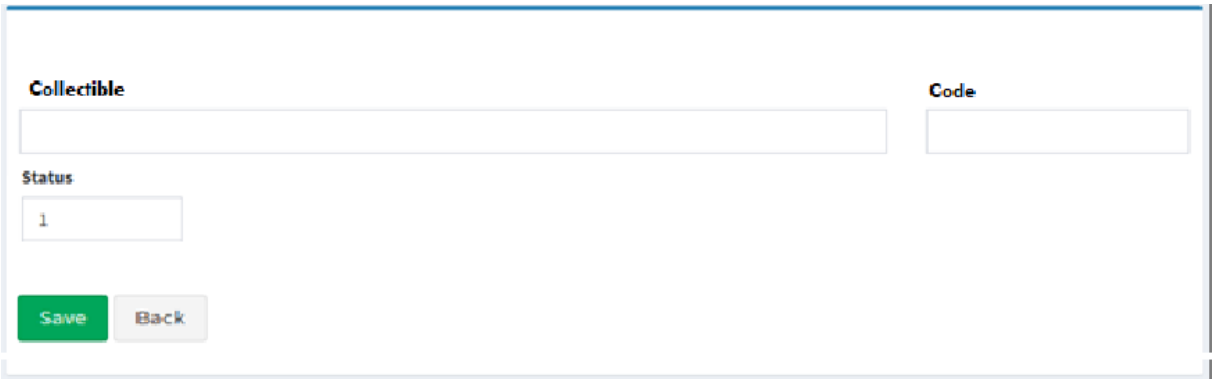


O cadastro de livros, permite a inclusão de obras literárias no sistema, sendo este cadastro disponível a todas as bibliotecas do sistema.

Figura 23 – Cadastro de Reserva



A inclusão da reserva, e realizada pelo formulário acima, sendo informado os dados do leitor, bibliotecário, início e fim da reserva.
Figura 22 – Itens da Reserva.



Os itens de uma reserva são incluídos através do formulário acima, sendo estes associados a um item do acervo de uma biblioteca.