UNIVERSIDADE FEDERAL DO PARÁ
NÚCLEO DE DESENVOLVIMENTO AMAZÔNICO EM ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

LUIZ ANTONIO LEÃO LISBOA JUNIOR

KIRN: PROPOSTA DE ABORDAGEM DE EVOLUÇÃO TECNOLÓGICA DE SISTEMAS LEGADOS

Tucuruí - Pará
2020

LUIZ ANTONIO LEÃO LISBOA JUNIOR

KIRN: PROPOSTA DE ABORDAGEM EVOLUÇÃO TECNOLÓGICA DE SISTEMAS LEGADOS

Dissertação apresentada ao Programa de Pós-Graduação em Computação Aplicada do Núcleo de Desenvolvimento Amazônico em Engenharia, da Universidade Federal do Pará, como requisito para a obtenção do título de Mestre em Computação Aplicada.

Orientador: Prof. Dr. Rodrigo Quites Reis

Tucuruí - Pará

2020

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

L769k Lisboa Junior, Luiz Antonio Leão.
KIRN: PROPOSTA DE ABORDAGEM DE EVOLUÇÃO TECNOLÓGICA DE SISTEMAS LEGADOS / Luiz Antonio Leão Lisboa Junior. — 2020.
121 f. : il. color.

Orientador(a): Prof. Dr. Rodrigo Quites Reis
Dissertação (Mestrado) - Universidade Federal do Pará, Núcleo de Desenvolvimento Amazônico em Engenharia, Programa de Pós-Graduação em Computação Aplicada, Tucuruí, 2020.

1. Software Legado. 2. Reengenharia de Software. 3. Engenharia de Software Experimental. I. Título.

CDD 620.001171

LUIZ ANTONIO LEÃO LISBOA JUNIOR

KIRN: PROPOSTA DE ABORDAGEM EVOLUÇÃO TECNOLÓGICA DE SISTEMAS LEGADOS

Dissertação apresentada ao Programa de Pós-Graduação em Computação Aplicada do Núcleo de Desenvolvimento Amazônico em Engenharia, da Universidade Federal do Pará, como requisito para a obtenção do título de Mestre em Computação Aplicada.

Orientador: Prof. Dr. Rodrigo Quites Reis

Aprovada em 16 de Dezembro de 2020.

BANCA EXAMINADORA:

(Assinado digitalmente em 18/12/2020 15:37)
RODRIGO QUITES REIS
PROFESSOR DO MAGISTERIO SUPERIOR
1224877

(Assinado digitalmente em 18/12/2020 16:02)
BRUNO MERLIN
PROFESSOR DO MAGISTERIO SUPERIOR
2810321

(Assinado digitalmente em 18/12/2020 16:05)
GUSTAVO HENRIQUE LIMA PINTO
PROFESSOR DO MAGISTERIO SUPERIOR
1277614

SUMÁRIO

Dedico este trabalho à minha filha Maria Luiza,
que me motiva a ser uma pessoa melhor todos os dias.

AGRADECIMENTOS

Agradeço aos meus pais, Luiz e Filomena, pelo suporte fundamental na minha caminhada acadêmica.

Agradeço a minha esposa Heloisa, pelo apoio, carinho e compreensão em todos os momentos da minha vida.

Agradeço ao meu orientador Prof. Rodrigo Quites, pelos aconselhamentos e pela disponibilidade em me auxiliar na elaboração desse trabalho.

Agradeço aos professores, colaboradores e coordenação do Programa de Pós-graduação em Computação Aplicada da UFPA (PPCA/UFPA) que, mesmo à distância em muitos momentos, forneceram apoio e atenção fundamentais para o desenvolvimento das minhas atividades acadêmicas.

Agradeço aos colegas da turma PPCA 2017, pelo convívio sempre cordial e pelos amigos que conquistei para além da vida acadêmica.

RESUMO

Com a finalidade de manter os sistemas de informação das instituições em conformidade funcional com os seus processos organizacionais, surge q necessidade de modernizá-los frequentemente. Porém, dificuldades como a falta de recursos diversos (técnicos, financeiros, humanos e outros), tornam essa tarefa difícil de ser executada. Diante das adversidades que as instituições encontram em modernizar os seus sistemas legados, esse trabalho propõe uma abordagem, definida como Kirn, para a evolução tecnológica desses sistemas, que tem como premissa a reengenharia de software de forma ágil, com uso eficiente dos recursos disponíveis objetivando a otimização dessa prática. Para auxiliar nessa atividade, foi desenvolvida uma ferramenta chamada Kirn Framework, que ajuda no mapeamento de requisitos dos sistemas legados e na criação de uma aplicação base, que será o ponto de partida da evolução tecnológica. Como proposta para a avaliação dessa abordagem, foi desenvolvido um experimento na Superintendência do Desenvolvimento da Amazônia (SUDAM), utilizando conceitos de engenharia de software experimental, no qual foram comparados os protótipos desenvolvidos com a abordagem institucional e a abordagem Kirn, através do uso de indicadores pré-estabelecidos no planejamento do experimento.

Palavras-chave: Software Legado, Reengenharia de Software, Engenharia de Software Experimental.

ABSTRACT

In order to keep the institutions' information systems in functional compliance with their organizational processes, there is a need to modernize them frequently. However, difficulties such as the lack of diverse resources (technical, financial, human and others), make this task difficult to perform. In view of the adversities that institutions find in modernizing their legacy systems, this work proposes an approach, defined as Kirn, for the technological evolution of these systems, which has as premise the reengineering of software in an agile way, with efficient use of the available resources aiming the optimization of this practice. To assist in this activity, a tool called Kirn Framework was developed, which helps in mapping the requirements of legacy systems and in creating a base application, which will be the starting point of technological evolution. As a proposal for the evaluation of this approach, an experiment was developed at the Superintendência do Desenvolvimento da Amazônia (SUDAM), using experimental software engineering concepts, in which the prototypes developed with the institutional approach and the Kirn approach were compared through the use of pre-established indicators in the experiment’s planning.

Keywords: Legacy Software, Software Reengineering, Experimental Software Engineering.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

API	*Application Programming Interface*
APF	Análise de Pontos de Função
BD	Banco de Dados
BPMN	*Business Process Model Notation*
CBSE	*Component-Based Software Engineering*
CSS	*Cascading Style Sheets*
CTI	Coordenação de Tecnologia da Informação da SUDAM
DEVOPS	*Development & Operation*
DBMS	*Database Management Systems*
ESSE	Engenharia de Software Experimental
HTML	*Hypertext Markup Language*
IDE	*Integrated Development Environment*
IEC	*International Electrotechnical Commission*
IFPUG	*International Function Point Users Group*
ISO	*International Organization for Standardization*
JDL	*JHipster Domain Language*
KML	*Kirn Markup Language*
MDA	*Model-Driven Architecture*
MDD	*Model-Driven Development*
MDE	*Model-Driven Engineering*
MDR	Ministério do Desenvolvimento Regional
MVC	*Model-View-Controller*
NOSQL	*Not Only SQL*
N/A	Não se aplica
ORM	*Object-Relational Mapping*
PDO	*PHP Data Object*
PF	Ponto de Função
PHP	*Hypertext Preprocessor*
PSW-SISP	Processo de Software para o SISP
RSL	Revisão Sistemática da Literatura
SGBD	Sistema de Gestão de Banco de Dados

SISP	Sistema de Administração dos Recursos de Tecnologia da Informação
SO	Sistema Operacional
SUDAM	Superintendência do Desenvolvimento da Amazônia
SVN	*Subversion*
TI	Tecnologia da Informação
TIC	Tecnologia da Informação e Comunicação
TRF4	Tribunal Regional Federal da 4ª Região
UFPA	Universidade Federal do Pará
VO	*Value Object*
XMI	XML *Metadata Interchange*

1. INTRODUÇÃO

Ao definir sistemas legados, surgem as mais variadas definições, pois, para cada cenário, surgem caracterizações complementares. Para Feathers (2013), uma das constatações mais importantes é a que são sistemas que não possuem testes definidos na maior parte do código. Para o autor, levando em consideração as necessidades institucionais observadas no decorrer do tempo, sistemas legados são aqueles que não atendem aos processos organizacionais como quando eles foram concebidos para atender. Onde o caminho natural é um processo de reengenharia ou troca por um sistema mais moderno.

Porém, a atividade de reengenharia de software toma tempo, tem um custo financeiro alto e absorve recursos que poderiam ser usados em necessidades mais imediatas (PRESSMAN, 2016). Some-se a isso o fato de o software requerer modificações periódicas, as quais podem degradar sua estrutura e tornar o desenvolvimento de tais mudanças mais caro ainda (SOMMERVILLE, 2011). Sabe-se que compreender os requisitos de um problema é uma das atividades mais difíceis para um engenheiro de software (PRESSMAN, 2016) e, consequentemente, evoluir um software demanda uma dedicação especial, pois requer atenção simultânea à perspectiva de negócio e à perspectiva técnica do software (WARREN, 2002).

A necessidade de manutenções frequentes nos sistemas demanda diversos recursos - técnicos, humanos, financeiros, entre outros - que podem não estar sempre disponíveis na organização. Tais sistemas em operação tornam-se então legados, pois atendem de forma precária as necessidades estabelecidas pelos processos institucionais.

Sommerville (2011) apresenta um esquema de classificação de sistemas legados, na relação “Valor de Negócio x Qualidade de Sistema”, como pode ser visto no diagrama representado pela Figura 1.

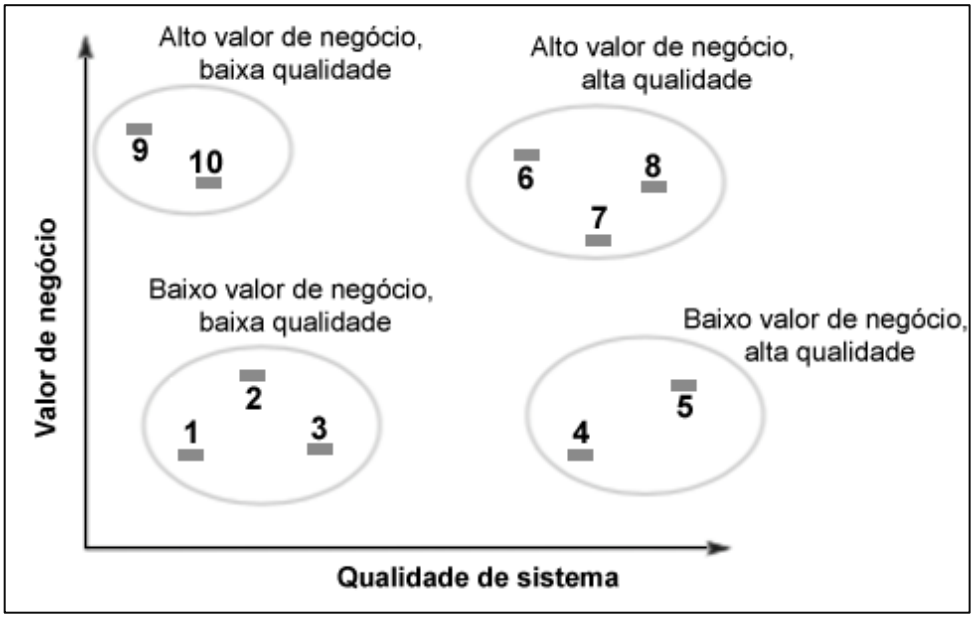


Figura 1 - Exemplo de avaliação de sistemas legados (SOMMERVILE, 2011)

Sommerville (2011) classifica os sistemas legados de alto valor de negócio e baixa qualidade na seguinte maneira:

> Esses sistemas dão uma contribuição importante para o negócio, portanto, eles não podem ser descartados. Contudo, sua baixa qualidade significa que seu custo de manutenção é alto. Esses sistemas devem ser reestruturados para melhorar a qualidade. Eles podem ser substituídos, caso um sistema de prateleira esteja disponível.

Os sistemas legados de alto valor de negócio e baixa qualidade não podem ser descartados. No entanto, devido à ausência de documentação, obsolescência tecnológica e/ou baixa qualidade geram alto custo de manutenção. Para atender às necessidades da organização, esses sistemas tornam-se candidatos prioritários ao processo evolução tecnológica através de reengenharia.

Existem abordagens para o processo de reengenharia com e sem apoio ferramental. No âmbito teórico, diversos autores descreveram soluções para essa finalidade, que foram abordadas na seção ‘TRABALHOS RELACIONADOS’. No entanto, há uma carência de abordagens que utilizam ferramentas, que tenham sido desenvolvidas

especificamente para apoiar as atividades dessas abordagens, o que agilizaria o processo de reengenharia.

Como a maioria dos sistemas legados possuem um esquema de banco de dados modelado, implementado e utilizado, essa pode ser uma importante fonte de requisitos a fim de auxiliar na compreensão dos processos organizacionais atendidos por esse software. Com esses requisitos estruturados à disposição, diversas ferramentas foram desenvolvidas para tornar a atividade de desenvolvimento de software mais eficiente ao utilizarem de MDD (*Model-Driven Development*), que é mais como abordagem auxiliar a esse processo, pois possibilita a leitura de modelos documentais com o objetivo de transformá-los em artefatos de software.

Dentre as soluções disponíveis no mercado temos Dubois (2013). É uma ferramenta que oferece a possibilidade de se construir modelos de software baseado na linguagem JDL e a partir dele gerar uma aplicação base, utilizando diversas tecnologias disponíveis no mercado, além da possibilidade de se efetuar integração contínua com infraestruturas DevOps (*Development & Operation*). A sua desvantagem em relação a solução buscada é que ele não oferece a possibilidade de geração automática do modelo a partir de requisitos existentes, sejam eles escritos em notações técnicas ou a partir de artefatos de software já implementados, que possam ser acessados através de engenharia reversa. Tolitech (2020) é uma ferramenta de geração de código a partir do banco de dados construído. Ele oferece a possibilidade de customizar a exibição dos campos das tabelas mapeadas, através de formatações e validações. Um ponto fraco identificado é o custo, por se tratar de uma ferramenta proprietária. Dependendo dos recursos financeiros disponíveis na organização, esse pode ser um fator de inviabilidade da atividade de reengenharia do sistema legado. Netmake (2000) apresenta o mesmo problema e a ferramenta Changevision (2006), além de proprietária, apresenta limitações nos artefatos gerados, podendo apenas oferecer aos desenvolvedores a conversão do diagrama de classes em classes de negócio, apenas.

Com base nas ferramentas pesquisadas, identificando suas vantagens e desvantagens, o autor deste trabalho desenvolveu a ferramenta chamada *Kirn*

Framework, que tem como objetivo gerar um modelo de projeto a partir da engenharia reversa do esquema de banco de dados de uma aplicação legada e, em seguida, através de um processo de engenharia avante, gerar uma aplicação que serve de base para o processo de modernização dos sistemas legados, através da customização dessa aplicação.

Nesse trabalho é apresentada uma abordagem conhecida como *Kirn* que tem como objetivo organizar as atividades de engenharia de software, definindo as fases do processo, papeis e insumos a serem utilizados na modernização de sistemas legados. Para isso, foram pesquisadas ferramentas e técnicas que possam ajudar na execução dessa atividade em diversos contextos.

Para avaliar a aplicabilidade dessa abordagem, foi elaborado um experimento, baseado na estrutura elaborada em Travassos (2002), que utiliza de preceitos da engenharia de software experimental, cuja finalidade é comparar a abordagem *ad-hoc* institucional com a abordagem aqui proposta, através de indicadores previamente definidos no planejamento desse experimento.

1.1. MOTIVAÇÃO

A necessidade de tornar a atividade de reengenharia mais acessível às instituições demandam um aprofundamento das pesquisas acerca do tema. Com relação à experimentação em engenharia de software, uma motivação pertinente é lembrada por Travassos (2002):

> O crescimento do número de trabalhos científicos com uma validação empírica significativa possui a boa chance de acelerar o processo de formação da Engenharia de Software como ciência. As ideias duvidosas serão rejeitadas mais rapidamente e os pesquisadores poderão concentrar-se nas abordagens promissoras.

A elaboração da pesquisa e a execução do experimento é necessária pois busca resolver problemas reais enfrentados pelas organizações e que podem ser resolvidos com o aperfeiçoamento das práticas de reengenharia de software. As nomeadas boas práticas de engenharia de software foram alçadas à essa denominação em

decorrência de experimentações envolvendo tecnologias promissoras, resolvendo problemas reais.

1.2. PROBLEMA

Software que não pode ser mantido não é um problema novo (PRESSMAN, 2016). Após anos atendendo a instituição, o software inevitavelmente sofre depreciações decorrentes das manutenções que não utilizaram as boas práticas de engenharia de software, não necessariamente por negligência por parte dos desenvolvedores de software, mas por necessidade de resolução de problemas imediatos.

Sistemas legados são difíceis de serem compreendidos e mudados (SOMMERVILLE, 2011). As inúmeras manutenções efetuadas durante o ciclo de vida do software depreciam a sua arquitetura, dificultando cada vez mais a inclusão de novas funcionalidades e consecutivamente, o funcionamento da aplicação. Apensadas às dificuldades técnicas, estão os custos. Cerca de 85% a 90% dos custos organizacionais de software são custos de evolução (SOMMERVILLE, 2011). Com certeza, os custos para mudança de software podem inviabilizar a sua modernização.

1.3. CONTEXTO

Sobre o contexto do trabalho, foi escolhida a Superintendência do desenvolvimento da Amazônia (SUDAM), órgão vinculado até o momento ao Ministério do Desenvolvimento Regional, pertencente ao governo federal, como missão promover o desenvolvimento includente e sustentável da Amazônia Legal, por meio do planejamento, articulação e fomento, contribuindo para a redução das desigualdades regionais. A Coordenação de Gestão de Tecnologia da Informação (CTI) é a unidade institucional que tem como competência, segundo Brasil (2017):

> coordenar, orientar e supervisionar a execução das atividades relativas à gestão de tecnologia da informação e comunicação, de acordo com as políticas, diretrizes, planos, normas e padrões emanados pelo órgão central do Sistema de Administração de Recursos de Informação e Informática - SISP;

Vinculada à CTI, há a Divisão de Sistemas, Documentação e Informações Bibliográficas (DSIB), que também de acordo com Brasil (2017) é responsável por:

> executar as atividades de sistemas de tecnologia da informação e comunicação, conforme políticas, diretrizes, planos, normas e padrões, no âmbito da Sudam;

O autor ocupa na instituição o cargo de Analista Técnico Administrativo – Área: Ciência da computação e, quando surgiu a necessidade de desenvolver a abordagem proposta no trabalho, o ele ocupava o cargo de chefe de divisão da DSIB, assumindo assim a responsabilidade de capitanear o processo de modernização dos sistemas da instituição.

O processo de desenvolvimento de software adotado pela SUDAM foi definido por Brasil (2012), no qual pode ser conferido todo o detalhamento das fases, atores e artefatos da abordagem. Ele é utilizado na instituição desde o ano de 2016, quando foi contratada uma fábrica de software para o desenvolvimento de soluções para atender as áreas finalísticas da instituição, no desenvolvimento de novos softwares ou na manutenção de softwares pré-existentes. A instituição guarda um extenso histórico de desenvolvimento de software, os quais são voltados tanto para atender a área finalística da instituição, quanto a área meio.

O requerimento para o desenvolvimento da pesquisa, desenvolvimento da abordagem e o experimento acerca da sua aplicabilidade foi devidamente protocolado e autorizado, conforme autorização apresentada no Apêndice B deste trabalho.

Na Tabela 1, segue a lista de servidores lotados na CTI (identificados por S1 a S14) e nas demais unidades que compõem a coordenação.

Tabela 1 – Lista de servidores da CTI

Servidor	Cargo	Nível	Lotação	Atribuições
S1	Coordenador	Superior	CTI	Coordenar as ações de TIC
S2	Chefe de Divisão	Superior	DSIB	Coordenar ações relacionadas ao desenvolvimento e manutenção de software
S3	Analista de Sistemas	Superior	CTI	
S4	Analista Técnico Administrativo	Superior	CTI	Atuar na elaboração de termos de referência para editais de contratação de serviços e aquisição de equipamentos
S5	Analista Técnico Administrativo – Área: Ciência da Computação	Superior	DSIB	Desenvolver atividades de desenvolvimento e manutenção de software
S6	Analista Técnico Administrativo – Área: Ciência da Computação	Superior	DTEC	Desenvolver atividades relacionadas a manutenção da infraestrutura de comunicação da instituição
S7	Programador	Médio	DTEC	Desenvolver atividades relacionadas a manutenção da infraestrutura de comunicação da instituição
S8	Programador	Médio	DSIB	Desenvolver atividades de desenvolvimento e manutenção de software
S9	Agente Administrativo	Médio	DTEC	Desenvolver atividades relacionadas a manutenção da infraestrutura de comunicação da instituição

Servidor	Cargo	Nível	Lotação	Atribuições
S10	Operador de Computador	Médio	DTEC	Desenvolver atividades relacionadas a manutenção dos equipamentos da instituição
S11	Operador de Computador	Médio	DTEC	Desenvolver atividades relacionadas a manutenção dos equipamentos da instituição
S12	Operador de Computador	Médio	DTEC	Desenvolver atividades relacionadas a manutenção dos equipamentos da instituição
S13	Bibliotecário	Superior	DSIB	Gerenciar o acervo bibliográfico da instituição
S14	Bibliotecário	Superior	DSIB	Gerenciar o acervo bibliográfico da instituição

A Tabela 2 mostra um quadro com os softwares utilizados na instituição, apresentando suas características tecnológicas, forma de aquisição e início de operação.

Tabela 2 – Lista de sistemas da SUDAM

Sistema	Sistema Operacional	Linguagem de Programação	Banco de Dados	Desenvolvimento	Início da Operação
ASM - Sistema de Gestão de Serviços de TI	Windows Server 2008 R2	C#	SQL Server 10.50.1600	Adquirido (Proprietário)	2010
BIBLIOTECA - Sistema de Informação da Biblioteca	Windows Server 2008 R2	PHP 5.6	MySQL 6.0	Open Source	Desconhecido
COS - Central de Ordem de Serviços	Windows Server 2008 R2	PHP 5.6	MySQL 6.0	Open Source	2019
GESPRO - Administração de Almoxarifado e Patrimônio	Windows Server 2008 R2	ASP	SQL Server 10.50.1600	Cedido por órgão governamental (MDR)	2000
PROTOCOLO - Sistema de Gestão de Protocolo	Windows Server 2008 R2	PHP 5.6	MySQL 6.0	Próprio	2017
RH - Sistema de Gestão de Recursos Humanos	Windows Server 2008 R2	PHP 5.6	SQL Server 10.50.1600	Próprio	2015
SEI - Sistema Eletrônico de Informações	Windows Server 2008 R2	PHP 5.6	MariaDB	Cedido por órgão governamental (TRF4)	2017
SIAC - Sistema de Apoio aos Convênios	Windows Server 2008 R2	PHP 5.6	MySQL 6.0	Terceirizado	2018
SICAS - Sistema de Controle de Assistência a Saúde	Windows Server 2008 R2	Java 8	SQL Server 10.50.1600	Terceirizado	2013
SIAV	Windows Server 2008 R2	PHP 5.6	MySQL 6.0	Terceirizado	2018
SICAS Legado	Win98	GOL	BTRIEVE	Terceirizado	2000
SigFDA - Sistema de Gestão do Fundo de Desenvolvimento da Amazônia	Linux CentOS	Ruby	PostgreSQL	Próprio	2014

Sistema	Sistema Operacional	Linguagem de Programação	Banco de Dados	Desenvolvimento	Início da Operação
SIN - Sistema de Incentivos e Benefícios Fiscais	Windows Server 2008 R2	PHP 5.6	MySQL 6.0	Terceirizado	2017
Sistema de Ponto Eletrônico Secullum	Windows Server 2008 R2	C#	SQL Server 10.50.1600	Adquirido (Proprietário)	2018
Visitantes – Controle dos visitantes da SUDAM	Windows Server 2008 R2	PHP 5.6	PostgreSQL	Próprio	2017

Sobre o contexto apresentado e analisando a força de trabalho da DSIB, que é a divisão responsável pelo desenvolvimento e manutenção dos sistemas de informação da instituição, poucos colaboradores atendem as necessidades institucionais em termos de capacitação. Há uma necessidade constante de reforço de mão-de-obra, seja através da contratação de empresas terceirizadas, seja pela contratação de estagiários. Ambas as modalidades geram um quadro de incerteza para a instituição. No caso das empresas contratadas, ao final da vigência dos contratos, não é garantida a manutenção da equipe que estava habituada a rotina laboral da instituição. Além disso, no caso dos estagiários em decorrência do curto período de permanência na instituição acabam desfalcando a equipe periodicamente.

Por todas as fragilidades de atendimento às necessidades institucionais são necessários estudos que promovam o funcionamento dos seus sistemas de informação de forma adequada, utilizando os recursos disponíveis para tal.

1.4. OBJETIVOS

Constitui objetivo geral deste trabalho avaliar a aplicabilidade da abordagem proposta, através de um experimento que utiliza de preceitos da engenharia de software experimental, e identificar pontos que auxiliem na construção de uma alternativa de processo de reengenharia de software de sistemas legados.

Para atingir o objetivo geral proposto neste trabalho é fundamental alcançar os seguintes objetivos específicos:

- Identificar as experiências e dificuldades da instituição envolvida na pesquisa;
- Investigar as melhores práticas para a reengenharia de sistemas legados, a fim de desenvolver uma abordagem específica para essa finalidade;
- Identificar os requisitos necessários para o desenvolvimento de modelo de projeto a ser utilizado no experimento de análise da abordagem;
- Planejar um experimento utilizando as abordagens propostas e institucional;
- Avaliar os protótipos desenvolvidos em ambas as abordagens com o intuito de verificar os indicadores comparativos, assim como as impressões da equipe envolvida no experimento sobre os projetos desenvolvidos.

1.5. ORGANIZAÇÃO DO TRABALHO

O trabalho está organizado nos seguintes capítulos:

- Capítulo 2 – FUNDAMENTAÇÃO DO TRABALHO: apresenta os critérios e o resultado da pesquisa que reiterem a validade do trabalho proposto.
- Capítulo 3 – ABORDAGEM KIRN: detalha o processo de desenvolvimento de software, voltado para a evolução tecnológica de sistemas legados.
- Capítulo 4 – KIRN FRAMEWORK: apresenta a ferramenta *Kirn Framework*, que auxilia no desenvolvimento das aplicações, através da abordagem Kirn de desenvolvimento de software.
- Capítulo 5 – METODOLOGIA: detalhamento do processo metodológico do trabalho, visando a ratificar a relevância científica do estudo proposto.
- Capítulo 6 – EXPERIMENTO: detalha o planejamento, a execução e a análise dos dados coletados do experimento.
- Capítulo 7 – CONCLUSÃO: descreve a conclusão do trabalho, com o relato das observações acerca de todas as fases de desenvolvimento do trabalho.

- Capítulo 8 – TRABALHOS FUTUROS: enumera as intenções de continuidade do trabalho, através de novas pesquisas e implementações.

Além de contar com o referencial bibliográfico, elemento documental obrigatório para trabalhos dessa natureza, estão presentes diversos apêndices que além de complementarem esse registro acadêmico, ratificam as ações necessárias para a validação desse trabalho, tais como o protocolo de autorização da pesquisa na instituição; o endereço do repositório dos artefatos de software produzidos no experimento; entre outros artefatos documentais.

2. METODOLOGIAS

Nesse capítulo serão abordados os procedimentos metodológicos utilizados na pesquisa que fundamentou o presente trabalho, direcionando e ratificando as atividades de planejamento e execução do experimento proposto. O capítulo está organizado através das seguintes seções:

- 2.1 SURVEY: Pesquisa elaborada com o objetivo de identificar a necessidade das instituições em utilizar uma abordagem para reengenharia de sistemas legados;
- 2.2 MÉTODO UTILIZADO: Apresenta a estratégia de pesquisa utilizada;
- 2.3 REVISÃO SISTEMÁTICA DA LITERATURA: Apresenta os critérios da pesquisa científica, bem como os resultados obtidos;
- 2.4 FUNDAMENTAÇÃO TEÓRICA: Exibe os assuntos, abordados pelos principais autores dos temas, que embasam o trabalho proposto.

2.1. *SURVEY*

Com o objetivo de identificar a real preocupação das instituições em evoluir os seus sistemas legados, foi elaborada um *survey*, com diversas instituições da administração pública, sediadas na cidade de Belém-PA, no período entre 12/03/2019 e 03/04/2019. As questões usadas no questionário podem ser vistas no Apêndice A do trabalho. Participaram 18 (dezoito) profissionais da área de TI lotados em quatro instituições públicas. Após a aplicação do formulário de pesquisa, foram obtidos os seguintes dados, referentes às experiências das instituições públicas relacionadas com a gestão de software legados, que podem ser conferidos nas figuras a seguir:

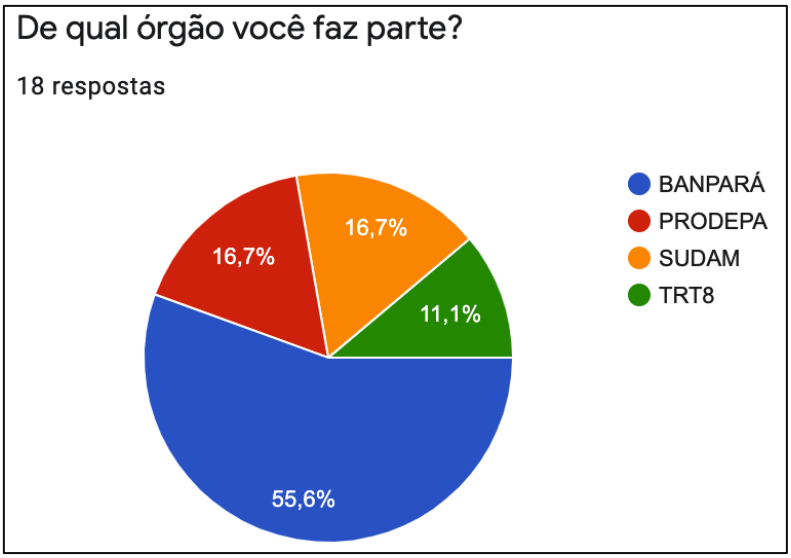


Figura 2 - Gráfico da pergunta 1 do Apêndice A

De acordo com as respostas da pergunta 1, representadas na Figura 2, a grande maioria dos participantes da pesquisa é originária do Banpará (55,6%), seguidos por Prodepa (16,7%), Sudam (16,7%) e TRT8 (11,1%).

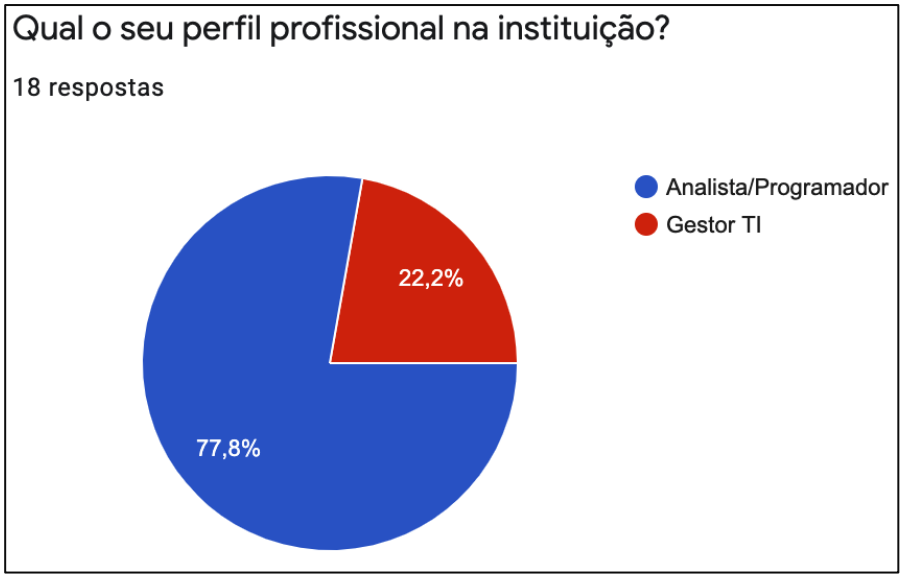


Figura 3 - Gráfico da pergunta 2 do Apêndice A

A Figura 3, representa a categorização dos profissionais que participaram da pesquisa quanto ao cargo que ocupam nas instituições. A maioria foi classificada como Analista/Programador (77,8%), seguida por Gestor de TI (22,2%).

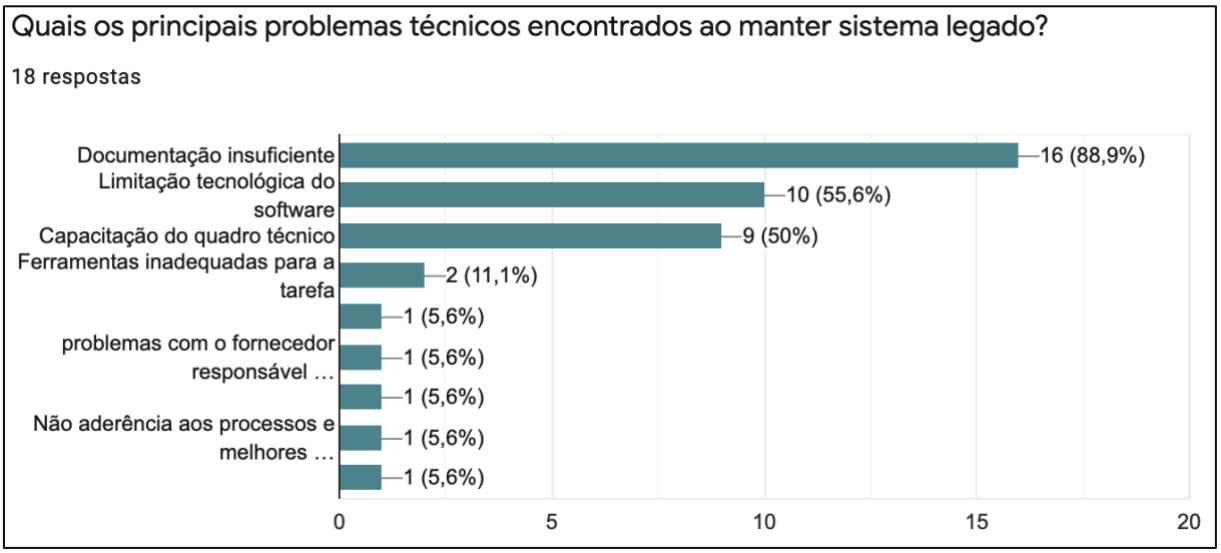


Figura 4 - Gráfico da pergunta 3 do Apêndice A

Foi observado que, os principais problemas técnicos que prejudicam o processo de manutenção ou evolução tecnológica encontrados nas instituições estão relacionados à documentação insuficiente, limitação tecnológica do software e a capacitação do quadro técnico funcional (Figura 4).

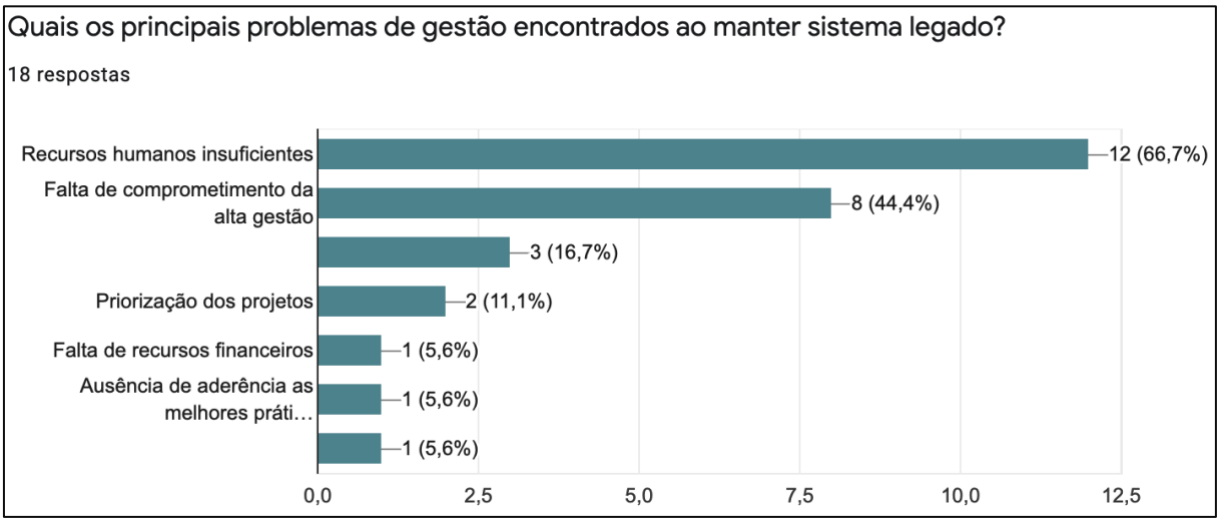


Figura 5 - Gráfico da pergunta 4 do Apêndice A

Pela ótica da gestão, o principal problema está relacionado aos recursos humanos insuficientes para desenvolver as tarefas relacionadas à manutenção desses sistemas, seguido pela falta de comprometimento da alta gestão institucional (Figura 5).

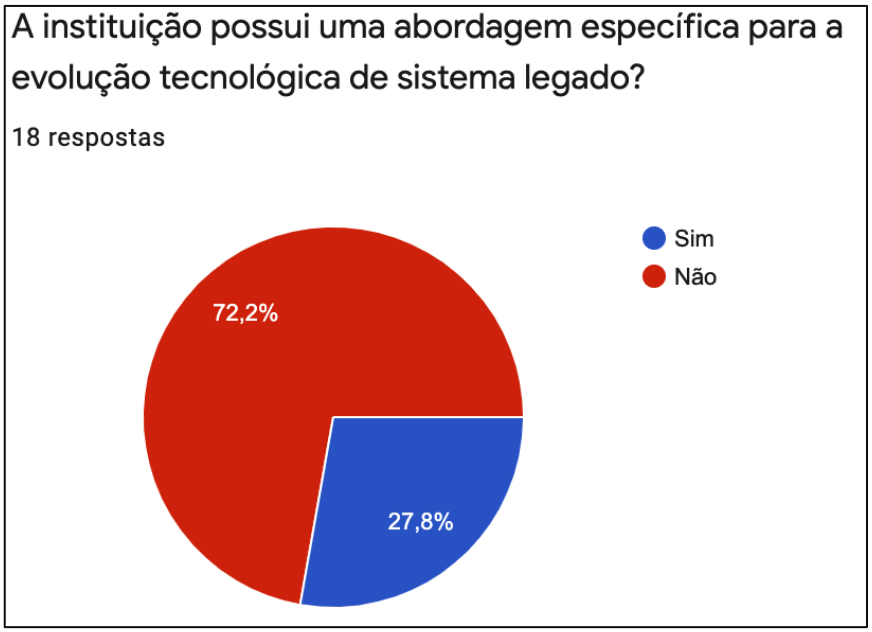


Figura 6 - Gráfico da pergunta 5 do Apêndice A

A maioria das instituições entrevistadas não possuem uma abordagem própria para lidar com a evolução tecnológica de sistemas legados (72,2%), embora haja um pequeno conjunto de instituições que já se buscou uma alternativa específica para esse problema (27,8%) (Figura 6).

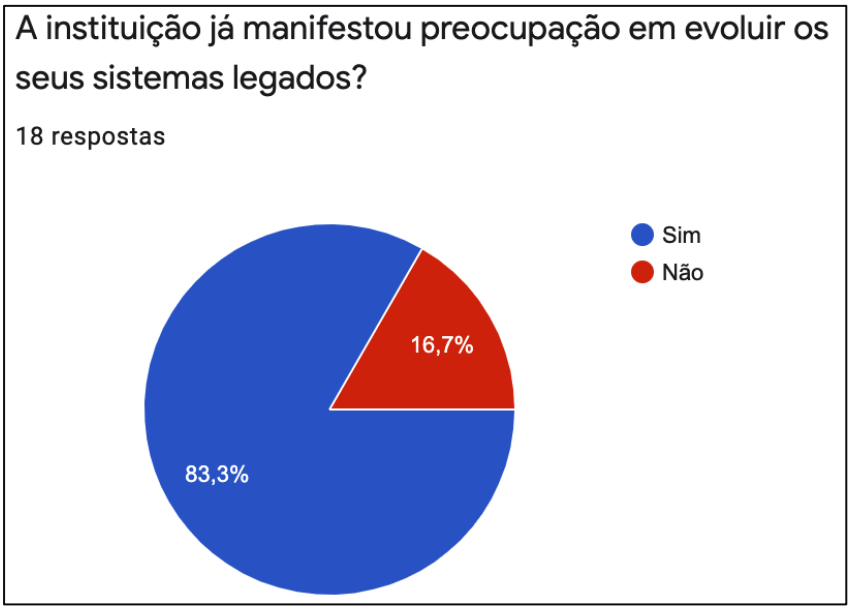


Figura 7 - Gráfico da pergunta 6 do Apêndice A

A maioria das instituições já manifestou preocupação com a evolução tecnológica dos seus sistemas legados (Figura 7).

Figura 8 - Gráfico da pergunta 7 do Apêndice A

Sobre ter desenvolvido atividades de evolução tecnológica em seus sistemas legados, a Figura 8 representa as respostas coletadas. A grande maioria já desenvolveu

trabalhos relacionados (88,9%), em comparação com as instituições que nunca se envolveram nesse tipo de empreitada (11,1%).

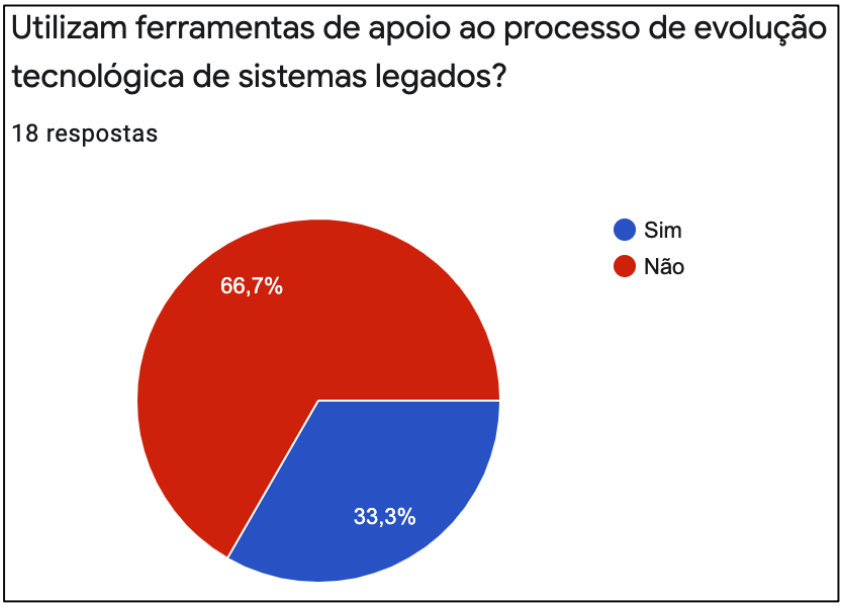


Figura 9 - Gráfico da pergunta 8 do Apêndice A

A maior parte dos profissionais consultados não fazem uso de ferramentas de apoio à atividade de reengenharia dos sistemas legados (66,7%) em comparação com os que utilizam (33,3%) (Figura 9).

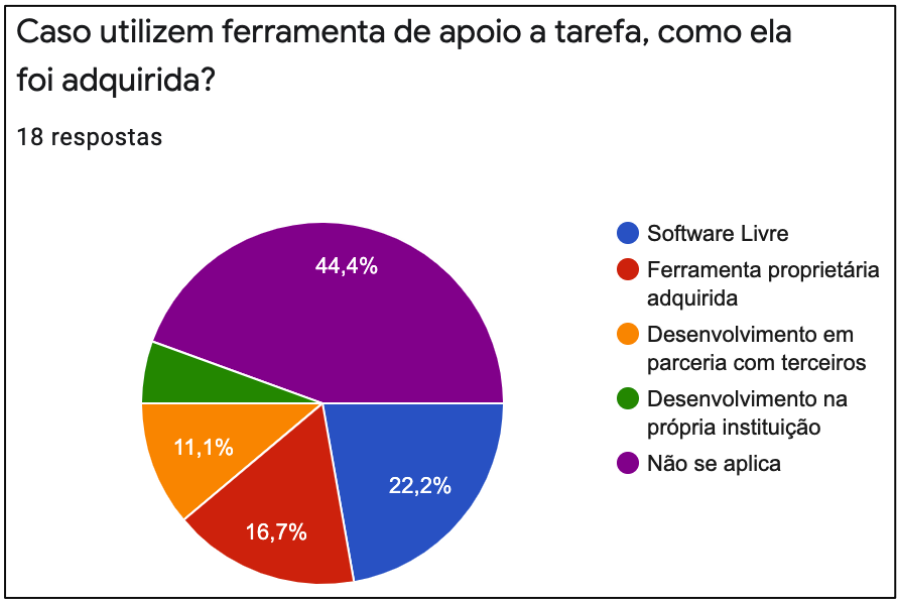


Figura 10 - Gráfico da pergunta 9 do Apêndice A

De acordo com a Figura 10, as instituições que já desenvolveram esse tipo de atividade, na sua maioria, utilizaram software livre para desempenhá-la (22,2%). Outra parcela utilizou ferramentas proprietárias através de aquisição (16,7%). Um percentual de 11,1% dos participantes desenvolveu ferramentas através de parceria com terceiras e, cerca de 5,6% dos participantes optaram por desenvolver uma ferramenta na própria instituição para essa finalidade. Mas a informação mais latente é que 44,4% das instituições envolvidas na pesquisa nunca utilizaram qualquer ferramenta de apoio a esse tipo de tarefa.

Ao analisar o resultado da pesquisa, de acordo com os profissionais participantes, a grande maioria das instituições, em alguma circunstância, necessitaram desenvolver atividades de reengenharia de sistemas legados, porém, adotaram abordagens e ferramentas que são utilizadas normalmente para a engenharia de software tradicional. Apesar disso, manifestaram o interesse em adotar estratégias apropriadas para esse tipo de projeto.

2.2. MÉTODO UTILIZADO

O método de pesquisa adotado será o exploratório, pois há a necessidade de estudo sobre diversas técnicas de reengenharia de software e todas as subatividades

relacionadas nesse processo. Será adotado também o método experimental, para a análise da aplicabilidade da abordagem proposta. Segundo Travassos (2002):

> O método experimental sugere o modelo, desenvolve o método qualitativo e/ou quantitativo, aplica um experimento, mede e analisa, avalia o modelo e repete o processo.

A finalidade dessas escolhas é definir uma análise de estudos primários sobre as técnicas de reengenharia de software, ferramentas de apoio a essa tarefa e como planejar o experimento a partir dos conceitos de engenharia de software experimental abordados em Travassos (2002).

2.3. REVISÃO SISTEMÁTICA DA LITERATURA

A revisão sistemática da literatura (RSL) estabeleceu o protocolo de pesquisa a cerca da problemática do trabalho, assim como os resultados obtidos após a apreciação da pesquisa aos critérios definidos. O detalhamento desse protocolo é apresentado nas subseções a seguir.

2.3.1. Questão de Pesquisa

A questão da pesquisa está relacionada a seguinte pergunta: Como realizar a evolução tecnológica de software legado, em órgãos da administração pública, de forma eficiente, assegurando o funcionamento institucional? Para responder esta questão, foram definidas as questões apresentadas na Tabela 3. A primeira coluna apresenta o código da questão da pesquisa, e a segunda a sua descrição.

Tabela 3 – Questões da Revisão de Literatura

Questões	Descrição
QP1	Que tipo de sistema é candidato a sofrer processo de reengenharia?
QP2	Há trabalhos relacionados a essa prática?
QP3	Quais os riscos identificados para esse trabalho?
QP4	Que metodologias estão sendo utilizadas na evolução/reengenharia de sistemas legados no contexto analisado?

2.3.2. *String* de Busca

As palavras chaves utilizadas para obtenção dos estudos primários desta revisão são: Software Legado e Reengenharia de Software. A partir desse conjunto de palavras foi gerada uma *string* de busca utilizada na RSL (Tabela 4), cujos objetivos são de encontrar publicações que abordem processos de reengenharia de sistemas legados e ferramentas que apoiaram esses processos. Para maximizar a quantidade de estudos encontrados na busca, foram utilizados sinônimos de algumas palavras.

Tabela 4 – *Strings* de Busca

String	Objetivo
(Process OR Approach) AND Reengineering AND (Legacy Software)	Abordagens de reengenharia de sistemas legados
MDD (Tools OR Software)	Ferramentas voltadas para a abordagem de desenvolvimento orientada a modelos
Experimental software engineering	Trabalhos envolvendo o uso de engenharia de software experimental

2.3.3. Critérios de Inclusão e Exclusão

Para a definição dos estudos primários, encontrados por meio da *string* de busca. Estes foram submetidos a análise das palavras chaves e leitura do resumo, a fim de identificar aqueles relevantes para responder às questões de pesquisa. Estes artigos pré-selecionados foram submetidos às listas de Critérios de Inclusão (CI) e Critérios de Exclusão (CE) que constam na Tabela 5.

Tabela 5 – Critérios de Inclusão e Exclusão de Artigos

Critérios de Inclusão	Critérios de Exclusão
CI1: Os artigos devem estar disponíveis na web; **CI2**: Os artigos devem apresentar textos completos dos estudos em formato eletrônico; **CI3:** Os artigos devem contemplar a execução de estudos experimentais investigando processos de reengenharia de sistemas legados; **CI4:** Artigos publicados entre 2010 e 2019. **CI5:** Periódicos revisados por pares.	**CE1:** Divergência significativa entre *string* de busca e palavras-chaves; **CE2:** Trabalhos sobre reengenharia em áreas distintas a sistemas legados; **CE3:** Estudos com texto completo não disponível em formato eletrônico; **CE4:** Artigos que apenas propõem uma abordagem ou descrevem hipóteses sobre o processo de reengenharia de sistemas legados, sem um experimento comprobatório.

Implicitamente os estudos duplicados foram ignorados. A partir de então, os demais trabalhos seguiram para a fase de extração e síntese dos dados.

2.3.4. Trabalhos Relacionados

Nesta fase, foram extraídos os seguintes dados sobre os estudos selecionados: título da obra, nomes dos autores, data de publicação, veículo de publicação, fonte, resumo, data da execução do estudo, local do estudo, tipo de estudo, descrição do estudo, hipóteses avaliadas, trabalhos relacionados à reengenharia de sistemas legados, técnicas utilizadas, número de participantes, projeto do estudo, ameaças a validade, resultados e referências relevantes. Esses dados extraídos foram armazenados em uma planilha, para fins de organização do referencial bibliográfico da pesquisa.

A resultante da busca e análise dos estudos é um total de 33 artigos encontrados, onde foram pré-selecionados 14 trabalhos. A Tabela 6 apresenta os resultados do processo de pré-seleção e inclusão dos artigos.

Tabela 6 - Resultado geral das buscas por publicações.

Fonte	Total de artigos	Pré-selecionados	Incluídos	Incluídos / Total artigos (%)
IEEE	11	5	3	27%
Science Direct	10	3	2	20%
Springer Link	12	6	3	25%
Total	**33**	**14**	**8**	**24%**

Após uma seleção criteriosa, os trabalhos selecionados foram divididos em 3 (três) grupos relacionados para alicerçar a fundamentação teórica do trabalho. Autores que abordaram esses temas em obras de notório conhecimento da comunidade acadêmica também foram citados. Essas áreas de pesquisa são apresentadas nas seções a seguir.

2.4. FUNDAMENTAÇÃO TEÓRICA

2.4.1. Aplicações de MDD

De acordo com Pressman (p. 807), as ferramentas de apoio automatizado a engenharia de software (CASE – *Computer Aided Software Engineering*) automatizam a atividade de gestão de projetos, gerenciam todos os produtos de trabalho produzidos ao longo do processo e assistem os engenheiros em seu trabalho de análise, projeto, codificação e teste. A ferramenta *Kirn Framework*, que será usada no processo de reengenharia do software legado, é um exemplo dessa categoria de ferramenta, pois trabalha na análise de modelos computacionais, baseados no banco de dados implementado, gerando uma aplicação base que será customizada no processo de modernização dos sistemas a serem evoluídos.

Sobre *Model-Driven Engineering* ou Engenharia orientada a modelos, Pressman (2016) aborda que:

> acopla linguagens de modelagem específicas de domino com mecanismo de transformação e geradores para facilitar a representação da abstração em altos níveis e, então, transforma-as em níveis mais baixos.

Muitos desses modelos encontram-se em padrões abertos como o XMI, o que possibilita a sua leitura através de ferramentas CASE, viabilizando assim a geração de artefatos de software de maneira automatizada, promovendo assim o MDD (*Model-Driven Development*), tornando o desenvolvimento de software uma atividade mais produtiva. A *Kirn Framework* utiliza um arquivo de modelo conhecido como KML, gerado através da leitura dos metadados dos sistemas legados, como principal modelo de projeto para apoio na geração da aplicação base, para fins de reengenharia, ou mesmo, para desenvolvimento de novas aplicações, onde a base de dados materializada na sua camada física possa ser usada como requisito.

A ferramenta *Kirn Framework,* desenvolvida para apoiar a abordagem *Kirn,* utiliza modelos a partir de um esquema de banco de dados modelado para gerar artefatos de software automaticamente. Os trabalhos relacionados nessa seção serviram de base para a construção dessa ferramenta, bem como justificar o uso de MDD em ferramentas de apoio às atividades de engenharia de software.

Alguns trabalhos abordam a utilização de modelos como requisitos para o processo de reengenharia automatizada de software, tais como Papotti (2012), que utiliza arquivos XMI (XML *Metadata Interchange*) como insumo para a geração de artefatos de software de um determinado projeto. O trabalho de Cerny (2010) apresenta uma proposta de analisar o código fonte de uma aplicação, usando esse código como modelo para a geração de artefatos de interface gráfica, figurando como importante referência no desenvolvimento de trabalhos voltados ao uso de MDD. O trabalho de Paige (2012) apresenta um relato de experiências no desenvolvimento de duas ferramentas. A ferramenta Epsilon (2010) possui um conjunto de linguagens textuais para descrição de modelos que são interoperáveis entre si. A ferramenta Viatra (2020) é uma ferramenta voltada para apoiar os ciclos de transformações dos modelos em artefatos de software. O relato das experiências em construir essas ferramentas foi fundamental para o desenvolvimento da *Kirn Framework* e determinante para os registros de melhorias previstos para trabalhos futuros.

Sobre o mapeamento objeto relacional, Bauer (2005) define que consiste na persistência automatizada e transparente de objetos em uma aplicação para as

tabelas em um banco de dados relacional, usando metadados que descrevem o mapeamento entre os objetos e o banco de dados. A ferramenta *Kirn Framework* utiliza esse recurso para extrair o modelo de dados, com o objetivo de criar uma aplicação orientada a objetos, adotando esse paradigma como padrão para o processo de modernização dos sistemas legados. Esse processo é detalhado no Capítulo 3.

2.4.2. Engenharia de Software Experimental

Os objetivos relacionados à execução de experimentos em Engenharia de Software, segundo Travassos (2002), são a caracterização, avaliação, previsão, controle e melhoria a respeito de produtos, processos, recursos, modelos, teorias entre outros. O uso de indicadores quantitativos assegura o aspecto científico do experimento. Para esse trabalho, é planejado um experimento cuja finalidade é observar os protótipos que serão desenvolvidos da abordagem *Kirn* em contrapartida à abordagem institucional de desenvolvimento de software, através dos indicadores estabelecidos, com a finalidade de verificar qual abordagem apresenta os melhores aspectos de eficiência. O detalhamento da estrutura e execução do experimento podem ser apreciados no Capítulo 6.

Em Lindgren (2016) pode ser visto uma pesquisa realizada em 13 (treze) empresas da Finlândia, cuja finalidade é identificar a prática de experimentação na indústria de software, identificando os principais desafios e fatores de sucesso associados à essa abordagem. Ela mostra que apesar da engenharia de software experimental está em evidência no debate de profissionais da indústria, é uma abordagem que ainda necessita de amadurecimento de suas práticas. Apenas com a propagação da cultura de experimentação, as abordagens podem ser validadas de forma mais constante nas organizações podemos melhorar os processos e o desenvolvimento/escolha de ferramentas adequadas para as atividades de engenharia ou reengenharia de software.

Uma das grandezas a serem medidas no experimento é o tamanho do software, e para isso este trabalho utiliza a análise de pontos de função (APF) que tem como medida de tamanho o ponto de função (PF). Citada pela primeira vez por Albrecht

(1979), é uma das principais abordagens de medida de esforço de desenvolvimento de software no mercado, e é mantido pelo IFPUG. Segundo Vazquez (2012), é um método padrão para medir o desenvolvimento de software do ponto de vista do usuário. As funcionalidades, assim como a abstração de armazenamento, conhecido como arquivo, são responsáveis pela quantificação do esforço de desenvolvimento do projeto.

2.4.3. Reengenharia de Software

Fusion/RE-I é um processo de desenvolvimento de software, dissertado por Figueiredo (1998), baseado no Fusion-RE, desenvolvido na Universidade de São Paulo (USP) em São Carlos, voltado para a modernização de sistemas legados, com o incremento da análise da interface gráfica e utilizando-a como requisito do projeto. Esse processo visa segmentar os requisitos que serão usados no processo de reengenharia em duas visões: funcional e estrutural. Na visão funcional serão extraídos requisitos oriundos da interface gráfica, dos documentos em geral e os diagramas técnicos, caso existam. Na visão estrutural serão analisados o código fonte e banco de dados. Após o levantamento desses requisitos, será feito o relacionamento entre as visões, a fim de serem identificadas as responsabilidades de cada artefato de software encontrado. O processo foi escolhido como a base da abordagem aqui apresentada pela forma organizada de identificar, classificar e utilizar os requisitos presentes no domínio do problema.

O trabalho de Goltz (2014) descreve um importante estudo sobre evolução contínua de software financiado pela *Deutsche Forschungsgemeinschaft* (DFG – Fundação de Pesquisas da Alemanha) ressaltando a necessidade de se desenvolver metodologias, conceitos e ferramentas inovadoras de engenharia de software que se concentram no suporte à melhoria contínua de sistemas complexos, assim como do seu ambiente. Uma das questões que torna complexa a atividade de reengenharia é o fato de que um software possui diversos componentes associados que não tiveram os seus ciclos de vida de software gerenciados de forma harmônica, depreciando todo o conjunto de software, middleware, plataformas e hardware. Ao todo, quinze (15) abordagens foram selecionadas para que seus resultados em estudos de caso fossem avaliados.

Em Pfleeger (2004) é apresentada uma proposta de processo de reengenharia que tem como foco analisar um sistema legado, visando organizar o conjunto de informações vinculadas, tornando-o mais inteligível, considerando os seguintes aspectos: redocumentação, reestruturação, engenharia reversa e reengenharia.

Em Pressman (2016) o autor identifica a necessidade de uma estratégia bem definida para apoiar a reengenharia de sistemas de informação em decorrência do seu custo significativo e absorção de recursos de TI durante muitos anos. O processo proposto é composto por seis (6) atividades, onde cada atividade pode ser revisitada e o processo pode ser encerrado após a realização de qualquer uma das atividades, caso apresente necessidades pontuais de modificação. As atividades desse processo são: análise do inventário, reestruturação de documentos, engenharia reversa, reestruturação de código, reestruturação dos dados e engenharia direta.

Em Sommerville (2011) é abordada a importância de se aplicar reengenharia de sistemas legados, visando a melhoria da sua estrutura e de sua inteligibilidade, em decorrência das diversas manutenções sofridas, que ocasionaram uma depreciação desses sistemas. O autor apresenta uma proposta de processo de reengenharia, cujas atividades são: Tradução de Código Fonte; Engenharia Reversa; Melhoria das Estruturas de Programa; Modularização de Programa; Reengenharia de Dados.

Em Gholami (2017) aborda a migração de sistemas legados para plataformas de computação em nuvem, e define uma estrutura organizada dos recursos a serem utilizados no experimento, tais como os profissionais envolvidos e suas experiências, assim como o processo crítico de transformação dos requisitos em artefatos do software a serem utilizados no processo de modernização. Esse processo é descrito de maneira resumida na Figura 11.

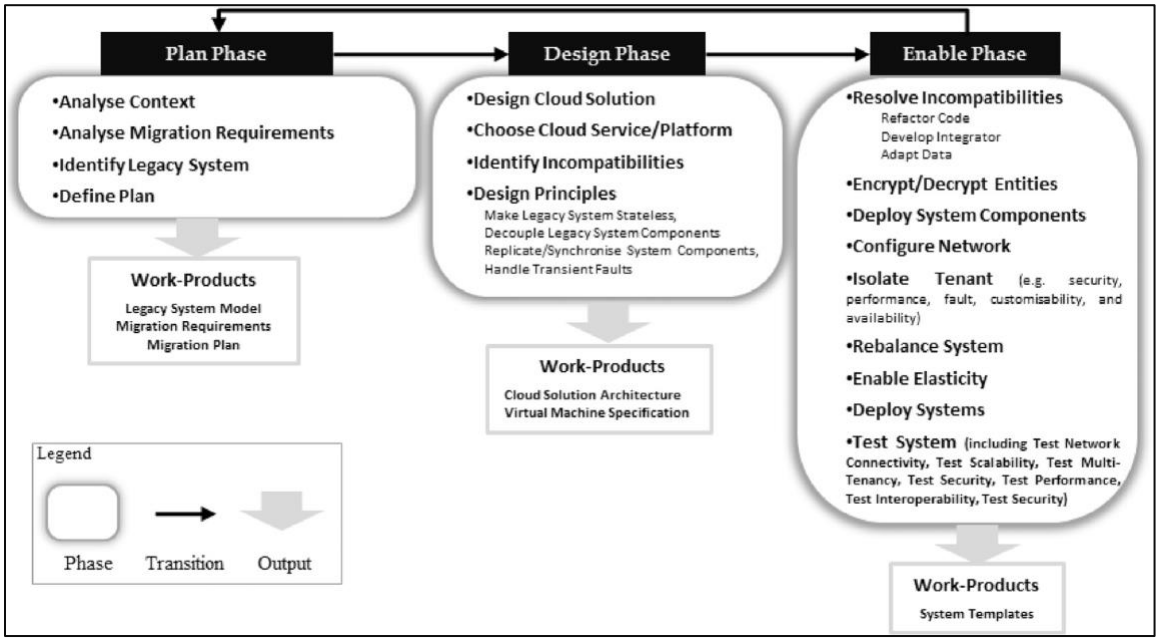


Figura 11 - Elementos críticos do processo em migrar sistemas legados para plataformas em nuvem (GHOLAMI, 2017)

2.4.4. Padrões de Projeto

Os padrões de projeto adotados na ferramenta foram baseados nos conhecidos catálogos encontrados em Gamma (1994) e Larman (2007). Esses padrões sugerem soluções amplamente utilizadas para problemas recorrentes em projetos de software. Dentre os diversos padrões adotados pela *Kirn Framework*, foram destacados os seguintes padrões a seguir.

O padrão *Adapter* converte a interface de uma classe em outra interface esperada pelos clientes (GAMMA, 1994). A classe BD, por padrão da ferramenta, é gerada para estender a classe BDBase, a qual contém as instruções para acessar um dos bancos de dados relacional que a ferramenta suporta. Porém, se a fonte de dados for diferente da previamente configurada, tal como, banco de dados noSQL, API externa, arquivo XML, entre outros, basta criar um adaptador adequado à necessidade apresentada.

O padrão *Façade* fornece uma interface unificada para um conjunto de interfaces em um subsistema (GAMMA, 1994). A classe Controller é uma classe abstrata que é acessada pelas classes Controller específicas de uma determinada entidade de

negócio. Dessa forma, a manutenção e evolução de funcionalidades específicas de uma determinada entidade, se dá de uma maneira mais eficiente e com qualidade.

O padrão de projetos *Proxy* fornece um objeto representante, ou um marcador de outro objeto, para controlar o acesso ao mesmo (GAMMA, 1994). Os relatórios desenvolvidos na aplicação gerada são classes Proxy, cuja finalidade é criar uma interface de acesso a diversas classes de negócio, sem a necessidade de manipulá-las para atender demandas estabelecidas, que serão implementadas na classe Proxy

O padrão *Low Coupling* tem a finalidade de atribuir responsabilidades de modo que acoplamento (desnecessário) permaneça baixo (LARMAN, 2007). A Kirn Framework é frequentemente avaliada em termos arquiteturais, o que gera mudanças rotineiras na sua forma de promover o desenvolvimento de softwares. Por conta dessa natureza, desenvolver uma arquitetura que possibilite mudanças que busque o mínimo de impacto auxilia nessa modernização regular da ferramenta. A aplicação gerada segue o mesmo princípio. Quanto menor o acoplamento entre os componentes, maior é a qualidade das manutenções no código, sejam elas corretivas, evolutivas, adaptativas ou preventivas (PLEEFGER, 2004).

O padrão de projetos *High Cohesion* tem como objetivo atribuir responsabilidades às classes de modo que a coesão permaneça alta (LARMAN, 2007). Em virtude de uma manutenibilidade mais facilitada, dividir as responsabilidades em classes específicas proporciona um contexto onde as alterações no software preservem a qualidade adquirida em alterações anteriores, causando o mínimo de impacto.

3. ABORDAGEM KIRN

Segundo Jamieson (1825), *Kirn* é uma antiga palavra de origem inglesa que pode ser associada ao mesmo tempo a colheita e a resultado, criando uma associação intuitiva com a abordagem, pois trata-se de um processo de análise de requisitos, que em algum momento foram produzidos para atender os sistemas legados, que serviram de insumo para o processo de sua modernização.

O propósito da abordagem *Kirn* é auxiliar na identificação automática de requisitos do software legado, a partir da sua base de dados e utilizá-la como insumo para a geração de uma aplicação base (através da ferramenta *Kirn Framework*) que será customizada no processo de reengenharia. Os demais requisitos, que não puderam ser gerados automaticamente, são mapeados a fim de organizar o *backlog* de funcionalidades que serão desenvolvidas no processo de customização pela equipe de desenvolvedores.

O processo *Kirn* descreve o conjunto de atividades, com os seus respectivos elementos de caracterização (papeis, insumos, fases, ferramentas, entre outros), objetivando o desenvolvimento da atividade de reengenharia dos sistemas legados.

Esse capítulo está organizado pelas seguintes seções:

- 3.1 KIRN FRAMEWORK: Ferramenta de apoio ao desenvolvimento de software adotada pela abordagem para atuar no processo de evolução tecnológica dos sistemas legados.
- 3.2 FASES DO PROCESSO: Informações relacionadas às atividades do processo de desenvolvimento de software e o que cada uma representa no contexto geral do projeto.
- 3.2 PAPEIS: Descreve as responsabilidades dos participantes do projeto de software;
- 3.3 INSUMOS E ARTEFATOS DE SOFTWARE PRODUZIDOS: Dados e Informações que serão utilizados na abordagem, assim como a relação de artefatos de software produzidos no decorrer das fases do processo de reengenharia do software.

Após a geração da aplicação base, o processo de customização se inicia com o trabalho da equipe de desenvolvimento. Através de reuniões de planejamento, serão definidas as iterações de desenvolvimento, agrupando os requisitos de software que irão compor cada um desses ciclos de natureza iterativo incremental (PRESSMAN, 2016).

3.1. *KIRN FRAMEWORK*

Kirn Framework é uma ferramenta desenvolvida pelo autor do trabalho em 2018 cuja finalidade é gerar aplicações a partir da leitura de um banco de dados relacional implementado. Atualmente a ferramenta possui suporte para a geração de aplicações em linguagem de programação PHP e preparadas para serem executadas em servidor de aplicações web.

A *Kirn Framework* não foi desenvolvida especificamente para atividades de reengenharia, mas como o seu principal requisito para a geração das aplicações é a existência de um esquema de banco de dados e como diversos sistemas de informação possuem esse componente, a ferramenta surge como uma alternativa para esse tipo de atividade.

Esse capítulo está organizado pelas seguintes seções:

- 3.2.1 APRESENTAÇÃO: Informações sobre o funcionamento da ferramenta e suas características.
- 3.2.2 KML: *Kirn Model Language*. Apresentação da estrutura do arquivo gerado pela ferramenta com as informações sobre o esquema de banco de dados que servirá de base para a geração da aplicação;
- 3.2.3 ARQUITETURA DA FERRAMENTA: Informações sobre a arquitetura da ferramenta e os componentes que a constituem;
- 3.2.4 ARQUITETURA DA APLICAÇÃO GERADA: Apresentação dos elementos arquiteturais da aplicação gerada;
- 3.2.5 TEMPLATES: Estrutura dos arquivos de modelo que servem de base para a geração das aplicações pela ferramenta;
- 3.2.6 PADRÕES DE PROJETO: Lista do conjunto de padrões de projeto utilizados pela ferramenta, com as respectivas motivações para as escolhas.

3.1.1. Apresentação

O funcionamento da ferramenta ocorre através do fluxo ilustrado na Figura 12 a seguir.

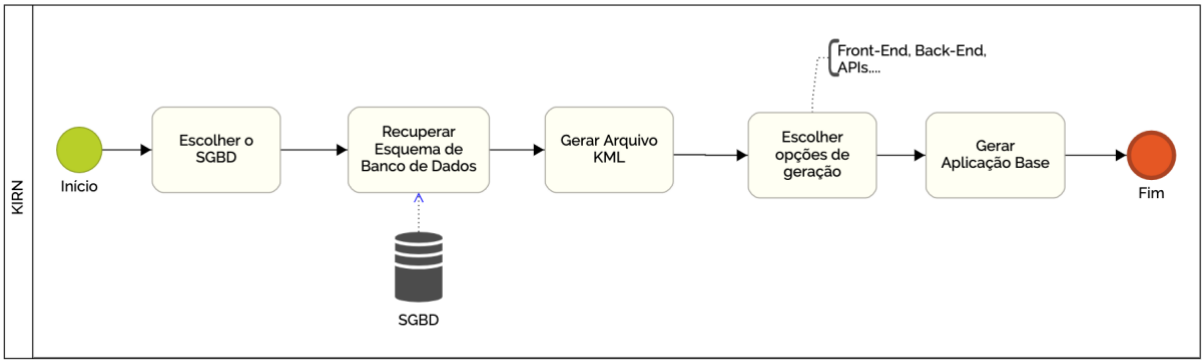


Figura 12 - Processo de geração de aplicações pela *Kirn Framework*

A ferramenta realiza a leitura dos metadados obtidos a partir dos sistemas gerenciadores de banco de dados (SGBDs) relacionais MySQL, MS SQL Server e PostgreSQL. Essa ferramenta foi desenvolvida em linguagem PHP, podendo ser utilizada em computadores que seguem as configurações apresentadas na Tabela 7.

Tabela 7 – Requisitos de sistemas para utilização da Kirn Framework

Processador	32 bits (x86) ou 64 bits (x64) de 1 GHz ou superior
Memória RAM	2GB de Memória RAM
Espaço em Disco	40MB
Sistema Operacional	-
Linguagem de Programação	PHP 5.6 ou Superior
Servidor de aplicação	Apache, IIS, Nginx, PHP Server
Bibliotecas	MySQL Nativo, PostgreSQL Nativo, PDO, libxml

A Figura 13 apresenta a tela inicial da ferramenta que é acessada através de um navegador web.

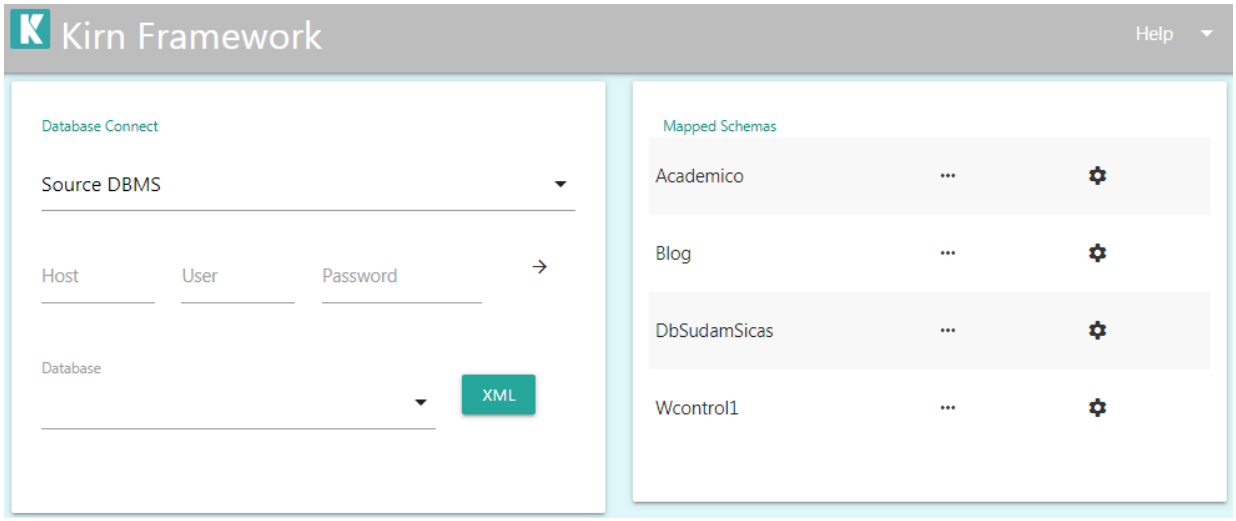


Figura 13 - Tela inicial da Kirn Framework

Ao acessar a ferramenta, ela apresenta duas áreas de interação. A primeira, que pode ser vista na Figura 14, permite que o usuário selecione um SGBD, para que possa acessar um determinado *schema* de dados.

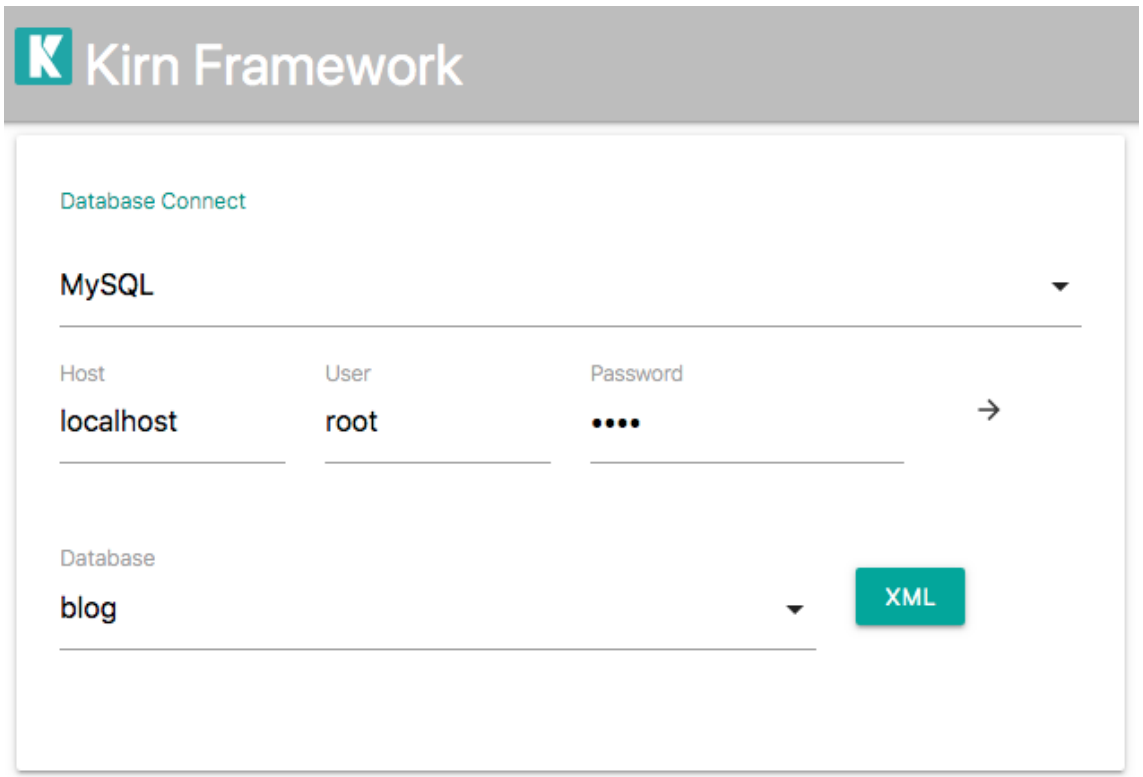


Figura 14 - Conexão com o banco de dados da Kirn Framework

Após a parametrização dos campos de entrada (*DBMS, Host, User, Password*), a conexão com o SGBD é efetuada. O campo *Database* será preenchido com os *schemas* disponíveis para o usuário do SGBD em questão. O usuário escolherá uma opção do campo *database* para gerar um arquivo XML com os metadados encontrados. Esse arquivo servirá de base para a geração de todos os artefatos de software da aplicação base. Os detalhes sobre a estrutura desse arquivo, serão explanados na seção a seguir.

Os arquivos XML gerados serão listados na área *Mapped Schemas*, conforme apresenta a Figura 15.

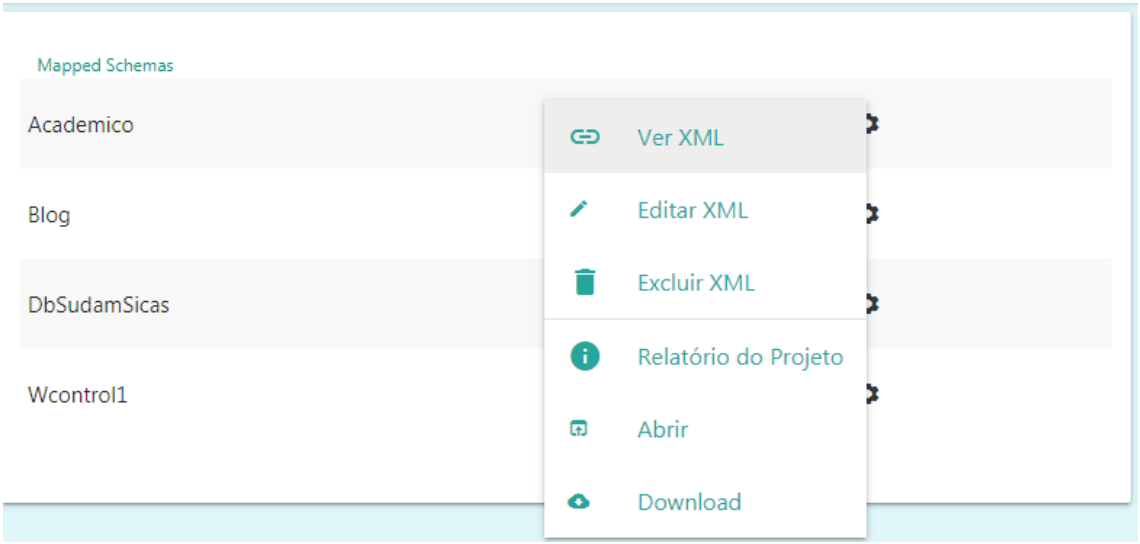


Figura 15 - Lista de base de dados extraídas dos SGBDs

Esses arquivos guardam as informações básicas necessárias para a geração da aplicação base. Para manipulá-los, são disponibilizadas as seguintes operações:

- **Ver XML:** Visualização do documento XML gerado pela leitura do modelo de dados;
- **Editar XML:** Edição do arquivo XML gerado.
- **Excluir XML:** Exclui o arquivo XML gerado;
- **Relatório do Projeto: Apresenta**
- **Abrir:** Visualizar a aplicação gerada, em uma janela a parte.
- **Download:** Efetuar o download da aplicação gerada.

As opções de configuração para a geração da aplicação base são apresentadas Figura 16.

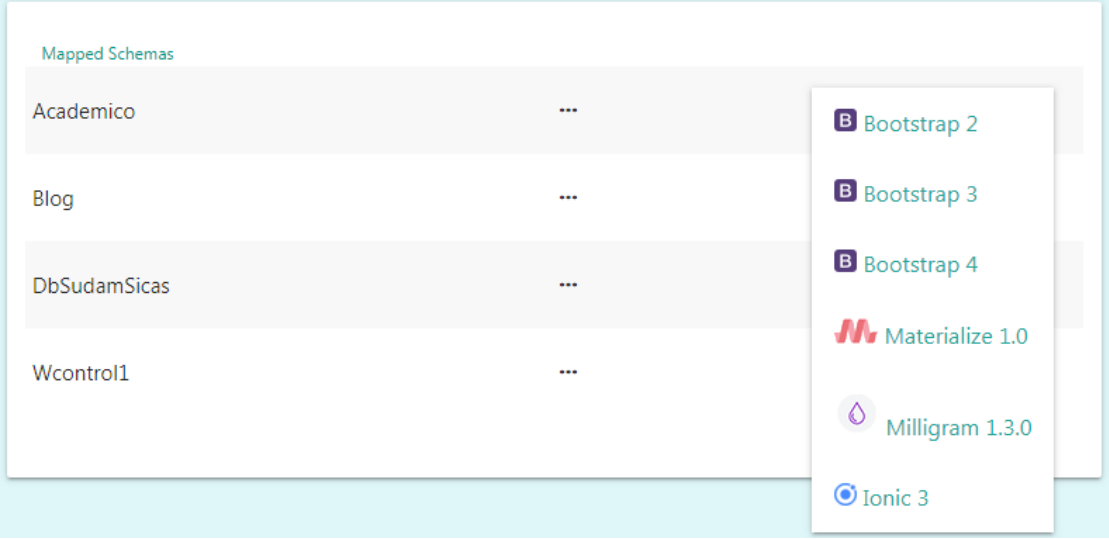


Figura 16 – Opções de geração de aplicação base

Essas opções estão ligadas ao tipo de *framework front-end*[1] que se deseja utilizar na aplicação gerada. As opções são:

- **Bootstrap 2:** Geração da aplicação base, tendo o Bootstrap (2012) como *framework* gráfico;
- **Bootstrap 3:** Geração da aplicação base, tendo o Bootstrap (2013) como *framework* gráfico;
- **Bootstrap 4:** Geração da aplicação base, tendo o Bootstrap (2017) como *framework* gráfico;
- **Materialize 1.0:** Geração da aplicação base, tendo como *framework* gráfico o Materialize (2014).
- **Milligram 1.3.0:** Geração da aplicação base, tendo como *framework* gráfico o Milligram (2015);

[1] Compõem os elementos da camada de visão (*View*) da aplicação.

- **Ionic 3.0:** Geração da aplicação base, tendo como *framework* de desenvolvimento de aplicações móveis híbridas o Ionic (2013);

3.1.2. KML

A KML (*Kirn Model Language*) é usada para descrever o modelo representativo da base de dados selecionada para o processo de desenvolvimento de software ou reengenharia de sistemas legados. Para aplicar os conceitos de MDD, ao escolher uma base de dados pela ferramenta, um arquivo KML é gerado pela ferramenta. Trata-se de um arquivo XML que contém todas as informações sobre a base de dados acessada, necessárias para a geração da aplicação base. A estrutura do arquivo é organizada de acordo com a Figura 17.

```
- <database name="blog" dbms="mysql" host="localhost" user="root" passwd="root">
  - <table name="Comentario" schema="" type="NORMAL">
      <column name="idComentario" type="int(11)" null="NO" pk="true" ai="true" fkTable="" fkField=""/>
      <column name="idPost" type="int(11)" null="NO" pk="false" ai="false" fkTable="Post" fkField="idPost"/>
      <column name="idPessoa" type="int(11)" null="NO" pk="false" ai="false" fkTable="Pessoa" fkField="idPessoa"/>
      <column name="webpage" type="varchar(45)" null="YES" pk="false" ai="false" fkTable="" fkField=""/>
      <column name="descricao" type="text" null="YES" pk="false" ai="false" fkTable="" fkField=""/>
      <column name="dataHoraCadastro" type="datetime" null="YES" pk="false" ai="false" fkTable="" fkField=""/>
    </table>
  - <table name="Pessoa" schema="" type="NORMAL">
      <column name="idPessoa" type="int(11)" null="NO" pk="true" ai="true" fkTable="" fkField=""/>
      <column name="nome" type="varchar(200)" null="YES" pk="false" ai="false" fkTable="" fkField=""/>
      <column name="email" type="varchar(45)" null="YES" pk="false" ai="false" fkTable="" fkField=""/>
    </table>
  - <table name="Post" schema="" type="NORMAL">
      <column name="idPost" type="int(11)" null="NO" pk="true" ai="true" fkTable="" fkField=""/>
      <column name="idPessoa" type="int(11)" null="NO" pk="false" ai="false" fkTable="Usuario" fkField="idPessoa"/>
      <column name="titulo" type="varchar(200)" null="YES" pk="false" ai="false" fkTable="" fkField=""/>
      <column name="descricao" type="text" null="YES" pk="false" ai="false" fkTable="" fkField=""/>
      <column name="dataHoraCadastro" type="datetime" null="YES" pk="false" ai="false" fkTable="" fkField=""/>
    </table>
  - <table name="Usuario" schema="" type="1:1">
      <column name="idPessoa" type="int(11)" null="NO" pk="true" ai="false" fkTable="Pessoa" fkField="idPessoa"/>
      <column name="login" type="varchar(45)" null="YES" pk="false" ai="false" fkTable="" fkField=""/>
      <column name="senha" type="varchar(45)" null="YES" pk="false" ai="false" fkTable="" fkField=""/>
      <column name="ativo" type="tinyint(1)" null="YES" pk="false" ai="false" fkTable="" fkField=""/>
      <column name="grupo" type="enum('ADMIN','USUARIO')" null="YES" pk="false" ai="false" fkTable="" fkField=""/>
    </table>
  </database>
```

Figura 17 - Estrutura do arquivo KML

Na próxima subseção, o detalhamento das *tags* e atributos que compõem a estrutura do documento KML.

3.1.2.1. Tag database

Agrupa as informações de conexão com o *database.* Está localizado no nó raíz do documento KML e orienta as configurações iniciais de acesso à aplicação base gerada pela ferramenta. Os detalhes sobre seus atributos são apresentados na Tabela 8.

Tabela 8 - Estrutura da tag "database"

Atributo	Descrição	Domínio
name	Nome do *database*	N/A
dbms	O SGBD de origem do *database.* Caso haja a necessidade de se utilizar outro SGBD para a aplicação, esse atributo pode ser alterado manualmente no arquivo XML	N/A
host	Endereço de acesso ao *database*	- Endereço IP -Localhost (127.0.0.1)
user	Usuário autorizado a acessar os dados	N/A
passwd	Senha do usuário	N/A
frontEnd	*Framework Front-End* selecionado para a aplicação gerada	- Bootstrap 2 - Bootstrap 3 - Bootstrap 4 - Materialize 1.0
backEnd	*Framework Back-End* selecionado para a aplicação gerada	Kirn Framework
totalFP	Quantidade de Pontos de Função identificados na aplicação gerada	N/A
numLineCode	Quantidade de linhas de código quantificadas na aplicação gerada	N/A

3.1.2.2. Tag table

Representa as tabelas contidas na base de dados selecionada. Tem no elemento "*database*" o seu nó raiz. Os detalhes e descrições quem compõem a lista de seus atributos podem ser conferidos na Tabela 9.

Tabela 9 - Estrutura da tag "table"

Atributo	Descrição	Domínio
Name	Nome da tabela	N/A
Schema	Nome do *schema* que a tabela faz parte. Em SGBDs que não utilizam essa estrutura, como o MySQL, ele não é utilizado;	N/A
Type	A ferramenta Kirn trabalha com três tipos de tabelas:	▪ **1:1:** Contém chave primária oriunda de outra tabela, criando uma relação onde um registro de uma tabela da relação só pode ter uma única referência na outra tabela; ▪ **N:M:** Conhecida como entidade fraca que, segundo Elmasri (2011), não possuem atributos-chave próprios. Possuem chave primaria composta, oriunda da relação de duas outras tabelas, com chave primária própria; ▪ **NORMAL:** Tabelas que não se enquadram nas definições acima.

3.1.2.3. Tag column

Representa as colunas de uma tabela da base de dados selecionada. Tem no nó "*table*" a sua raiz. Os dados sobre os atributos podem ser verificados na Tabela 10.

Tabela 10 - Estrutura da tag "column"

Atributo	Descrição	Domínio
Name	Nome da coluna	N/A
Type	Tipo de dado da coluna, definido baseado no tipo oriundo do SGBD	- Integer - String - Text - Date - Datetime - Boolean - Enum
Null	Possibilidade de assumir valores nulos	- True - False
Pk	Possibilidade da coluna representar uma chave primária da tabela	- True - False
Ai	Coluna alimentada por auto incremento do SGBD	- True - False
fkTable	Caso seja uma chave estrangeira, vem preenchido com o nome da tabela na qual mantém uma relação	Tabela contida no modelo
fkColumn	Caso seja uma chave estrangeira, vem preenchido com o nome da coluna da tabela, no qual faz relação	Atributo da Tabela contida no modelo, que desempenha o papel de chave primária nessa tabela

3.1.3. Arquitetura da Ferramenta

A Figura 18 exibe o diagrama de classes que representa a arquitetura funcional da ferramenta *Kirn Framework*.

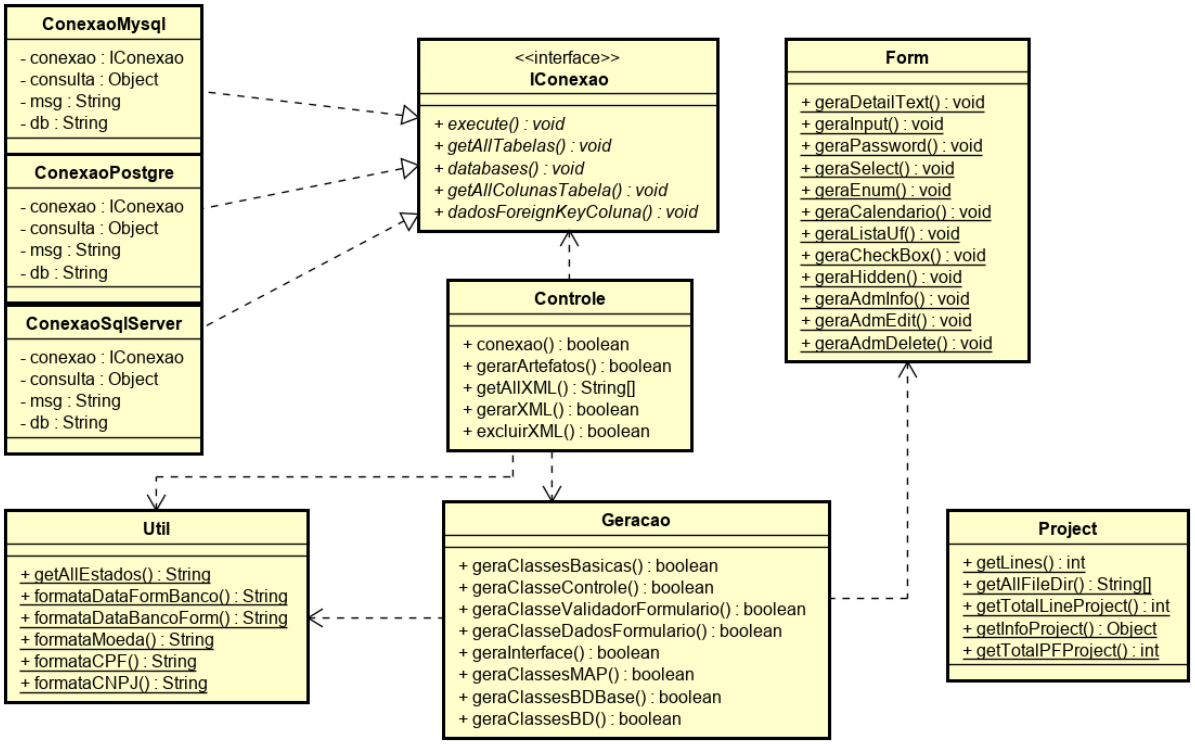


Figura 18 - Diagrama de classes da *Kirn Framework*

A seguir, um breve detalhamento sobre cada classe presente na arquitetura da *Kirn Framework* com suas respectivas responsabilidades para o funcionamento da ferramenta:

- **Controle:** Representa a fachada da ferramenta**.** Concentra as chamadas de métodos das demais classes da ferramenta

- **ConexaoMysql, ConexaoPostgre e ConexaoSqlServer:** Classes de conexão com o banco de dados, que implementam a interface **Iconexao**. Diferem entre si pelas singularidades existentes nos SGBDs Mysql, PostgreSQL e MS SQL Server, com relação a forma de acessar e manipular os dados.

- **Form:** Concentra as rotinas de geração de interface gráfica da ferramenta. Seus métodos são acionados pela classe **Geracao**;

- **Geracao:** Principal classe do gerador. Concentra as rotinas de geração dos artefatos de software;

- **IConexao:** Interface que apresenta os métodos que as classes de conexão com os SGBDs devem implementar para a manipulação dos dados;

- **Project:** Gerencia indicadores que são calculados automaticamente pela ferramenta. Os métodos são responsáveis pelas respectivas informações:

    - **getLines(file : String):** Retorna a quantidade de linhas de um determinado arquivo do código fonte do projeto.

    - **getAllFileDir(path : String):** Retorna todos os arquivos do diretório do projeto selecionado, recursivamente.

    - **getTotalLineProject(path : String):** Retorna o total de linhas de código do projeto selecionado.

    - **getInfoProject(kml : Resource):** Retorna informações do projeto selecionado, contidas no arquivo KML respectivo.

    - **getTotalPFProject(kml : Resource):** Calcula, através de análise de ponto de função, o esforço de desenvolvimento da aplicação base gerada.

- **Util:** Contém diversos métodos estáticos auxiliares, que auxiliam nas operações da ferramenta, tais como, formatação de data, validação de dados, entre outros.

Sobre o processo de captura do esquema do banco de dados que a *Kirn Framework* acessa, segue o diagrama de atividade representado na Figura 19 que detalha os passos dessa atividade.

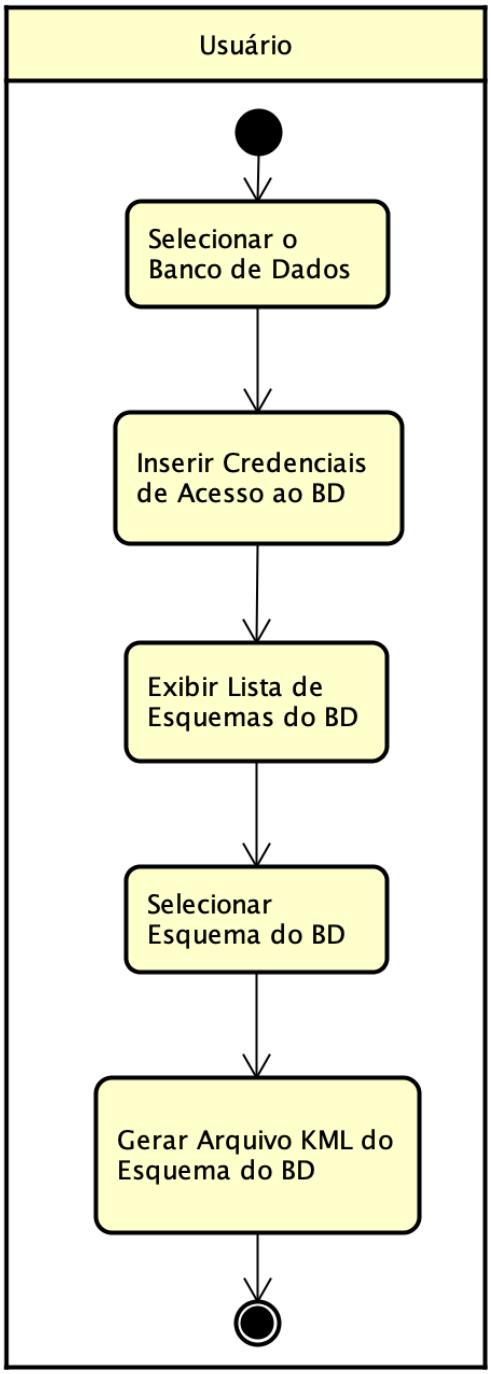


Figura 19 - Diagrama de classes da *Kirn Framework*

3.1.4. Arquitetura da Aplicação Gerada

A Figura 20 apresenta o diagrama de classes do software gerado pela *Kirn Framework* que, por sua vez, foi construído a partir do modelo XML gerado através da leitura da base de dados analisada. O diagrama de classes pode sofrer modificações de acordo com as necessidades que forem apresentadas quando o processo de customização do software for iniciado.

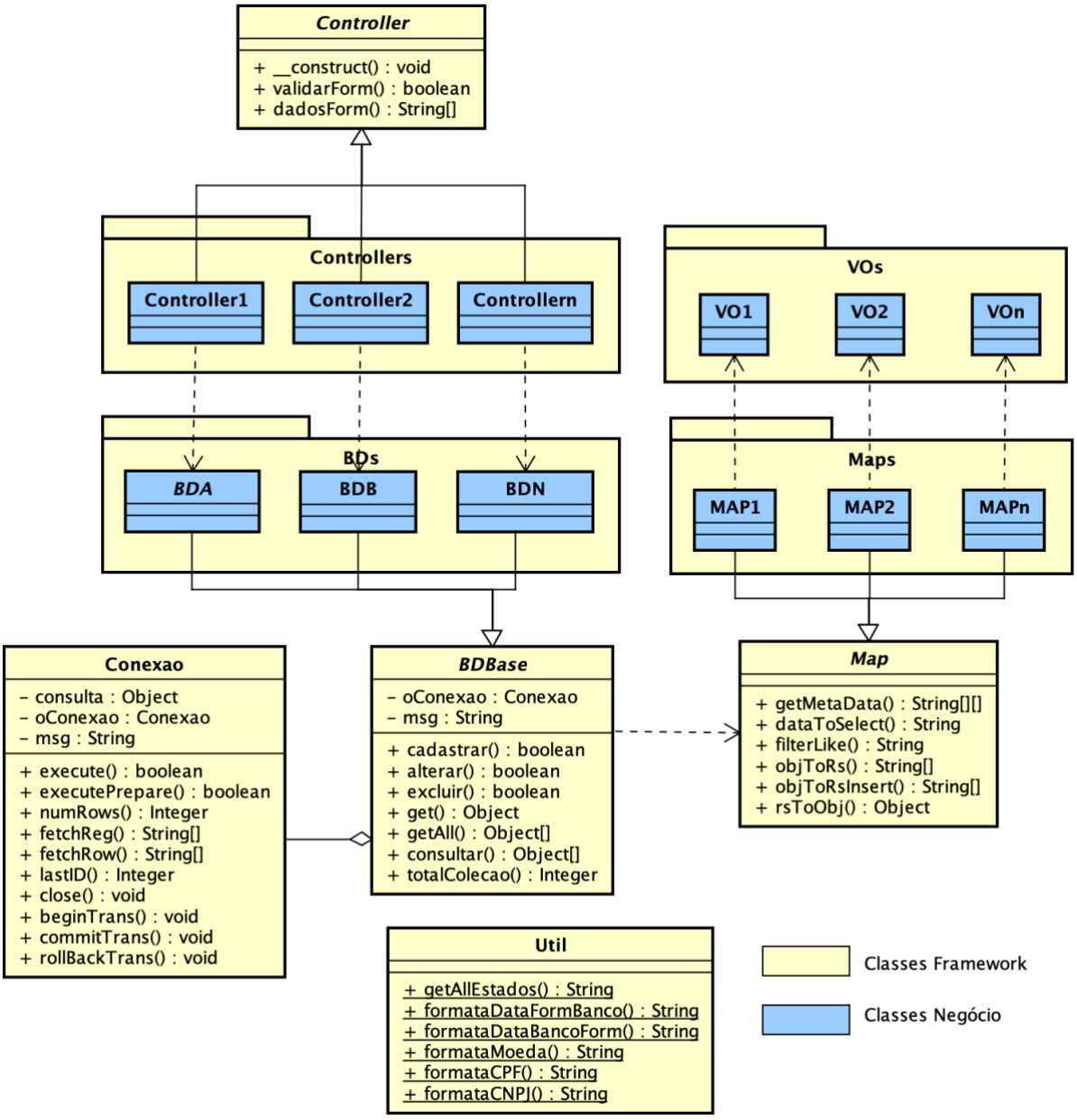


Figura 20 - Diagrama de classes da aplicação gerada pela *Kirn Framework*

As classes fornecidas pelo framework estão na cor amarela. Essas classes fornecem o suporte de funcionalidade no processo de customização da aplicação gerada. As classes de negócio estão ilustradas na cor azul. Elas são geradas a partir da leitura do esquema de banco de dados acessado e também representam as demais classes de negócio que devem implementadas manualmente pela equipe de desenvolvimento. A seguir, uma descrição das responsabilidades de cada classe:

- **Controller**: Classe abstrata, utilizada como fachada para os métodos das entidades do sistema. Contem os métodos “validarForm()”, responsável pela validação dos dados oriundos da camada de visão e o método “dadosForm()”, usado para a formatação dos dados de entrada. Cada entidade de negócio implementa a sua classe Controller para manter a alta coesão esperada em projetos orientados à objetos;

- **BD:** Extensão da classe BDBase, as classes BD recebem os métodos, que são inseridos manualmente pelo programador, que manipulam da camada de persistência. Seus métodos são acessados pelas classes concretas Controller;

- **BDBase:** Contém os métodos de manipulação da camada de persistência;

- **VO:** Classes geradas a partir das tabelas da base de dados usada como requisitos primários da aplicação gerada. Os atributos da classe têm relação direta com as colunas das tabelas encontradas, evidenciando a necessidade do uso de ORM para estabelecer essa ligação;

- **MAP:** Aplicando conceitos de ORM, as classes concretas MAP relacionam os atributos das classes com os seus atributos correspondentes no modelo de dados, que pode ser: banco de dados relacional, banco de dados noSQL, arquivos XML, APIs Rest, APIs SOAP, entre outros. As classes MAP possuem 2 métodos que resumem\ a atividade de ORM:

    - **objToRs():** Trata de converter os dados contidos na camada de controle (*Controller*), que estão em formato de objeto, para o formato que o banco de dados relacional (*Resultset*) interpreta na camada de modelo (*Model*);

- **rsToObj():** Atividade inversa ao método anterior, quando é necessário converter os dados oriundos do banco de dados relacional para objeto, no qual será usado tanto na camada de controle quanto na camada de visão (*View*);

- **Conexao:** Intermedia as chamadas das funções vindas das Classes **BD** e **BDBase**. Responsável pela conexão com o banco de dados.

3.1.5. *Templates*

A ferramenta Kirn Framework utiliza diversos *templates* que servem de base para a geração dos artefatos de software. Esses modelos são armazenados em arquivos de extensão ".tpl" e contêm códigos implementados, em princípio, somente na linguagem PHP. Como não há limitações técnicas quem impeçam a criação de *templates* para outras linguagens, esse objetivo tornou-se uma meta para trabalhos futuros, apresentado no capítulo 6. Esses modelos são constituídos por partes fixas, que compõem todos os documentos gerados, e de uma parte variável, que é modificada de forma customizada, no momento da geração dos artefatos de software, apresentado de forma exemplificada na Figura 21.

```php
/**
 * Cadastrar %%NOME_CLASSE%%
 *
 * @access public
 * @param $post
 * @return bool
 */
public function cadastrar($post = NULL){
    // recebe dados do formulario
    $post = DadosFormulario::form%%NOME_CLASSE%%($post);

    $_SESSION["post"] = $post;
    // valida dados do formulario
    $oValidador = new ValidadorFormulario();
    if(!$oValidador->validaForm%%NOME_CLASSE%%($post)){
        $this->msg = $oValidador->msg;
        return false;
    }
    // cria variaveis para validacao com as chaves do array
    foreach($post as $i => $v) $$i = utf8_encode($v);
    // cria objeto para grava-lo no BD
    %%MONTA_OBJETO%%
    %%MONTA_OBJETOBD%%
    if(!$o%%NOME_CLASSE%%BD->cadastrar($o%%NOME_CLASSE%%)){
        $this->msg = $o%%NOME_CLASSE%%BD->msg;
        return false;
    }
    unset($_SESSION["post"]);
    return true;
}
```

Figura 21 - Exemplo de *template* da ferramenta Kirn Framework

As variáveis são identificadas pela delimitação estabelecida pelos caracteres “%%” (exemplo: %%NOME_CLASSE%%, que representa o nome da classe a ser gerada pela ferramenta). O gerador de código fará uma varredura do modelo para alterar as variáveis para código customizado automaticamente.

3.2. FASES DO PROCESSO

As fases que compõem o processo proposto de reengenharia de sistemas legados estão organizadas em um framework de organização das informações tais como as etapas, insumos, atores e resultado produzido como saída. Os diagramas que representam o sequenciamento das fases do processo estão presentes nas Figura 22, representando o ciclo total da abordagem e na Figura 23, o ciclo iterativo/incremental de desenvolvimento, que é o subprocesso onde as atividades de implementação dos requisitos identificados do software submetido ao processo de reengenharia são organizadas, sequenciadas e executadas, até que o conjunto desses requisitos seja implementado na sua totalidade.

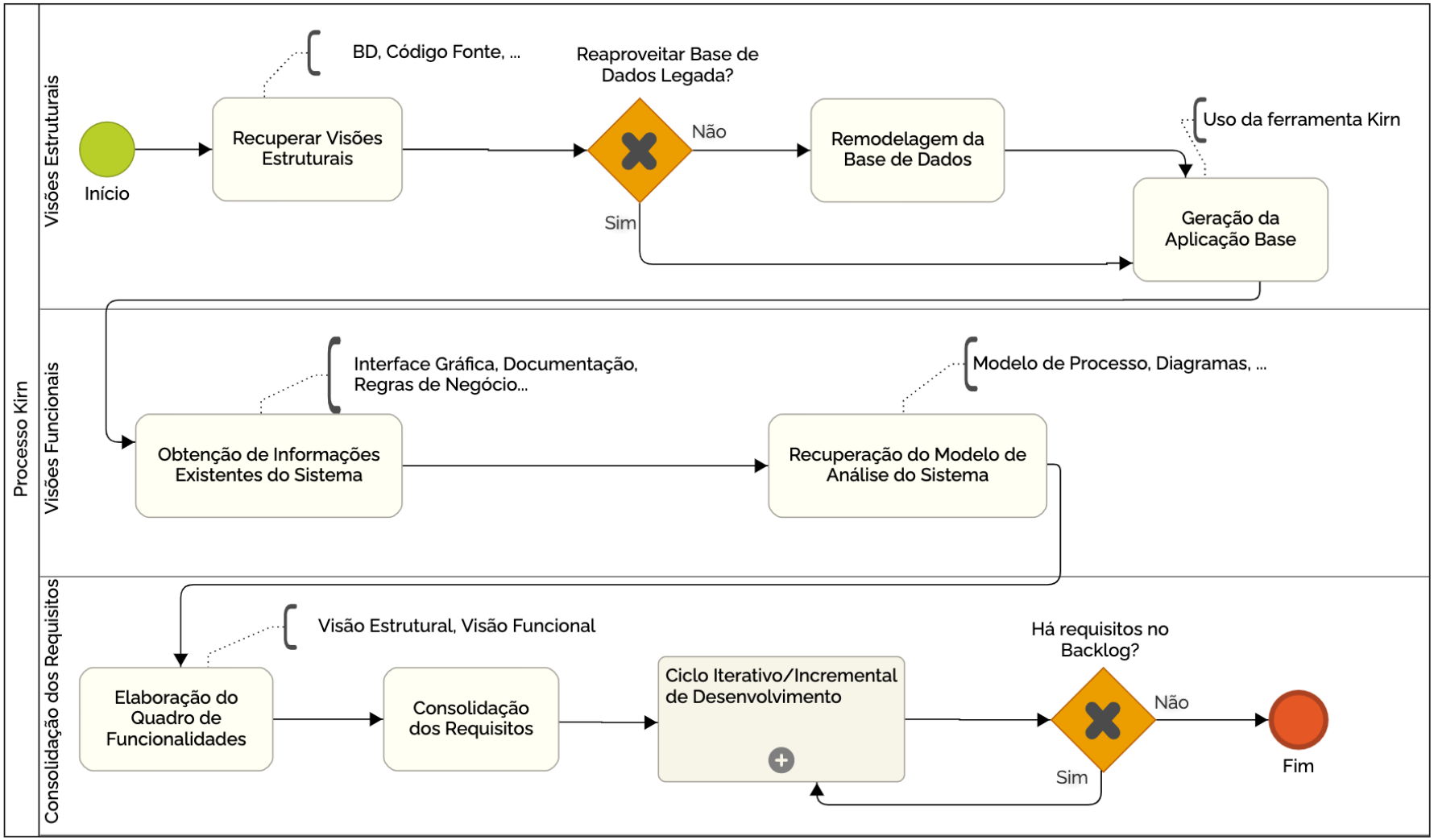


Figura 22 - Notação BPMN da abordagem Kirn

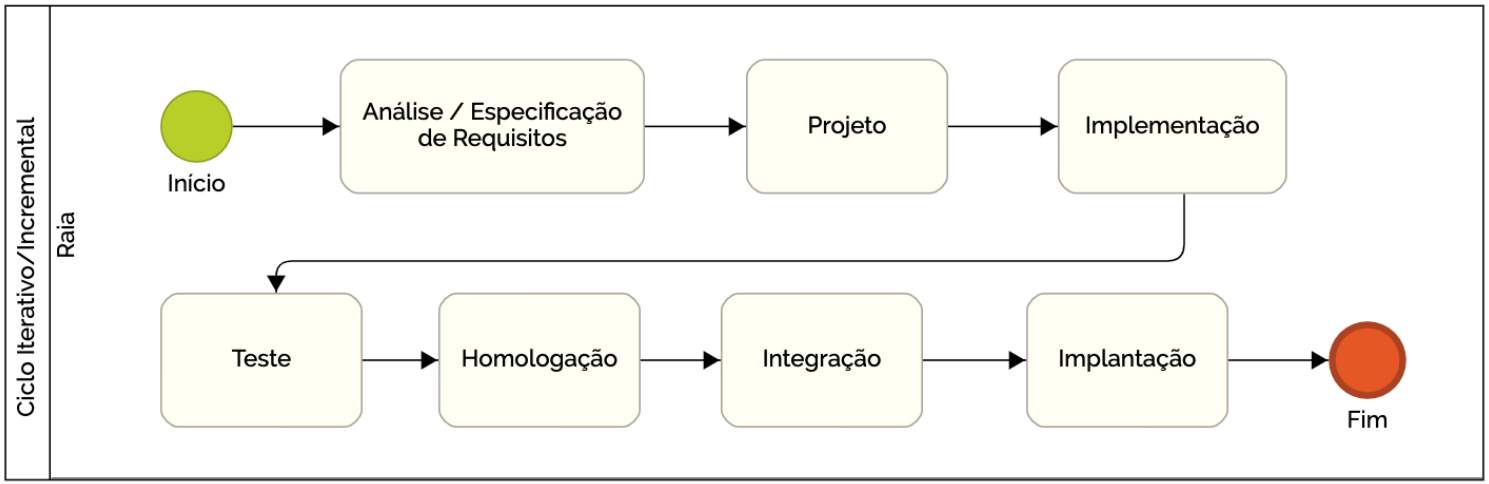


Figura 23 - Atividades do Ciclo Iterativo de Desenvolvimento

A Tabela 11 contém a explicação sobre os elementos das fases do processo. Na Tabela 12 são encontradas as informações relacionadas a cada fase do processo:

Tabela 11 - Elementos das fases do processo

Objetivo	**Detalhamento sobre os objetivos da fase do processo;**
Insumos	Requisitos necessários para a produção do artefato de software correspondente a fase;
Atores	Papéis desempenhados pelos profissionais envolvidos no projeto;
Saída	Resultado do trabalho desenvolvido na fase. Na maioria dos casos, servirá de insumo para outras fases do processo;

Tabela 12 - Fases do Processo

Fase	Objetivo	Atores	Insumos	Saídas
Recuperação da Visão Estrutural	Identificar quais os elementos estruturais do sistema legado serão usados na elaboração dos artefatos iniciais do processo de reengenharia. A ferramenta *Kirn Framework* é utilizada para ler o esquema de banco de dados afim de gerar o arquivo KML que será insumo para a geração da aplicação base.	- Analista de Requisitos; - Analista Desenvolvedor; - Gerente de Projetos;	- Esquema de Banco de Dados; - Código fonte original.	- Arquivo KML.
Remodelagem da Base de Dados	Caso seja necessário, seja por adequações à novos requisitos, seja por se tratar de uma base de dados legada, essa atividade será executada.	- Analista de Requisitos; - Analista Desenvolvedor; - Gerente de Projetos;	Esquema de Banco de Dados;	Base de dados remodelada.
Geração da Aplicação Base	Gerar a aplicação que será a base da modernização do sistema legado. É utilizada a ferramenta *Kirn Framework* para executar essa ação.	Analista Desenvolvedor.	- Base de dados; - Arquivo KML.	Aplicação base gerada.

Fase	Objetivo	Atores	Insumos	Saídas
Elaboração do Quadro de Funcionalidades	Criar uma lista de funcionalidades a serem implementadas no novo sistema.	Analista de Requisitos.	- Esquema de Banco de Dados; - Código Fonte; - Interface Gráfica; - Modelos e Diagramas.	Quadro de Funcionalidades.
Consolidação dos Requisitos	Unificar a documentação coletada ou criada para a customização da aplicação base.	Analista de Requisitos.	- Documento de Visão Estrutural; - Quadro de Funcionalidades.	Requisitos de Software Consolidados.
Planejamento da Iteração	Definir o conjunto de requisitos que irão compor o *backlog* da iteração em questão.	- Analista de Requisitos; - Analista Desenvolvedor; - Gerente de Projetos;	Requisitos de Software Consolidados.	*Backlog* de requisitos da iteração.
Ciclo Iterativo / Incremental de Desenvolvimento	Iniciar o desenvolvimento do software, através de ciclos de desenvolvimento, que será reiniciado tantas vezes forem necessárias para implementar todos os requisitos de software especificados.	- Analista de Requisitos; - Analista Desenvolvedor; - Gerente de Projetos;	*Backlog* de requisitos da iteração.	Entrega da iteração implementada.

3.3. PAPEIS

Os papeis envolvidos no processo proposto são relatados nessa seção.

- **Gerente de Projetos:** Responsável por acompanhar continuamente as atividades do processo de desenvolvimento de software, trabalhando como facilitador na execução dessas atividades, alocando e modificando os recursos utilizados no projeto, com a finalidade de concluir o projeto de software com a qualidade esperada.
- **Analista de Requisitos:** Designado para analisar os requisitos encontrados, relacionados ao sistema legado e especificá-los se necessário. Também é responsável pelo processo de consolidação dos requisitos identificados, oriundos de diversas fontes, para viabilizar o processo de reengenharia;
- **Analista Desenvolvedor:** Responsável por codificar o software, baseado nos requisitos, gerando o produto de software esperado.

Vale ressaltar que um profissional envolvido no projeto pode, eventualmente, desempenhar mais de um papel, dependendo da fase em que o projeto esteja. Um determinado papel pode conter mais de um participante desenvolvendo atividade similar.

3.1. INSUMOS E ARTEFATOS DE SOFTWARE PRODUZIDOS

Os insumos são as informações, representadas através de requisitos de software, que serão utilizados para o desenvolvimento dos artefatos de software nas fases do processo. Estão previstos nesse modelo de processo os seguintes insumos:

3.1.1. SGBD

O SGBD (Sistema de Gerenciamento de Banco de Dados) é um dos principais requisitos no processo de reengenharia do software legado porque se trata de um modelo estruturado, que reflete o resultado do trabalho de análise que resultou nas primeiras versões do sistema a ser evoluído. Através de uma rotina de leitura feita pela ferramenta *Kirn Framework*, é gerado um arquivo KML, detalhado na seção 4.2, do capítulo 4.

3.1.2. KML

O arquivo KML guarda informações relevantes como o esquema de banco de dados do software a ser evoluído. Ele guarda as informações necessárias para a geração da

aplicação base que sofrerá as customizações esperadas para o desenvolvimento do projeto. Após a geração, o *Kirn Framework* retroalimenta esse arquivo com novas informações, tais como o número de linhas geradas, tamanho do projeto gerado em PF e outras.

3.1.3. Quadro de Funcionalidades

Reúne as funções/procedimentos (linguagens estruturadas) ou métodos (linguagens orientadas a objetos) extraídos do código fonte com o objetivo de identificar quais foram gerados pela *Kirn Framework* e quais necessitam de implementação manual no software novo. O quadro deve contar com as colunas apresentadas no exemplo da Tabela 13.

Tabela 13 – Exemplo de Quadro de Funcionalidade

Classe / Entidade	Função / Método	Objetivo	Parâmetros	Saída	Gerado
Cliente	getCliente()	Recuperar dados de uma instância de Cliente	id: Integer	oCliente: Cliente	Sim
Cliente	getClientePorEmail()	Recuperar dados de uma instância de Cliente através do atributo email	Email: String	oCliente: Cliente	Não

3.1.4. Requisitos de Software Consolidados

Ao identificar os elementos da visão estrutural (SGBD, código fonte etc.) e a visão funcional (interface gráfica, modelos, diagramas etc.) do software legado em questão,

essas informações serão usadas na especificação dos requisitos a serem consolidados para que a fase de customização do novo software seja iniciada.

3.1.5. Aplicação base gerada

Aplicação gerada pela *Kirn Framework*, a partir de requisitos inicialmente levantados do software legado, que em seguida será submetida a customização para atender os demais requisitos não gerados automaticamente. Mais detalhes sobre a geração desse artefato no Capítulo 4.

3.1.6. Especificação de Requisitos

Documento de especificação de requisitos adotado. Pode ser conferido no Apêndice F do trabalho.

3.1.7. *Backlog* de requisitos da iteração

Conjunto de requisitos elencados para compor cada iteração da reengenharia do software escolhido.

3.1.8. *Release* incremental de software

Software desenvolvido após cada ciclo iterativo/incremental de modernização do software legado. Cada *release* deve passar pelo processo de validação junto ao cliente que demandou a tarefa.

4. EXPERIMENTO

Para submeter a abordagem *Kirn* a avaliações quanto a sua relevância no contexto de reengenharia de sistemas legados foi desenvolvido um experimento de engenharia de software a fim de cientificar a avaliação, bem como tornar esse processo repetível em outros cenários institucionais.

A organização do experimento foi elaborada a partir do modelo apresentado em Wolin et al (2000). As fases do experimento são descritas a seguir e detalhadas nas suas respectivas seções:

- 5.1 DEFINIÇÃO: Segundo Travassos (2002), é a fase onde o experimento é expresso em termos dos problemas e objetivos;
- 5.2 PLANEJAMENTO: Logo após a fase de definição, A fase de planejamento vem em seguida onde o projeto do experimento é determinado, a instrumentação é considerada e os aspectos da validade do experimento são avaliados. A execução do experimento segue o planejamento. Nesse momento os dados experimentais são coletados.
- 5.3 ANÁLISE E INTERPRETAÇÃO: Nesse momento os dados experimentais são coletados para serem analisados e avaliados na fase de análise e interpretação;
- 5.4 APRESENTAÇÃO E EMPACOTAMENTO: Os resultados são apresentados e empacotados durante a fase da apresentação e empacotamento

4.1. DEFINIÇÃO

A definição do experimento é dividida em quatro etapas, tendo o detalhamento do que representa cada etapa descritos a seguir. As etapas são: objetivo global, objetivos da medição, objetivos de estudo, questões para alcançar os objetivos e as métricas associadas às questões.

4.1.1. Objetivo Global

Definir se a abordagem *Kirn* é capaz de auxiliar no processo de reengenharia de software legado, de uma forma mais eficiente que a abordagem confrontada nesse trabalho. A eficiência será medida pela análise dos indicadores de produção do software estabelecidos para esse experimento.

4.1.2. Objetivos da Medição

Com base na observação do experimento, identificar:

1. Qual abordagem apresenta a melhor forma de executar a atividade de evolução tecnológica de sistemas legados;
2. Qual a abordagem melhora a atividade de reengenharia de sistemas legados, resultando em produtos de melhor qualidade;
3. Qual a abordagem atendeu melhor as expectativas da equipe para a atividade de reengenharia de sistemas legados.

4.1.3. Objetivos do Estudo

A Tabela 14 apresenta de forma estruturada os objetivos do estudo apresentado neste texto.

Tabela 14 – Objetivos do estudo

Analisar:	O desenvolvimento de protótipos, utilizando o processo de desenvolvimento de software usado na instituição e abordagem *Kirn*, proposta nesse trabalho.
Com o propósito de:	Comparar
Com respeito à:	Indicadores de produção de software coletados
Do ponto de vista:	Do gerente de projeto de software.
No contexto de:	Desenvolvimento de software da SUDAM

4.1.4. Questões e Métricas

A tabela 15 apresenta as questões de pesquisa relacionadas para este trabalho.

Tabela 15 – Questões e métricas

Questão	Métrica(s)
Q1. Os projetos desenvolvidos utilizam técnicas que visam melhorar a forma de se desenvolver software?	• Tempo de Desenvolvimento do Software • Produtividade Média da Equipe. • Tamanho do Projeto (PF) • Quantidade de Linhas de Código Geradas • Quantidade de Linhas de Código Programadas.
Q2. O projeto desenvolvido buscou utilizar técnicas que visam à garantia da qualidade de software do produto desenvolvido?	• Quantidade de erros produzidos
Q3. A abordagem proposta atendeu as expectativas da equipe?	• Nível de satisfação da equipe

As formas de coleta, cálculo e fase de coleta das métricas citadas acima, serão melher detalhadas na subseção 4.2.5 que fala sobre seleção de variáveis.

4.2. PLANEJAMENTO

De acordo com a proposta de Wohlin (2000), esta subseção foi dividida em: definição de hipóteses, descrição da instrumentação, seleção do contexto, seleção dos participantes, seleção das variáveis e validade.

4.2.1. Definição das Hipóteses

Hipótese nula (H0): A abordagem proposta não obteve resultados relevantes para se tornar uma opção de substituição da abordagem institucional, para o desenvolvimento de atividades de reengenharia de sistemas legados.

Hipótese alternativa (H1): A abordagem proposta obteve resultados significativos para se tornar uma opção de substituição da abordagem institucional, para o desenvolvimento de atividades de reengenharia de sistemas legados, porém a equipe avaliou a adoção da *Kirn* abaixo do esperado.

Hipótese alternativa (H2): A abordagem proposta obteve resultados significativos para se tornar uma opção de substituição da abordagem institucional, para o desenvolvimento de atividades de reengenharia de sistemas legados e a equipe avaliou positivamente a adoção da abordagem *Kirn*.

4.2.2. Descrição da Instrumentação

O experimento conta uma lista de participantes, quem pode ser conferida na subseção ‘Seleção dos Participantes’ desse capítulo. Cada participante utilizou computadores fornecidos pela SUDAM com as seguintes configurações:

- Modelo: HP EliteDesk 800 G1 SFF
- Processador: Intel(R) Core (TM) i7-4770 CPU @ 3.40GHz Octacore
- Ano de fabricação: 2014
- Memória RAM: 16 GB
- Memória HD: 1.0 TB
- Sistema Operacional: Windows 7 Professional

Cinco (5) computadores foram usados no experimento. Em cada um deles foram instaladas as ferramentas que podem ser conferidas na Tabela 16. Com relação a *Kirn Framework*, ela foi instalada nos mesmos computadores, porém, somente após o desenvolvimento do protótipo com a abordagem institucional e após a capacitação promovida para o desenvolvimento do protótipo na abordagem *Kirn*.

Tabela 16 – Lista de ferramentas utilizadas no projeto

Nome	Categoria	Versão	Função	Abordagem
Apache	Servidor de Aplicação	2.4	Portar a aplicação para o ambiente web.	Ambas
Redmine	Gestão de Projetos	4.1.0	Gerenciar o desenvolvimento dos protótipos propostos	Ambas
Kirn Framework	Ferramenta CASE	1.0	Responsável pela geração de uma aplicação base a ser customizada	Kirn
MySQL	SGBD	5.6	Armazenar dados	Ambas
PHP	Interpretador da linguagem	7.0	Interpretar o código desenvolvido.	Ambas
Tortoise SVN	Controle de versão	1.8.8	Controlar o versionamento da aplicação desenvolvida	Ambas
VS Code	IDE	1.34	Codificar a aplicação	Ambas

4.2.3. Seleção do Contexto

A seleção do contexto é o de evolução tecnológica do “SICAS Legado”, mais especificamente o módulo “Despesas Médicas”, que controla as despesas dos servidores com o plano de saúde institucional. Esse sistema foi o escolhido para o experimento porque se trata de uma aplicação amplamente utilizada pela área meio da instituição, além do fato de que essa aplicação, como pode ser verificado na Tabela 1, foi desenvolvido em linguagem GOL, incompatível com as demais tecnologias utilizadas atualmente na instituição.

Em Brasil (2019), não há previsão orçamentária para o desenvolvimento de sistemas dessa natureza. Com nesse contexto, foi planejado um experimento, utilizando a abordagem *Kirn* como alternativa para a evolução dos sistemas não contemplados pelo contrato de fábrica de software vigente, no qual será utilizada mão-de-obra da instituição, entre servidores e estagiários, para a evolução do software mencionado.

Os protótipos que serão desenvolvidos foram baseados no mesmo conjunto de requisitos, onde um será desenvolvido usando o processo da instituição e o outro na abordagem *Kirn*. O treinamento da equipe de desenvolvimento deve ser efetuado para a execução do experimento de forma adequada.

Sobre a caracterização do contexto através de quatro dimensões, proposta por Travassos (2002), a Tabela 17 apresenta uma adaptação que se adequa ao experimento desse trabalho.

Tabela 17 – Dimensões do contexto

Dimensão	Características	Decisão
In-vitro vs. *In-vivo*	O primeiro refere-se à experimentação no laboratório sob as condições controladas. O segundo considera o estudo de um projeto real.	O experimento realizado é de carácter ***in-vivo***, pois se trata de uma atividade de desenvolvimento de software rotineira na instituição.
Estagiários vs. Servidores	Define a equipe que vai executar o experimento.	Os participantes são **Estagiários** e **Servidores** da SUDAM, onde ambos representam a força de trabalho disponível.
Problema de sala de aula vs. Problema real	Mostra o tamanho do problema que está sendo estudado.	O estudo é um **problema real**, pois o projeto desenvolvido está relacionado a um contexto real de evolução tecnológica de um sistema legado e de necessidade urgente para a instituição.
Específico vs. Geral	Mostra se os resultados do experimento são válidos para um contexto particular ou para o domínio global da Engenharia de Software.	O contexto possui caráter **específico**, pois o experimento se propõe a resolver uma necessidade institucional, a priori, porém, visando submeter à abordagem proposta para avaliação em outros cenários.

4.2.4. Seleção dos Participantes

Ao todo, cinco foram selecionados como participantes do experimento. Dados como cargo, função no experimento, formação e habilidades são apresentados na Tabela 18. Todos os participantes além de terem vínculo com a instituição que acolherá o experimento, também estão diretamente ligados à execução.

Tabela 18 – Lista de Participantes

	Cargo	Função	Formação	Habilidades
P1	Estagiário	Desenvolvedor de software	Estudante do curso de Engenharia da Computação – 6º semestre	PHP, Javascript, HTML, CSS, MySQL, PowerBI
P2	Estagiário	Desenvolvedor de software	Estudante do curso de Análise de Sistemas – 7º semestre	PHP, Javascript, HTML, CSS, MySQL, PowerBI
P3	Programador	Desenvolvedor de software e Analista de requisitos	Técnico em Processamento de Dados	Gol Soft, SGBD Btrieve
P4	Analista	Desenvolvedor de software e Analista de requisitos	Bacharel em Sistemas de Informação	PHP, Javascript, HTML, CSS, MySQL
P5	Gerente de Projeto	Observador e tabulador dos dados coletados	Bacharel em Sistemas de Informação	PHP, Javascript, HTML, CSS, MySQL, Scrum, XP, PMBOK

4.2.5. Seleção das Variáveis

A Tabela 19 apresenta a lista de variáveis independentes do experimento e, a tabela 20, a lista de variáveis dependentes do experimento.

Tabela 19 – Lista de Variáveis Independentes do Experimento

Variável	Origem
Documentos da visão funcional do sistema legado	Requisitos, Manuais, Interface Gráfica.
Documentos da visão estrutural do sistema legado	SGBD e Código Fonte.

Tabela 20 – Lista de Variáveis Dependentes do Experimento

Alias	**Variável**	**Unidade**	**Forma de Coleta**	**Cálculo**	**Periodicidade da Coleta**	**Fase**
T_D	Tempo de Desenvolvimento do Software	Dias	Automática	-	Diária	Implementação
$T_{Projeto}$	Tamanho do Projeto	PF	Manual	-	-	Consolidação dos Requisitos
Q_E	Quantidade de Erros Produzidos	Unidade	Manual	-	Diária	Todas
QL_{Ger}	Quantidade de Linhas de Código Geradas	Unidade	Automática	Fornecido pela Kirn Framework	Na geração da aplicação base	Implementação
P_{EQ}	Produtividade Média da Equipe	PF/Dia	Manual	$P_{EQ} = QT_{total} / T_D$	Iteração	Todas
QL_{Prog}	Quantidade de Linhas de Código Programadas	Unidade	Automática	-	Diária	Implementação
QL_{Tot}	Quantidade Total de Linhas de Código	Unidade	Manual	$QL_{Tot} = QL_{Ger} + QL_{Prog}$	Final do projeto	-

4.2.6. Validade

- **Validade interna**: A equipe envolvida no experimento deve analisar se a abordagem proposta é capaz de apresentar resultados esperados no tocante de uso eficiente dos recursos institucionais para a evolução tecnológica dos sistemas legados. A cada participante selecionado serão distribuídas tarefas, de acordo com as suas competências, visando uma melhor produtividade e motivação da equipe.
- **Validade de conclusão**: Ao final do experimento, os dados serão coletados, calculados, analisados e empacotados para apresentação. Em seguida, será efetuada a verificação das hipóteses, utilizando os dados como instrumento de apoio à conclusão.
- **Validade de construção**: Para que a equipe envolvida no experimento apresentasse a maior isenção possível com relação as abordagens avaliadas, primeiramente devem desenvolver o protótipo utilizando a abordagem institucional. Em seguida, passarão por uma capacitação na qual irão conhecer a abordagem *Kirn* e como ela deve ser conduzida, para que em seguida, o protótipo seja desenvolvido nessa abordagem.
- **Validade externa**: A abordagem proposta deve ser, em momento oportuno, testada em outros contextos similares aos encontrados na instituição SUDAM. Com base no planejamento elaborado, o experimento deve ser executado em outros cenários, em condições análogas, para a comprovação das hipóteses levantadas.

4.3. EXECUÇÃO

O experimento foi executado no período de 29/07/2019 a 30/08/2019, com a equipe definida no planejamento, desenvolvendo protótipos a serem avaliados através dos indicadores, sendo que um deles foi desenvolvido usando a abordagem institucional e o outro com a abordagem *Kirn*. O escopo de requisitos usado foi o mesmo para ambas as abordagens. A documentação técnica dos protótipos desenvolvidos, assim como o código fonte da ferramenta *Kirn Framework*, estão disponíveis nos repositórios

onde estão presentes os protótipos desenvolvidos com a abordagem institucional[2] e com a abordagem *Kirn*[3].

Foram listadas as funcionalidades presentes no sistema legado através de entrevista com os servidores que utilizam o sistema e da análise das telas do sistema legado. Essas funcionalidades foram adicionadas em documentos de requisitos, segmentados por em módulos.

O participante P3 foi o responsável por extrair a estrutura de dados do sistema legado originalmente implementada no banco de dados *BTrieve*, armazenando em arquivos TXT, que podem ser vistos no Apêndice G do trabalho. Para atender as necessidades de negócio, e diante das limitações de uso do SGBD legado, o banco de dados foi remodelado. O banco de dados escolhido pela instituição para substituir a base de dados legada e compor o processo de modernização do sistema legado foi o MySQL versão 8.0, por além de ser de um banco de dados adotado em diversos sistemas do órgão, logo há uma experiência institucional em utilizá-lo, a *Kirn Framework* tem suporte para a geração de aplicações tendo como base de dados de referência o MySQL. A nova modelagem é apresentada na Figura 24. No período de 29/07/2019 à 02/08/2019, foi efetuado um levantamento de requisitos assim como a sua especificação por parte da equipe. A documentação e o código fonte das implementações podem ser conferidos no Apêndice D desse trabalho.

---

[2] https://github.com/luizleao/despesasMedicas

[3] https://github.com/luizleao/despesasMedicasKirn

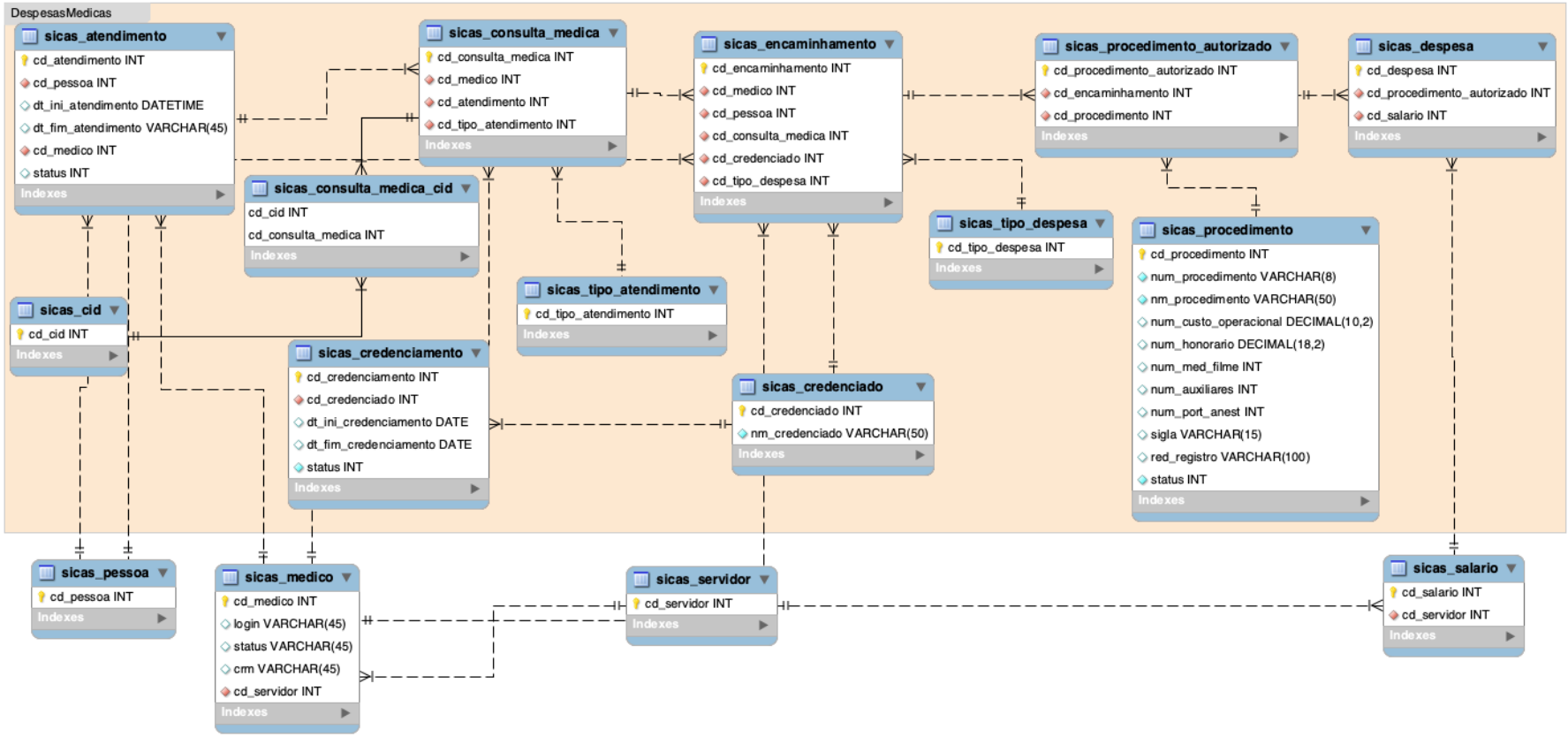


Figura 24 – Modelo relacional para o módulo de Despesas Médicas

Após a modelagem do banco de dados, rotinas de carga foram desenvolvidas e executadas para alimentar o esquema com os dados provenientes do sistema legado. A partir de então, foi iniciada a elaboração dos protótipos

4.3.1. Protótipo com Abordagem SUDAM

O desenvolvimento do protótipo com a abordagem SUDAM ocorreu no período de 06/08/2019 à 21/08/2019, com a equipe definida no planejamento. A equipe utilizou as ferramentas que estavam habituados a utilizar para desenvolver e corrigir os demais sistemas institucionais. Diariamente a equipe reunia com o gerente de projeto (e observador), com o propósito de informar para qual o status do desenvolvimento e eventuais problemas a serem corrigidos. Os dados sobre o andamento do projeto foram registrados para fins de acompanhamento da produção do protótipo. Como a equipe estava adaptada ao processo institucional, a produtividade foi constante, sem muitas oscilações. Ao final da implementação de cada caso de uso, um roteiro de testes era executado. Alguns erros foram encontrados e corrigidos em tempo hábil. Ao final do período mencionado, o protótipo foi apresentado e atualizado no servidor de controle de versão.

4.3.1. Protótipo com Abordagem Kirn

Após o desenvolvimento do protótipo usando a abordagem SUDAM, foi promovida uma capacitação sobre a proposta de abordagem *Kirn* no período de 22/08/2019 e 26/08/2019, com a mesma equipe que desenvolveu o primeiro protótipo. A equipe, em um primeiro momento, apresentou uma certa estranheza ao adotar uma nova abordagem de desenvolvimento, mas se mostraram dispostos a auxiliar no que fosse necessário para melhorar os processos de modernização dos sistemas legados da instituição. No primeiro dia, a equipe utilizou a ferramenta *Kirn Framework* para gerar os artefatos de software iniciais do projeto. A Figura 25 mostra a lista dos esquemas mapeados dos servidores de banco de dados que a ferramenta acessou. O menu apresentado representa as opções de *frameworks front-end* (interface gráfica da aplicação) que a ferramenta têm para a geração da aplicação baseada no esquema selecionado.

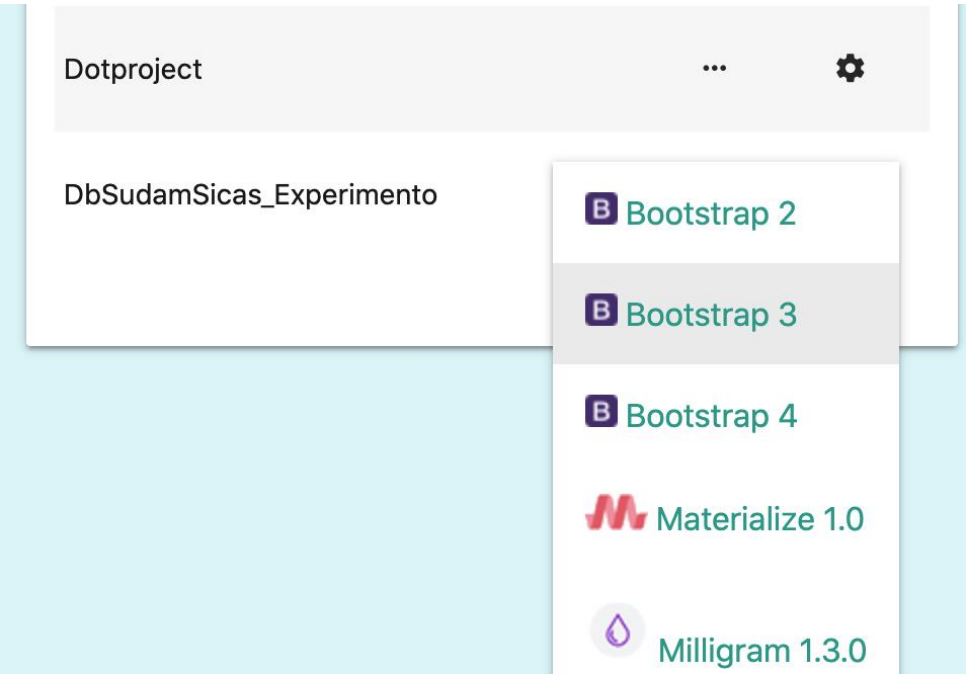


Figura 25 – Tela de geração dos artefatos por Framework Front-End escolhido

Após a escolha da opção do menu, é exibido um relatório de geração da aplicação base, que pode ser visto na Figura 26. Ela apresenta um relatório sobre o sucesso ou falha na geração dos componentes do código fonte, bem como métricas quantitativas relacionadas.

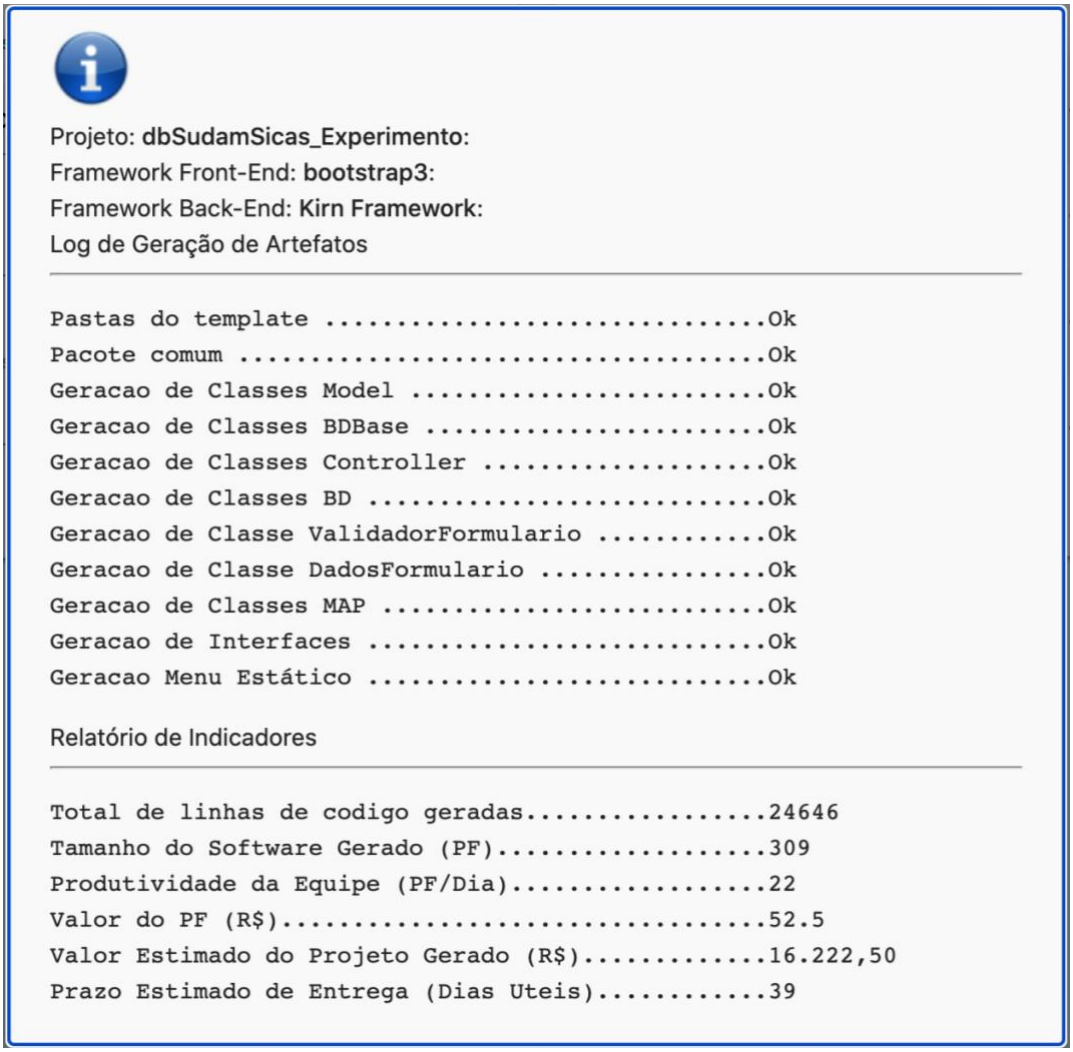


Figura 26 – Relatório de geração dos artefatos

Atualmente a ferramenta só possibilita a geração do *Back-End* da aplicação no formato do *Kirn Framework*. Para o *Front-End*, foi escolhido o *framework Bootstrap* 3. Algumas telas da aplicação gerada estão reproduzidas no Apêndice E.

Após a geração desses artefatos, o processo de customização foi iniciado. Como foram geradas automaticamente as implementações de diversos requisitos do sistema legado, a customização foi efetuada nos requisitos restantes. Foi observado que, excetuando o primeiro dia de desenvolvimento, a produtividade média de desenvolvimento na nova abordagem foi inferior à produtividade média computada da abordagem institucional, muito por conta da ambientação da equipe nesse novo processo. Apesar das dificuldades, a equipe concluiu o protótipo e apresentou a implementação dos casos de uso definidos no planejamento.

4.4. APRESENTAÇÃO E EMPACOTAMENTO

O empacotamento padronizado dos dados experimentais pode servir como base para a criação das bibliotecas de experimentação (TRAVASSOS, 2002). Após a execução do experimento, os dados foram coletados para que as análises sobre o desempenho das abordagens pudessem ser verificadas. A seguir, são apresentadas as visualizações de informações coletadas durante o experimento.

4.4.1. Análise Quantitativa

Para comparar quantitativamente as abordagens utilizadas no experimento, as variáveis independentes (indicadores) foram coletadas após a execução do experimento.

4.4.1.1. Tempo de Desenvolvimento do Software (T_D)

As atividades relacionadas ao desenvolvimento dos protótipos foram coletadas pelo autor no trabalho, que desempenhou o papel de gerente de projeto do experimento. Os dados estão representados em gráficos de *Burndown*, como podem ser visualizadas nas Figuras 27 a seguir.

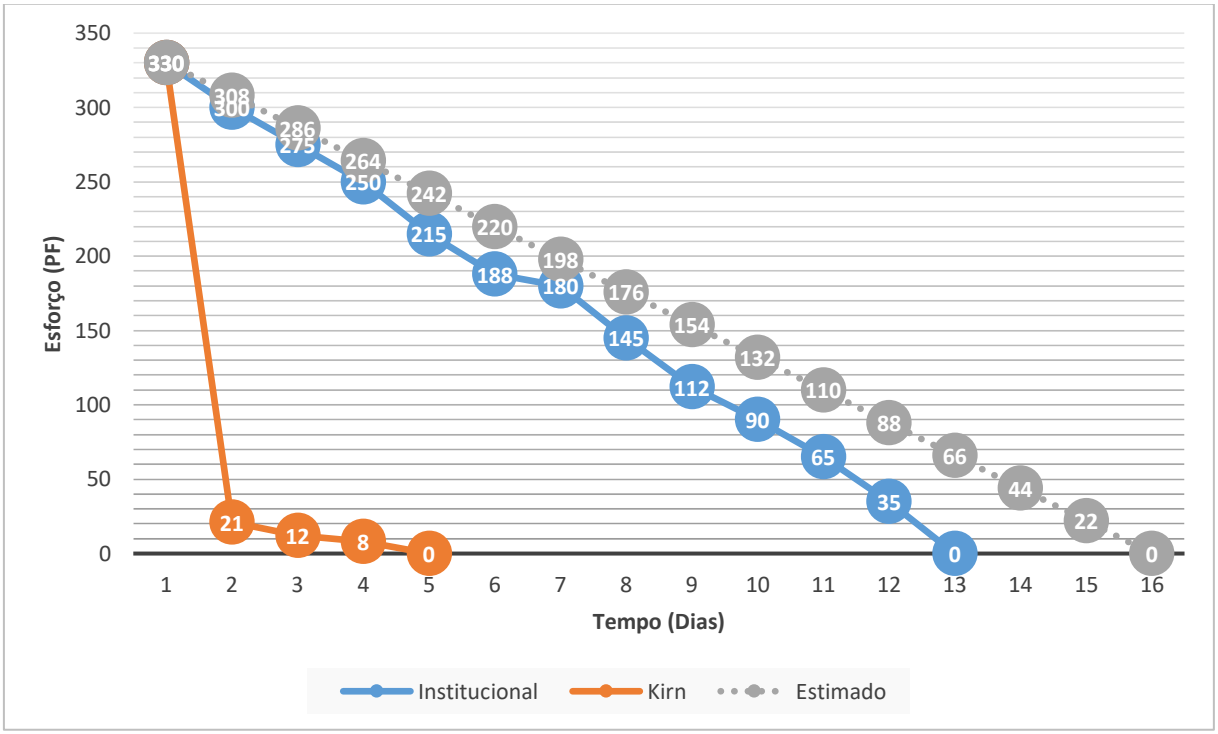


Figura 27 - Gráfico *Burndown* – Tempo de desenvolvimento

Ao comparar as duas abordagens no gráfico de acompanhamento do desenvolvimento dos protótipos, é observado que o protótipo desenvolvido através da abordagem institucional foi concluído após 14 (quatorze) dias de implementação do projeto. Já o protótipo desenvolvido pela abordagem Kirn, foi concluído após 5 (cinco) dias de desenvolvimento do projeto.

4.4.1.2. Tamanho do Projeto ($T_{Projeto}$)

O tamanho do projeto, utilizando a métrica PF, após a consolidação dos requisitos, gera o dado quantitativo referente ao total de esforço do projeto. Tanto a abordagem institucional quanto a Kirn resultaram em 330 pontos de função (PF).

4.4.1.3. Quantidade de Erros Produzidos (Q_E)

A quantidade de erros produzidos é um indicador determinante para a avaliação da qualidade do software produzido. Os valores são apresentados na Figura 28.

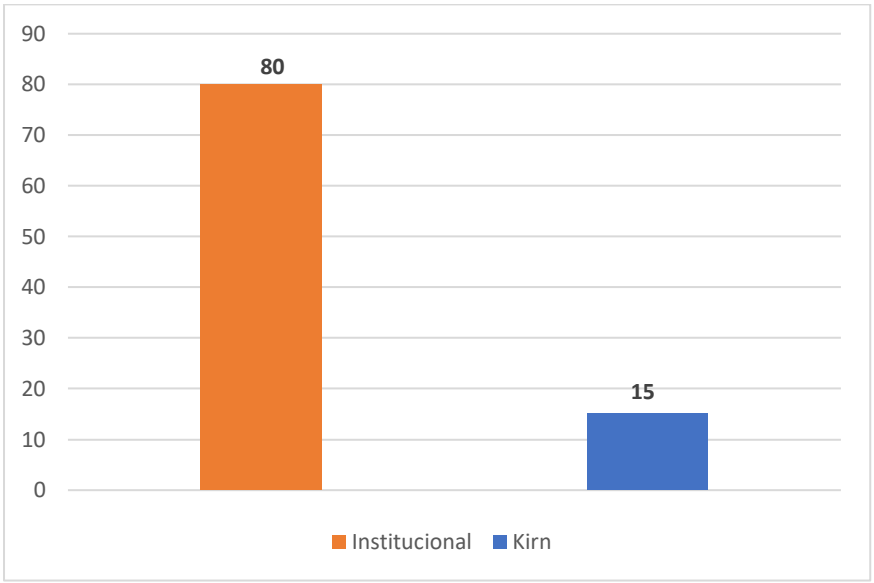


Figura 28 - Quantidade de Erros Produzidos (Q_E)

O gráfico apresenta os seguintes números: O desenvolvimento do protótipo, através da abordagem utilizada comumente na SUDAM, produziu 80 (oitenta) erros identificados nos primeiros testes após a entrega. Por outro lado, os desenvolvedores que utilizaram a abordagem Kirn, introduziram 15 erros de implementação, identificados também nos primeiros testes.

4.4.1.4. Produtividade Média da Equipe (P_{EQ})

A produtividade média da equipe está diretamente ligada a quantidade de requisitos de software, mensurados em PF (pontos de função) produzidos pela equipe por dia útil trabalhado. Esses números são apresentados na Figura 29.

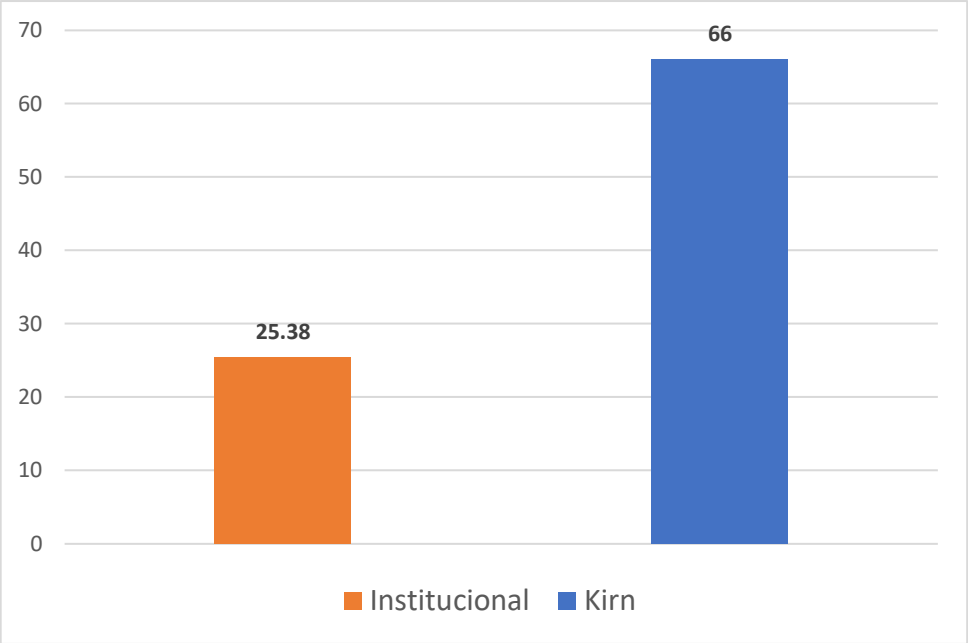


Figura 29 - Produtividade Média da Equipe (P_{Eq})

A equipe que utilizou as técnicas e ferramentas estabelecidas na abordagem institucional teve a produtividade na ordem de 25,38 PF/Dia. Ao utilizar o processo Kirn, a equipe obteve uma produtividade média de 66 PF/Dia.

4.4.1.5. Quantidade de Linhas de Código Programadas (QL_{Prog})

A quantidade de linhas programadas está relacionada diretamente com a customização da aplicação base, pois é uma métrica que é quantificada pelo trabalho manual dos desenvolvedores ligados ao projeto. A Figura 30 exibe os números quantificados pela equipe.

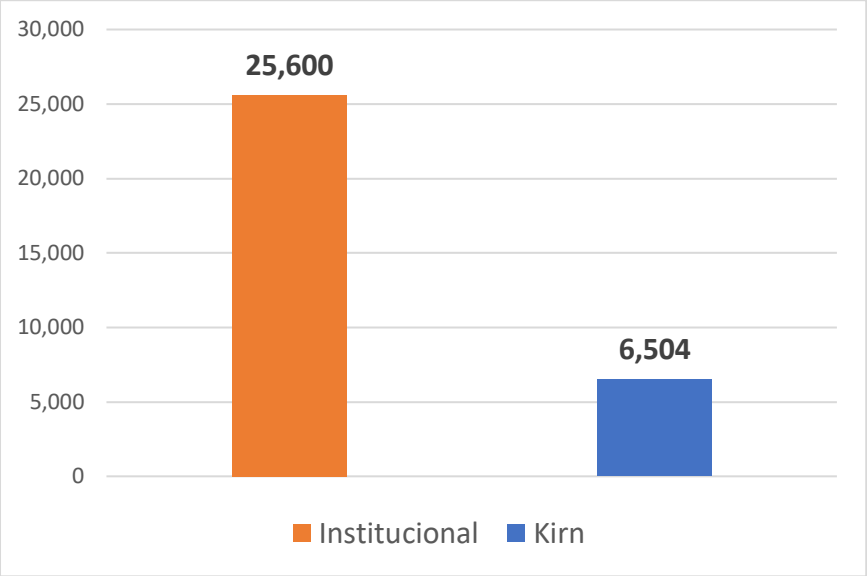


Figura 30 - Quantidade de Linhas de Código Programadas (QL_{Prog})

Através da abordagem da SUDAM, a equipe precisou programar 25.600 linhas de código para concluir o protótipo. Por sua vez, ao se utilizar a abordagem proposta pelo trabalho, os desenvolvedores programaram 6.504 linhas de código da aplicação base gerada.

4.4.1.6. Quantidade de Linhas de Código Geradas (QL_{Ger})

A quantidade de linhas de código geradas é o resultado do uso de ferramentas *CASE*, que auxiliam a equipe de desenvolvimento a automatizar parte do processo de criação de artefatos de software, que nesse caso, está representado pelo código fonte gerado. Esses números podem ser vistos na Figura 31.

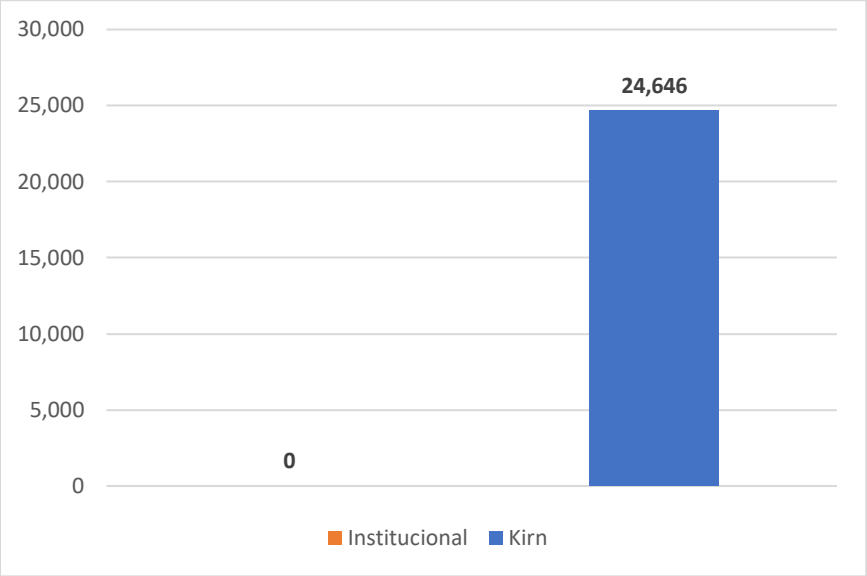


Figura 31 - Quantidade de Linhas de Código Geradas (QL_{GER})

A *Kirn Framework*, que compõe as ferramentas utilizadas pela abordagem do trabalho, produziu automaticamente no momento da geração da aplicação base, 24.646 linhas de código. A abordagem institucional, por não utilizar ferramentas de geração automatizada de código, não gerou qualquer linha de código. Esse gráfico foi adicionado ao trabalho com o intuito que reforçar que, uma abordagem que faz uso de uma ferramenta de geração de artefatos de software apresenta um diferencial evidente em uma abordagem que não utiliza tal recurso.

4.4.1.7. Quantidade Total de Linhas de Código (QL_{Tot})

Através da fórmula $QL_{Tot} = QL_{Ger} + QL_{Prog}$, é deduzido que a quantidade total de linhas código produzido com a abordagem *Kirn* é de 24646 linhas geradas pela ferramenta e 6.504 programadas, totalizando 31.150 linhas de código, ao passo que com a abordagem adotada pela SUDAM, foram produzidas 25.600 linhas, sendo zero (0) linhas programadas.

4.4.1.8. Resumo da coleta das variáveis

Após o fim do experimento, foi gerada uma tabela comparativa dos indicadores coletados. Os dados coletados, analisados e calculados podem ser conferidos em resumo na Tabela 21.

Tabela 21 – Resumo dos Indicadores Coletados das Variáveis Dependentes do Experimento

Variável	Abordagem Kirn	Abordagem SUDAM
T_D	5 dias	14 dias
$T_{Projeto}$	330	330
Q_E	15	80
P_{EQ}	20	12
QL_{Ger}	24.646	0
QL_{Prog}	6.504	25.600
QL_{Tot}	31.150	25.600

4.4.2. Análise Qualitativa

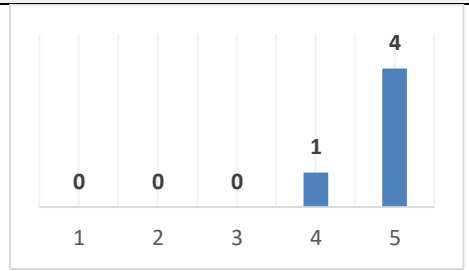
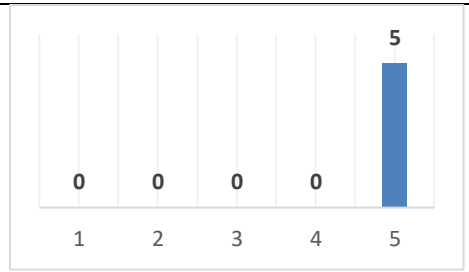
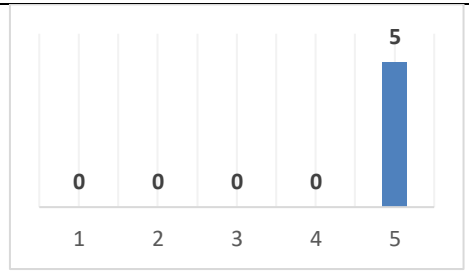
Foi aplicado um questionário, baseado na norma para avaliação da qualidade do software produzido (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2003) através da abordagem Kirn, reproduzido no Apêndice C, além de avaliar a satisfação da equipe em utilizar essa abordagem. São 9 (nove) perguntas que tem como respostas os números de 1 a 5, que, baseada na escala de *Likert*, têm relação com as respostas apresentadas na Tabela 22.

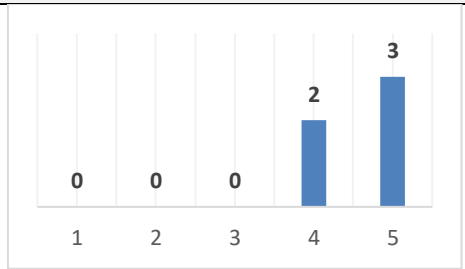
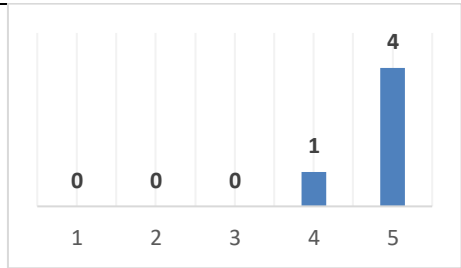
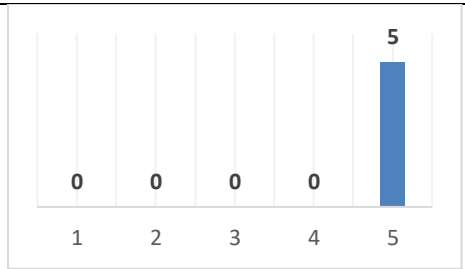
Tabela 22 – Respostas Objetivas da Pesquisa Qualitativa

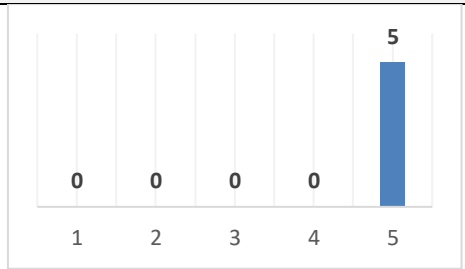
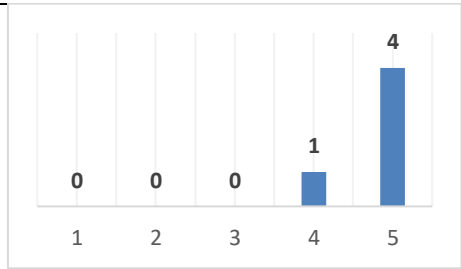
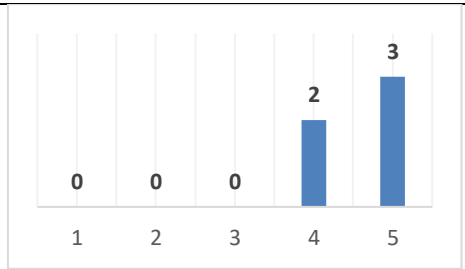
1	Não atende
2	Atende insuficientemente
3	Indiferente
4	Atende parcialmente
5	Atende totalmente

Foram elaboradas também 2 (duas) perguntas subjetivas nas quais os participantes relatam os pontos positivos e negativos que identificaram na da abordagem. Os dados coletados da pesquisa podem ser conferidos na Tabela 23 a seguir:

Tabela 23 - Respostas da pesquisa da análise qualitativa

Pergunta	Respostas
1. Como você avalia o critério Funcionalidade (capacidade do software de prover funcionalidades que satisfaçam o usuário) da aplicação gerada?	
2. Como você avalia o critério Confiabilidade (capacidade do software em manter um nível de desempenho, em condições esperadas) da aplicação gerada?	
3. Como você avalia o critério Usabilidade (capacidade do software de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições especificadas) da aplicação gerada?	

Pergunta	Respostas
4. Como você avalia o critério Eficiência (o tempo de execução e os recursos envolvidos são compatíveis com o nível de desempenho) da aplicação gerada?	
5. Como você avalia o critério Manutenibilidade (capacidade do software ser modificado, incluindo tanto as melhorias ou extensões de funcionalidade quanto as correções de defeitos, falhas ou erros) da aplicação gerada?	
6. Como você avalia o critério Portabilidade (capacidade do sistema ser transferido de um ambiente para outro) da aplicação gerada?	

Pergunta	Respostas
7. A ferramenta contribuiu para o seu aprendizado em desenvolvimento de software?	
8. Como você avalia o nível de produtividade da Kirn Framework?	
9. Como você avalia a arquitetura de software proposta pela Kirn Framework?	

Pergunta	**Respostas**
10. Pontos Positivos	▪ Produtividade na geração de artefatos; ▪ Fácil manuseio da ferramenta; ▪ Abordagem organizada.
11. Pontos Negativos	▪ Pouco suporte da comunidade; ▪ Problema de customização do projeto quando é necessário gerar a aplicação base novamente; ▪ Acesso a repositórios públicos de bibliotecas apenas com ajuste na arquitetura. Não vem preparado nativamente para isso.

4.4.3. Verificação das Hipóteses

A Hipótese nula (H0) não se confirmou à medida que os dados coletados sobre o experimento demonstraram favoráveis à adoção da abordagem *Kirn*. A Hipótese alternativa (H1), que ratifica que abordagem proposta obteve resultados significativos, porém, ao avaliar qualitativamente a satisfação da equipe na adoção da abordagem, o resultado seria negativo, foi refutada. Logo, a hipótese que foi confirmada é a H2, visto que a equipe mostrou um grau de satisfação significativo ao desenvolver o projeto com a abordagem apresentada.

5. CONCLUSÃO

Diante da necessidade de submeter os sistemas legados das instituições a um processo de reengenharia, quando for constatada a sua depreciação em relação ao atendimento dos processos organizacionais, foi desenvolvida uma abordagem específica para essa finalidade, onde o seu grande diferencial é a rapidez na construção de aplicações, com uma baixa quantidade de erros incluídos no processo de customização, velocidade de desenvolvimento, entre outros indicadores que foram observados e relatados no capítulo anterior. Os indicadores do experimento apresentado neste texto ratificam a aplicabilidade da abordagem *Kirn*, representando uma alternativa viável para a evolução dos sistemas legados de instituições públicas que se assemelham com a instituição que foi objeto do estudo de caso.

A elaboração dessa abordagem foi um trabalho personalista do autor, baseado no contexto de aplicação do experimento e de suas experiências, logo, a inclusão de outras visões de análise em forma de modelos que possam enriquecer o processo de reengenharia foram postergadas para trabalhos futuros. As evoluções almejadas compreendem tanto o modelo de processo, quanto a ferramenta *Kirn Framework*.
Apesar dos resultados positivos no contexto do trabalho, é necessário que outros cenários, tecnológicos ou institucionais, sejam testados para reforçar a relevância da abordagem desenvolvida. Ao analisar os dados qualitativos do experimento, foi constatada a aceitação da equipe em utilizar a abordagem em novos projetos. Atualmente, a abordagem está sendo aplicada em projetos da área meio da instituição, e com isso, frequentes contribuições para a melhoria da ferramenta e do processo estão sendo reportadas.

A equipe que participou do experimento relatou um *trade-off* a ser resolvido na ferramenta. Quando a aplicação base é gerada, um processo de customização é iniciado. Porém, pode surgir a necessidade de se “re-gerar” a aplicação, e da maneira que a ferramenta foi construída, boa parte do trabalho customizado pode ser perdido, logo, essa é uma demanda a ser melhorada em trabalhos futuros. Observa-se que esta é uma oportunidade de melhoria que também ocorre no uso da JHipster, uma das principais ferramentas congêneres disponível atualmente.

Os sistemas legados guardam um extenso histórico de informações que ajudam na compreensão dos processos organizacionais ao qual eles atendem. Porém, nem sempre há a disposição da equipe de desenvolvimento de software ferramentas e técnicas que auxiliem na identificação de requisitos de uma maneira prática. Uma das motivações em evoluir esse trabalho reside na necessidade de tornar esse processo, que remota da identificação dos requisitos, customização da aplicação e a implantação do novo software, completando o ciclo de reengenharia, objetivo tão almejado pelas organizações ao se dedicarem a esse tipo de empreitada.

Sobre o trabalho foi elaborado um artigo de título "*Empirical assessment of the Kirn approach to support the reengineering of legacy software systems*", que foi submetido para "*XLVI Latin American Computing Conference*" (CLEI 2020), tendo como resultado da submissão a rejeição do artigo. Segundo os avaliadores, o trabalho está bem escrito, porém alguns elementos devem ser corrigidos, como a forma das citações, alguns erros gramaticais e mencionar o significado de uma sigla antes de utilizá-la no texto pela primeira vez. Foi questionado também o uso de duas imagens utilizadas no texto. Uma por faltar explicações sobre a finalidade dela no artigo (Figura 12) e outra pela ilegibilidade, que representa um trecho do arquivo KML usado na abordagem *Kirn*.

Pela ótica da engenharia de software experimental, foram criticadas algumas referências por serem antigas sendo a mais recente do ano de 2017. O planejamento do experimento foi concluído no fim do ano de 2018, logo as referências ainda eram atuais naquele momento. Comentam também que o trabalho apresenta testes de hipóteses, mas em nenhum momento descrevem qual teste foi usado e se foram testes paramétricos ou não paramétricos.

Um dos avaliadores analisou a proposta de abordagem de evolução tecnológica de sistemas legados, e julgou a proposta como interessante. Sugeriu que no resumo do artigo já fosse inseridos os resultados do experimento para dar uma visão geral sobre o artigo. Na seção II, que fala sobre trabalhos relacionados, sugeriu descrever quais as contribuições da abordagem Kirn e qual o diferencial para os demais trabalhos apresentados.

Como o trabalho aborda reengenharia de software, *Model-Driven Development* (MDD) e engenharia de software experimental está sendo avaliado junto ao orientador qual a melhor abordagem para a submissão de um novo artigo para outro veículo da área de Engenharia de Software que ratifique a relevância do estudo efetuado.

6. TRABALHOS FUTUROS

Em relação à abordagem *Kirn*, algumas ações podem resultar em melhorias nas atividades relacionadas:

- Análise detalhada nas atas das reuniões de retrospectiva com a equipe de desenvolvimento, para identificar pontos de melhoria no processo de reengenharia;
- Verificar os indicadores de produção dos protótipos e, através de uma análise quantitativa, relacioná-los a potenciais melhorias e oportunidades a serem com a abordagem experimentada.

Sobre a ferramenta *Kirn Framework*, está em curso uma pesquisa para aumentar o portfólio de funcionalidades oferecidas, tais como:

- Desenvolver um analisador (*parser*) automático do código fonte de sistemas legados, desenvolvido em linguagens com um grau de depreciação tecnológica reconhecido no mercado como *Cobol*, *Fortran* e *Clipper*, por exemplo, e que estão ligados a plataformas de software legados ainda em funcionamento, com o objetivo de criar um quadro de funcionalidades, destacando as operações que serão convertidas automaticamente pela Kirn Framework e as que serão customizadas pelos desenvolvedores do sistema gerado;
- Utilizar outros padrões de modelagem, como JDL e XMI para a geração automática de artefatos de software, buscando uma melhor comunicação entre os requisitos e a implementação do sistema em processo de reengenharia.
- Quantificar a aplicação base gerada em unidades de PF e oferecer a opção para o usuário da ferramenta de inserir parâmetros como valor do PF e produtividade média da equipe (PF/Dias) para que haja um cálculo de estimativa para custo e tempo de desenvolvimento do projeto.

Sobre o experimento, são vislumbradas as seguintes oportunidades:

- Submeter a abordagem à prova em cenários distintos, buscando identificar outros fatores de risco que ameassem a validade da abordagem, para que sejam desenvolvidas melhorias no modelo de processo como um todo, com a finalidade observar como a abordagem resolve os problemas, vislumbrando o

processo de maturidade ao corrigir seus pontos fracos e potencializar e aperfeiçoar seus pontos positivos.

- Aplicar experimentos da abordagem utilizando outras ferramentas CASE, para verificar como os indicadores se apresentam e investigar eventuais adequações no processo que vislumbre torná-lo maduro o suficiente para a utilização de outras ferramentas de apoio, mitigando eventuais riscos nessa nova possibilidade.

REFERÊNCIAS BIBLIOGRÁFICAS

ALBRECHT, A. J. ***Measuring Application Development Productivity***. Proceeding of Joint SHARE/GUIDE/IBM Application Development Symposium, 1979. Disponível em <https://www.dropbox.com/s/1hv3npu0q2sclk3/MeasuringApplicationDevelopmentProductivity.pdf?dl=0>. Acesso em 2019-03-22.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO/IEC 9126-1 Engenharia de software - Qualidade de produto - Parte 1: Modelo de qualidade**. 2003.

BAUER, C.; KING, G. **Hibernate em Ação**. Rio de Janeiro: Ciência Moderna, 2005.

BRASIL. Ministério do Planejamento, Orçamento e Gestão. Secretaria de Logística e Tecnologia da Informação. **Processo de Software para o SISP (PSW-SISP)**. Brasília: MPOG, 2012. Disponível em <https://www.gov.br/governodigital/pt-br/sisp/guiaPsw.pdf/@@download/file/guiaPsw.pdf>. Acesso em 2020-12-15.

BRASIL. Superintendência do Desenvolvimento da Amazônia (SUDAM). **Regimento Interno**. Belém, 2017. Disponível em < https://www.gov.br/sudam/pt-br/acesso-a-informacoes/institucional/regimento_interno_versao_novembro_2019.pdf>. Acesso em 2020-11-18.

BRASIL. Superintendência do Desenvolvimento da Amazônia (SUDAM). **Plano Diretor de Tecnologia da Informação e Comunicação Biênio 2019/2020** . Belém-PA, 2019. Disponível em <http://antigo.sudam.gov.br/conteudo/menus/referencias/documentosinstitucionais/arquivos/PDTI_SUDAM_2019_2020_vf.pdf>. Acesso em 2020-11-09.

BOOTSTRAP. **Versão 2.3.2**, 2012. Disponível em <https://getbootstrap.com/2.3.2/>. Acesso em 2020-11-09.

BOOTSTRAP. **Versão 3.4.1**, 2013. Disponível em <https://getbootstrap.com/docs/3.4/>. Acesso em 2020-11-09.

BOOTSTRAP. **Versão 4.5**, 2017. Disponível em <https://getbootstrap.com>. Acesso em 2020-11-09.

CERNY, T.; SONG, E. ***A Profile Approach to Using UML Models for Rich Form Generation***. IEEE, 2010.

CHANGEVISION. ***Astah.*** Versão Community Edition. Japão, 2006. Disponível em <https://astah.net>. Acesso em 2020-10-29.

DUBOIS, JULIEN. et al. ***JHIPSTER***. França, 2013. Disponível em: <https://www.jhipster.tech>. Acesso em 2020-10-29.

TOLITECH. ***Code Generator.*** Brasil, 2020. Disponível em <https://www.codegenerator.com.br>. Acesso em 2020-10-29.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. 6ª Edição. Brasil: Pearson, 2011.

EPSILON. 2010. Disponível em <http://www.eclipse.org/gmt/epsilon/>. Acesso em 2020-10-29

FEATHERS, M. C. **Trabalho Eficaz Com Código Legado.** 1ª Edição. Brasil: Bookman, 2013

FIGUEIREDO, R. M. C. **Fusion-Re-I - Um Método de Engenharia Reversa para Auxiliar a Manutenção de Software"**. 1997. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 1997.

GAMMA, ERICH et al. ***Design Patterns: Elements of Reusable Object-Oriented Software***. Addison-Wesley Professional. 1ª Edição. Estados Unidos da América: Addison-Wesley, 1995.

GHOLAMI, M. F. et al. ***Challenges in migrating legacy software systems to the cloud: An empirical study***. Information Systems 67:100–113: Elsevier, 2017;

GOLTZ, URSULA et al. B. **Design for future: managed software evolution**. Computer Science – Research and Development 30, 2014.

IONIC. **Versão 3.9.2, 2013**. Disponível em <https://github.com/ionic-team/ionic-framework/archive/v3.9.2.zip>. Acesso em 2020-11-10.

JAMIESON, J. ***Supplement to the Etymological Dictionary of the Scottish Language***, Volume 2, Pag. 25. UK: Oxford University, 1825. <Disponível em https://play.google.com/books/reader?id=amAJAAAAQAAJ&hl=pt&pg=GBS.PA25>. Acesso em 2020-11-09.

LARMAN, C. **Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo**. Porto Alegre: Bookman, 2007.

LINDGREN, E.; MÜNCH, J. ***Raising the Odds of Success: The Current State of Experimentation in Product Development***. Information and Software Technology 77, 2016.

MATERIALIZE. **Versão 1.0.0**, 2014. Disponível em <https://materializecss.com/>. Acesso em 2020-10-29.

MILLIGRAM. **Versão 1.4.1**, 2015. Disponível em <https://milligram.io/>. Acesso em 2020-11-01.

NETMAKE. ***Script Case.*** Versão 9.0. Brasil, 2000. Disponível em <https://www.codegenerator.com.br/>. Acesso em 2020-10-29.

PAIGE, R.F.; VARRÓ, D. ***Lessons learned from building model-driven development tools.*** Springer Link, 2012.

PAPOTTI, P. E.; FRANCISCO, A.; LOPES, W. ***An approach to support legacy systems reengineering to MDD using metaprogramming***. IEEE, 2012.

PFLEEGER, S. L. **Engenharia de Software: teoria e prática**. 2. ed. São Paulo: Prentice Hall,2004.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software**. 8a edição. Brasil:McGraw-Hill, 2016.

SOMMERVILLE, IAN. **Engenharia de Software**. 9a edição. São Paulo: Pearson, 2011.

TRAVASSOS, G.H.; GUROV, D.; AMARAL, E.A.G.D. **Introdução à Engenharia de Software Experimental**, Rio de Janeiro, Brasil: RT-ES-590/02, COPPE, UFRJ, 2002.

VARRÓ, D., BALOGH, A.: ***The model transformation language of the VIATRA2 framework***. Sci. Comput. Program, 2007.

VAZQUEZ, C. E.; SIMÕES, G. S. ; ALBERT, R. M. **Análise de Pontos de Função**. 12a edição. Brasil: Érica, 2012.

VIATRA. ***Visual Automated model TRAnsformations***, 2020. Disponível em <http://www.eclipse.org/viatra/>. Acesso em 2020-11-09.

WARREN, I.; RANSOM, J. ***Renaissance: a method to support software system evolution***, Proceedings 26th Annual International Computer Software and Applications, Oxford, UK, 2002, pp. 415-420.

WOHLIN, C; RUNESON, P; HÖST, M; OHLSSON, M.C; REGNELL, B.; WESSLÉN, ***A. Experimentation In Software Engineering: An Introduction***. 1.ed. Boston: The Kluwer Academic Publishers, 2000.

APÊNDICE A – PESQUISA SOBRE SISTEMA LEGADO NAS INSTITUIÇÕES PÚBLICA

Avaliação de aspectos relevantes sobre sistema legado presentes nas instituições, que serão usadas para compor uma base de conhecimento, de trabalho de mestrado do Programa de Pós-Graduação em Computação Aplicada da UFPA - PPCA/UFPA, que pode ser acessado pelo site https://docs.google.com/forms/d/e/1FAIpQLSfC3bMWtJKimzWsjYyL2DcAnsacT7Id3VDUi6Wo6Lies1dFuw/viewform ou pelo QR Code a seguir:

*Obrigatório

1. De qual órgão você faz parte? *

Marcar apenas uma oval.

- ◯ BANPARÁ
- ◯ PRODEPA
- ◯ SERPRO
- ◯ SUDAM
- ◯ TRT8
- ◯ Outro: _____

2. Qual o seu perfil profissional na instituição? *

Marcar apenas uma oval.

- ◯ Analista/Programador
- ◯ Gestor TI

3. Quais os principais problemas técnicos encontrados ao manter um sistema legado? *

Marque todas que se aplicam.

- [ ] Documentação insuficiente
- [ ] Capacitação do quadro técnico
- [ ] Ferramentas inadequadas para a tarefa
- [ ] Limitação tecnológica do software
- [ ] Outro: _____

4. Quais os principais problemas de gestão encontrados ao manter em sistema legado? *

Marque todas que se aplicam.

- [ ] Falta de comprometimento da alta gestão
- [ ] Falta de investimentos financeiros
- [ ] Falta de recursos financeiros
- [ ] Recursos humanos insuficientes

5. A instituição possui uma abordagem específica para a evolução tecnológica de sistema legado? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

6. A instituição já manifestou preocupação em evoluir os seus sistemas legados? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

7. A instituição já desenvolveu atividades de evolução tecnológica de algum sistema legado? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

8. Utilizam ferramentas de apoio ao processo de evolução tecnológica de sistemas legados? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

9. Caso utilizem ferramenta de apoio a tarefa, como ela foi adquirida? *

Marcar apenas uma oval.

- ◯ Software Livre
- ◯ Ferramenta proprietária adquirida
- ◯ Desenvolvimento na própria instituição
- ◯ Desenvolvimento em parceria com terceiros
- ◯ Não se aplica
- ◯ Outro: _____

APÊNDICE B – AUTORIZAÇÃO DA SUDAM PARA DESENVOLVIMENTO DA PESQUISA E EXPERIMENTO DO TRABALHO

A seguir, o requerimento de solicitação da pesquisa na unidade referida, acompanhado da resposta da coordenação sobre o desenvolvimento da atividade proposta na instituição. Ambos são documentos assinados eletronicamente, acompanhados dos seus respectivos códigos verificadores.

SUDAM

MINISTÉRIO DO DESENVOLVIMENTO REGIONAL
SUPERINTENDÊNCIA DO DESENVOLVIMENTO DA AMAZÔNIA

REQUERIMENTO

Eu, Luiz Antonio Leão Lisboa Junior, Analista Técnico Administrai vo - Área: Ciência da Computação, lotado na DSIB/CTI, venho solicitar a autorização para desenvolvimento de ai vidade de pesquisa e experimentação nesta unidade, ai vidades essa relacionadas ao meu trabalho de dissertação de mestrado do Programa de Pós-Graduação em Computação Aplicada da Universidade Federal do Pará (PPCA/UFPA).

Luiz Antonio Leão Lisboa Junior, Analista Técnico Administrativo - Área: Ciência da Computação

Ciente.

Luzio Filho, Coordenador Substituto da CTI.

sei! assinatura eletrônica

Documento assinado eletronicamente por **Luiz Antonio Leão Lisboa Junior, Analista Técnico Administrai vo - Ciência da Computação**, em 17/05/2019, às 10:27, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenı cidade deste documento pode ser conferida no site hı p://sei.sudam.gov.br /sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0156105** e o código CRC **C0386CA0**.

Referência: Processo nº 59004.000997/2019-29 SEI nº 0156105

MINISTÉRIO DO DESENVOLVIMENTO REGIONAL
SUPERINTENDÊNCIA DO DESENVOLVIMENTO DA AMAZÔNIA
COORDENAÇÃO DE GESTÃO DE TECNOLOGIA DA INFORMAÇÃO

Ao senhor Luiz Antonio Leão Lisboa Junior,

1. Com cópia para: Divisão de Sistemas e Informações Bibliográficas e Diretoria de Administração.

2. Considerando que a referida pesquisa está alinhada aos objeı vos tái cos da Coordenação de Gestão de Tecnologia da Informação;

3. Considerando as reuniões internas nos quais confirmamos a viabilidade técnica do trabalho;

4. Mediante a condição de que o produto desenvolvido será usado em beneı cio da SUDAM, autorizo a coni nuidade e aprofundamento do trabalho no que diz respeito ao estudo de caso - abordagem de evolução tecnológica de sistemas legados, apoiada por ferramentas CASE.

Documento assinado eletronicamente por **Luzio Santana da Silva Filho**, **Coordenador Subsı tuto(a)**, em 17/05/2019, às 10:35, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenı cidade deste documento pode ser conferida no site hı p://sei.sudam.gov.br /sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0156149** e o código CRC **C01A6E64**.

Referência: Processo nº 59004.000997/2019-29 SEI nº 0156149

APÊNDICE C – QUESTIONÁRIO SOBRE AS EXPERIÊNCIAS DA EQUIPE EM RELAÇÃO À ABORDAGEM KIRN

O formulário da pesquisa pode ser acessado pelo endereço https://docs.google.com/forms/d/e/1FAIpQLSffnOn8qrdX1Uk9hhOawaN4aHbGx1aKmh6b9F7u3DcW86tu1Q/viewform ou pelo QR Code a seguir:

1. Como você avalia o critério Funcionalidade (capacidade do software de prover funcionalidades que satisfaçam o usuário) da aplicação gerada?		1	2	3	4	5	
	Não atende	○	○	○	○	○	Atende totalmente
2. Como você avalia o critério Confiabilidade (capacidade do software em manter um nível de desempenho, em condições esperadas) da aplicação gerada?		1	2	3	4	5	
	Não atende	○	○	○	○	○	Atende totalmente
3. Como você avalia o critério Usabilidade (capacidade do software de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições especificadas) da aplicação gerada?		1	2	3	4	5	
	Não atende	○	○	○	○	○	Atende totalmente
4. Como você avalia o critério Eficiência (o tempo de execução e os recursos envolvidos são compatíveis com o nível de desempenho) da aplicação gerada?		1	2	3	4	5	
	Não atende	○	○	○	○	○	Atende totalmente
5. Como você avalia o critério Manutenibilidade (capacidade do software ser modificado, incluindo tanto as melhorias ou extensões de funcionalidade quanto as correções de defeitos, falhas ou erros) da aplicação gerada?		1	2	3	4	5	
	Não atende	○	○	○	○	○	Atende totalmente

6. Como você avalia o critério Portabilidade (capacidade do sistema ser transferido de um ambiente para outro) da aplicação gerada?	Não atende 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ Atende totalmente
7. De que forma a ferramenta contribuiu para o seu aprendizado em desenvolvimento de software?	Não atende 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ Atende totalmente
8. Como você avalia o nível de produtividade da Kirn Framework?	Não atende 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ Atende totalmente
9. Como você avalia a arquitetura de software proposta pela Kirn Framework?	Não atende 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ Atende totalmente
10. Pontos positivos	
11. Pontos negativos	

APÊNDICE D – REPOSITÓRIOS DOS ARTEFATOS DE SOFTWARE PRODUZIDOS PELO EXPERIMENTO

Segue abaixo a lista dos repositórios mencionados:

- Ferramenta *Kirn Framework*: https://github.com/luizleao/kirn
- Protótipo feito com a abordagem institucional (SUDAM): https://github.com/luizleao/despesasMedicasSudam
- Protótipo feito com a abordagem Kirn: https://github.com/luizleao/despesasMedicasKirn

APÊNDICE E – TELAS DA APLICAÇÃO GERADA PELA KIRN FRAMEWORK

Segue abaixo algumas telas capturada da aplicação gerada pela ferramenta.

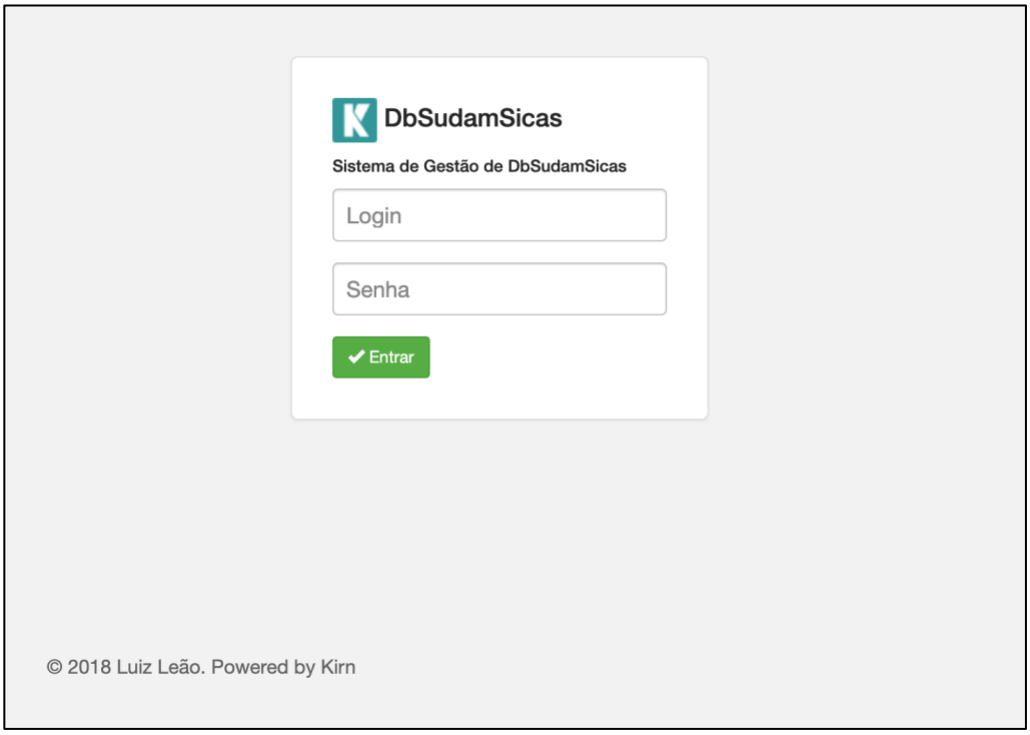


Figura E.1 – Tela de Login da aplicação gerada

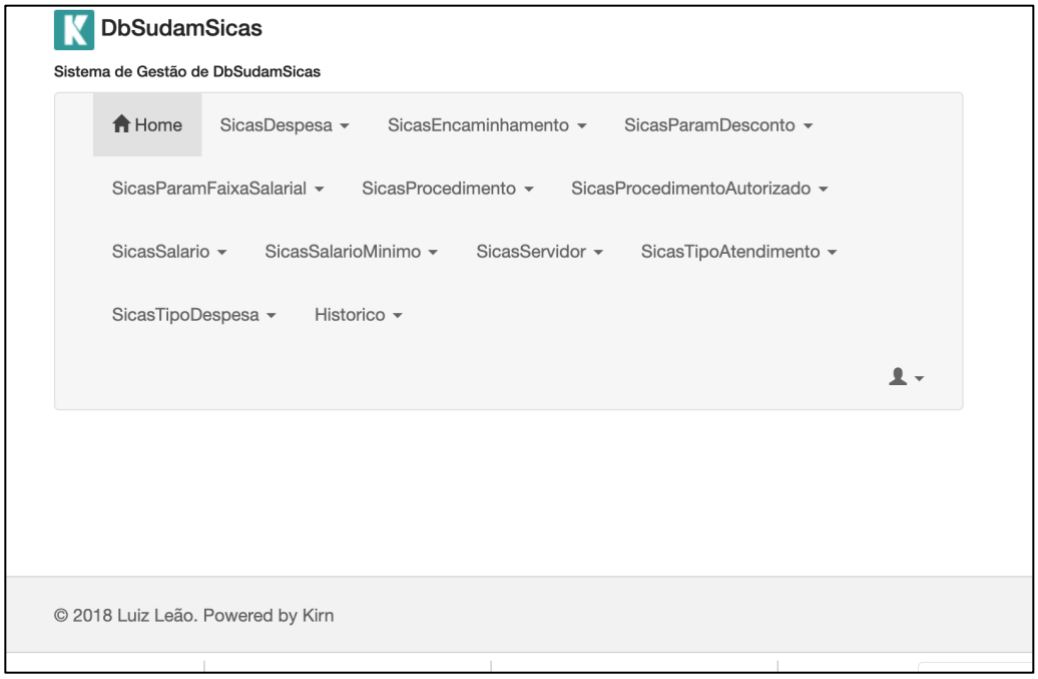


Figura E.2 – Tela de inicial da aplicação gerada

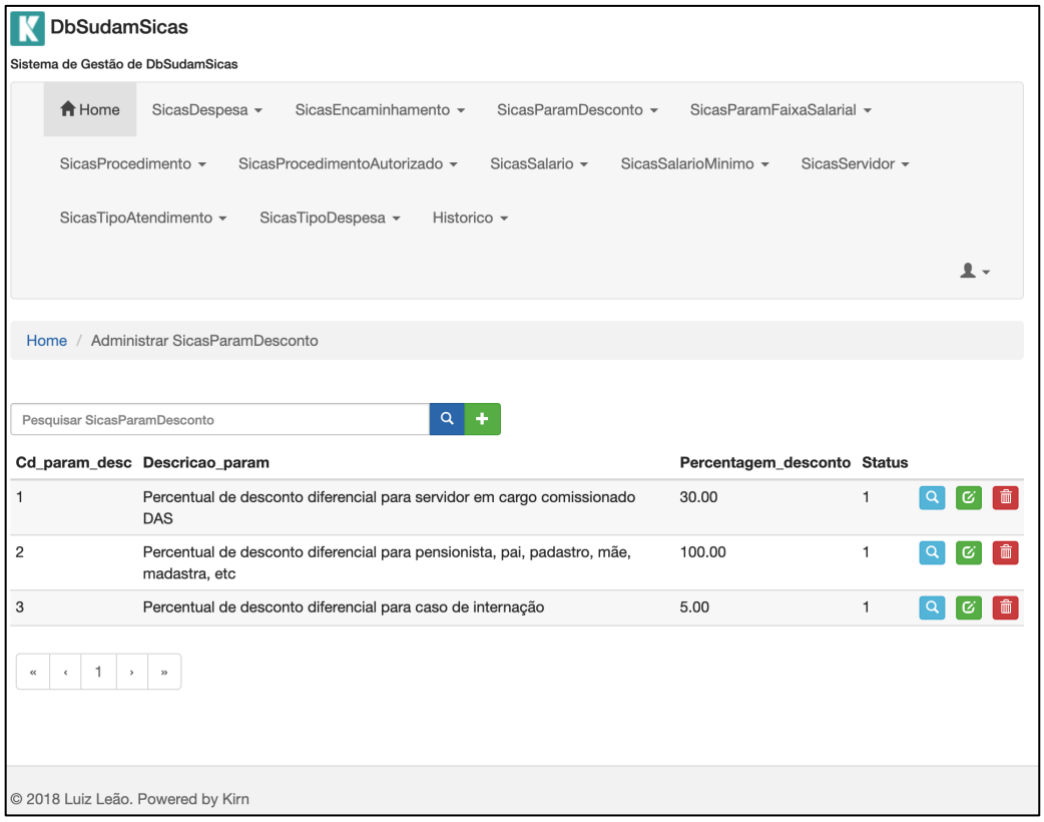


Figura E.3 – Exemplo de Tela de administração de dados de uma determinada tabela da aplicação gerada

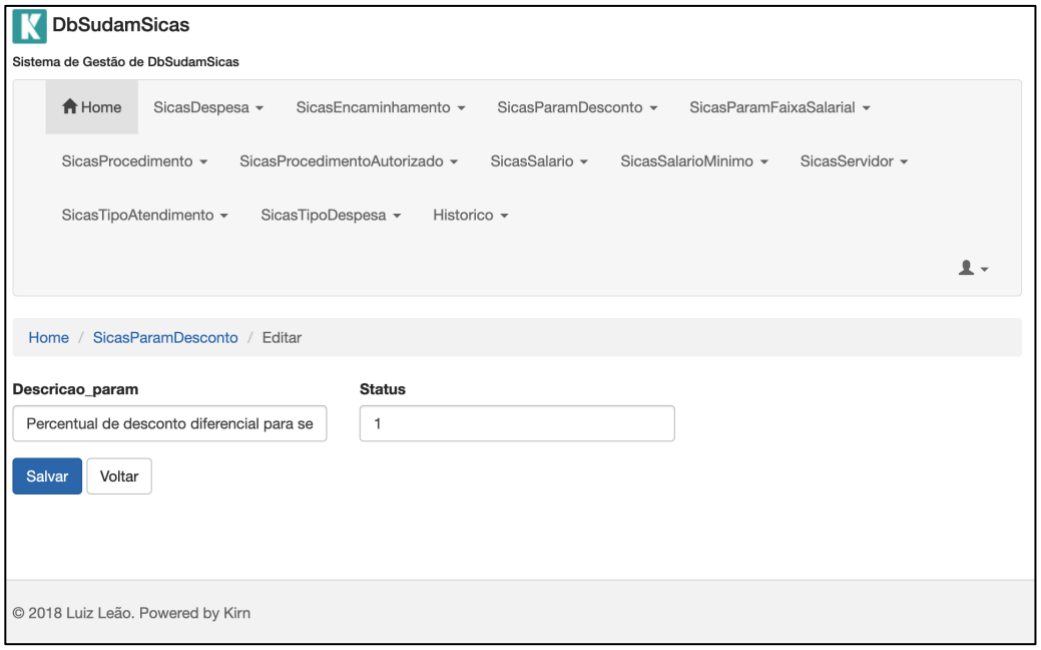


Figura E.4 – Exemplo de Tela de administração de dados de uma determinada tabela da aplicação gerada

APÊNDICE F – DOCUMENTO DE ESPECIFICAÇÃO DE REQUISITOS

Os arquivos usados na migração do sistema legado encontram-se em: https://github.com/luizleao/despesasMedicasKirn/docs/requisitos

APÊNDICE G – ARQUIVOS DE MIGRAÇÃO DA BASE LEGADA

Os arquivos usados na migração do sistema legado encontram-se em: https://github.com/luizleao/despesasMedicasKirn/docs/migracao .