UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
FACULDADE DE CIÊNCIAS EXATAS E TECNOLÓGICAS
SISTEMAS DE INFORMAÇÃO

DESENVOLVIMENTO DE SOFTWARE DE GESTÃO PARA A BIBLIOTECA DO CESEC JUSCELINO KUBITSCHEK DE OLIVEIRA DO MUNICÍPIO DE DIAMANTINA-MG

EDUARDO AUGUSTO COSTA TRINDADE

DIAMANTINA
2018

EDUARDO AUGUSTO COSTA TRINDADE

DESENVOLVIMENTO DE SOFTWARE DE GESTÃO PARA A BIBLIOTECA DO CESEC JUSCELINO KUBITSCHEK DE OLIVEIRA DO MUNICÍPIO DE DIAMANTINA-MG

Trabalho de Conclusão de Curso apresentada como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri - UFVJM, Faculdade de Ciências Exatas e Tecnológicas.

Orientador: Prof. Dr[a]. Claudia Beatriz Berti

Diamantina

2018

EDUARDO AUGUSTO COSTA TRINDADE

DESENVOLVIMENTO DE SOFTWARE DE GESTÃO PARA A BIBLIOTECA DO CESEC JUSCELINO KUBITSCHEK DE OLIVEIRA DO MUNICÍPIO DE DIAMANTINA-MG

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação do Universidade Federal dos Vales do Jequitinhonha e Mucuri - UFVJM, Faculdade de Ciências Exatas e Tecnológicas.

Data de Apresentação: 02/03/2018

Orientação

Prof. Drª. Claudia Beatriz Berti

Membros da Banca

Prof. Me. Cinthya Rocha Tameirão

Prof. Me. Marcelo Ferreira Rego

TRINDADE, Eduardo Augusto Costa. **Desenvolvimento de Software de Gestão para a Biblioteca do CESEC Juscelino Kubitschek de Oliveira do Município de Diamantina-MG**. Diamantina, 2018. 96. Trabalho de Conclusão de Curso Superior em Sistemas de Informação. Faculdade de Ciências Exatas, Universidade Federal dos Vales do Jequitinhonha e Mucuri. Diamantina, 2018.

Aos meus pais, por não medirem esforços
e nunca deixarem de acreditar em mim!

AGRADECIMENTOS

A Deus por me conceder o dom da vida com saúde e sabedoria em sua plenitude. Aos meus pais, Renato Trindade e Lourdes Trindade, por toda a motivação, amor e apoio incondicional perante às dificuldades. Ao meu irmão Marcelo Trindade e familiares pelos anos de convivência e aconselhamentos.

Agradeço à minha namorada Laís Santos pelo incentivo, amor, ajuda e estar ao meu lado em todos os momentos. Aos meus amigos Cadu Guedes, Grhégory Lima, Michel Cordeiro, Lucas Campos e Hugo Carvalho pela amizade, convivência e companheirismo ao longo destes anos.

À minha orientadora Claudia Berti por sua paciência, dedicação, disponibilidade e conhecimento, fundamentais nesta etapa. A todos aqueles que direta ou indiretamente contribuíram para minha formação, como o corpo docente e técnico do Departamento de Computação da UFVJM, amigos do IPSEMG, SEPLAG, INSS, CESEC e ATS durante períodos de estágio e trabalho.

A todos vocês, meu muito obrigado!

Se fosse fácil achar o caminho das pedras, tantas pedras no caminho não seria ruim.

Humberto Gessinger

RESUMO

A modernização das bibliotecas está diretamente ligada a automação de rotinas e serviços, com a finalidade de implantar uma infraestrutura de comunicação para agilizar e ampliar o acesso à informação pelo usuário. Para isso é necessário haver uma ampla visão da tecnologia da informação e sua aplicação nas organizações. Este trabalho apresenta o desenvolvimento de um sistema web cujo objetivo principal é gerenciar o acervo de livros da Biblioteca do CESEC Juscelino Kubitschek de Oliveira, de Diamantina, MG. Atualmente, a biblioteca não dispõe de um controle de seu acervo, e os empréstimos são registrados manualmente pelos bibliotecários, limitando seu acesso e controle. Para auxiliar o desenvolvimento do sistema foi utilizada a metodologia ágil Scrum. Com o sistema espera-se contribuir para o trabalho dos bibliotecários e possibilitar todo o registro e controle do acervo de livros da biblioteca, registrando suas movimentações de entrada e saída.

Palavras-chave: Biblioteca, *web*, Scrum, Metodologia ágil, Engenharia de Software.

ABSTRACT

The modernization of libraries is directly linked to the automation of routines and services, in order to deploy a communication infrastructure to streamline and expand access to information by the user. This requires a broad vision of information technology and its application in organizations. This work presents the development of a web system whose main objective is to manage the book collection of the CESEC Library Juscelino Kubitschek de Oliveira, from Diamantina, MG. Currently, the library does not have a control of its collection, and the loans are manually registered by the librarians, limiting their access and control. To support the development of the system, the agile Scrum methodology was used. The system is expected to contribute to the work of the librarians and enable all the registration and control of the library's collection of books, recording their movements of entry and exit.

Palavras-chave: Library, *web*, Scrum, agile Methodology, Software Engineering.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CESEC	Centro Estadual de Educação Continuada
TU	Teste de Unidade
PC	Personal Computer
TV	Abreviação de televisão
XP	Extreme Programming
HTTP	Hypertext Transfer Protocol
UML	Unified Modeling Language
HTML	HyperText Markup Language
XML	EXtensible Markup Language
PHP	PHP: Hypertext Preprocessor
SQL	Structured Query Language
CSS	Cascading Style Sheets
JS	JavaScript
API	Application Programming Interface
CGI	Common Gateway Interface
OOP	Object-oriented programming
SGBD	Sistema Gerenciador de Banco de Dados
SGBDR	Sistema Gerenciador de Banco de Dados Relacional

SUMÁRIO

Capítulo 1

INTRODUÇÃO

Com o avanço das tecnologias de informação, e a constante necessidade do ser humano em se organizar, encontra-se um cenário amplo para o desenvolvimento de trabalhos de informatização e a inserção de práticas de gestão por meio de ferramentas e sistemas de informação. A Internet tornou-se um importante meio de comunicação e aprendizagem, sendo possível conciliar sua usabilidade com o desenvolvimento e a utilização de sistemas *web*, que beneficiam a prática de gestão em qualquer segmento que seja aplicado.

Neste trabalho será apresentado o desenvolvimento de um sistema *web* para uso em uma biblioteca de ensino, para otimizar as práticas de gestão. O produto de *software* resultante deste trabalho será doado ao CESEC Juscelino Kubitschek de Oliveira do Município de Diamantina-MG que atualmente controla e cataloga seu grande acervo bibliográfico de maneira totalmente manual.

O CESEC - Centro Estadual de Educação Continuada, é uma escola mantida pelo Governo do Estado, com certificado válido em todo o território nacional, que lida com a educação de jovens e adultos que, por alguma razão, não concluíram o ensino fundamental ou médio na idade própria, oferecendo a estes uma oportunidade de concluírem seus estudos em um período abreviado. Com uma carga horária acessível, encontra-se disponível para alunos do ensino fundamental a partir dos 15 anos, e alunos do ensino médio a partir dos 18 anos. O atendimento ocorre de segunda a sexta-feira em horários flexíveis, de acordo com a disponibilidade do aluno e horários do professor. Em Diamantina, encontra-se atualmente situado à Praça Dom Joaquim, nº 76, Centro (Figura 1.0.1).

O trabalho foi desenvolvido utilizando o método ágil Scrum, a fim de comprovar sua eficácia e validar seu uso em ambientes de desenvolvimento de tecnologias educacionais.

A escolha da aplicação *web* não surgiu apenas pela não necessidade de instalação e

Figura 1.0.1 – Sede atual do CESEC em Diamantina, MG

Fonte: https://www.youtube.com/watch?v=x2nRbQPOtCE

manutenção local, ou pela facilidade de acesso e disponibilidade do serviço, mas pela possibilidade de uma nova experiência de uso aos usuários. O sistema *web* proposto e desenvolvido oferecerá, além de uma nova experiência para as bibliotecárias do CESEC JK, maior tranquilidade no trabalho e praticidade com os conteúdos armazenados digitalmente - necessidade já sinalizada pelos funcionários da escola.

1.1 JUSTIFICATIVA E MOTIVAÇÃO

Após trabalhar no CESEC como Professor de Educação Básica do Curso Técnico em Informática, oferecido pelo PRONATEC, ministrando disciplinas como Programação para a Internet, Redes de Computadores e Banco de Dados, durante aproximadamente 12 meses, convivendo com a rotina escolar, alunos e servidores quase que diariamente, foi possível conhecer as estruturas da escola e identificar algumas rotinas que poderiam ser melhoradas.

No CESEC o aluno cursa de modo semi-presencial, ou seja, o aluno não cumpre carga horária, apesar de ser registrado sua presença para controle da escola, sendo necessário que o aluno vá à escola para que o professor o oriente quanto ao conteúdo de estudos e seja tirado

suas dúvidas referente à matéria estudada, preparando-o assim para submeter-se a avaliação chamada Teste de Unidade (T.U.)[1], ao qual o aluno está autorizado a prestar mediante autorização do professor. As matrículas ocorrem durante todo o ano.

Toda essa flexibilidade permitiu observar também, que os horários da biblioteca nem sempre condizem com o período em que o aluno ou professor se encontram na escola. Com isso, muitas vezes, no período noturno (horário ao qual estive lotado no quadro funcional da escola), percebeu-se uma indisponibilidade nos serviços da biblioteca, uma vez que o horário dos bibliotecários é flexível e os mesmos podiam não estar presentes ou dispôr de informações referentes ao empréstimo de livros dentro da própria escola, totalmente dependente de anotações e serviços manuais feitos pelos mesmos.

Este trabalho justifica-se pela necessidade de um controle automatizado do acervo bibliotecário. Necessidade notada durante os meses de trabalho e declarada pelos funcionários do setor.

Além disso, a realização deste projeto poderá incentivar mais experiências como essa dentro do município e da Universidade, oferecendo um trabalho eficaz, de qualidade e que atinja os requisitos necessários ao bom desempenho de um setor da sociedade.

O sistema *web* de gestão para biblioteca se caracterizará pela principal atividade dos bibliotecários da escola que é o controle do acervo de livros do local. Diferente de outros sistemas com este propósito, os bibliotecários, por meio de entrevistas e pelo Questionário para Levantamento dos Requisitos aplicado (ver Apêndice B), optaram por não permitir que o cadastro de alunos ou outros usuários no sistema fosse realizado por eles mesmos, desejando assim centralizar todo o controle em suas mãos, seja no cadastro ou no empréstimo dos livros. Tais características podem garantir à Biblioteca do CESEC JK um primeiro e importante passo no sentido de informatizar trabalhos gerenciáveis, reforçados pela ideia que os próprios funcionários desejam ingressar neste caminho.

1.2 OBJETIVOS

Os objetivos deste trabalho são divididos em duas sessões. A seguir são apresentadas cada uma delas.

[1] Fonte: http://cesecgv.blogspot.com.br/2012/08/como-estudar-no-cesec.html.

1.2.1 Objetivo geral

Oferecer ao CESEC JK o controle de sua biblioteca por meio de um sistema *web*, de modo que o bibliotecário possa administrar o acervo de livros e situações de empréstimos automaticamente. Atualmente este controle é feito manualmente.

1.2.2 Objetivos específicos

- Compreender a importância da Engenharia de *Software* e do planejamento no desenvolvimento de um sistema para *web*.
- Levantar um acervo de estudo a respeito de programação para a *web*, aprendendo a utilizar melhor as linguagens de programação propostas, *frameworks* e as ferramentas a serem utilizadas como IDE.
- Implementar o método ágil Scrum, a fim de compreender, testar e comprovar sua eficácia, por vezes já definida como eficaz em ambientes de desenvolvimento de *software*.
- Desenvolver banco de dados, diagramas e gráficos que possibilitem auxilio no desenvolvimento e visualização do sistema.
- Implementar um sistema *web* para uso bibliotecário com os conceitos e aprendizados citados anteriormente e que atenda aos requisitos levantados.

1.3 ESTRUTURA DO TRABALHO

Este trabalho está dividido em nove capítulos.

O Capítulo dois é fundamentado o referencial teórico que contribuiu no estudo deste trabalho, apresentando desde os princípios da Engenharia de *Software* às Aplicações para a *web* desenvolvidas por metodologias ágeis.

O Capítulo três apresenta trabalhos relacionados ao tema proposto, modelos de *software* desenvolvidos para fins semelhantes, metodologias aplicadas que contribuíram tanto para a tomada de decisões quanto a compreensão de ambientes organizacionais.

O Capítulo quatro descreve as ferramentas, linguagens utilizadas no desenvolvimento deste trabalho e a metodologia aplicada neste desenvolvimento, o Scrum. Este capítulo apresenta ainda as práticas fundamentais do Scrum e análises com o ambiente.

O Capítulo cinco apresenta desenvolvimento aplicando a metodologia Scrum, com resultados e análises de cada etapa.

O Capítulo seis apresenta o resultado visual e final da metodologia e desenvolvimento. Telas do sistema são exibidas a fim de demonstrar resultados concretos dos objetivos propostos e atendidos. Por fim, conclui-se no Capítulo sete a importância de uma metodologia no desenvolvimento de qualquer atividade, deixando sugestões futuras para melhorias do sistema.

Os Apêndices trazem o Termo de Abertura do Projeto (A), o Questionário para levantamento de requisitos (B) utilizado, um Modelo de Quadro Scrum (C) para aplicação que esclarece alguns conceitos apresentados no trabalho, algumas das telas mais importantes do sistema (D) e o *script* PHP e SQL para empréstimo de livro desenvolvido (E).

Capítulo 2

REFERENCIAL TEÓRICO

Este capítulo apresenta conceitos e características importantes sobre todo o conteúdo a ser abordado, que serve de base para o desenvolvimento do trabalho.

2.1 ENGENHARIA DE *SOFTWARE*

Compreender o significado de um *software* envolve muitas definições. O *software* consiste em instruções (programas de computador) que, quando executadas, fornecem características, funções e desempenho desejados, bem como estruturas de dados que possibilitam aos programas manipularem informações adequadamente, conforme relata Pressman (2011).

Segundo Pressman (2011), o *software* distribui o produto mais importante da nossa era, a informação. Essas e muitas outras questões demonstram a preocupação com o *software* e a maneira como é desenvolvido, o que tem levado à adoção da prática da engenharia de *software*.

Independente da necessidade da aplicação, o *software* precisa ser cuidadosamente estudado antes de sua produção. Para Sommervile (2011), engenharia de *software* é uma disciplina de engenharia cujo foco está em todos os aspectos da produção de *software*, desde os estágios iniciais da especificação do sistema até sua manutenção, quando o sistema já está sendo usado.

Além disso, a engenharia de *software* se preocupa com o *software* como produto. Estão fora de seu escopo programas feitos unicamente por diversão do programador. “Estão fora também pequenos programas descartáveis, feitos por alguém exclusivamente como meio para resolver um problema dessa pessoa, e que não serão utilizados por outros” (FILHO, 2009, p. 5).

2.1.1 Requisitos do Sistema

Segundo Sommerville (2011), os requisitos de um sistema são as descrições do que o sistema deve executar, os serviços que oferece e as restrições a seu funcionamento. Esses requisitos refletem as necessidades dos clientes para um sistema que serve a uma finalidade determinada, como controlar um dispositivo, colocar um pedido ou encontrar informações.

Os requisitos de software são frequentemente classificados como requisitos funcionais e não funcionais.

2.1.1.1 Requisitos funcionais

Os requisitos funcionais de um sistema descrevem os serviços que ele deve fornecer. Eles dependem do tipo de *software* a ser desenvolvido, de quem são seus possíveis usuários e da abordagem geral adotada pela organização ao escrever os requisitos (SOMMERVILLE, 2011).

Em princípio, para Sommerville (2011), a especificação dos requisitos funcionais de um sistema deve ser completa e consistente. Completude significa que todos os serviços requeridos pelo usuário devem ser definidos. Consistência significa que os requisitos não devem ter definições contraditórias. Na prática, para sistemas grandes e complexos, é praticamente impossível alcançar completude e consistência dos requisitos. Todavia, uma vez que o sistema bibliotecário aqui apresentado não exige tamanha complexidade, é perfeitamente possível definir os requisitos funcionais do sistema.

2.1.1.2 Requisitos não funcionais

Os requisitos não funcionais, como desempenho, proteção ou disponibilidade, normalmente especificam ou restringem as características do sistema como um todo.

Sommerville (2011) define requisitos não funcionais da seguinte forma:

> Os requisitos não funcionais são requisitos que não estão diretamente relacionados com os serviços específicos oferecidos pelo sistema a seus usuários. Eles podem estar relacionados às propriedades emergentes do sistema, como confiabilidade, tempo de resposta e ocupação de área. (SOMMERVILLE, 2011, p. 74).

É recomendado que os requisitos não funcionais devam ser escritos quantitativamente, para que possam ser objetivamente testados. A Tabela 2.1.1 mostra as métricas que podem ser utilizadas para especificar as propriedades não funcionais do sistema. Sommerville (2011) explica que pode-se medir essas características quando o sistema está sendo testado para verificar se ele tem cumprido ou não seus requisitos não funcionais.

Tabela 2.1.1 – Métricas para especificar requisitos não funcionais

Propriedade	Medida
Velocidade	Transações processadas/segundo. Tempo de resposta de usuário/evento. Tempo de atualização de tela.
Tamanho	Megabytes. Número de chips de memória ROM.
Facilidade de uso	Tempo de treinamento. Número de *frames* de ajuda.
Confiabilidade	Tempo médio para falha. Probabilidade de indisponibilidade. Taxa de ocorrência de falhas. Disponibilidade.
Robustez	Tempo de reinício após falha. Percentual de eventos que causam falhas. Probabilidade de corrupção de dados em caso de falha.
Portabilidade	Percentual de declarações dependentes do sistema-alvo. Número de sistemas-alvo.

Fonte: (SOMMERVILLE, 2011, p. 63, adaptado)

2.2 APLICAÇÕES PARA A *WEB*

Um dos maiores avanços da era atual é sem dúvidas, a Internet. No início, ela era basicamente um armazenamento de informações acessível universalmente e tinha pouco efeito nos sistemas de *software*. Esses sistemas executavam em computadores locais e eram acessíveis apenas dentro da organização. De acordo com Sommervile (2011), baseado nisso, o avanço da internet contribuiu diretamente para que sistemas *web* pudessem ser desenvolvidos e que, em vez de ter uma interface de usuário específica, poderiam ser acessados por um navegador. Isso levou ao desenvolvimento de uma enorme quantidade de novos produtos de *software* que ofereciam serviços inovadores e que eram acessados através da internet.

Com essa evolução, desenvolvedores conseguiram suprir algumas dificuldades apresentadas nos primeiros *software*. Além disso, a preocupação em manter o nível de qualidade e disponibilidade de serviços, permitiu que os engenheiros pudessem dividir o *software* em categorias, para melhor poderem trabalhar em seus desenvolvimentos.

Paralelamente a este desenvolvimento, Sommervile (2011) afirma que o desenvolvimento de navegadores *web* capazes de executar programas pequenos e fazer algum processamento local levou a uma evolução no *software* corporativo e organizacional. Em vez de escrever o *software* e instalá-lo nos computadores dos usuários, o *software* era implantado em um servidor *web*.

A esse tipo de *software*, Pressman (2011) define como aplicações para *web*:

> [...] essa categoria de *software* centralizada em redes abarca uma vasta gama de aplicações. Em sua forma mais simples, as *web Apps* podem ser pouco mais

> que um conjunto de arquivos de hipertexto interconectados, apresentando informações por meio de texto e informações gráficas limitadas. Entretanto, com o aparecimento da *web* 2.0, elas têm evoluído e se transformado em sofisticados ambientes computacionais que não apenas fornecem recursos especializados, funções computacionais e conteúdo para o usuário final, como também estão integradas a banco de dados corporativos e aplicações comerciais. (PRESSMAN, 2011, p. 35)

Não obstante, o próximo estágio no desenvolvimento de sistemas *web* foi a noção de *web services*, como define Sommerville (2011).

> *Web services* são componentes de *software* acessados pela Internet e fornecem uma funcionalidade específica e útil. Aplicações são construídas integrando esses *web services*, os quais podem ser fornecidos por empresas diferentes. A princípio, essa ligação pode ser dinâmica, para que a aplicação possa usar *web services* diferentes toda vez que é executada. (SOMMERVILE, 2011, p. 8)

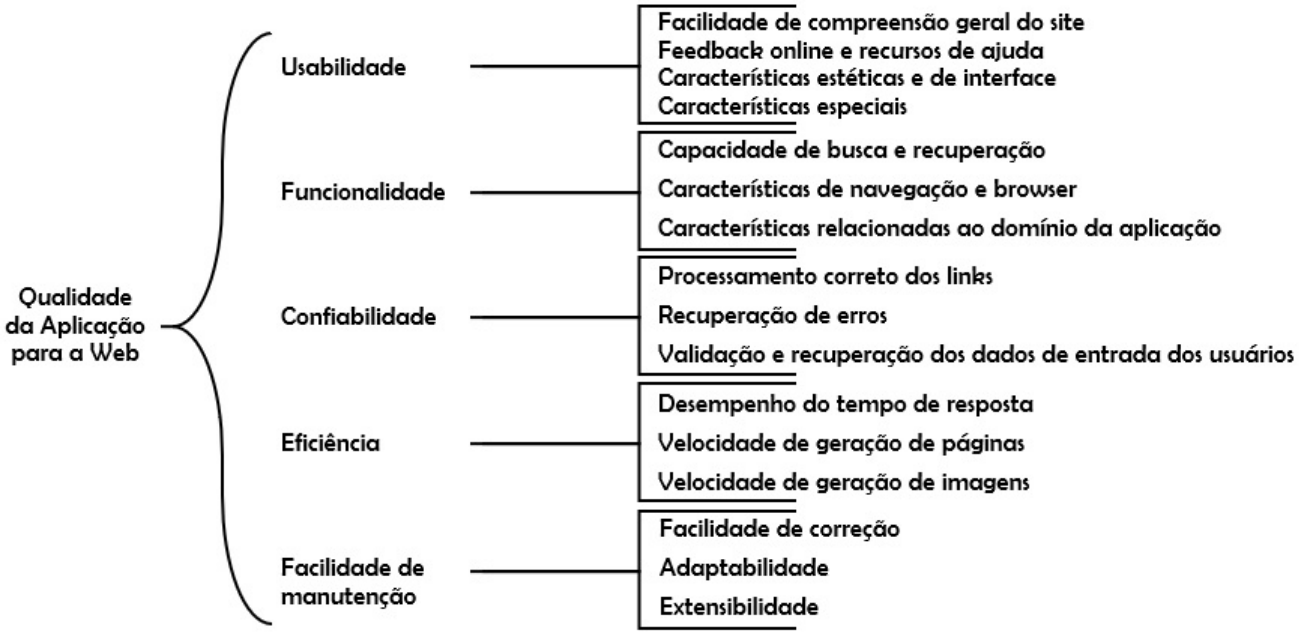
2.2.1 A Qualidade nas Aplicações para *web*

Com diversas opções por processos de desenvolvimento de *software* disponíveis, a preocupação com a qualidade do projeto é constante. Embora a percepção do usuário sempre varie, resultando na aceitação ou rejeição da aplicação, trabalhar a qualidade de um *web App*[1] é importantíssimo para conseguir atender às funções propostas pelo projeto.

Segundo Pressman (2009, p. 339, *apud* Olsina, 1999) uma árvore de requisitos técnicos identifica um conjunto de atributos técnicos – usabilidade, funcionalidade, confiabilidade, eficiência e facilidade de manutenção – que levam *web* Apps de alta qualidade. Juntos, fornecem uma base útil para se avaliar a qualidade de sistemas baseados na *web*. A Figura 2.2.1 sintetiza o trabalho destes pesquisadores, identificando características para cada atributo técnico.

[1] Conjunto de páginas *web* interligadas com funções computacionais, com as quais os usuários interagem através de browsers (navegadores) (SOUZA, 2005).

Figura 2.2.1 – Árvore de Requisitos de Qualidade



Fonte: (PRESSMAN, 2011, p. 340, adaptado)

Paralelamente relacionado à qualidade da aplicação para *web*, a segurança é fator importantíssimo. Para Pressman (2011), à medida que a crítica sobre os sistemas e aplicações *web* aumentam, a segurança da informação tem se tornado cada vez mais importante. Ou seja, um *software* que não apresente alta qualidade é mais fácil de ser copiado e, como consequência, o *software* de baixa qualidade pode aumentar indiretamente o risco de segurança, assim como todos os problemas e custos associados.

Sendo assim, a qualidade de *software* não aparece simplesmente do nada. Ela é o resultado de um bom gerenciamento de projeto e uma prática consistente de engenharia de *software*, conforme afirma Pressman (2011). Por menor que seja o *software* ou o projeto envolvido, um mínimo de gerenciamento e adoção de boas práticas de qualidade para aplicações *web* ampliam a chance de se obter um produto satisfatório, eficiente, seguro e de qualidade no final.

2.3 ENGENHARIA DE *SOFTWARE* APLICADA EM *WEB APPS*

Com a implementação de metodologias de engenharia focadas nas características das aplicações *web* é possível obter uma melhor estrutura no desenvolvimento de *web Apps*, e

assim permitir a redução de custo de manutenção destas (SOMMERVILLE, 2011).

De acordo com Pressman (2011, p. 50), o desenvolvimento para *web* acelerou de maneira incontrolável, sem nenhuma estruturação ou planejamento. Assim, "para evitar que o desenvolvimento na *web* se torne caótico, é necessário que exista uma estrutura e planejamento, que proporcione sustentabilidade ao desenvolvimento de aplicativos *web*, baseados nos conceitos da própria Engenharia de *software*".

Neste sentido, Sommervile (2011) afirma que os *web Apps* trouxeram mudanças radicais na organização de *software*, o que obviamente causou mudanças na maneira como os sistemas *web* são projetados. Por exemplo:

1. O reuso de *software* tornou-se a abordagem dominante para a construção de sistemas *web*. Quando construímos esses sistemas, pensamos em como podemos montá-los a partir de componentes e sistemas de *software* preexistentes.

2. Atualmente, aceita-se que é impraticável especificar todos os requisitos para tais sistemas antecipadamente. Sistemas *web* devem ser desenvolvidos e entregues incrementalmente.

3. Interfaces de usuário são restringidas pela capacidade dos navegadores. Formulários *web* com *scripts* locais são mais usados. Interfaces das aplicações em sistemas *web* são normalmente mais "pobres" do que interfaces projetadas especialmente para produtos de *software* que executam em PCs. De modo a trabalhar essas mudanças e planejar o desenvolvimento dos *web Apps*, juntamente com os conceitos da engenharia de *software*, adaptando-os a cada necessidade, surge a Engenharia *web* (*webE*).

2.4 ENGENHARIA *WEB*

Segundo Lowe et al. (2009, p. 15), "a engenharia *web* compreende o panorama inteiro das práticas e fluxo de processo da engenharia de *software*, mas de uma maneira que se adapte e retifique o processo e cada prática aos atributos e características especiais das *web Apps*. A engenharia *web* propõe um arcabouço ágil, porém disciplinado para a montagem de *web* Apps de qualidade industrial". Os termos **arcabouço** e **ágil** são tratados a partir daqui como chave para interligarmos a engenharia *web* à engenharia de *software*.

Pressman et al. (2009) sustentou uma posição que acopla engenharia *web* e engenharia de *software*:

> *Praticamente qualquer produto ou sistema importante merece engenharia. Antes que você comece a construí-lo, é melhor entender o problema, projetar uma solução funcional, implementá-la de uma forma sólida e testá-la totalmente. Você provavelmente também deve controlar as mudanças enquanto trabalha e ter algum mecanismo para garantir a qualidade do resultado final. Muitos desenvolvedores web não discutem isso; eles simplesmente acham que seu mundo é realmente diferente e que as técnicas convencionais de engenharia de software simplesmente não se aplicam. (PRESSMAN, Roger, LOWE, David. 2009, p. 16)*

Com isso, Pressman claramente argumentou que os princípios, conceitos e métodos da engenharia de *software* podem ser aplicados ao desenvolvimento *web*, mas sua aplicação requer uma técnica de certa forma diferente de seu uso.

2.4.1 O Arcabouço

Um arcabouço estabelece o alicerce para um processo completo de engenharia *web*, identificando um pequeno número de atividades de arcabouço que se aplicam a todos os projetos de *web Apps*, independentemente de seu tamanho e complexidade (LOWE et al., 2009.). Lowe et al. (2009) ainda definem cinco principais atividades de engenharia *web* que fazem parte de um arcabouço genérico e se aplicam à grande maioria dos projetos *web App*, conforme Figura 2.4.1:

Figura 2.4.1 – Fluxo do Processo com ações de *webE*

Fonte: (Lowe et al. 2009, p.25)

As atividades deste ciclo, são explicadas por pelos autores da seguinte maneira:

- **Comunicação**: envolve interação e colaboração intensas com o cliente (e outros interessados), e abrange o levantamento de requisitos e outras atividades relacionadas.

- **Planejamento**: estabelece um plano incremental[2] para o trabalho de engenharia *web*. Ele descreve as ações de *webE* que ocorrerão, as tarefas técnicas a serem realizadas, os riscos que são prováveis, os recursos que serão exigidos, os produtos de trabalho a serem produzidos e um cronograma de trabalho.

- **Modelagem**: abrange a criação de modelos que auxiliam o desenvolvedor e o cliente a entenderem melhor os requisitos de *web* App e o projeto que satisfará esses requisitos.

- **Construção**: combina a geração de HTML, XML Java e código semelhante, juntamente com o teste necessário para revelar erros no código.

- **Implantação**: entrega um incremento[3] de *web App* ao cliente, que o avalia e oferece com base na avaliação.

2.5 UML

A UML, que significa Linguagem de Modelagem Unificada (do inglês, UML - *Unified Modeling Language*) é uma linguagem-padrão para a elaboração da estrutura de projetos de *software*. Ela define um número de diagramas que permite dirigir o foco para aspectos diferentes do sistema de maneira independente (BOOCH et al., 2012). Se bem usados, os diagramas facilitam a compreensão do sistema que está sendo desenvolvido.

Segundo Silva et. al. (2001), trata-se de "uma linguagem para especificação, construção, visualização e documentação de artefatos de um sistema de *software*". A UML permite que a equipe de desenvolvedores de sistemas visualize o produto de seus trabalhos em diagramas padronizados.

Booch (2012) define no Guia do Usuário da seguinte forma:

> A UML é adequada para a modelagem de sistemas, cuja abrangência poderá incluir desde sistemas de informação corporativos a serem distribuídos a aplicações baseadas na *web* e até sistemas complexos embutidos de tempo real. É uma linguagem muito expressiva, abrangendo todas as visões necessárias ao desenvolvimento e implantação desses sistemas (BOOCH et al., 2012).

[2] Para Lowe e Pressman, um plano incremental assume que a *web App* deve ser entregue em uma série de "incrementos" que oferecem conjuntos de requisitos sucessivamente mais robustos a cada entrega.

[3] Lowe e Pressman definem que um incremento de *web App* entrega conteúdo e funcionalidade selecionados ao usuário final.

Entretanto, vale ressaltar que não estamos lidando com uma metodologia de desenvolvimento, o que significa que ela não diz para você o que fazer primeiro e em seguida ou como projetar seu sistema, mas ela lhe auxilia a visualizar seu desenho e a comunicação entre os objetos.

Neste trabalho serão apresentados dois tipos de diagramas utilizados, o diagrama de casos de uso e o de classes.

2.5.1 Diagrama de Casos de Uso

O diagrama de casos de uso é uma demonstração gráfica das funcionalidades oferecidas pelo sistema, sendo constituído por atores, casos de uso e os relacionamentos entre eles. O objetivo principal deste diagrama é demonstrar as funções do sistema que são utilizadas por um ator, visando auxiliar a comunicação entre o analista e o cliente.

Pender (2004 *apud* Cheli 2013) resume:

> O diagrama de caso de uso modela as expectativas dos usuários para usar o sistema. As pessoas e os sistemas que interagem com o sistema alvo são chamados de atores, e os recursos do sistema que os atores utilizam são chamados de casos de usos. Alguns casos de uso interagem com outros casos de uso, em relacionamento modelado por meio de setas de dependência (CHELI, 2013, p. 18).

Na Figura 2.5.1 a definição de alguns elementos do diagrama:

Figura 2.5.1 – Definição de figuras do diagrama de casos de uso

Fonte: (https://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408)

2.5.2 Diagrama de Classes

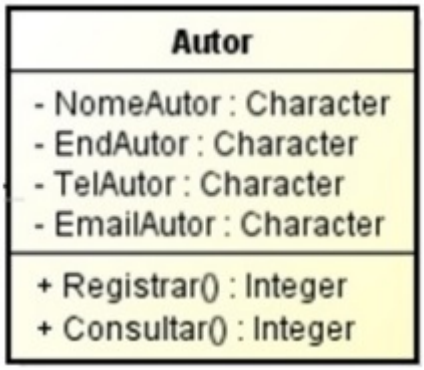
Considerado um dos diagramas mais importantes da linguagem de modelagem UML, o diagrama de classes apresenta uma visão da estrutura do sistema, servindo de origem tanto para geração quanto conversão do código para o modelo.

Para Booch (2005 *apud* Cheli 2013):

> Os diagramas de classes são os diagramas encontrados com maior frequência na modelagem de sistemas orientados a objetos. Um diagrama de classes mostra um conjunto de classes, interfaces e colaborações e seus relacionamentos. [...] São usados para fazer a modelagem estática do projeto de um sistema. São importantes não só para a visualização, a especificação e a documentação de modelos estruturais, mas também para a construção de sistemas executáveis por intermédio de engenharia de produção e reversa. (CHELI, 2013. p. 20)

O diagrama de classes é representado por um retângulo que contém três compartimentos empilhados verticalmente, como é mostrado na Figura 2.5.2. O compartimento superior mostra o nome da classe. O compartimento do meio lista os atributos da classe. Já o compartimento inferior lista as operações da classe.

Figura 2.5.2 – Exemplo de classe UML



Fonte: (GUEDES, 2006)

2.6 METODOLOGIAS ÁGEIS EM PROJETOS DE TI

A expressão “Metodologias Ágeis” tornou-se conhecida em 2001, quando especialistas em processos de desenvolvimento de *software* representando entre outros, os métodos Scrum e *Extreme Programming* (XP), definiram princípios e características comuns destes métodos. Na área da tecnologia da informação, metodologias ágeis possuem como objetivo acelerar o desenvolvimento de *software*, visando melhoria contínua no processo.

A abordagem ágil é muito utilizada em projetos orientados a valor. Estes projetos geralmente são realizados por profissionais do conhecimento e possuem elevado grau de incerteza, por grande indefinição do escopo e elevado número de mudanças (RIBEIRO et al., 2015). Os métodos ágeis diferem largamente no que diz respeito à forma de serem gerenciados, dependendo sempre do projeto em questão. Além disso, o desenvolvimento ágil tem pouco em comum com o modelo em cascata. O modelo em cascata é uma das metodologias com maior ênfase no

planejamento, seguindo seus passos através da captura dos requisitos, análise, projeto, codificação e testes em uma sequência pré-planejada e restrita, enquanto o método ágil abre novas portas para o desenvolvimento de um projeto, sempre baseado no lado adaptativo e contínuo (COSTA et al., 2007).

Na busca por seguir uma metodologia eficiente, existem inúmeros *frameworks* de processos para desenvolvimento de *software*. Embora apresentem diferenças em suas formas práticas, estão sempre compartilhando informações importantes e objetivos bem definidos, como a minimização de riscos, custo e tempo. Neste trabalho foi escolhido o método Scrum, auxiliado de algumas funcionalidades do modelo em cascata.

2.6.1 A Metodologia Scrum

Scrum é um método ágil empírico, iterativo e adaptativo. Por todas estas características é usado no desenvolvimento de sistemas de modo incremental, onde os requisitos sofrem constantes mudanças durante o ciclo de vida do produto, "resultando em uma abordagem que reintroduz as ideias de flexibilidade, adaptabilidade e produtividade" (SCHWABER; SUTHERLAND, 2017).

Schwaber et al. (2009 *apud* Leitao 2010) explicam que a metodologia Scrum "não é um processo ou uma técnica para a construção de produtos, mas sim um *framework* estrutural que está sendo empregado para suportar o desenvolvimento e gerenciar o trabalho em produtos complexos desde o início dos anos 90". Na verdade ele simplesmente fornece uma estrutura para entrega, mas não diz como fazer práticas específicas, deixando isso para a equipe de determinar, baseado na previsibilidade e o controle de riscos.

De acordo com o Guia Oficial do Scrum (SCHWABER; SUTHERLAND, 2017), três pilares apoiam a implementação de controle de processo empírico: transparência, inspeção e adaptação.

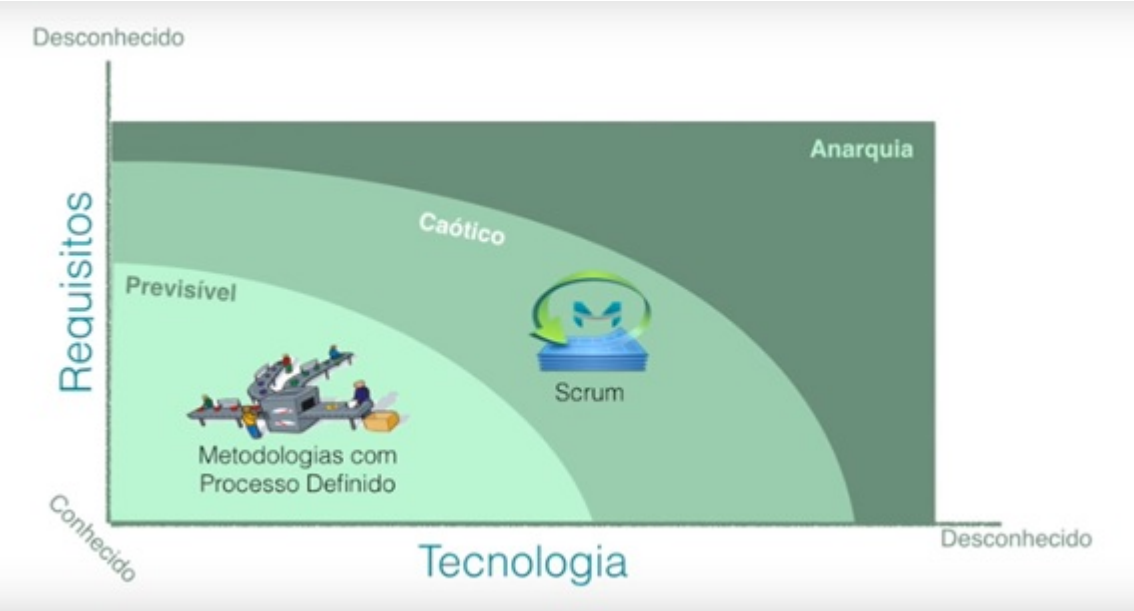
- **Transparência**: Aspectos significativos do processo devem estar visíveis aos responsáveis pelos resultados, com transparência dos processos, requisitos de entrega e status.
- **Inspeções**: Os usuários Scrum devem, frequentemente, inspecionar os artefatos Scrum e o progresso em direção ao objetivo da *Sprint* para detectar variações indesejadas.
- **Adaptação**: Se um inspetor determina que um ou mais aspectos de um processo desviou para fora dos limites aceitáveis, e que o resultado do produto será inaceitável, o processo

ou o material sendo produzido deve ser ajustado. O ajuste deve ser realizado o mais breve possível para minimizar mais desvios.

2.6.1.1 Escolha da metodologia Scrum

Considerando que o Scrum é um *framework* simples para projetos complexos, a Figura 2.6.1 auxilia na avaliação da aplicabilidade desta metodologia dentro de um projeto, ou seja, serve para identificar se um projeto é complexo o bastante ou insuficiente para o Scrum.

Figura 2.6.1 – O uso do Scrum: Requisitos x Tecnologia



Fonte: https://www.youtube.com/watch?v=XfvQWnRgxG0

A Figura 2.6.1 representa um gráfico que demonstra uma relação entre o conhecimento dos requisitos e das tecnologias envolvidas no projeto. Segundo as normas do Scrum, disponíveis no guia oficial em seu *site*[4], quando tem-se um total domínio da tecnologia e conhecimento dos requisitos, o nível de previsibilidade é elevado, e o Scrum torna-se dispensável. Neste primeiro nível (Metodologias com Processo Definido) utiliza-se modelos como o modelo em cascata, onde uma fase é iniciada seguida da outra, e atendem perfeitamente às necessidades.

O guia do Scrum afirma ainda que o Scrum é melhor aplicado na segunda camada, conforme o nível de conhecimento de requisitos ou tecnologia diminui, ao passo que se dificulte o cenário encontrado, tornando-o caótico. Todavia, se os requisitos e as tecnologias são desco-

[4] Disponível em: https://www.scrum.org

nhecidos e sem domínio, considera-se um caso de anarquia, e o Scrum não traria resultados. No capítulo quatro será relatado em maiores detalhes o *framework* Scrum e sua utilização.

2.7 DESENVOLVIMENTO RESPONSIVO

Uma página *web* com conteúdo responsivo pode ser acessada nos computadores convencionais, nos *laptops*, nos *smartphones*, *tablets*, TV e qualquer outro dispositivo, que tenha acesso à internet, de forma bem apresentada (PROSTT, 2013).

Nos meados de 2010, Ethan Marcotte escreveu um artigo no site *A List Apart* onde o termo *web Design* Responsivo surgiu e que mudaria a filosofia e os métodos de desenvolvimento de *layouts*. A palavra "responsivo" foi retirada da arquitetura onde é referente a técnicas cujos materiais utilizados se adaptam ao ambiente que interagem (ARAÚJO, 2015).

Evidentemente, não é de agora que existe a preocupação com layouts. John Allsopp publicou em seu artigo *A Dao of web Design* em 2000, citado por Araujo (2015), uma sugestão que viria ser foco dos *web* designers anos depois:

> Faça páginas que são acessíveis, independentemente de navegador, plataforma ou tela que seu leitor escolha ou tenha que usar para acessar suas páginas. Isso significa páginas que são legíveis independentemente da resolução ou tamanho da tela, ou do número de cores. (*A Dao of web Design*, 2000 *apud* Araújo, 2015).

Com esta preocupação, o sistema aqui apresentado foi desenvolvido com o auxílio do *framework* Bootstrap, totalmente focado e representado com a linguagem CSS, responsável pelo aspecto da página, do *layout* e da leitura da página *web* pelo navegador.

2.8 INFORMATIZAÇÃO NAS BIBLIOTECAS

Conforme a tecnologia avança e possibilita novas descobertas e criações por parte do homem, os programadores agem de maneira semelhante. Com a inserção da informática na sociedade, os *software* para bibliotecas começaram a surgir como ferramenta de auxílio às tarefas mais básicas. Primeiramente, eles apenas emitiam listagens em forma de referências ou fichas catalográficas, para serem utilizadas nos catálogos de fichas das bibliotecas.

Segundo Corte et al. (1999, p. 242) citado por Damasio et al. (2006, p. 7):

> "A modernização das bibliotecas esta diretamente ligada a automação de rotinas e serviços, com o intuito de implantar uma infraestrutura de comunicação para

> agilizar e ampliar o acesso a informação pelo usuário, tornando-se necessário haver uma ampla visão da tecnologia da informação e sua aplicação nas organizações".

Conforme relata Damasio et al. (2006, p. 7), no final da década de 80 surgiu o MicroIsis – CDS-ISIS, um *software* desenvolvido pela Unesco e que se tornou "um dos *software* mais popularizados para bibliotecas, no Brasil e no mundo, devido a sua licença *freeware*[5], não sendo necessário o pagamento de licenças". No Brasil foi desenvolvido o *software* GNUTECA – Sistema de Gestão de Acervo Empréstimo e Colaboração para Bibliotecas, primeiramente instalado e desenvolvido na UNIVATES (Universidade do Vale do Taquari), bastante aceito por ser um sistema para automação de todos os processos de uma biblioteca, independentemente do tamanho de seu acervo ou da quantidade de usuários.

Porém, existem diversos *software* livres para bibliotecas sendo utilizados pelo mundo, bem como *software* com licenças para compra. Dos mais simples aos mais complexos, a escolha deve ser baseada de acordo com a necessidade da biblioteca em questão.

[5] É qualquer programa de computador cuja utilização não implica no pagamento de licenças.

Capítulo 3

TRABALHOS RELACIONADOS

Neste capítulo são descritos alguns trabalhos relacionados e que contribuíram à realização do trabalho proposto, correlacionando-os de maneira a justificar a necessidade de um novo produto. Em particular, considerou-se os trabalhos de (Machado, 2014), onde o mesmo desenvolveu um sistema bibliotecário para fins semelhantes, (Leitao, 2010), que avalia o uso da metodologia Scrum e (Aguiar, 2004), onde a transformação organizacional é citada mediante à informatização tecnológica.

3.1 SISTEMA *WEB* PARA GERENCIAMENTO DE ACERVO MULTIMÍDIA

Em Machado (2014) é proposto e desenvolvido um sistema *web* para gerenciamento de acervo multimídia. A ideia geral consiste em fornecer uma solução para a catalogação e gerenciamento do conteúdo presente na Biblioteca do Instituto Casa da Glória de Diamantina, MG, avaliando a interação entre o usuário final e o sistema.

O autor utilizou conceitos da engenharia *web* e a metodologia Scrum para o desenvolvimento. A importância de uma metodologia a ser seguida é enfatizada algumas vezes, sendo comprovada no estudo de seu trabalho. Além disso, foi observado o nível de satisfação e interação dos usuários da cidade com o uso do sistema, consideráveis satisfatórios.

Embora algumas rotinas fossem semelhantes, o trabalho relacionado se difere do aqui proposto pelo modo como o acervo é administrado. Enquanto Machado (2014) solucionou o problema em fornecer os arquivos multimídia aos usuários da Biblioteca, seja por download[1] ou consulta, o sistema bibliotecário aqui proposto visa o controle de empréstimos para os usuários.

[1] Ato de fazer cópia de uma informação, ger. de um arquivo, que se encontra num computador remoto. Fonte: Dicionário.

3.2 APLICAÇÃO DE SCRUM EM AMBIENTE DE DESENVOLVIMENTO DE *SOFTWARE* EDUCATIVO

O trabalho apresentado em Leitao (2010), avalia o uso da metodologia Scrum na solução de problemas como: baixa qualidade, aumento de custos e falta de gerenciamento de processos no desenvolvimento de *software*. Para exemplificar o uso da metodologia supracitada, a autora realizou um estudo de caso em uma empresa real de desenvolvimento de *software* educativos, analisando e incluindo recursos de suporte com uso da metodologia. De acordo com o trabalho, nesta empresa foram analisadas mudanças de papéis e responsabilidades nos recursos humanos; inclusão de recursos e aplicativos de suporte ao uso da metodologia e alterações na distribuição de tarefas pelo tempo de desenvolvimento.

O ponto forte dessa relação é um trabalho rico em conteúdo capaz de apresentar na prática, todas as fases da metodologia Scrum. Com algumas adaptações, é possível compreender o ganho de tempo a partir do momento em que os roteiros foram definidos e passaram a ser abraçados. Os problemas iniciais encontrados pela autora lidam com uma descentralização de informações, consequência de uma falta de padronização na coordenação da equipe responsável por fornecer objetos de aprendizagem. Com a implantação do Scrum, a organização de tarefas possibilitou ferramentas mais dinâmicas, padronizadas e com menor índice de correções apresentados.

Essa abordagem serviu de motivação para a escolha da metodologia ágil implantada. Segundo Leitao (2010), comprovou-se que o método é válido em diversos tipos de ambientes, inclusive para a gestão educacional. Neste sentido, foi importante compreender a aplicação em relacionamentos humanos e na prática de ferramentas didáticas, afirmando a boa relação entre estas duas partes.

3.3 SISTEMA DE INFORMAÇÃO PARA GESTÃO EDUCACIONAL: SISTEMATIZAÇÃO DE UMA PROPOSTA DE MODELO E AVALIAÇÃO DO PROCESSO DE SUA CONSTRUÇÃO

Em Aguiar (2004) é proposto um modelo de utilização das novas tecnologias na gestão educacional. O trabalho analisa a relação entre Gestão, Tecnologia da Informação e Cidadania. Por meio de um estudo de caso, o autor implementou na Secretaria da Educação do Estado do Ceará, os projetos Sistema Integrado de Gestão Educacional, Matrícula Humanizada,

Internet nas Escolas e Programa de Melhoria da Educação. A modernização da tecnologia nesta Secretaria, sinalizou o desenvolvimento de ferramentas do processo de administração escolar e envolvimento da comunidade.

Analogamente, esta dissertação traz uma visão de cultura organizacional que merece destaque quando se deseja interferir em um ambiente, seja ele escolar ou não, com a implantação de um novo *software* e metodologia. Para Moreira (1994 *apud* Aguiar 2004), a empresa moderna precisa ser flexível à mudança.

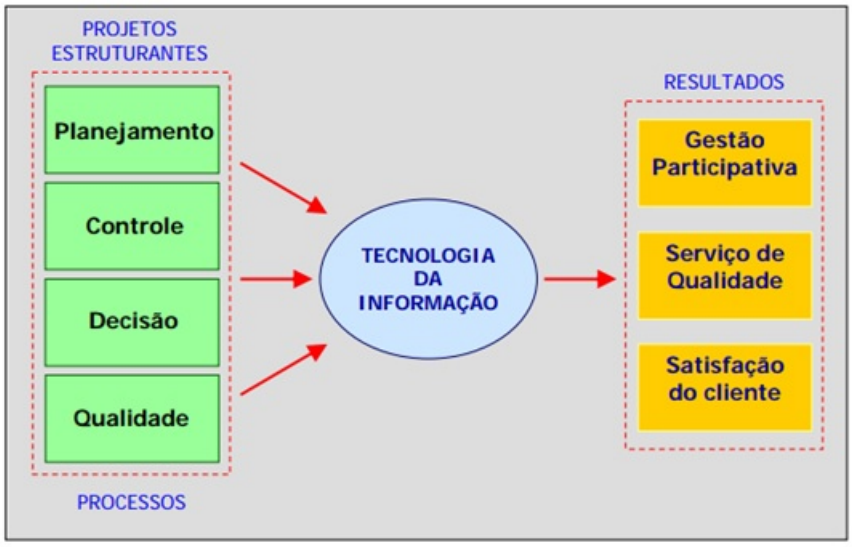
> Das várias consequências que possam emergir do conceito de flexibilidade, duas delas nos parecem mais relevantes, no sentido de serem mais abrangentes e, em certa medida, coordenarem todas as demais. A primeira diz respeito à criação e manutenção de uma cultura organizacional voltada para a mudança; a segunda enfatiza a necessidade de estruturas organizacionais mais flexíveis. (Moreira, 1994 *apud* Aguiar, 2004. p. 28).

O trabalho de Aguiar (2004) direciona seu estudo para um modelo de transformação organizacional baseado na reengenharia. Ele define reengenharia como:

> "Um modelo de mudança organizacional, conduzido do alto para baixo em uma organização, que se baseia na reestruturação radical dos processos de trabalho, buscando aumentar indicadores de desempenho: aumento da produtividade, qualidade dos serviços ou produtos e eficácia do atendimento ao cliente".

A Figura 3.3.1 apresenta o modelo proposto para a utilização da tecnologia da informação na área educacional, com ênfase nos processos.

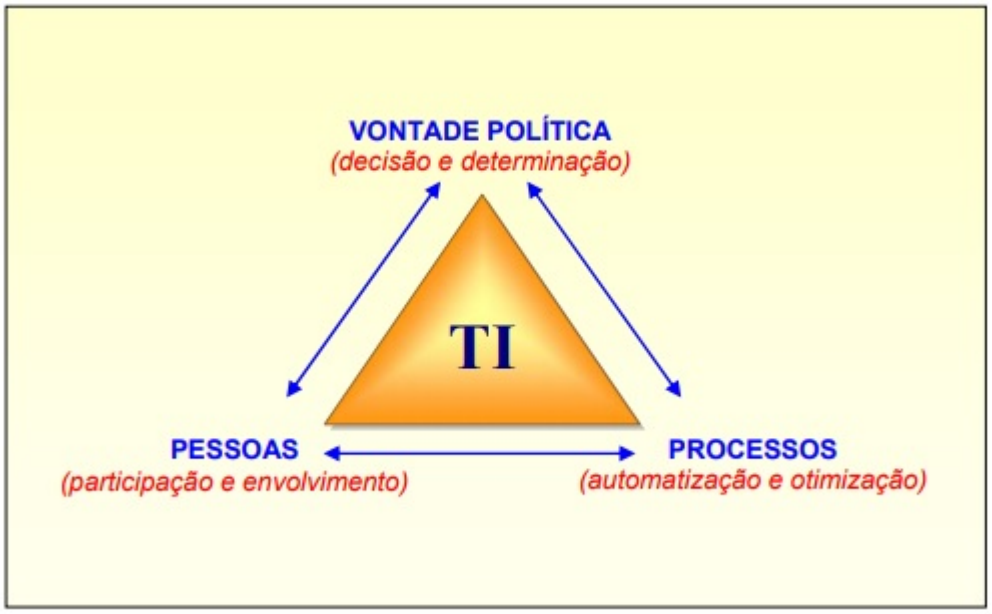
Figura 3.3.1 – Modelo proposto para tecnologia da informação com enfoque nos processos



Fonte: (AGUIAR, 2004, p. 103)

O estudo indica ainda, uma trilogia dos elementos que envolvam Tecnologia da Informação (Figura 3.3.2), no ambiente organizacional. Todos esses elementos são abordados pelo autor em seu trabalho, e deixam claro a mudança constatada na Secretaria da Educação, onde processos de automatização e otimização foram implantados.

Figura 3.3.2 – Trilogia dos elementos de sucesso no ambiente organizacional



Fonte: (AGUIAR, 2004, p. 147)

3.4 CONSIDERAÇÕES PARA O DESENVOLVIMENTO DO BIBLIOTECA JK

Analisando os trabalhos relacionados, opções disponíveis no mercado, e as necessidades da Biblioteca do CESEC JK que buscou-se atender, foi tomada a decisão em que se faria necessário desenvolver um sistema específico para este caso.

Embora no mercado da Internet fosse possível identificar sistemas robustos e completos no âmbito da administração bibliotecária escolar, algumas particularidades direcionaram para um sistema de simples implantação e aprendizado, seja pelas funcionalidades desejadas, ou pela cultura organizacional presente. No sistema desenvolvido, conforme o levantamento das histórias de usuário e requisitos, seja por meio de entrevistas ou questionário (Apêndice B) descritos nos próximos capítulos, as bibliotecárias optaram por centralizar a tomada de decisões em suas mãos. Todo o empréstimo e devolução de livros será controlado pelas mesmas, o que

torna o sistema “diferente” de modelos oferecidos em que o leitor possui acesso ao sistema para realizar empréstimos e devoluções, pendentes apenas da aceitação do administrador.

Além disso, uma solicitação atendida foi um cadastro separado de alunos, professores, bibliotecários e funcionários da escola. Outra vantagem em se desenvolver o sistema está ligada diretamente aos custos com o mesmo. Uma vez que o sistema será fornecido gratuitamente à escola, a mesma não precisará arcar com licenças de software pagos.

Capítulo 4

METODOLOGIA E FERRAMENTAS COMPUTACIONAIS UTILIZADAS

Este Capítulo esclarece a metodologia aplicada, informando as ferramentas computacionais que foram utilizadas para o desenvolvimento do trabalho.

4.1 EQUIPAMENTOS

Os equipamentos utilizados para o desenvolvimento deste trabalho foram um computador *Deskop* modelo AMD A-8 3800 APU 2.40 GHz, que possui 8GB de Memória RAM, placa gráfica AMD Radeon HD 5700 *Series* com 2GB de Memória RAM, e sistema operacional *Windows* 10 Pro 2017 64bit. Foi utilizado também um *MacBook* modelo Intel Core 2 Duo 2.16 GHz, com 4GB de Memória RAM, gráficos Intel GMA 950 64MB, com distribuição *Mac* OS X *Lion* 10.7.5 (11G63).

4.2 AMBIENTE DE DESENVOLVIMENTO

Nesta seção, destaca-se o ambiente de desenvolvimento escolhido. O XAMPP[1], presente e consolidado no mercado há mais de uma década para fornecer todo o servidor *web* e gerenciar o banco de dados, e o Sublime Text, responsável pela manipulação, edição e escrita do código fonte, as páginas e telas do sistema.

[1] Disponível em: https://www.apachefriends.org/pt$_br/about.html.Acessoem$: $21dejulhode2017$.

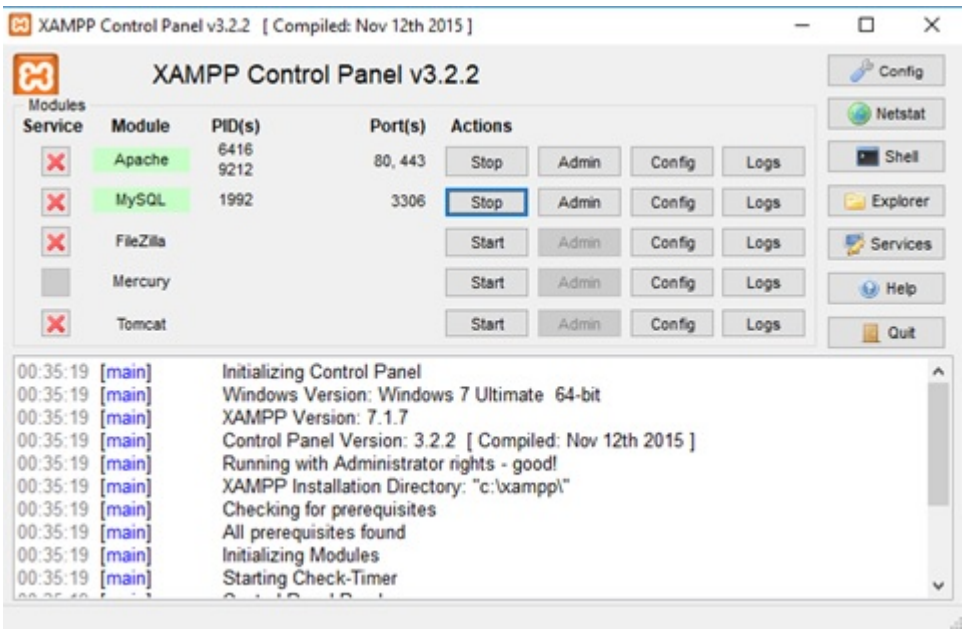
4.2.1 XAMP

O XAMPP é um dos ambientes de desenvolvimento PHP mais popular. Ele é completamente gratuito, de fácil instalação e distribuição Apache[2] , contendo MySQL, PHP e Perl. Está disponível tanto para *Windows*, quanto para *Linux* e OS X.

Segundo o *site* oficial do XAMPP[3], não é fácil instalar um servidor *web* Apache e torna-se mais difícil se você quer acrescentar MySQL e PHP. O objetivo do XAMPP é construir uma distribuição fácil de instalar para desenvolvedores entrarem no mundo do Apache. Para torná-lo conveniente para os desenvolvedores, o XAMPP é configurado com todos os recursos ativados.

O XAMPP possui um painel de controle bastante intuitivo, e de fácil utilização. Como se pode observar na Figura 4.2.1, tudo o que é preciso é ativar no botão *Start* os módulos desejados para trabalhar. O servidor Apache e o MySQL tornam-se fundamentais e necessários no desenvolvimento de uma aplicação *web*.

Figura 4.2.1 – Painel de Controle do XAMPP (versão 7.1.7)



Fonte: Imagem extraída da execução em tempo real do XAMPP (2017)

[2] É o servidor *web* livre mais utilizado no mundo, um *software* responsável por aceitar pedidos HTTP de clientes, geralmente os navegadores, e servi-los com respostas HTTP. Fonte: https://www.apache.org/.

[3] Disponível em: https://www.apachefriends.org/pt$_br/index.html$

4.2.2 Sublime Text

O Sublime Text tem sido um editor de texto muito escolhido pelos desenvolvedores por ser leve, simples e com interface agradável. Ele foi desenvolvido em Python[4] e foi feito para ser simples. É um editor de texto bastante fácil de usar, mas com muitos recursos e funcionalidades que podem ser adicionadas para complementar seu uso[5].

No *site* oficial[6] é possível realizar o *download* para as versões 2 e 3, ambas com opções para Windows (*portable* e .exe), Linux (em uma seção separada) e OS X. A versão do Sublime Text utilizada neste trabalho é a terceira.

Alguns motivos que justificam sua escolha são: diversidade de atalhos disponíveis para agilizar o processo na digitação e edição de texto, edição personalizada para HTML que inclui o fechamento de *tags* e atributos automaticamente, recursos para auto completar comandos pré-definidos, e interface amigável, além de ser facilmente utilizado em conjunto com *frameworks*.

4.3 LINGUAGENS UTILIZADAS

4.3.1 HTML

Segundo FERREIRA et al. (2012), HTML é uma abreviação de *Hypertext Markup Language*, que traduzindo para o português significa Linguagem de Marcação de Hipertexto. Resumindo em uma frase: o HTML é uma linguagem para publicação de conteúdo (texto, imagem, vídeo, áudio e etc.) para a *web*.

Neste trabalho, a versão utilizada é o HTML5. Um de seus principais objetivos é facilitar a manipulação do elemento possibilitando o desenvolvedor a modificar as características dos objetos de forma não intrusiva e de maneira que seja transparente para o usuário final. Ao contrário das versões anteriores, o HTML5 fornece ferramentas para a CSS e o Javascript fazerem seu trabalho da melhor maneira possível. O HTML5 permite por meio de suas APIs[7] a manipulação das características destes elementos, de forma que o *website* ou a aplicação continue leve e funcional, sem a necessidade de criações de grandes blocos de *scripts* (FERREIRA; EIS,

[4] Python é uma linguagem de programação de alto nível, interpretada, de *script*, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Foi lançada por Guido van Rossum em 1991. Fonte: https://www.python.org/.

[5] Fonte: http://www.devmedia.com.br/sublime-text-ide-introducao-a-melhor-ide-para-desenvolvimento Acesso em: 21 de julho de 2017.

[6] Disponível em: https://www.sublimetext.com

[7] API é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de *software* ou plataforma baseado na *web*. Fonte: https://canaltech.com.br/*software*/o-que-e-api/.

2012).

A estrutura básica do HTML5 continua sendo praticamente a mesma das versões anteriores, mas com menos código. Na Figura 4.3.1 a seguir apresenta-se a estrutura básica que pode ser seguida:

Figura 4.3.1 – Estrutura básica do HTML

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Título da página</title>
    <meta charset="utf-8">
  </head>
  <body>
    Aqui vai o código HTML que fará seu site aparecer.
  </body>
</html>
```

Fonte: http://tableless.github.io/iniciantes/manual/html/estruturabasica.html

4.3.2 PHP

Conforme explica Converse e Park (2003), o PHP, que significa *PHP: Hypertext Preprocessor* é a linguagem de desenvolvimento *web* escrita por desenvolvedores *web* e para desenvolvedores *web*. Trata-se de uma linguagem de criação de *scripts* do lado servidor, que pode ser incorporada em HTML ou utilizada como um binário independente.

O PHP é mundialmente conhecido como uma linguagem de fácil aprendizado, em comparação com outras maneiras de obter funcionalidades semelhantes. Muitas das funções específicas mais úteis, como abrir uma conexão a um banco de dados ou buscar um correio eletrônico a partir de um servidor já são predefinidas. Além disso, o PHP não custa nada, é surpreendentemente veloz em sua execução, e possui código fonte aberto (CONVERSE; PARK, 2003).

O código PHP segue um conjunto básico de regras sintáticas, a maioria emprestada de linguagens de programação como C e Perl[8]. Seus requisitos sintáticos são mínimos, e quando possível, o PHP tenta exibir resultados em vez de gerar um erro. Porém, em vez de muitos

[8] Perl é uma linguagem de programação multiplataforma usada em aplicações de CGI para a *web*, para administração de sistemas linux e por várias aplicações que necessitam de facilidade de manipulação de strings.

comandos para mostrar HTML (como acontece com C ou Perl), as páginas PHP contêm HTML em código mesclado que faz "alguma coisa". O código PHP é delimitado pelas instruções de processamento (*tags*) de início e fim **<?php e ?>** que permitem que você pule para dentro e para fora do "modo PHP"[9].

A estrutura básica do PHP é demonstrada em seu site como um exemplo introdutório bastante intuitivo, conforme a Figura 4.3.2 abaixo:

Figura 4.3.2 – Estrutura básica do PHP

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Exemplo</title>
    </head>
    <body>

        <?php
            echo "Olá, eu sou um script PHP!";
        ?>

    </body>
</html>
```

Fonte: http://www.google.com

Por fim, o desenvolvimento do PHP também é constante e está sempre em progresso, e conforme ressalta Converse e Park (2003), o PHP não é a resolução para todos os problemas de desenvolvimento *web*, mas tem inúmeras vantagens. Com o PHP, portanto, você tem liberdade de escolha de sistema operacional e de servidor *web*. Do mesmo modo, você pode escolher entre utilizar programação estruturada ou programação orientada a objeto (OOP), ou ainda uma mistura das duas. Ele oferece o melhor tipo de conectividade para todos os tipos de servidores de *back-end*[10].

4.3.3 JavaScript

A linguagem JavaScript é uma tecnologia para melhoria da *web*. Quando empregada no computador cliente, a linguagem pode ajudar a transformar uma página de conteúdo estático

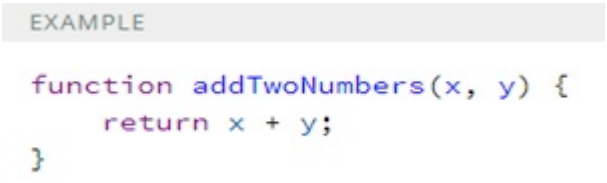
[9] Explanação extraída do site oficial do PHP, disponível em: https://secure.php.net/manual/pt$_BR/intro - whatis.php$.

[10] O desenvolvedor *back-end* trabalha na parte de "trás"da aplicação. Ele é o responsável, em termos gerais, pela implementação da regra de negócio, podendo trabalhar várias linguagens, como *Go*, *Clojure*, *C*, *PHP*, *Java*, *Python*, *Ruby*, entre outras. Fonte: https://www.treina*web*.com.br/blog/o-que-e-front-end-e-back-end/.

em uma experiência atraente, interativa e inteligente (GOODMAN, 2001).

Sendo assim, Goddman (2001) afirma que o JavaScript se caracteriza como uma linguagem de programação interpretada. Foi originalmente implementada como parte dos navegadores *web* para que *scripts* pudessem ser executados do lado do cliente e interagissem com o usuário sem a necessidade deste *script* passar pelo servidor, controlando o navegador, realizando comunicação assíncrona e alterando o conteúdo do documento exibido. É atualmente a principal linguagem para programação *client-side*[11] em navegadores *web*. A Figura 4.3.3, mostra uma estrutura básica de uma função[12] em JavaScript para retornar a soma de dois números:

Figura 4.3.3 – Exemplo de uma função em JavaScript



Fonte: http://www.google.com

Neste contexto, o JavaScript assume um papel importantíssimo para facilitar a experiência do lado do cliente no ambiente *web*, onde seu uso adequado pode facilitar a execução de tarefas, agilizando diversas atividades.

Por fim, Goodman (2001) explica que o JavaScript é a ferramenta certa para o trabalho certo, onde uma parte importante para se tornar um autor de página da *web* habilidoso é saber como combinar uma ferramenta de autoria com uma tarefa de criação de solução. Dentre as características trabalhadas pelo JavaScript, destaca-se:

- Fazer com que a página *web* responda ou reaja diretamente à interação do usuário com elementos do formulário e *links* de hipertexto.
- Distribuir pequenas coleções de informações tipo banco de dados e oferecer uma interface amigável para esses dados.
- Controlar a navegação por vários *frames*, *plugins*, ou *applets* Java com base nas escolhas do usuário no documento HTML.

[11] São linguagens onde apenas o seu navegador vai entender. Quem vai processar essa linguagem não é o servidor, mas o seu *browser* (*Chrome*, IE, *Firefox*, etc...). Fonte: https://www.gigasystems.com.br/artigo/60/client-side-e-server-side.

[12] Uma função é uma definição de um conjunto de ações adiadas, chamadas por manipuladores de evento ou instruções em outro lugar no script. (Goodman, 2001. p. 67).

- Pré-processar dados no cliente antes do envio a um servidor.
- Alterar o conteúdo e os estilos nos *browsers* modernos dinamicamente e instantaneamente em resposta à interação com o usuário.

4.3.4 SQL

Os bancos de dados não são simplesmente lugares eletrônicos para guardar dados. Conforme explica Suehring (2002), os bancos de dados mantêm as informações em tabelas. Uma tabela é uma estrutura que consiste em pelo menos uma coluna, mas normalmente muito mais. Um banco de dados é, portanto, uma coleção de uma ou mais tabelas de informações relacionadas. Classificação, indexação e consultas organizam dados para ajudar o usuário do banco de dados. Para a criação de um banco de dados, requer-se a utilização de um sistema de gerenciamento para o mesmo.

O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês *Structured Query Language*) como interface. É atualmente um dos mais populares[13].

A linguagem SQL é um tipo de linguagem de programação especializada desenvolvida para trabalhar com banco de dados relacionais como MySQL, Oracle, Microsoft SQL *Server*, PostgreSQL, Informix e outros (SUEHRING, 2002).

A escolha do MySQL como ferramenta deste trabalho é impulsionada por uma definição de Suehring (2002), quando decide listar tarefas que o MySQL faz especialmente bem:

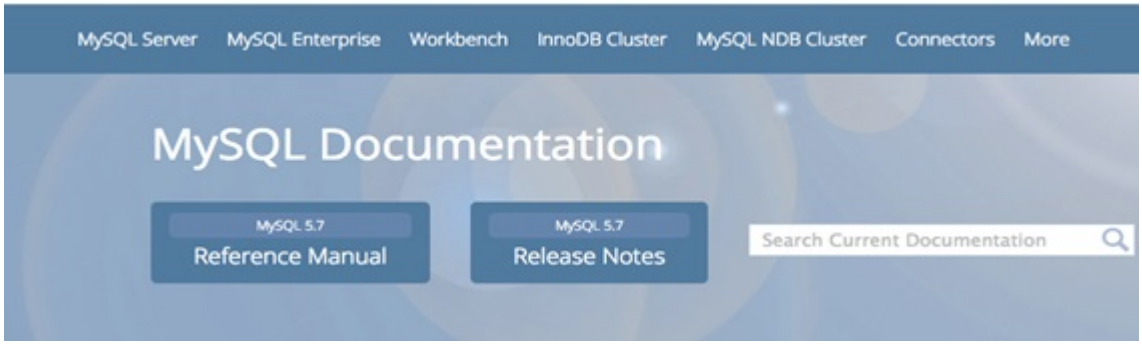
> As aplicações *web* em geral apresentam muitas leituras e poucas gravações. O MySQL é rápido e pode atender às demandas de velocidade na internet. Em minha experiência, o MySQL tem provado repetidas vezes que supera outros produtos de SGBDR em aplicações *web* (SUEHRING, 2002, p.20).

A Figura 4.3.4 demonstra que é possível realizar o *download* do manual da versão mais atual do MySQL no *site* oficial[14] do MySQL, bastando clicar sobre *Reference Manual*.

[13] Definição extraída do *site* oficial do MySQL, disponível em: https://www.mysql.com/. Acesso em 21 de julho de 2017.

[14] Disponível em: https://www.mysql.com/

Figura 4.3.4 – Encontrando o manual do MySQL



Fonte: http://www.mysql.com (2017)

4.4 *FRAMEWORKS*

Segundo Fayad et al. (1997 *apud* Costa et al. 2007), *frameworks* representam uma estrutura formada por blocos pré-fabricados de *software* que os programadores podem usar, estender ou adaptar para uma solução específica. Em outras palavras, um *framework* de desenvolvimento é uma "base" de onde se pode desenvolver algo maior ou mais específico. É uma coleção de códigos-fonte, classes, funções, técnicas e metodologias que facilitam o desenvolvimento de novos *softwares*.

Frameworks possuem vantagens, tais como: maior facilidade para a detecção de erros, por serem peças mais concisas de *software*; concentração na abstração de soluções do problema que estamos tratando; eficiência na resolução dos problemas e otimização de recursos.

4.4.1 Bootstrap

Bootstrap é o mais popular *framework* HTML, CSS, e JS para desenvolvimento de projetos responsivos e focado para dispositivos móveis na *web*. Ele torna o desenvolvimento *front-end*[15] *web* mais rápido e fácil, sendo feito para pessoas de todos os níveis e dispositivos de qualquer forma ou tamanho[16].

O Bootstrap utiliza CSS tradicional, mas seu código fonte utiliza os dois pré-processadores CSS mais populares, *Less* e *Sass*[17]. Além disso, ele possui código aberto, hospe-

[15] O desenvolvedor *front-end* é responsável por "dar vida"à interface. Trabalha com a parte da aplicação que interage diretamente com o usuário. Este profissional foca em HTML, CSS e JavaScript. Fonte: https://www.treina*web*.com.br/blog/o-que-e-front-end-e-back-end/.

[16] Fonte: http://getbootstrap.com.br/.

[17] Verificar o *site* oficial do Bootstrap. Disponível em: http://getbootstrap.com.br/

dado, desenvolvido e gerenciado no GitHub.

Para a realização deste trabalho, foi efetuado o download da versão v3.3.7 no mesmo link onde é possível observar exemplos do *framework*, conforme a Figura 4.4.1:

Figura 4.4.1 – Página para *download* da versão atual do Bootstrap

Fonte: http://getbootstrap.com.br/getting-started/examples

O download do código fonte Bootstrap inclui o CSS, JavaScript pré-compilados e *assets fonts*, junto com o código fonte do *Less*, JavaScripts e documentação. Mais precisamente, ele inclui a estrutura representada na Figura 4.4.2.

Figura 4.4.2 – Estrutura do Bootstrap incluso em um projeto

```
bootstrap/
├── less/
├── js/
├── fonts/
├── dist/
│   ├── css/
│   ├── js/
│   └── fonts/
└── docs/
    └── examples/
```

Fonte: http://getbootstrap.com.br/getting-started/examples

4.4.2 CakePHP

O CakePHP torna a construção de aplicativos da *web* mais simples, mais rápidos, enquanto requerem menos código. É descrito como uma moderna estrutura PHP 7 que oferece

uma camada de acesso flexível ao banco de dados e um poderoso sistema de andaimes que torna o desenvolvimento de sistemas pequenos e complexos mais simples, fáceis e, claro, mais saborosos[18].

Atualmente na versão 3.5, o CakePHP foi escolhido para agilizar processos no desenvolvimento do sistema *web*, visando reduzir o tamanho do código. Os requisitos para sua instalação foram apenas um servidor *web*, neste caso o Apache, por meio do XAMPP já instalado, e a utilização de uma versão superior ao PHP 5.6.0.

O CakePHP utiliza *Composer*, uma ferramenta de gerenciamento de dependências para PHP 5.3+, como o método suportado oficial para instalação. O mesmo pode ser facilmente baixado em seu *site* oficial[19].

Depois de instalar o CakePHP seguindo as orientações de seu *site* oficial, a estrutura de sua configuração de produção será parecida com o sistema de arquivos da Figura 4.4.3.

Figura 4.4.3 – Exemplo de sistema de arquivos do CakePHP

```
/cake_install/
    bin/
    config/
    logs/
    plugins/
    src/
    tests/
    tmp/
    vendor/
    webroot/ (esse diretório é definido como DocumentRoot)
    .gitignore
    .htaccess
    .travis.yml
    composer.json
    index.php
    phpunit.xml.dist
    README.md
```

Fonte: https://book.cakephp.org/3.0/pt/installation.htmlinstalando-o-cakephp

4.5 METODOLOGIA

Esta seção tem por objetivo caracterizar a metodologia Scrum, apresentar seu *framework* tendendo fornecer informações suficientes que justifiquem a escolha de sua utilização como metodologia adequada para desenvolvimento de *software* educativos.

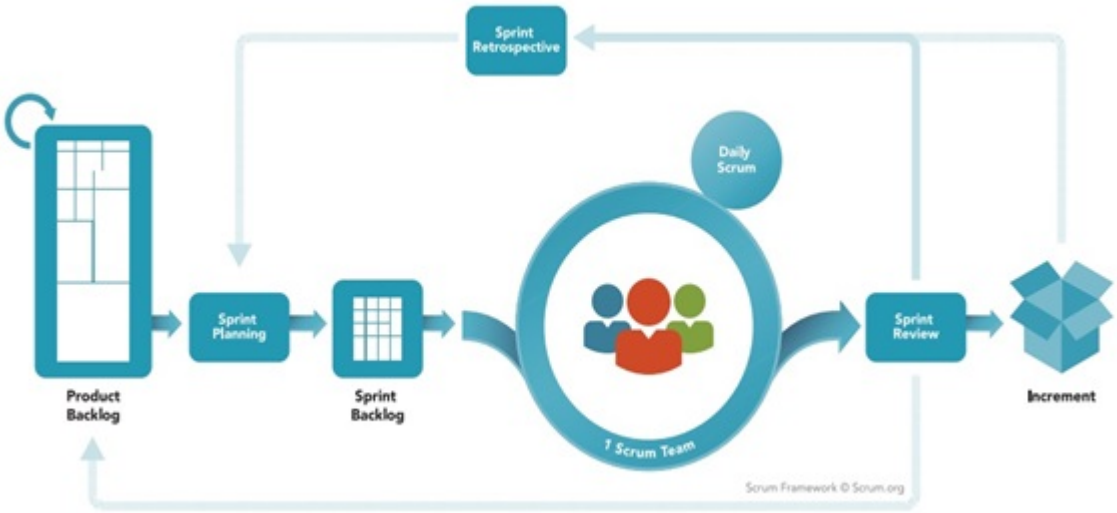
[18] Fonte: https://cakephp.org/.
[19] Fonte: https://getcomposer.org/download/

4.5.1 O *Framework* Scrum

O Scrum possui cinco eventos prescritos, todos com duração máxima predefinida (*Time-Boxed*), que são: *Sprint, Sprint Planning, Daily Scrum, Sprint Review e Sprint Retrospective* (Retrospectiva).

O *framework* Scrum é esquematizado em seu *site* oficial conforme a Figura 4.5.1.

Figura 4.5.1 – *Framework* Scrum



Fonte: http://www.scrum.org

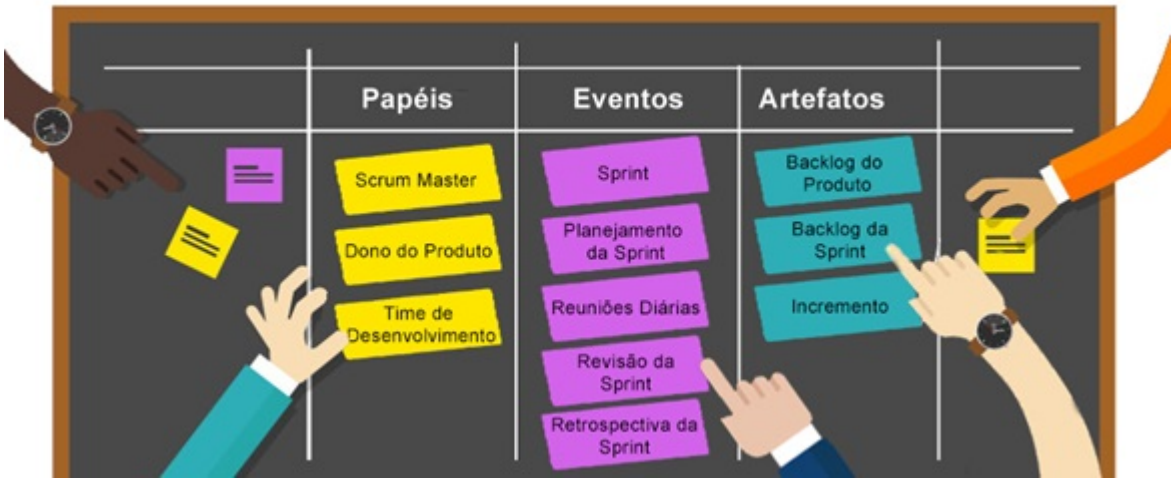
4.5.1.1 Práticas fundamentais do Scrum

O Scrum é composto por papéis, eventos e artefatos bem definidos. A perfeita utilização deste *framework* auxilia pessoas para que possam tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível. Ele é leve, simples de entender e difícil de dominar[20].

A Figura 4.5.2 ilustra uma pequena introdução ao *framework* Scrum com seus papéis, eventos e artefatos.

[20] Guia do Scrum

Figura 4.5.2 – Scrum: papéis, eventos e artefatos



Fonte: https://www.smarti.blog.br/scrum-desenvolvimento-agil-*software*/

4.5.1.2 Papéis

Os componentes de uma organização no Scrum são auto organizáveis e multifuncionais. Segundo Schwaber e Sutherland (2017) o Scrum é composto por três papéis fundamentais: o *Product Owner* (Dono do Produto), o *Scrum Master* e o *Development Team* (Time de Desenvolvimento), sendo:

- **Product Owner**: é o ponto central com poder de manter liderança sobre os outros grupos. É o único responsável por decidir quais recursos e funcionalidades serão construídos e qual a ordem de prioridade que devem ser feitos. Deve conhecer as necessidades dos clientes e ser claro o suficiente para que toda a sua equipe compreenda de imediato o que se espera alcançar com o projeto.

- **Scrum Master**: é o responsável pelo gerenciamento do projeto, devendo garantir como um coach, garantindo que o trabalho da equipe seja funcional e produtivo. Ele acompanha o desenvolvimento e remove os impedimentos.

- **Development Team**: o Time de Desenvolvimento consiste de profissionais que realizam o trabalho de entregar um incremento potencialmente liberável do produto "Pronto" ao final de cada *Sprint*. Um incremento "Pronto" é requerido na Revisão da *Sprint*. Somente integrantes do Time de Desenvolvimento criam incrementos (SCHWABER; SUTHERLAND, 2017). O tamanho ideal do Time de Desenvolvimento é pequeno o suficiente para

se manter ágil e grande o suficiente para completar um trabalho significativo dentro da *Sprint*.

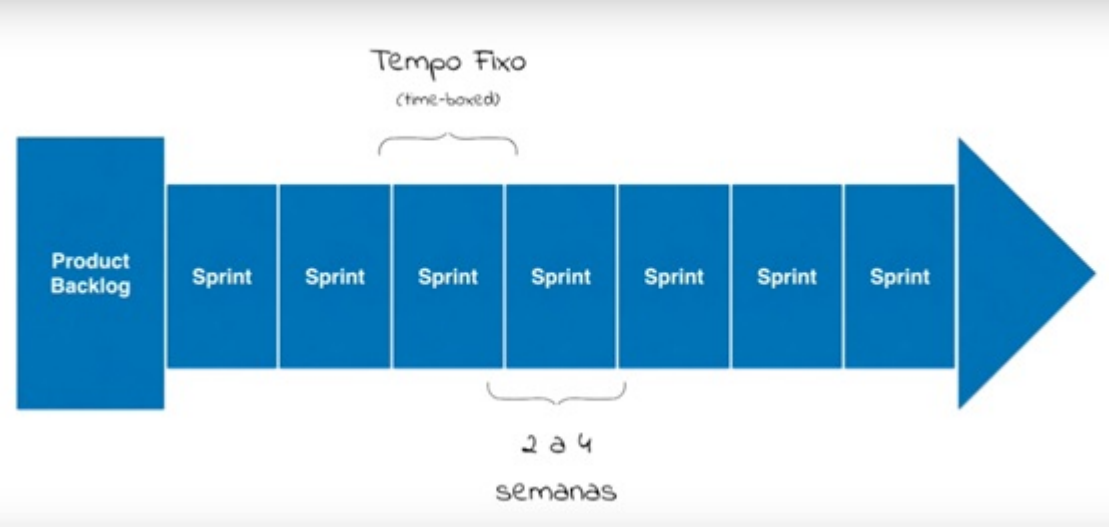
4.5.1.3 A dinâmica do Scrum: visão do produto

O *Product Owner* é o responsável por prover essa visão. Qualquer tipo de macroplanejamento pode ser aplicável, desde que saiba "o que você quer" e "onde você quer chegar".

O *Product Owner* organiza uma lista que pode variar de acordo com cada time, e pode ser formado por casos de uso, requisitos, ou *user stories*. Na técnica de *user stories*, bastante comum em metodologias ágeis, o *Product Owner* descreve o que precisa ser feito, identifica quem usará a funcionalidade e por que isto ajuda a não gerar itens que não agregam valor ao negócio (SUTHERLAND, 2014).

Em seguida, deve-se desmembrar todas as funcionalidades que serão necessárias. A lista resultante deste desmembramento de funcionalidades é o *Product Backlog* (*Backlog* do Produto). A Figura 4.5.3 descreve a execução do *Product Backlog* por meio de *Sprints*.

Figura 4.5.3 – Execução do *Product Backlog* por meio de *Sprints*



Fonte: https://www.youtube.com/watch?v=XfvQWnRgxG0

Os *Sprints* são períodos de tempo onde alguns itens selecionados do *Product Backlog* serão consumidos e entregues. Para planejar os *Sprints*, obedece-se uma regra básica do Scrum: eventos de duração fixa. O *Sprint* é o coração do Scrum, um caixa de tempo de um mês ou menos, durante o qual um incremento de produto potencialmente liberável é criado. Uma nova

Sprint inicia imediatamente após a conclusão da *Sprint* anterior (SCHWABER; SUTHERLAND, 2017).

4.5.1.4 Artefatos

Segundo seu guia oficial[21], o Scrum prevê alguns artefatos que nos permite ter uma visão sobre o andamento do projeto e da *Sprint*. Estes artefatos são conhecidos como *Backlog*. Conforme ilustrado anteriormente na Figura 4.5.2, dentre os artefatos temos o *Backlog* do Produto, o *Backlog* da *Sprint* e os incrementos.

O *Product Backlog* (*Backlog* do Produto) é uma lista de itens priorizados elencando o que deve ser desenvolvido na *Sprint*, associada a um valor de negócio, e que pode ser composta de requisitos funcionais ou não. Esses itens são organizados de acordo com suas prioridades, nas quais podem mudar durante o decorrer do projeto (SUTHERLAND, 2014). Desse modo, o *Product Backlog* permite controle e gerenciamento do processo.

É importante ressaltar a importância do *Backlog* da *Sprint*. Ribeiro et al. (2015) define como “um conjunto de itens selecionados para serem implementados (e entregues como incremento ao produto) durante a *Sprint*”. O objetivo do *Backlog* da *Sprint* é tornar o trabalho visível e tangível para que o Time atinja a meta da *Sprint*.

O incremento, por sua vez, “é a soma de todos os itens completados do *Backlog* do Produto. No final de um *Sprint*, o novo incremento deve estar ‘pronto’, o que significa que ele está em uma condição utilizável e atende à definição da Equipe do Scrum"(SCHWABER; JEFF, 2018).

4.5.1.5 Eventos

Os eventos do Scrum são definidos em volta da *Sprint*. Desta forma, o guia do Scrum os define como o passo a passo das execuções do Scrum, desde a reunião de planejamento da *Sprint*, às reuniões diárias e revisões, organizadas pelo *Scrum Master*, com retrospectiva do que foi feito. Devido ao fato deste trabalho ser realizado por uma única pessoa, não tem-se o Scrum aplicado em uma equipe com determinado número de pessoas, mas sim uma adaptação para que possa ser realizado por apenas uma. Neste caso, as reuniões diárias e revisões são substituídas por reflexões e análises do que foi e será feito, a fim de identificar possíveis mudanças e prevenir erros.

[21] Disponível em: https://www.scrum.org

Antes de um *Sprint* começar, é feito uma reunião de planejamento do *Sprint*, o *Sprint Planning*. Nesta etapa planeja-se a iteração a ser executada, sendo selecionadas funcionalidades a serem implementadas durante o *Sprint*, com base no *Product Backlog* pré-definido e priorizado[22]. A quantidade de funcionalidades incluídas num *Sprint* depende diretamente da capacidade e velocidade da equipe Scrum. Neste trabalho, dispensa-se o *Sprint Planning* em equipe, com as definições sendo feitas por apenas uma pessoa.

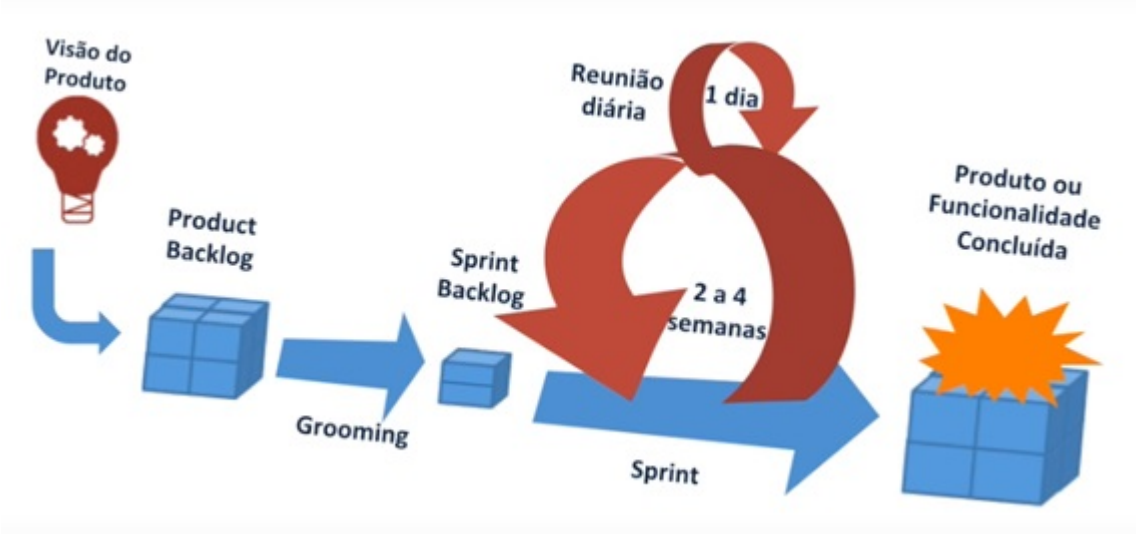
No final do *Sprint*, tem-se duas atividades fundamentais, o *Sprint Review* e a Retrospectiva. Primeiramente, o *Sprint Review* possui o objetivo de validar e adaptar o produto que está sendo construído. É uma reunião informal, onde a equipe, monitorada pelo *Scrum Master* exibe ao *Product Owner* e interessados os itens de *Backlog* considerados prontos e inicia-se uma nova *Sprint*[23].

A Retrospectiva (*Sprint Retrospective*), por sua vez, é o momento em que se deve inspecionar como foi desenvolvida o *Sprint* passado no que diz respeito às pessoas, relações de processos e ferramentas. Verifica necessidades de adaptação no processo, o que devemos melhorar e parar de fazer.

A Figura 4.5.4 apresenta todos os eventos do Scrum. Entendendo melhor todo o fluxo, tudo começa com a visão, o esboço atual, que se desdobra em *Product Backlog*. Este *Backlog* sofre *Grooming*, onde ocorre toda a sua priorização, ou estimativa de tamanhos. No planejamento de *Sprints*, cria-se o *Sprint Backlog*, que é a lista de funcionalidades que serão criadas durante o *Sprint*. Este, por sua vez, conforme dito anteriormente tem entre 2 e 4 semanas. O *Dailly Scrum* é onde acontecem as reuniões diárias. Por fim, ao final do *Sprint* atinge-se o produto ou funcionalidade concluída[24]. .

[22] Disponível em: www.desenvolvimentoagil.com.br/scrum/
[23] Disponível em: https://www.scrum.org
[24] Disponível em: https://www.scrum.org

Figura 4.5.4 – Eventos do Scrum



Fonte: http://www.mindmaster.com.br/scrum/

4.5.1.6 Ferramenta adicional: O *Burndown Chart*

O *Burndown chart* ou gráfico de *Burndown* é o gráfico utilizado pelas equipes Scrum para representar diariamente o progresso do trabalho em desenvolvimento. Ou seja, após cada dia de trabalho o gráfico apresenta a porção de trabalho finalizada em comparação com o trabalho total planejado[25].

O gráfico *Burndown* marca:

- no eixo horizontal os dias da *Sprint*, do 1º dia ao último e,
- no eixo vertical os pontos que foram planejados para compor a *Sprint*, partindo do máximo de pontos da *Sprint* (velocidade da equipe) até zero[26].

Este gráfico pode ser elaborado tanto pela quantidade de dias restantes que a equipe tem, quanto pelo trabalho realizado[27]. Neste trabalho foi optado por trabalhar com os dias restantes na variável.

[25] Fonte: http://blog.myscrumhalf.com/2012/01/burndown-chart-medindo-o-progresso-de-sua-sprint-e-trazendo-indicativos-do-processo-de-trabalho-da-equipe/
[26] Fonte: https://confluence.atlassian.com/jirasoftwarecloud/burndown-chart-777002653.html
[27] Disponível em: http://www.mindmaster.com.br/e-book-como-implantar-scrum/

4.5.2 Caracterização do Ambiente de Implantação do Scrum

Considerando o Sistema Bibliotecário “Biblioteca JK” como objetivo principal e produto deste trabalho, dividiu-se por módulos semelhantes suas estruturas (cadastrar, pesquisar, atualizar, editar, excluir). O projeto vigente é um material a ser fornecido tanto para alunos, funcionários, professores e bibliotecários da instituição de ensino que irá usufruir do produto, no caso o CESEC Juscelino Kubitschek de Oliveira, com foco no empréstimo e controle do acervo bibliotecário.

4.5.3 Análise e Correlação: Scrum x Ambiente

Após análise do ambiente, e avaliar a Metodologia na Seção 4.5, mediante os conceitos de Engenharia de Software apresentados no Referencial Teórico (Capítulo dois) deste trabalho, como base para pesquisas e conhecimento, pôde-se concluir ser a metodologia Scrum a mais adequada para a implantação neste caso.

Uma justificativa para essa decisão está ilustrada na Figura 2.6.1 da Seção “2.6.1.1 Escolha da metodologia Scrum”. Uma vez que, no início deste projeto não se tinha total conhecimento dos requisitos e tecnologias de desenvolvimento, a metodologia Scrum absorve este ambiente caótico ao passo que se adquire estes conhecimentos, permitindo um desenvolvimento incremental e adaptativo.

Capítulo 5

DESENVOLVIMENTO

Este capítulo tem por objetivo apresentar o desenvolvimento da metodologia Scrum com seus conceitos apresentados de *Product Backlog*, *Sprint*, *Sprint Backlog* e *Sprint Review* das tarefas executadas.

5.1 HISTÓRIAS DE USUÁRIO

Dando início ao processo de desenvolvimento através dos *Sprints*, conforme descrito e definido no Capítulo quatro, o desenvolvimento do sistema seguiu as práticas e artefatos da metodologia Scrum. A Tabela 5.1.1 apresenta as *user stories* levantadas através de entrevistas e questionário.

No sistema bibliotecário, visando atender os requisitos do usuário e garantir a eficiência na utilização do sistema, são apresentados além das histórias de usuário, que assumem o papel de requisitos funcionais, os seguintes requisitos não funcionais na Tabela 5.1.2.

Definidos os requisitos do sistema, já se tem informações suficientes para iniciar o Scrum, e preparado para dar início à execução dos *Sprints*.

5.2 *PRODUCT BACKLOG*

Ao término do levantamento de requisitos, é possível definir uma lista de prioridades, primeiro esboço do *Product Backlog* (Tabela 5.2.1).

Com o *Product Backlog* definido, a etapa seguinte é a reunião de planejamento. Esta reunião torna-se impossível neste trabalho pelo fato do desenvolvimento não dispôr de uma equipe com integrantes para tal. Entretanto, nesta etapa foi definido por meio de uma reflexão, a quantidade de dias que cada tarefa deveria ocupar com base nas perspectivas de desenvolvimento

Tabela 5.1.1 – Histórias de Usuário

Histórias de Usuário	
HU01	O sistema deve permitir que seja possível cadastrar novos bibliotecários.
HU02	Como administrador do sistema, preciso cadastrar os alunos da escola.
HU03	Como administrador do sistema, preciso cadastrar os professores da escola.
HU04	Como administrador do sistema, preciso cadastrar os funcionários da escola.
HU05	Como administrador do sistema, preciso cadastrar os livros da biblioteca.
HU06	Como usuário, o sistema deve permitir a consulta de alunos.
HU07	Como usuário, o sistema deve permitir a consulta de professores.
HU08	Como usuário, o sistema deve permitir a consulta de funcionários.
HU09	Como usuário, o sistema deve permitir a consulta de livros.
HU10	O sistema deve permitir que eu altere ou exclua os leitores cadastrados.
HU11	O sistema deve permitir que eu altere ou exclua os livros cadastrados.
HU12	Desejo que os empréstimos sejam gerenciados apenas pelos bibliotecários.
HU13	Preciso que os leitores tenham acesso somente à consulta de livros e empréstimos.
HU14	Os empréstimos são feitos com prazo inicial de 07 dias, prorrogados igualmente.
HU15	Gostaria de escolher no empréstimo primeiro o livro, depois o leitor.
HU16	Seria interessante emitir relatórios simples de consultas executadas no sistema.

Fonte: Elaborado pelo autor

Tabela 5.1.2 – Requisitos não funcionais do Sistema

Requisitos Não Funcionais	
RNF01	O sistema deve estar disponível 24hrs por dia, 7 dias por semana, ininterruptos.
RNF02	O sistema deve ser utilizável para qualquer pessoa.
RNF03	A base de dados deve estar protegida, indisponível para usuários finais.
RNF04	O sistema pode ser executado em qualquer plataforma ou sistema operacional.
RNF05	O sistema deve executar igualmente em qualquer navegador (*browser*).
RNF06	O sistema é interativo o suficiente para dispensar treinamentos de rotinas.
RNF07	Devem ser escolhidas cores leves para não cansar ou poluir visualmente a tela.
RNF08	A velocidade de upload das páginas dependerá da banda de internet do usuário.
RNF09	O sistema será corrigido caso o usuário encontre algum erro de execução.

Fonte: Elaborado pelo autor

e capacidade de adaptações. O resultado foi um *Product Backlog* que agora tem suas estimativas de custo/dias apresentados na Tabela 5.2.2.

5.3 *SPRINTS*

Esta seção esboçará os *Sprints* realizados e os resultados obtidos com base no planejamento do *Product Backlog*.

Tabela 5.2.1 – Primeiro *Product Backlog*: Funcionalidades x Prioridade

Funcionalidade	Prioridade
Login e recuperação de senha	1
Cadastro e Gerenciamento de Usuários	2
Cadastro e Gerenciamento de Leitores	3
Cadastro e Gerenciamento do Acervo de Livros	4
Sistema de Empréstimo e Devoluções de Livros	5
Páginas de Consulta, Histórico e Acompanhamento de Empréstimos	6
Integração do sistema para gerar relatórios em PDF	7

Fonte: Elaborado pelo autor

Tabela 5.2.2 – Estimativa de Custo x Dia do *Product Backlog*

Funcionalidade	Prioridade	Custo/Dias
Login e recuperação de senha	1	10
Cadastro e Gerenciamento de Usuários	2	5
Cadastro e Gerenciamento de Leitores	3	21
Cadastro e Gerenciamento do Acervo de Livros	4	7
Sistema de Empréstimo e Devoluções de Livros	5	21
Páginas de Consulta, Hist. e Acomp. de Empréstimos	6	15
Integração do sistema para gerar relatórios em PDF	7	5

Fonte: Elaborado pelo autor

5.3.1 *Login* e recuperação de senha

A criação das telas iniciais do sistema para *login* e recuperação de senha foi o primeiro *Sprint* definido no *Product Backlog*. Aqui foi desenvolvido a tela inicial do sistema, onde o bibliotecário teria seu primeiro contato com o mesmo.

O *Sprint Backlog* é a subdivisão das etapas a serem desenvolvidas nesta *Sprint*. A Tabela 5.3.1 esclarece a ordem de execução da modelagem de dados.

Tabela 5.3.1 – *Sprint Backlog* do *Sprint* 1

Sprint 1	Prioridade	Custo/Dias
✓ *Login* e recuperação de senha	1	10
➢ Modelagem e criação do Banco de Dados	1.1	2
➢ Desenvolvimento da tela inicial do sistema	1.2	5
➢ Elaboração da recuperação de senha	1.3	3

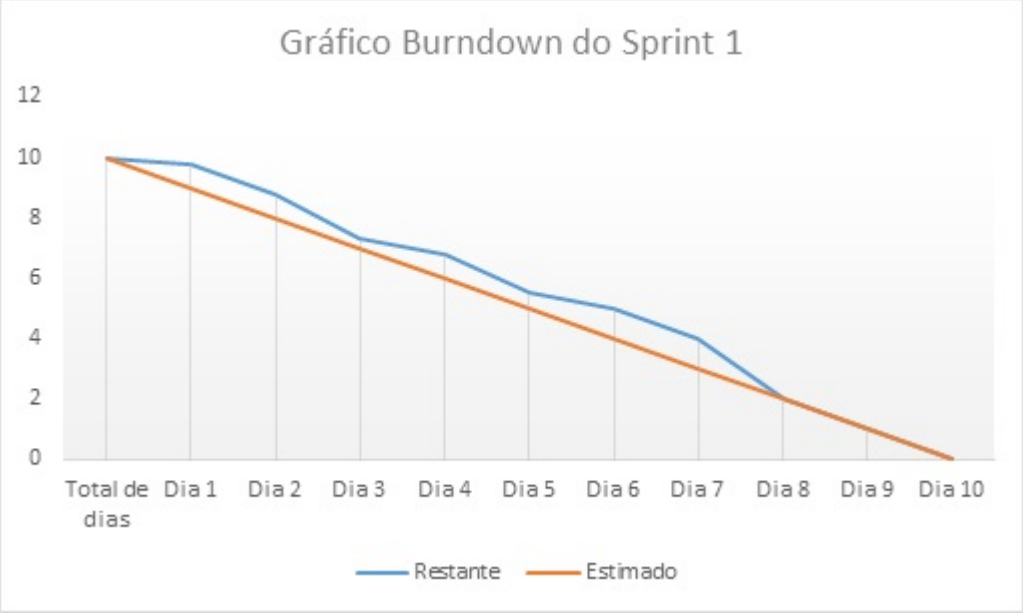
Fonte: Elaborado pelo autor

Ao final do ciclo do *Sprint* foram revisados todos os processos. Foi constatado que todas as especificações do *Sprint* foram produzidas. Os resultados obtidos nessa *Sprint* foram a tela inicial do sistema, uma página para recuperação de senha e o Banco de Dados.

5.3.1.1 *Burndown Chart* do *Sprint* 1

Como demonstra a Figura 5.3.1, nota-se um pequeno atraso no início da *Sprint*. Após definir como seria realizado o *login* no sistema e o modo como as senhas poderiam ser recuperadas através do *e-mail* cadastrado, a *Sprint* foi concluída dentro do prazo.

Figura 5.3.1 – *Burndown Chart* do *Sprint* 1



Fonte: Elaborado pelo autor

5.3.2 Cadastro e Gerenciamento de Usuários

Segundo *Sprint* do *Product Backlog*, o cadastro e gerenciamento de usuários tratou da forma como o usuário (bibliotecário) acessaria o sistema. Ainda sem um *layout* definido, foi desenvolvido páginas de Cadastro de Bibliotecário, alteração de dados, exclusão e métodos de segurança e conexão.

O custo estimado para concluir este *Sprint* foi exatamente o suficiente para entregar a funcionalidade “pronta”. Novamente foi definido o *Sprint Backlog* deste *Sprint*, subdividindo-o em tarefas menores, como pode ser observado na Tabela 5.3.2.

Tabela 5.3.2 – *Sprint Backlog* do *Sprint* 2

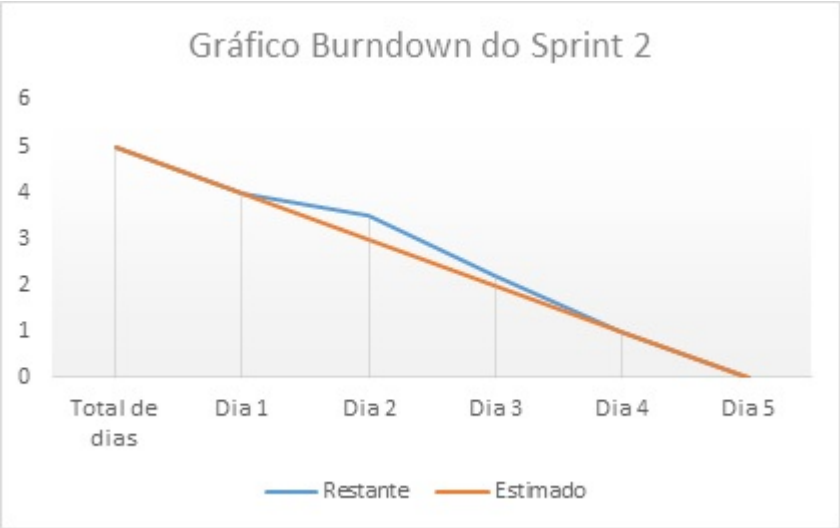
Sprint 2	Prioridade	Custo/Dias
✓ Cadastro e Gerenciamento de Usuários	1	5
➢ Criação do script para login	1.1	1
➢ Criação do script do painel administrativo	1.2	2
➢ Configurações de acesso ao Banco de Dados	1.3	1
➢ Criação do script para cadastro de usuário	1.4	1

Fonte: Elaborado pelo autor

5.3.2.1 *Burndown Chart* do *Sprint* 2

A Figura 5.3.2 que representa o gráfico do *Burndown* deste *Sprint* nos traz a afirmação das tarefas sendo executadas dentro do prazo estipulado, com um período um pouco maior para a criação do painel administrativo, que envolveu mais funções do que apenas cadastrar e logar no sistema.

Figura 5.3.2 – *Burndown Chart* do *Sprint* 2



Fonte: Elaborado pelo autor

Ao final do ciclo deste *Sprint*, foi realizado uma revisão de todos os processos executados. Foi possível verificar que toda a especificação inicial na *Sprint* foi produzida com sucesso. Neste caso, os seguinte *Backlog Sprint* foram concluídos: Elaboração do *script* para *login*, criação do painel administrativo, configurações de acesso ao banco de dados das páginas criadas, criação de cadastro para usuários administrativos (bibliotecários) e alteração de dados pelo administrador (já com acesso ao banco de dados).

5.3.3 Cadastro e Gerenciamento de Leitores

Dando início ao terceiro *Sprint*, o objetivo nesta etapa era desenvolver o cadastro e gerenciamento de leitores no sistema. Estes leitores, por sua vez, se subdividem em alunos, professores e funcionários. Sendo assim, seria preciso elaborar três diferentes cadastros no sistema, cada um com suas particularidades. O *Sprint Backlog* ficou representado da seguinte maneira na Tabela 5.3.3.

Tabela 5.3.3 – *Sprint Backlog* do *Sprint* 3

Sprint 3	**Prioridade**	**Custo/Dias**
✓ Cadastro e Gerenciamento de Leitores	1	21
➢ Criação dos três formulários de cadastro	1.1	10
➢ Conexão dos cadastros ao banco de dados	1.2	3
➢ Criação dos três formulários de alteração	1.3	7
➢ Criação dos scripts para exclusão de leitor	1.4	1

Fonte: Elaborado pelo autor

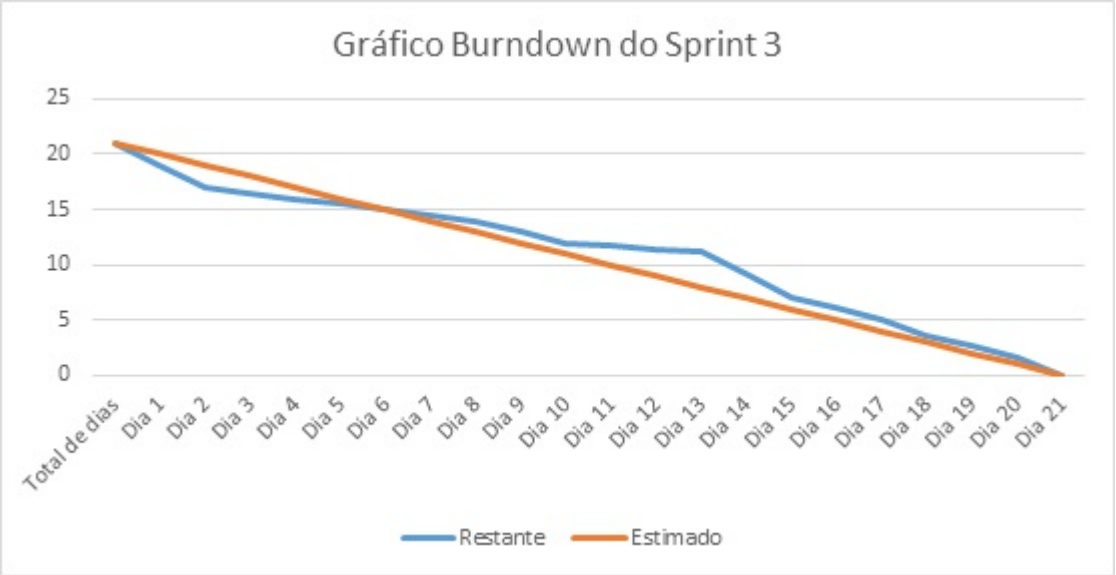
De acordo com a execução do *Sprint Backlog* deste *Sprint*, evidentemente que as telas de formulários de cadastro demandaram maior tempo. Embora os conhecimentos de HTML já fossem suficientes, e nesta fase ainda não houve uma preocupação em definir o *layout* final, foi definido um padrão tanto no alinhamento do formulário como tabela quanto no nome das variáveis de modo que facilitasse o trabalho mais adiante no sistema.

Além disso, houve uma preocupação com os dados que deveriam ser solicitados, e quais aqueles que seriam obrigatórios ou não para o Bibliotecário inserir. Nesta etapa vale ressaltar a solicitação dos Bibliotecários de que todos os cadastros seriam feitos por eles, e não pelos leitores. Com isso, cada formulário foi elaborado de acordo com as informações solicitadas nas entrevistas realizadas com os bibliotecários.

5.3.3.1 *Burndown Chart* do *Sprint* 3

A Figura 5.3.3 representa o *Burndown Chart* do *Sprint* 3, sem maiores dificuldades, mas com um tempo relativamente grande na primeira tarefa. Para compensar o atraso, houve uma aceleração na terceira etapa, pois assim que se finalizou uma tela de alteração, foi possível replicar a mesma aos outros leitores trabalhando somente suas variáveis.

Figura 5.3.3 – *Burndown Chart* do *Sprint* 3



Fonte: Elaborado pelo autor

Ao final do ciclo deste *Sprint*, foi realizado uma revisão de todos os processos executados. Foi possível verificar que toda a especificação inicial no *Sprint* foi produzida. Neste caso, os seguintes *Backlog Sprint* foram concluídos: criação dos formulários de cadastro para leitores (alunos, funcionários, professores e bibliotecários); conexão e execução dos formulários ao banco de dados; criação dos formulários para alteração de dados nas tabelas dos leitores e *scripts* para exclusão do leitor.

5.3.4 Cadastro e Gerenciamento do Acervo de Livros

O próximo *Sprint* a ser trabalhado foi o Cadastro e Gerenciamento de Livros. As tarefas executadas foram definidas e divididas no *Sprint Backlog* da Tabela 5.3.4.

Tabela 5.3.4 – *Sprint Backlog* do *Sprint* 4

Sprint 4	Prioridade	Custo/Dias
✓ Cadastro e Gerenciamento do Acervo de Livros	1	7
➢ Funcionalidade de cadastrar livro no acervo	1.1	3
➢ Funcionalidade de editar o cadastro do livro	1.2	3
➢ Funcionalidade de excluir um livro do acervo	1.3	1

Fonte: Elaborado pelo autor

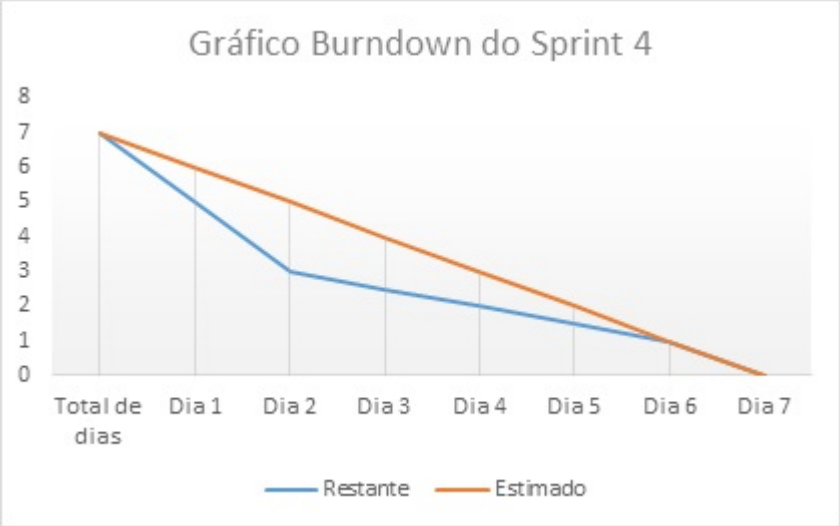
Esta talvez seja uma das funcionalidades mais importantes de todo o sistema, pois

aqui serão catalogados todo o acervo da biblioteca. Por este motivo, o Cadastro e Gerenciamento de Acervo de Livros está definido separadamente no *Product Backlog*.

5.3.4.1 *Burndown Chart* do *Sprint* 4

Observando o *Sprint Backlog* do *Sprint* 4, foi estabelecido prazos iguais para o cadastro de um livro e a edição de dados do mesmo. Contudo, como pode ser observado no gráfico do *Burndown Chart* do *Sprint* 4 na Figura 5.3.4, ganhou-se tempo nesta fase. Talvez pela prática recente com este tipo de tarefa (páginas de cadastro), o cadastro foi finalizado antes do prazo e aproveitou-se para dedicar mais à edição de dados por parte do administrador, visto a importância dessa funcionalidade para o sistema.

Figura 5.3.4 – *Burndown Chart* do *Sprint* 4



Fonte: Elaborado pelo autor

Finalizado o ciclo deste *Sprint*, foi realizado uma revisão de todos os processos executados. Novamente foi possível constatar que toda a especificação inicial no *Sprint* vinda do *Product Backlog* foi produzida. Neste caso, os seguintes *Backlog Sprint* foram concluídos: criação dos formulários de cadastro para o acervo de livros; armazenamento das informações cadastradas no banco de dados; criação dos formulários para alteração de dados de um livro cadastrado e funcionalidade para exclusão do livro.

5.3.5 Sistema de Empréstimo e Devolução de Livros

Definida como *Sprint* 5 do projeto, o sistema de empréstimo e devolução de livros desenvolvido é sem dúvidas o coração do sistema. Muito se refletiu e estudou a respeito da

melhor maneira de implementar essa funcionalidade. O *Sprint Backlog* desta fase é apresentado na Tabela 5.3.5.

Tabela 5.3.5 – *Sprint Backlog* do *Sprint* 5

Sprint 5	Prioridade	Custo/Dias
✓ Sistema de Empréstimo e Devolução de Livros	1	21
➢ Criação de janelas em Modal para execução	1.1	2
➢ Elaboração da página de empréstimo	1.2	2
➢ Funcionalidade de selecionar/pesquisar um livro para empréstimo	1.3	2
➢ Funcionalidade de empréstimo de livro para leitor	1.4	10
➢ Funcionalidade de renovação de empréstimo	1.5	3
➢ Funcionalidade de devolução de empréstimo	1.6	1
➢ Funcionalidade para cancelar um empréstimo	1.7	1

Fonte: Elaborado pelo autor

Com os cadastros do sistema finalizados, foi possível executar o *Sprint* 5, objetivo principal do sistema. Alguns problemas foram encontrados nesta tarefa, e a mesma demandou mais tempo do que o esperado. Foi preciso refazer os *scripts* em SQL que confirmariam um empréstimo de um livro a um leitor. Neste *Sprint* foi possível identificar uma escolha equivocada, porém não comprometedora no planejamento inicial do *Product Backlog*.

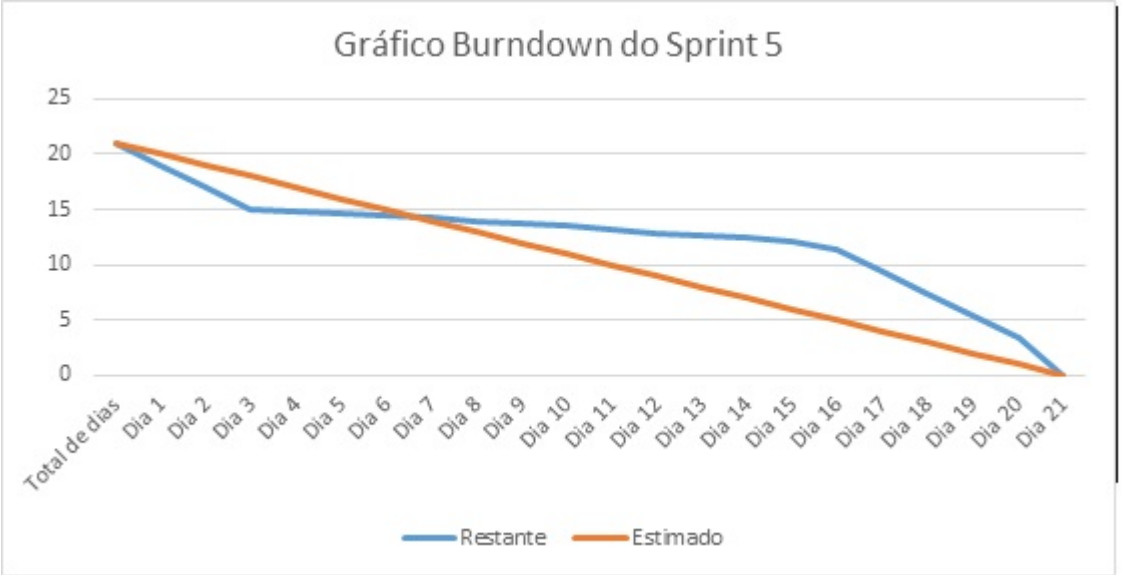
Ao elaborar o *Product Backlog* ficou definido que o sistema lidaria com cadastros de alunos, funcionários e professores separadamente. Talvez se tivesse sido aproveitado uma única classe com algumas condições especiais de cada caso, o ganho no tempo e uma maior facilidade de implementação seria evidente. O *script* básico para registrar um empréstimo de livros pode ser verificado no Apêndice E deste documento.

Todavia, antes do término do prazo da *Sprint* foi reescrito todo o código SQL responsável tanto por “emprestar” quanto por “devolver” um livro. O comando *UNION* da linguagem SQL solucionou este contratempo. O tempo gasto com a funcionalidade de empréstimo foi recuperado nas três tarefas seguintes.

5.3.5.1 *Burndown Chart* do *Sprint* 5

A Figura 5.3.5 comprova toda a dificuldade encontrada nesta *Sprint*. Por um momento, cogitou-se adiar o desenvolvimento de uma funcionalidade para outro *Sprint*, e se não fosse a solução rápida encontrada, seria difícil completá-la dentro do prazo.

Figura 5.3.5 – *Burndown Chart* do *Sprint* 5



Fonte: Elaborado pelo autor

5.3.6 Consultas, acompanhamentos e histórico de empréstimos

A *Sprint* 6 do projeto iniciou com a construção de páginas para consulta de leitores e livros. Foi definido o *Sprint Backlog* da tarefa conforme a Tabela 5.3.6.

Tabela 5.3.6 – *Sprint Backlog* do *Sprint* 6

Sprint 6	Prioridade	Custo/Dias
✓ Consultas, acompanhamentos e históricos	1	15
➢ Criação de páginas de consulta de leitores	1.1	2
➢ Criação de página de consulta de livros	1.2	2
➢ Criação de página de consulta de livros emprestados	1.3	2
➢ Funcionalidade para acompanhar devoluções e prazos vencidos de devolução	1.4	5
➢ Elaboração de janelas em *Modal* para visualizar informações de leitores e livros	1.5	4

Fonte: Elaborado pelo autor

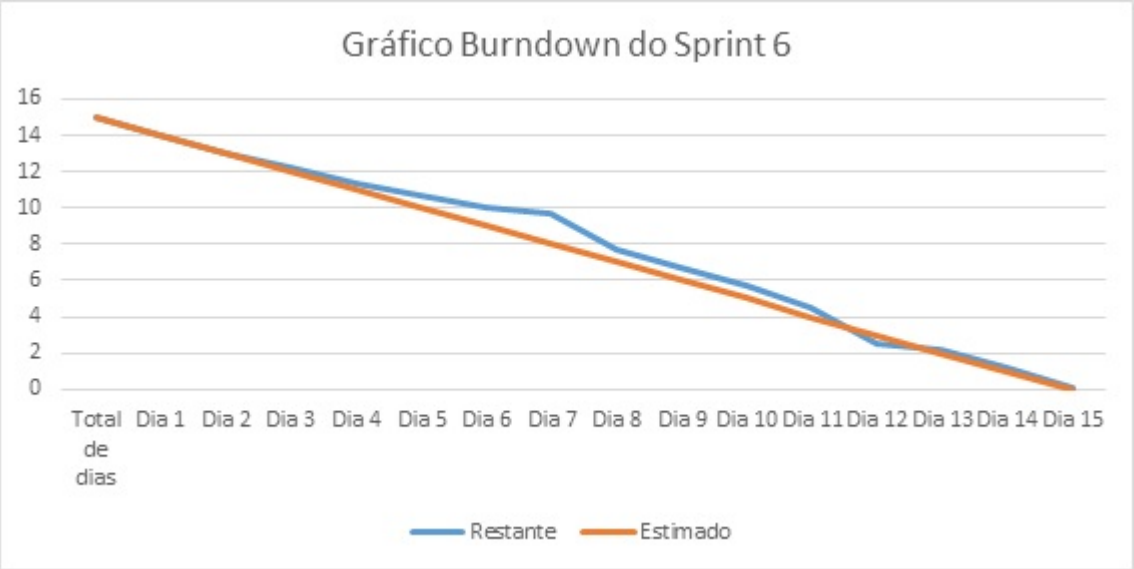
Ao realizar a Revisão do que foi executado no *Sprint Backlog* do *Sprint* 6, verificou-se que as consultas realizadas no sistema são de suma importância para os usuários. As tarefas foram cuidadosamente desenvolvidas com o objetivo de facilitar o uso de qualquer usuário. As janelas em *Modal* deram uma visão resumida dos dados que seriam mais importantes levantados pelos bibliotecários.

O custo em dias levado para executar essa *Sprint* foi considerável, tendo em vista o sucesso atendido nos requisitos propostos. Foi entregue nesta etapa um menu de consultas pronto, onde é possível pesquisar alunos, funcionários, professores, livros, empréstimos e devoluções. Ficou “pendente” apenas a estilização das páginas para uma melhor interação de interface do usuário com o sistema, mas nada considerado um problema, visto que é a última prioridade do *Product Backlog*.

5.3.6.1 *Burndown Chart* do *Sprint* 6

Definidos as consultas executadas no sistema no *Sprint Backlog* do *Sprint* 6 e finalizado seu desenvolvimento, a Figura 5.3.6 mostra um gráfico com resultados satisfatórios e sem problemas encontrados nesta etapa.

Figura 5.3.6 – *Burndown Chart* do *Sprint* 6



Fonte: Elaborado pelo autor

5.3.7 Integração do Sistema para gerar relatórios PDF

Com o *Product Backlog* se encerrando, o *Sprint* 7 do projeto dá início à execução de funcionalidades voltadas à visualização do sistema por parte do usuário. A decisão por exportar as consultas do sistema em um formato de arquivo PDF atende aos requisitos levantados como histórias de usuário, pois atende à necessidade do cliente referente aos relatórios.

O *Sprint Backlog* deste *Sprint* é apresentado na Tabela 5.3.7.

Durante a execução deste *Sprint*, ao realizar a Revisão do que foi executado no *Sprint Backlog*, verificou-se que as funcionalidades requeridas foram atendidas. Como resultado, o

Tabela 5.3.7 – *Sprint Backlog* do *Sprint* 7

Sprint 7	Prioridade	Custo/Dias
✓ Integração do sistema para gerar relatórios PDF	1	5
➢ Criação dos botões imprimir nas telas necessárias	1.1	0,5
➢ Implantação da classe DOMPDF em novas páginas para impressão	1.2	2,5
➢ Formatação dos relatórios em PDF.	1.3	2

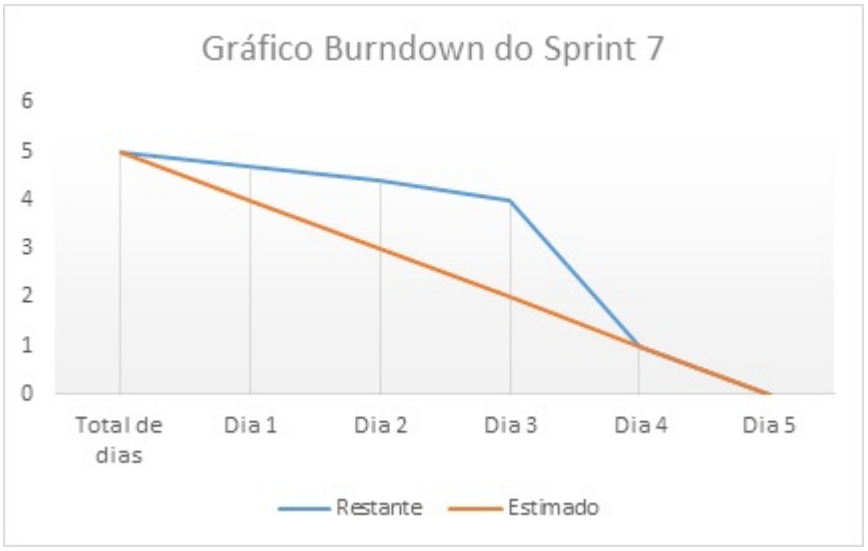
Fonte: Elaborado pelo autor

sistema agora é capaz de emitir um arquivo PDF com o resultado de uma busca selecionada.

5.3.7.1 *Burndown Chart* do *Sprint* 7

O início deste *Sprint* foi um pouco lento. A Figura 5.3.7 mostra que até o início do terceiro dia havia um grande atraso, e por vezes, cogitou-se adiar o segundo requisito "Implantação da classe DOMPDF em novas páginas para impressão" para um próximo *Sprint*.

Figura 5.3.7 – *Burndown Chart* do *Sprint* 7



Fonte: Elaborado pelo autor

Tal fato se deu pela dificuldade em apurar a causa de erros que surgiam constantemente no momento de carregar a página PDF. Apenas a mensagem de "Erro ao abrir o arquivo PDF" não era muito intuitiva, e foi preciso elaborar uma nova página que estivesse executando normalmente para comparar linha por linha do código e identificar o problema, que era um erro na declaração de variável a ser exibida.

Contudo, a partir do terceiro dia foi atingido o objetivo de imprimir uma consulta em PDF com a classe DOMPDF sem erro. A partir deste momento, restaram apenas ajustes, e o *Sprint* foi finalizado exatamente dentro do prazo.

5.4 MODELAGEM DE DADOS

A modelagem de dados teve início com a criação do Banco de Dados na *Sprint* 1 definida no *Product Backlog*. A ela se aplica todas as atividades relacionadas à criação do Banco de Dados, tais como a definição de dados, criação e organização de classes e relacionamentos entre tabelas. Por não se tratar de uma funcionalidade do sistema, sua classificação não se aplica como *Sprint Backlog*.

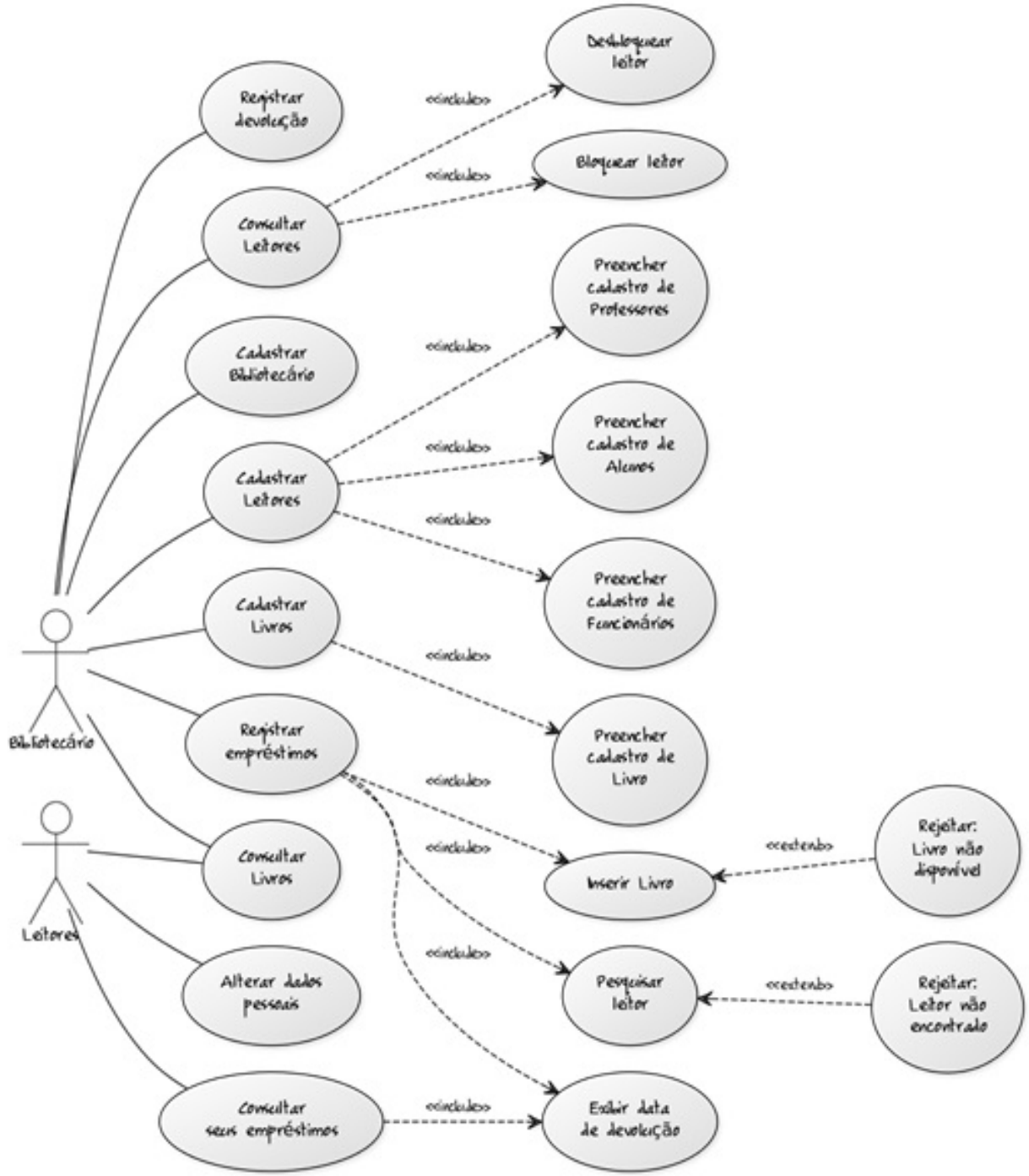
Sendo assim, trata-se neste trabalho de um produto do resultado final obtido. Uma vez que com as metodologias ágeis torna-se complicado a elaboração de diagramas a serem seguidos fielmente, dado sua capacidade de iteração e adaptação ao longo do desenvolvimento, ao término das *Sprints* foi possível definir os diagramas do sistema.

5.4.1 Diagrama de Casos de Uso do Sistema

Estando o sistema definido com dois tipos de atores, o bibliotecário que administrará todas as rotinas e o leitor, que poderão ser os alunos, professores e funcionários da escola, e atendendo às histórias de usuário expostas em forma de Funcionalidades no *Product Backlog* do projeto, foi elaborado ao término dos *Sprints*, o seguinte diagrama de caso de uso, apresentado na Figura 5.4.1.

A Figura 5.4.1 apresenta as ações possíveis que um bibliotecário pode executar no sistema, tais como: cadastrar bibliotecário, leitores, livros, registrar empréstimos e devoluções, bloquear e desbloquear leitores e realizar pesquisas no sistema. Já os leitores, possuem acesso à atualização dos dados de seu perfil, consulta de livros e empréstimos correntes, com a informação da data de devolução dos mesmos.

Figura 5.4.1 – Diagrama de Casos de Uso do Sistema



Fonte: Elaborado pelo autor

5.4.2 Diagrama de Classes do Sistema

O relacionamento entre as classes é demonstrado no Diagrama de Classes do Sistema. A Figura 5.4.2 apresenta os relacionamentos existentes.

Figura 5.4.2 – Diagrama de Classes do Sistema

Fonte: Elaborado pelo autor

Terminados os *Sprints*, os resultados do produto final obtido podem ser visualizados no capítulo sete e nos apêndices, onde o sistema Biblioteca JK é apresentado por meio de suas telas e comentários.

Capítulo 6

BIBLIOTECA JK E RESULTADOS

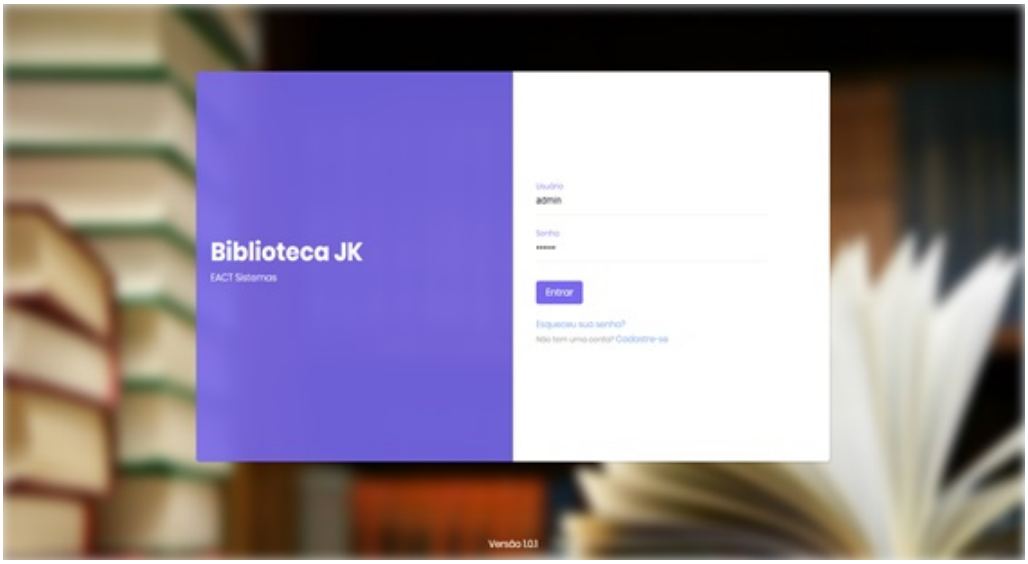
O objetivo deste capítulo é apresentar os resultados obtidos com a metodologia Scrum no desenvolvimento do sistema Biblioteca JK. Aqui serão apresentados os resultados obtidos em cada *Sprint* realizada no Capítulo anterior.

6.1 TELAS DO SISTEMA

Uma das formas de exibir resultados, é apresentar as telas do sistema. Para uma melhor organização dos resultados, as telas principais serão apresentadas nesta seção, enquanto as demais serão exibidas no Apêndice D.

6.1.1 Tela inicial: Bem-vindo ao Biblioteca JK

Figura 6.1.1 – Tela inicial do sistema



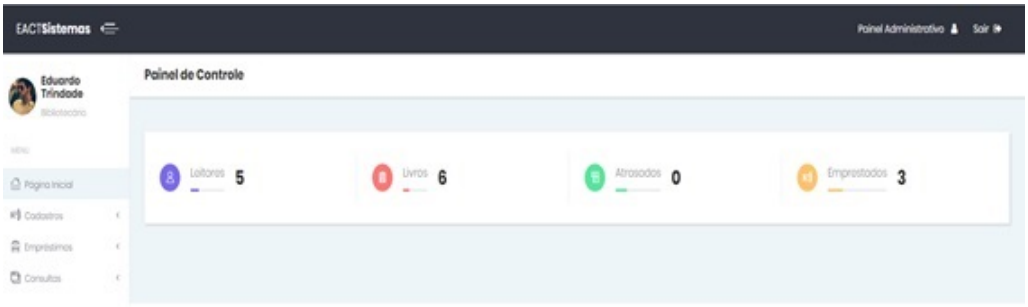
Fonte: Elaborado pelo autor

Ao executar o Biblioteca JK, o usuário se depara com uma tela de login, onde deverá informar seu usuário e senha, definidos no momento do cadastro feito pelo bibliotecário. A Figura 6.1.1 apresenta a tela inicial do sistema.

6.1.2 *Home* e *Dashboard*

Página inicial (Figura 6.1.2) do sistema após efetuar o *login*. O bibliotecário possui um painel de controle para acompanhar a quantidade de cadastros e empréstimos em seu sistema.

Figura 6.1.2 – *Home* e *Dashboard*

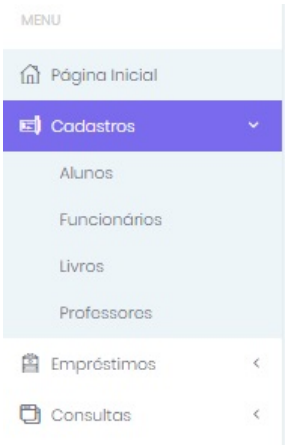


Fonte: Elaborado pelo autor

6.1.3 Cadastros

O cadastro de bibliotecários (administradores) é feito acessando o painel administrativo e clicando em "Cadastrar Bibliotecário". Sua tela pode ser vista na Figura D.0.3 Apêndice D). Já o menu de cadastros do sistema é subdividido em quatro opções. Aqui, o usuário decide se deseja cadastrar um aluno, um funcionário, um livro, ou um professor. A Figura 6.1.3 apresenta o menu Cadastros expandido.

Figura 6.1.3 – Menu do *Dashboard* (Cadastros expandido)



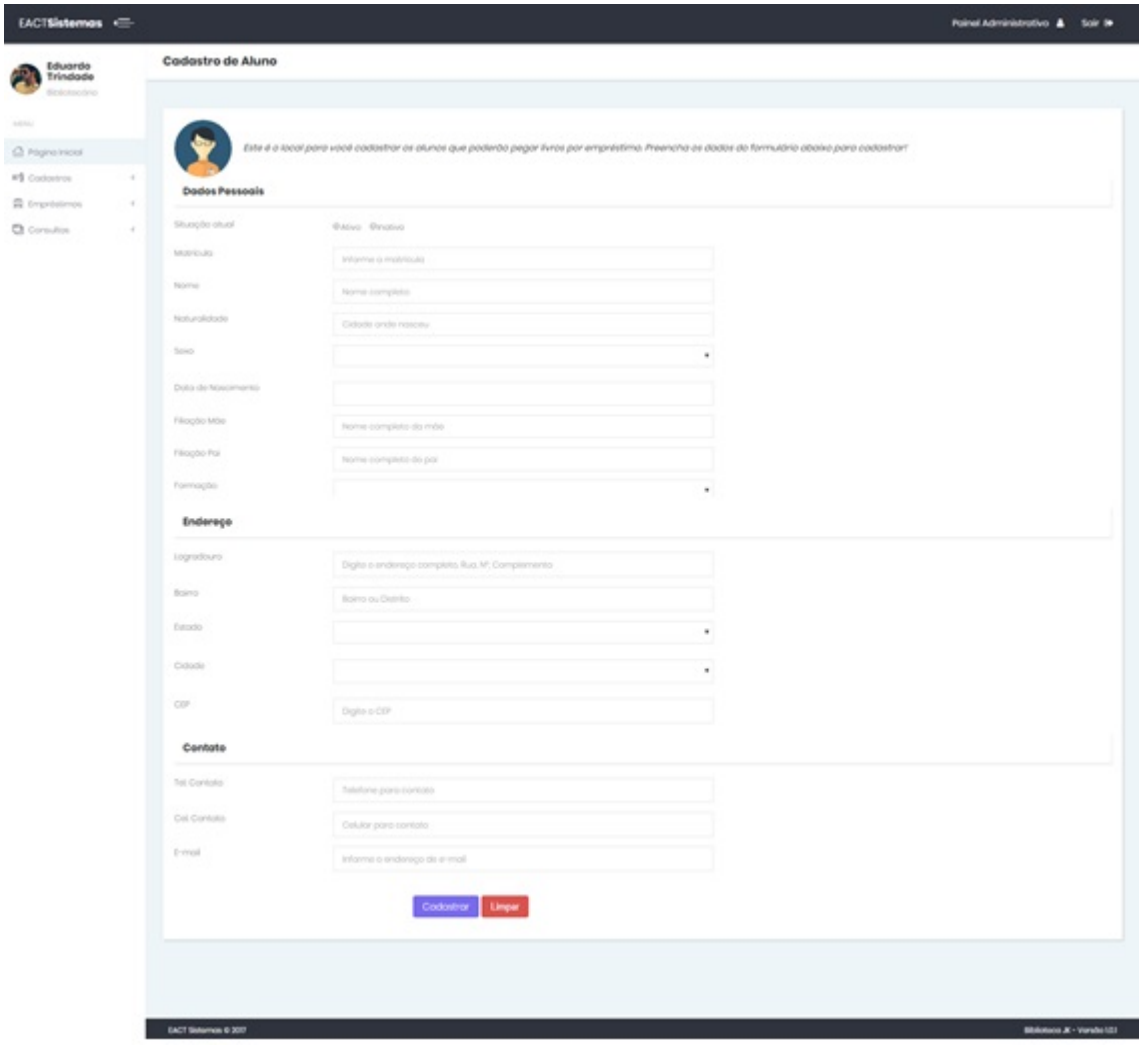
Fonte: Elaborado pelo autor

6.1.3.1 Cadastro de Leitores

O cadastro dos leitores é composto por três etapas que o bibliotecário deverá preencher. Inicialmente, os dados pessoais do leitor como: nome, sexo, data de nascimento, filiação e formação. Posteriormente, informam-se os dados referente ao endereço do leitor. Por fim, telefones e *e-mail* para contato.

Considerando diferenças mínimas nas telas de cadastro entre alunos, funcionários e professores, a Figura 6.1.4 ilustra o resultado final desta etapa.

Figura 6.1.4 – Cadastro de Leitor (exemplo: alunos)



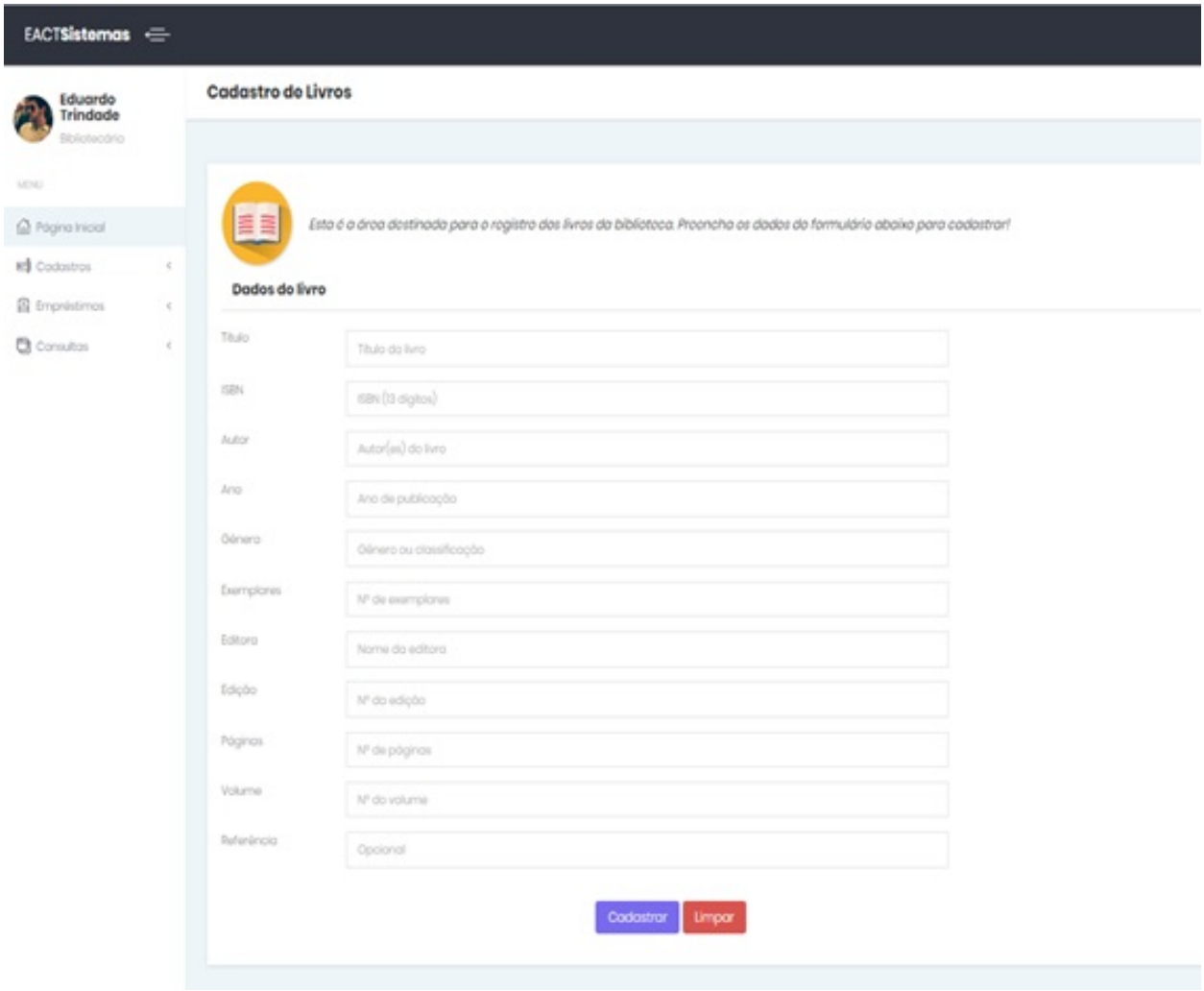
Fonte: Elaborado pelo autor

Ao término do cadastro, o bibliotecário deverá clicar no botão "Cadastrar"para confirmar a gravação das informações fornecidas. O botão "Limpar"oferece a opção de reiniciar o cadastro do início, apagando assim, informações já digitadas até o momento.

6.1.3.2 Cadastro de Livros

O cadastro de livros (Figura 6.1.5 é bastante simples. Neste cadastro, basta ao bibliotecário informar os dados do livro que deseja cadastrar, sendo 11 no total: título, ISBN, autor, ano, gênero, número de exemplares, editora, edição, quantidade de páginas, volume e uma referência (opcional). Com o livro em mãos, é possível obter todas essas informações facilmente.

Figura 6.1.5 – Cadastro de Livro



Fonte: Elaborado pelo autor

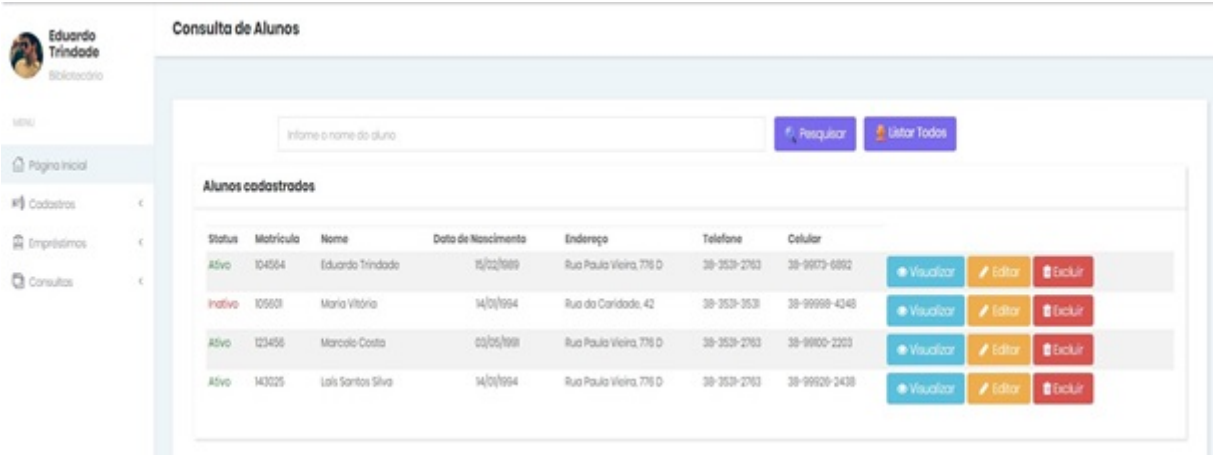
6.1.4 Consultas

O menu de consultas do sistema subdivide-se em três grupos: Livros, Leitores e Empréstimos. No primeiro, ao clicar em Livros o usuário será redirecionado a uma página de pesquisa dos livros cadastrados. O grupo Leitores abrange alunos, professores e funcionários cadastrados, podendo pesquisá-los separadamente. Já o grupo Empréstimos, possibilita ao usuário consultar os Livros atualmente emprestados e os prazos de devolução para os mesmos.

6.1.4.1 Consulta de Leitores

A consulta dos leitores segue uma mesma regra para alunos, professores e funcionários. A Figura 6.1.6 traz para o bibliotecário os alunos cadastrados, podendo visualizar e imprimir a ficha de cada um, editar informações de seu cadastro, ou excluí-los da base de dados.

Figura 6.1.6 – Consulta de Leitor (ex: alunos)



Fonte: Elaborado pelo autor

A visualização de dados de um leitor, traz uma ficha apresentada em uma janela *modal*[1]. Assim como na consulta de livros, o bibliotecário pode optar por imprimir os dados deste leitor.

Tanto os leitores quanto os livros possuem opções de visualização, edição e exclusão.

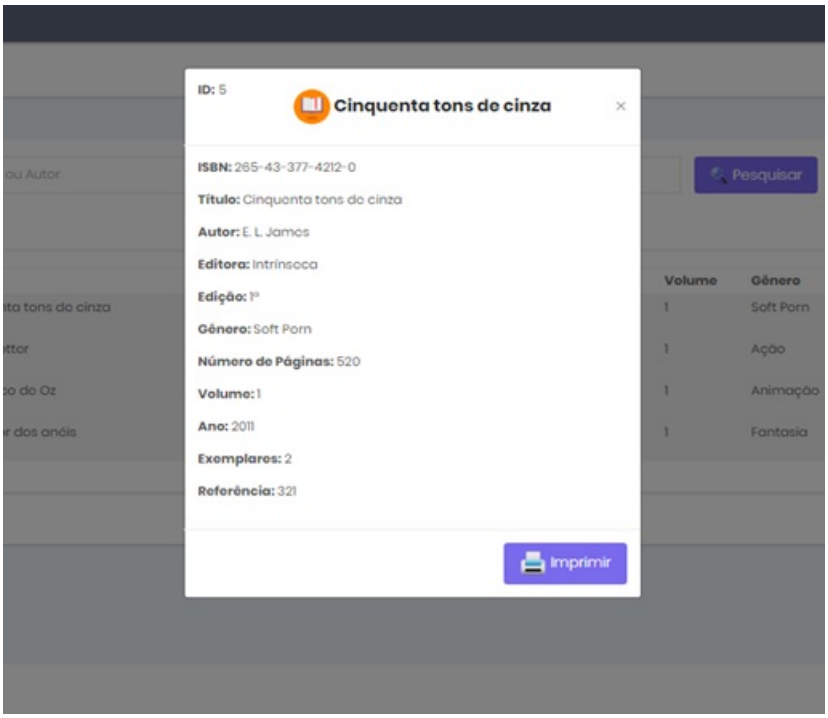
6.1.4.2 Consulta de Livros

A tela de consulta de livros permite ao usuário pesquisar um livro tanto pelo título quanto pelo autor. Há também a opção de listar todos os livros cadastrados. Cada livro é exibido em uma linha de uma tabela, permitindo ao bibliotecário três ações sobre ele: visualizar, editar ou excluir.

Ao visualizar um livro, uma janela no estilo *modal* é apresentada (Figura 6.1.7 com as informações do livro escolhido. O sistema permite que o bibliotecário imprima essas informações (gerando um arquivo PDF) que poderá ser utilizado da forma que bem entender para manter seus livros catalogados também de maneira impressa.

[1] Uma janela modal não perde foco enquanto estiver aberta. Isto significa que, depois de aberta, o utilizador deve necessariamente interagir com ela a fim de fechá-la, para então voltar a usar o sistema. Fonte: https://pt.wikipedia.org/wiki/Janela$(informtica).Acessoem12/12/2017$.

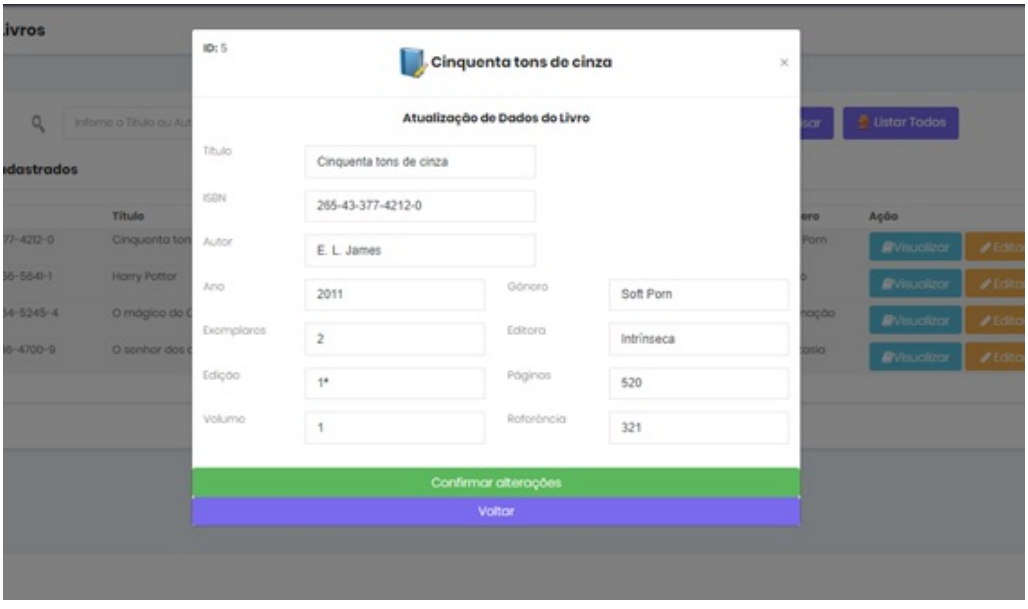
Figura 6.1.7 – Janela *Modal* para visualização de um Livro



Fonte: Elaborado pelo autor

Caso o bibliotecário opte por editar os dados de um livro, uma janela será aberta com os dados do livro selecionado, permitindo que se altere as informações que desejar (Figura 6.1.8). Clicando em “Confirmar alterações”, os dados do livro serão gravados/atualizados.

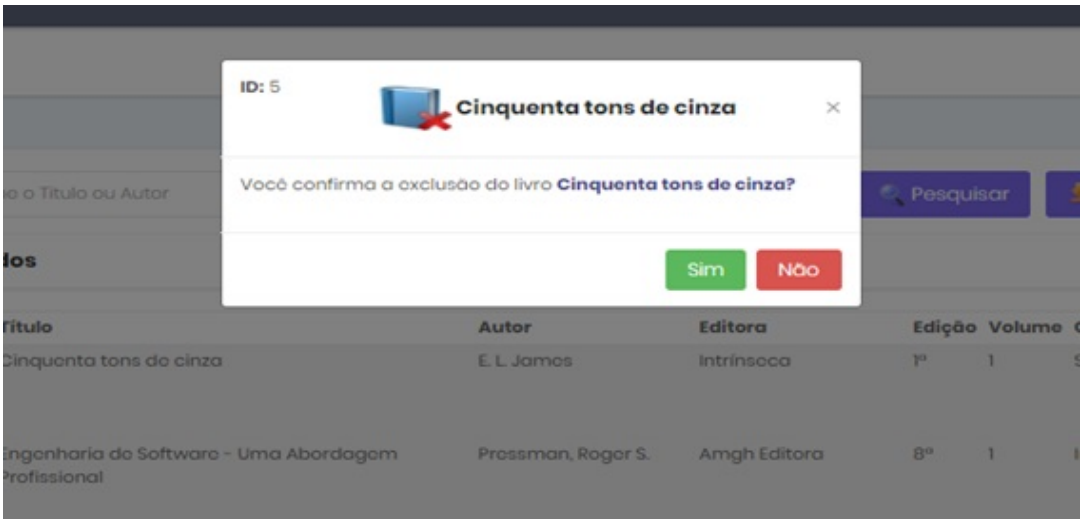
Figura 6.1.8 – Tela de edição dos dados de um Livro



Fonte: Elaborado pelo autor

A terceira funcionalidade disponível na consulta trata da exclusão de um livro. Selecionando esta opção, um alerta será emitido através de uma janela *modal* confirmando se o bibliotecário deseja realmente excluir aquele livro (Figura 6.1.9).

Figura 6.1.9 – Tela de exclusão de um Livro

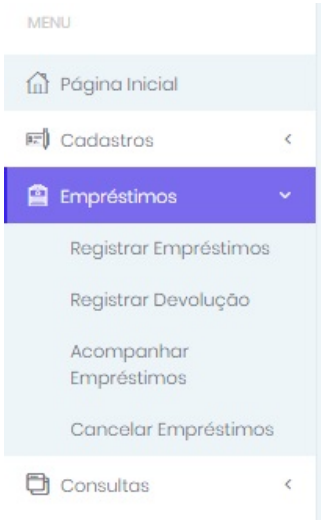


Fonte: Elaborado pelo autor

6.1.4.3 Empréstimos e Devoluções

Em um sistema bibliotecário, a função de emprestar um livro é sem dúvidas a chave para o perfeito funcionamento do sistema. Tal rotina pode ser desenvolvida de distintas maneiras, de acordo com a necessidade do cliente ou usuário. O menu de empréstimos e devoluções é apresentado na Figura 6.1.10.

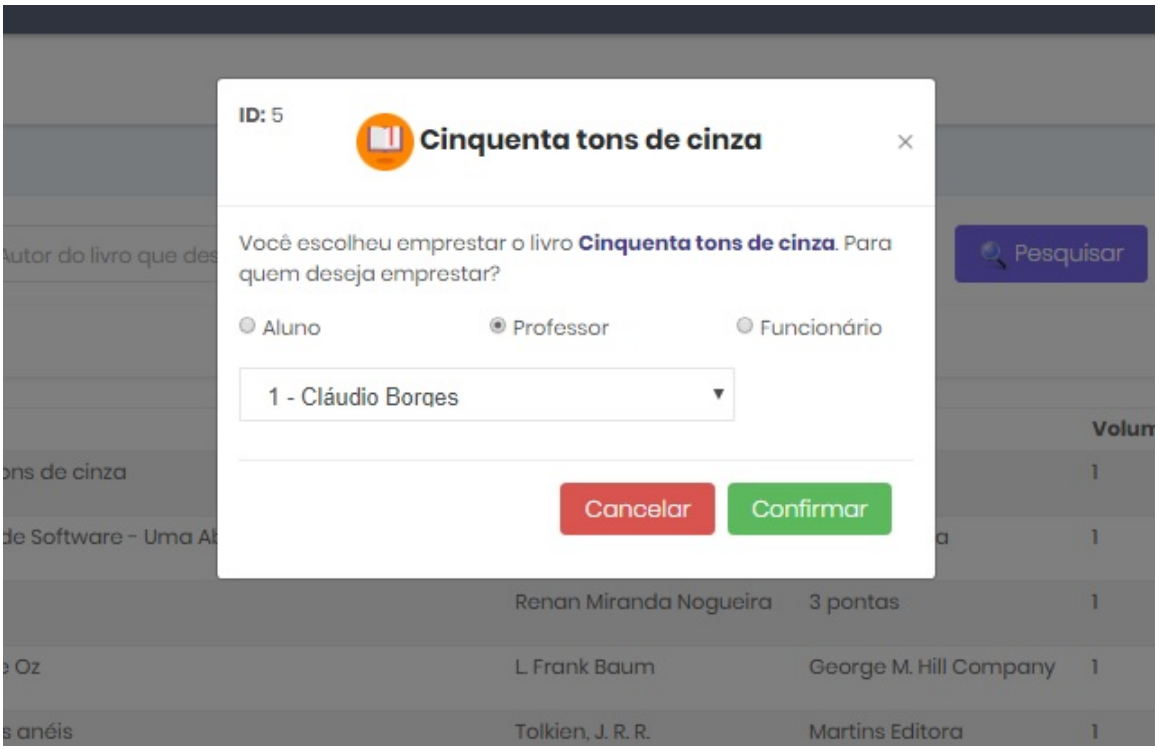
Figura 6.1.10 – Menu do *Dashboard* (Empréstimos expandido)



Fonte: Elaborado pelo autor

No Biblioteca JK, os livros são emprestados selecionando primeiro a obra, e em seguida, o leitor. O caminho a ser seguido é semelhante para a consulta de livros e leitores, uma vez que o resultado será uma consulta de livro (Empréstimo -> Registrar empréstimo). Com o resultado exibido na tela, o bibliotecário terá duas opções: visualizar o livro e emprestá-lo. Ao clicar no botão "Emprestar livro", uma janela *modal* surge na tela para que se escolha o leitor que irá receber o livro por empréstimo. A Figura 6.1.11 mostra o livro sendo emprestado para um professor selecionado.

Figura 6.1.11 – Empréstimo de Livro



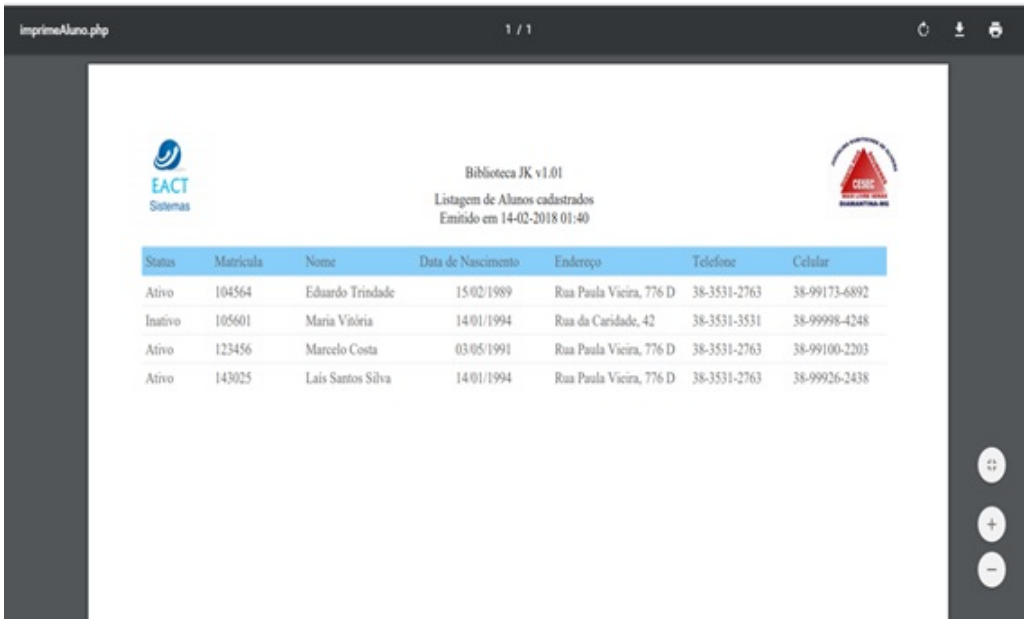
Fonte: Elaborado pelo autor

Para registrar a devolução de um livro no sistema, basta acessar o menu de Empréstimos e selecionar a opção "Registrar Devolução". Poderá ser pesquisado um livro ou um leitor que está com empréstimo vigente. Clicando em "Devolver livro"o empréstimo é finalizado.

6.2 ARQUIVOS DE SAÍDA

Além do resultado com as funcionalidades exibidas nas telas da seção anterior, outro resultado alcançado foi a emissão de relatórios em PDF, a fim de fornecer um material diferente ao bibliotecário. Um exemplo de relatório emitido é a Figura 6.2.1, exibido após uma consulta de alunos.

Figura 6.2.1 – Modelo de relatório impresso em PDF



Fonte: Elaborado pelo autor

Embora apresente ainda um *layout* simples, o relatório em PDF transmite ao usuário exatamente aquilo que ele consultou no sistema, facilitando sua divulgação dentro da instituição de ensino.

Capítulo 7

CONCLUSÃO

Este capítulo visa apresentar as considerações finais com a experiência deste trabalho e sugerir possíveis melhorias para o futuro.

7.1 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo propor uma solução para gerenciar o acervo de livros da Biblioteca do CESEC Juscelino Kubitschek de Oliveira de Diamantina/MG, através do desenvolvimento de um Sistema de Informação baseado na *web*.

A metodologia escolhida para acompanhar o desenvolvimento do trabalho gerenciando os processos foi o Scrum. Esta metodologia ágil se mostrou eficaz nas etapas aplicadas, contribuindo para uma organização e cumprimento das tarefas.

Inicialmente, encontrou-se dificuldade em iniciar a fase de implementação, pois foi preciso entender e definir o modo como os cadastros agiriam no sistema. A necessidade da biblioteca em questão contribuiu para que, após verificar os requisitos do sistema, a elaboração dos diagramas viesse a esclarecer o modo como as classes trabalhariam. Contudo, definido esta etapa, o desenvolvimento foi ágil e eficaz de acordo com o proposto.

Com a metodologia Scrum em prática, as etapas foram sendo realizadas sem maiores dificuldades, exceto pela confirmação de como seriam feitos os empréstimos no sistema. Neste momento, foi preciso um tempo um pouco maior para definir desde a elaboração de telas como consultas realizadas. O *Product Backlog* se mostrou fundamental nesta etapa, compreendendo sua importância dentro do planejamento.

Com o sistema finalizado, foi satisfatório perceber o interesse dos servidores do CESEC em cumprir com os objetivos propostos. A informatização nas instituições de ensino já se tornou uma situação real no âmbito educacional. Todavia, nem todas as bibliotecas possuem um

controle de acervo informatizado e virtual, pois não dispõem dessa ferramenta. Compreende-se e reitera a importância que este fato deve ser levado em consideração, pois suas vantagens superam as dificuldades.

Por fim, o desenvolvimento deste trabalho contribuiu para uma maior aprendizagem no planejamento de desenvolvimento de *software*, aliando teoria à prática, seguindo uma metodologia adequada ao processo. O trabalho foi finalizado com o sentimento de contribuição para a evolução da Biblioteca atendida, compreendendo e interagindo com sua cultura organizacional.

7.2 TRABALHOS FUTUROS

Para trabalhos futuros, a proposta é expandir o sistema e desenvolver a possibilidade de solicitação de empréstimo feita por qualquer usuário, bem como a reserva de livros, a fim de possibilitar o uso do sistema em outras instituições. Pretende-se também a realização de testes de usabilidade com os usuários, após uma maior familiarização com o sistema, o que possibilitará a implementação de melhorias funcionais e visuais. Além disso, uma validação dos requisitos poderia identificar novas funcionalidades ou possíveis correções na versão atual.

REFERÊNCIAS

AGUIAR, P. H. **Sistema de Informação para Gestão Educacional**: sistematização de uma proposta de modelo e avaliação do processo de sua construção. Dissertação do Curso de Mestrado Integrado Profissional em Computação do Centro de Ciência e Tecnologia. Fortaleza: Universidade Estadual do Ceará, 2004.

ARAÚJO, A. H. M. **Web Design Responsivo**. Curso de curta duração ministrado/Outra: [s.n.], 2015.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML, Guia do Usuário**. Tradução: Fábio Freitas da Silva. Rio de Janeiro: Campus, 2012.

CHELI, L. A. **E-Commerce Rede Onnix de Farmácias**. Trabalho de conclusão de curso (superior de tecnologia em Análise e Desenvolvimento de Sistemas). Ivaiporã: Faculdades Integradas do Vale do Ivaí, 2013.

CONVERSE, T.; PARK, J. **PHP: a Bíblia**. Tradução da 2ª ed. original de Edson Furmankiewicz. Rio de Janeiro: Elsevier, 2003.

COSTA, T. C.; GOMES, F. D.; CAGNIN, M. I. **Estudo para Adaptação de um Processo Ágil de Desenvolvimento baseado em Framework para apoiar o Desenvolvimento de Software baseado em Modelos**. Marília-SP: [s.n.], 2007.

DAMASIO, E.; RIBEIRO, C. E. N. **Software livre para bibliotecas, sua importância e utilização: o caso Gnuteca**. Campinas: Unicamp, 2006. Disponível em: <https://periodicos.sbu.unicamp.br/ojs/index.php/rdbci/article/view/2036>. Acesso em: 18 jul. 2017.

DATE, C. J. **Introdução a sistemas de banco de dados**. Rio de Janeiro: Elsevier, 2003.

FERREIRA, E.; EIS, D. **HTML5 e CSS3 com farinha e pimenta**: Desenvolvimento cliente-side inteligente. São Paulo: Tableless, 2012.

FILHO, W. d. P. P. **Engenharia de Software: fundamentos, métodos e padrões**. 3ª ed. Rio de Janeiro: LTC, 2009.

GOODMAN, D. **JavaScript: a Bíblia**. 5ª tiragem. Tradução de Daniel Vieira. Rio de Janeiro: Elsevier, 2001.

GUEDES, G. T. A. **UML: Uma abordagem prática**. São Paulo: Novatec, 2006.

LEITAO, M. d. V. **Aplicação de Scrum em Ambiente de Desenvolvimento de Software Educativo**. Monografia (Bacharel em Engenharia da Computação). Recife: Universidade de Pernambuco, 2010.

LOWE, D.; PRESSMAN, R. S. **Engenharia Web**. Rio de Janeiro: LTC ed, 2009.

MACHADO, J. P. d. F. **Desenvolvimento de um sistema web para gerenciamento do acervo multimídia do Instituto Casa da Glória de Diamantina, MG**. Trabalho de conclusão de curso (bacharelado em Sistemas de Informação). Diamantina: Universidade Federal dos Vales do Jequitinhonha e Mucuri, 2014.

PRESSMAN, R. S. **Engenharia de Software: uma abordagem profissional**. 7ª ed. Porto Alegre: AMGH: Dados eletrônicos, 2011.

PROSTT, M. E. **Interface Web utilizando Design Responsivo**: um estudo de caso aplicado a smartphones, tablets e televisores. Monografia (Especialização em Tecnologias Java e Desenvolvimento para Dispositivos Móveis). Curitiba: Universidade Tecnológica Federal do Paraná, 2013.

RIBEIRO, R. D.; RIBEIRO, H. d. C. e. S. **Gerenciamento de projetos com métodos ágeis**. 1ª ed. Rio de Janeiro: Direito Editorial, 2015.

SCHWABER, K.; JEFF, S. **O Guia do Scrum**. [s.n.], 2018. Disponível em: <https://www.scrum.org>. Acesso em: 12 jan. 2018.

SCHWABER, K.; SUTHERLAND, J. **Guia do ScrumMR. Um guia definitivo para o Scrum**: As regras do jogo. Oferecido por licença sobre a Attribution Share-Alike da Creative Commons: [s.n.], 2017. Disponível em: <http://creativecommons.org/licenses/by-sa/4.0/legalcode>. Acesso em: 15 jan. 2018.

SILVA, A. M. R. d.; VIDEIRA, C. A. E. **UML, Metodologias e Ferramentas CASE**. 1ª ed. Coleção: Tecnologias: Inova, 2001.

SOARES, W. **PHP 5: conceitos, programação e integração com banco de dados**. 6ª ed. São Paulo: Érica, 2012.

SOMMERVILLE, I. **Engenharia de Software**. 8ª ed. São Paulo: Pearson Addison Wesley, 2007.

SOMMERVILLE, I. **Engenharia de Software**. 9ª ed. São Paulo: Pearson Prentice Hall, 2011.

SOUZA, O. R. d. **Processos de apoio ao desenvolvimento de aplicações web**. Dissertação (Mestre em Ciências da Computação e Matemática Computacional. São Carlos-SP: USP, 2005.

SUEHRING, S. **MySQL, a Bíblia**. Tradução Edson Furmankiewicz. Rio de Janeiro: Elsevier, 2002, 6ª reimpressão.

SUTHERLAND, J. S. **The art of doing twice the work in half the time**. Tradução de Natalie Gerhardt. São Paulo: LeYa, 2014.

THOMSON, L.; WELLING, L. **PHP e MySQL: desenvolvimento Web**. Rio de Janeiro: Elsevier, 2005.

Apêndice A

Termo de Abertura do Projeto

EACT Sistemas	**Sistema de Gerenciamento para Biblioteca** **Termo de Abertura do Projeto**

Elaborado por:	Eduardo Augusto Costa Trindade	**Versão:**	1.0
Aprovador por:	Claudia Beatriz Berti	**Data de aprovação:**	02/06/2017

1. Informações Gerais

Gerente do Projeto	Eduardo Augusto Costa Trindade	**Email/Telefone:**	+55 38 9 9173-6892 eduardoepalmeiras@gmail.com

2. Justificativa do Projeto

Diante da necessidade constatada no CESEC Juscelino Kubitschek de Oliveira, no município de Diamantina-MG, de um sistema de informação para controle da biblioteca, surge a oportunidade de desenvolver o Trabalho de Conclusão de Curso de Graduação em Sistemas de Informação da UFVJM por meio do desenvolvimento deste software que poderá auxiliar o controle diário deste setor.
A motivação para a realização deste projeto é incentivar mais experiências como essa dentro do município e da Universidade, oferecendo um trabalho eficaz, de qualidade e que atinja os requisitos necessários ao bom desempenho do setor em questão.

3. Objetivos do Projeto

Oferecer à escola o controle de sua biblioteca por meio de um sistema web, de modo que o bibliotecário possa administrar seu acervo de livros e situações de empréstimos ou reservas, automaticamente. Atualmente este controle é feito manualmente.

4. Grupo de Interesse

Para o desenvolvimento do projeto, contamos com os seguintes participantes:

- Gerente do Projeto: Eduardo Augusto Costa Trindade;
- Gerente Sênior do Projeto (Professora orientadora): Claudia Beatriz Berti;
- Servidores do CESEC envolvidos diretamente com a biblioteca.
- Alunos do CESEC.

5. Descrição Geral

O projeto terá início em Dezembro de 2016, incluindo planejamento, organização e a realização do cronograma inicial com a análise de requisitos. Até o fim do período letivo escolar de Dezembro de 2016, deverá ser coletado dados que possibilitem o início do desenvolvimento do sistema. A previsão inicial para entrega do sistema será de seis meses.
Deverá ser entregue um sistema web de fácil manuseio e com treinamentos adequados para os funcionários que ficarem encarregados de suas funções administrativas.
O projeto não necessitará de recursos financeiros devido ao uso de ferramentas de acesso livre, e ao comprometimento do Gerente do Projeto em doar o produto ao CESEC, tratado a partir daqui como cliente. Possíveis modificações serão documentadas e atualizadas durante o andamento do projeto.
Após o encerramento do projeto, o cliente ficará responsável pelo manuseio do sistema, dispondo de um manual de instruções que servirá para auxiliá-lo nas diversas tarefas disponíveis. O Gerente do Projeto compromete-se ainda, em auxiliar o cliente em quaisquer dúvidas que possam surgir. Novas funcionalidades poderão ser avaliadas de acordo com a disponibilidade de ambas as partes.

6. Acompanhamento do Projeto

Serão realizadas reuniões com a Gerência Sênior onde dúvidas e sugestões serão avaliadas a fim de obter orientação adequada para o desenvolvimento do sistema.
Serão realizadas reuniões quinzenais também entre o Gerente do Projeto e servidores do CESEC, principalmente bibliotecários e servidores administrativos, salvo período de recesso escolar compreendido durante o mês de janeiro de 2017.

7. Aprovações

Diamantina, 02 de Junho de 2017.

Gerente Sênior do Projeto
Claudia Beatriz Berti

Gerente do Projeto
Eduardo A. Costa Trindade

Diretor do CESEC
"Juscelino Kubitschek de Oliveira"
Antônio Muniz Correa

Apêndice B

Questionário para levantamento de requisitos

EACT Sistemas	**Sistema de Gerenciamento para Biblioteca** ***Questionário para Levantamento de Requisitos***

Elaborado por:	Eduardo Augusto Costa Trindade	**Versão:**	1.0
Aprovador por:	Claudia Beatriz Berti	**Data de aprovação:**	15/06/2017

Respondido por: _____

É com imensa satisfação que iniciamos nossa primeira reunião a fim de identificar os primeiros requisitos do sistema web para Gestão da Biblioteca que será desenvolvido como Trabalho de Conclusão de Curso de Graduação em Sistemas de Informação, com o objetivo de oferecer à escola o controle de sua biblioteca por meio deste sistema. Considerando que deverá ser entregue um sistema web de fácil manuseio e com treinamentos adequados, agradecemos sua colaboração em esclarecer os requisitos que deverão ser tratados ao longo do projeto, por meio deste breve questionário.

Informações prévias da Biblioteca

1. Quais usuários (alunos, professores, servidores, etc.) possuem permissão para consulta e empréstimo de livros da biblioteca?

2. Como é feito o controle de empréstimo de livros atualmente?

3. Na ausência de livros na prateleira, é possível realizar uma reserva? Se sim, como é feita?

4. Existe prazo para a devolução dos livros? Se sim, qual o prazo (em dias)?

5. Há um limite de livros que podem ser emprestados a uma mesma pessoa? Se sim, Qual?

Informações complementares

6. Qual o seu atual nível de conhecimento em informática, no manuseamento de sistemas administrativos ou softwares (Word, Excel, PowerPoint)?

( ) Básico ( ) Intermediário ( ) Avançado

7. Como você prefere que seja realizado o cadastro de novos usuários no sistema?

( ) Pelo próprio usuário do Sistema.
( ) Pelo funcionário administrativo responsável pela biblioteca
(Ex: Bibliotecário).

8. Quantos livros, aproximadamente, a biblioteca possui? Será cadastrado mais algum tipo de item?

9. Quais os procedimentos que você gostaria de ver no sistema que facilitariam seu trabalho?

10. Há alguma consideração ou observação que queira mencionar?

Agradecemos a prestatividade e colaboração,

Diamantina, 15 de Junho de 2017.

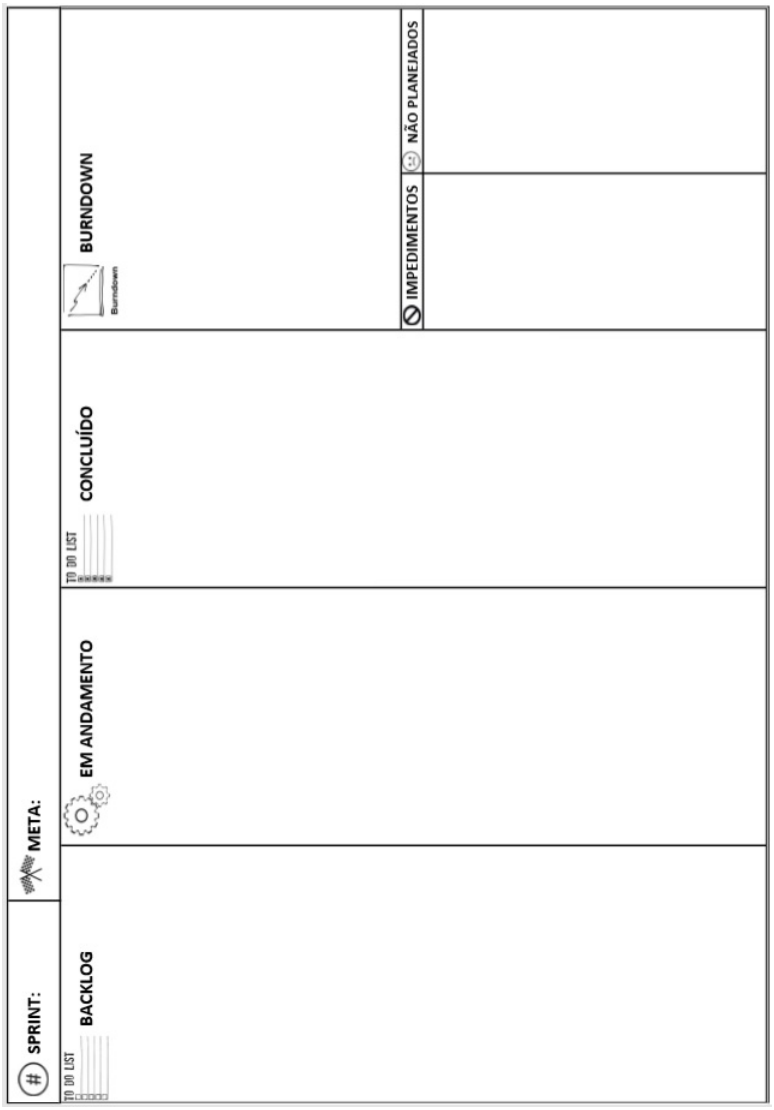
Gerente Sênior do Projeto
Claudia Beatriz Berti

Gerente do Projeto
Eduardo A. Costa Trindade

Apêndice C

Modelo de Quadro Scrum para aplicação

Figura C.0.1 – Modelo de Quadro Scrum para Projetos

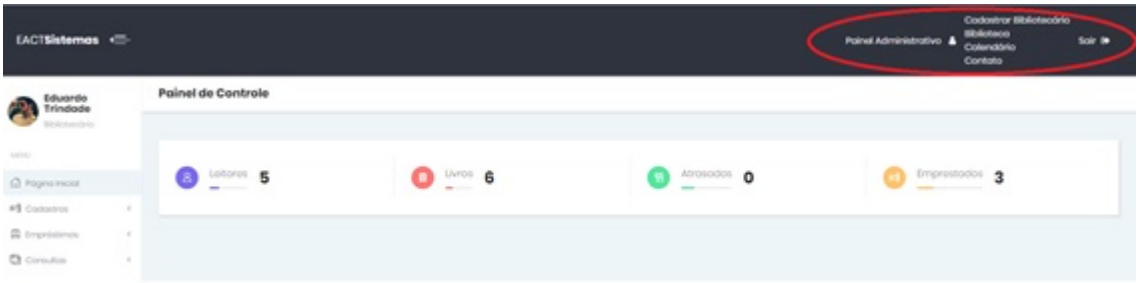


Fonte: https://www.guiadoexcel.com.br/quadro-scrum-quadro-de-tarefas-kanban-excel/ (Adaptado)

Apêndice D

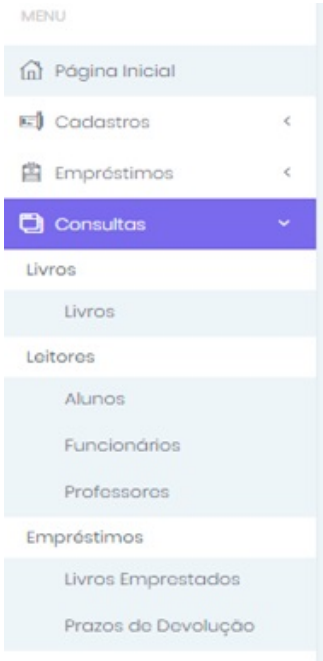
Telas do Sistema

Figura D.0.1 – Acessando o painel administrativo



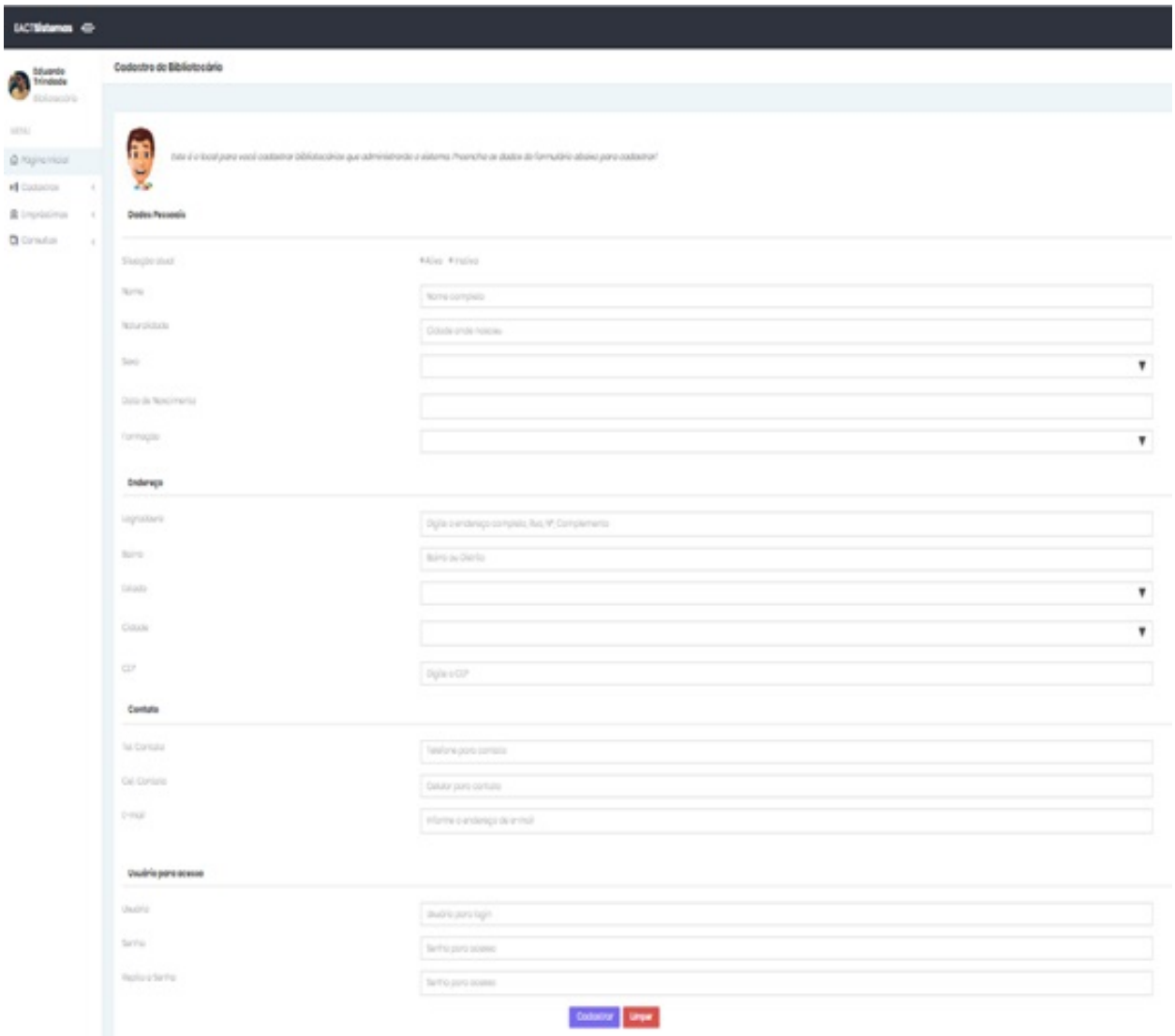
Fonte: Elaborado pelo autor

Figura D.0.2 – Menu do *Dashboard* (Consultas expandido)



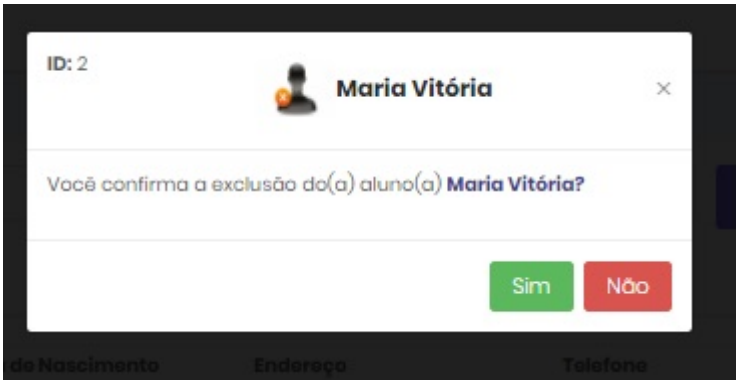
Fonte: Elaborado pelo autor

Figura D.0.3 – Tela para cadastro do Bibliotecário



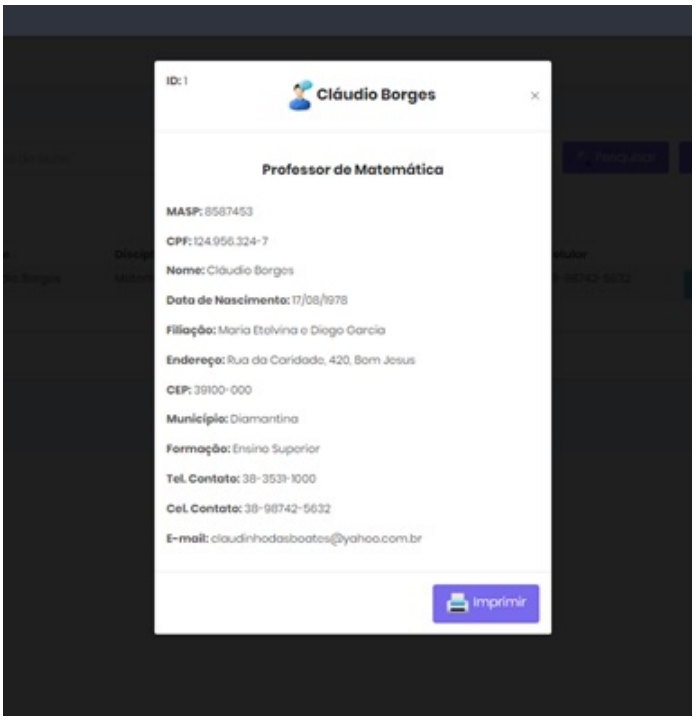
Fonte: Elaborado pelo autor

Figura D.0.4 – Exclusão de leitor



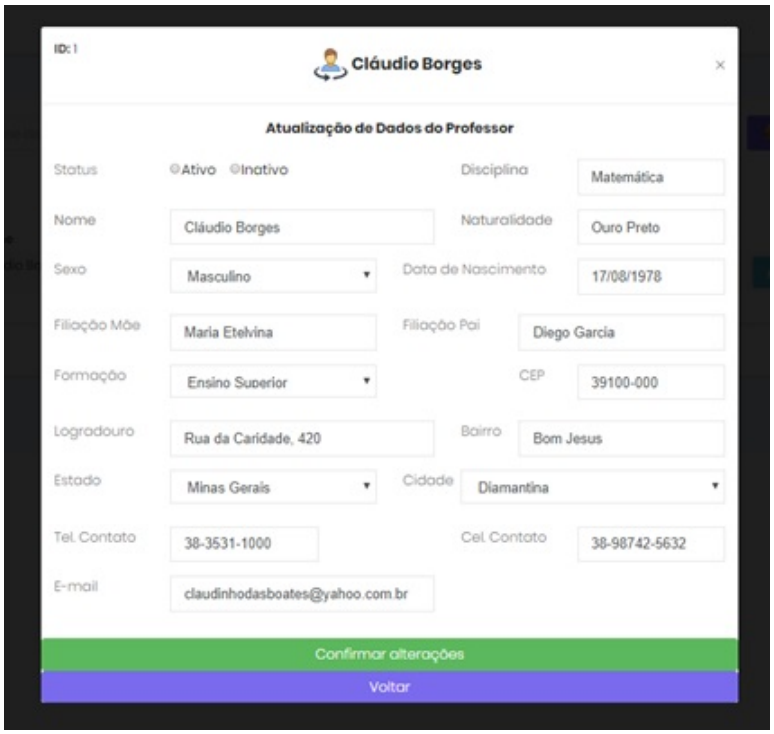
Fonte: Elaborado pelo autor

Figura D.0.5 – Visualiza dados do Leitor (ex: professor)



Fonte: Elaborado pelo autor

Figura D.0.6 – Edita dados do Leitor (ex: professor)



Fonte: Elaborado pelo autor

Apêndice E

Exemplo de *Script*: Empréstimo de Livro

```
<?php
require 'conexao.php';

// Atribui uma conexao PDO
$conexao = conexao::getInstance();
$situacao = 'E';

// Recebe os dados enviados pela submiss o
switch(isset($_POST))
{
        case @($_POST['id_alunos'] and $_POST['id_livros']):
                $id_alunos  = (isset($_POST['id_alunos'])) ? $_POST['id_alunos'] : '';
                $id_livros  = (isset($_POST['id_livros'])) ? $_POST['id_livros'] : '';

                $sql = "INSERT INTO emprestimo (data_emprestimo, data_devolucao, situacao,
                id_alunos, id_livros) VALUES(now(), DATE_ADD(CURDATE(),
                INTERVAL 7 DAY), :situacao, :id_alunos, :id_livros)";

                $stm = $conexao->prepare($sql);
                $stm->bindValue(':situacao', $situacao);
                $stm->bindValue(':id_alunos', $id_alunos);
                $stm->bindValue(':id_livros', $id_livros);
                $retorno = $stm->execute();

        break;

        case @($_POST['id_professores'] and $_POST['id_livros']):
                $id_professores  = (isset($_POST['id_professores']))
                                ? $_POST['id_professores'] : '';
                $id_livros  = (isset($_POST['id_livros'])) ? $_POST['id_livros'] : '';

                $sql2 = "INSERT INTO emprestimo (data_emprestimo, data_devolucao, situacao,
                id_professores, id_livros) VALUES(now(), DATE_ADD(CURDATE(),
                INTERVAL 7 DAY), :situacao, :id_professores, :id_livros)";

                $stm2 = $conexao->prepare($sql2);
                $stm2->bindValue(':situacao', $situacao);
                $stm2->bindValue(':id_professores', $id_professores);
                $stm2->bindValue(':id_livros', $id_livros);
                $retorno = $stm2->execute();
```

```
        break;

        case @($_POST['id_funcionarios'] and $_POST['id_livros']):
                $id_funcionarios  = (isset($_POST['id_funcionarios']))
                                    ? $_POST['id_funcionarios'] : '';
                $id_livros  = (isset($_POST['id_livros'])) ? $_POST['id_livros'] : '';

                $sql3 = "INSERT INTO emprestimo (data_emprestimo, data_devolucao, situacao,
                id_funcionarios, id_livros)     VALUES(now(), DATE_ADD(CURDATE(),
                INTERVAL 7 DAY), :situacao, :id_funcionarios, :id_livros)";

                $stm3 = $conexao->prepare($sql3);
                $stm3->bindValue(':situacao', $situacao);
                $stm3->bindValue(':id_funcionarios', $id_funcionarios);
                $stm3->bindValue(':id_livros', $id_livros);
                $retorno = $stm3->execute();

        break;
}

if ($retorno):
        echo "<div class='alert alert-success' role='alert'>
        Livro emprestado com sucesso!
        Aguarde enquanto voce esta sendo redirecionado...</div> ";
else:
        echo "<div class='alert alert-danger' role='alert'>
        Erro ao emprestar o livro!</div>";
endif;

echo "<meta http-equiv=refresh content='3;URL=emprestaLivro.php'>";

?>
```

.