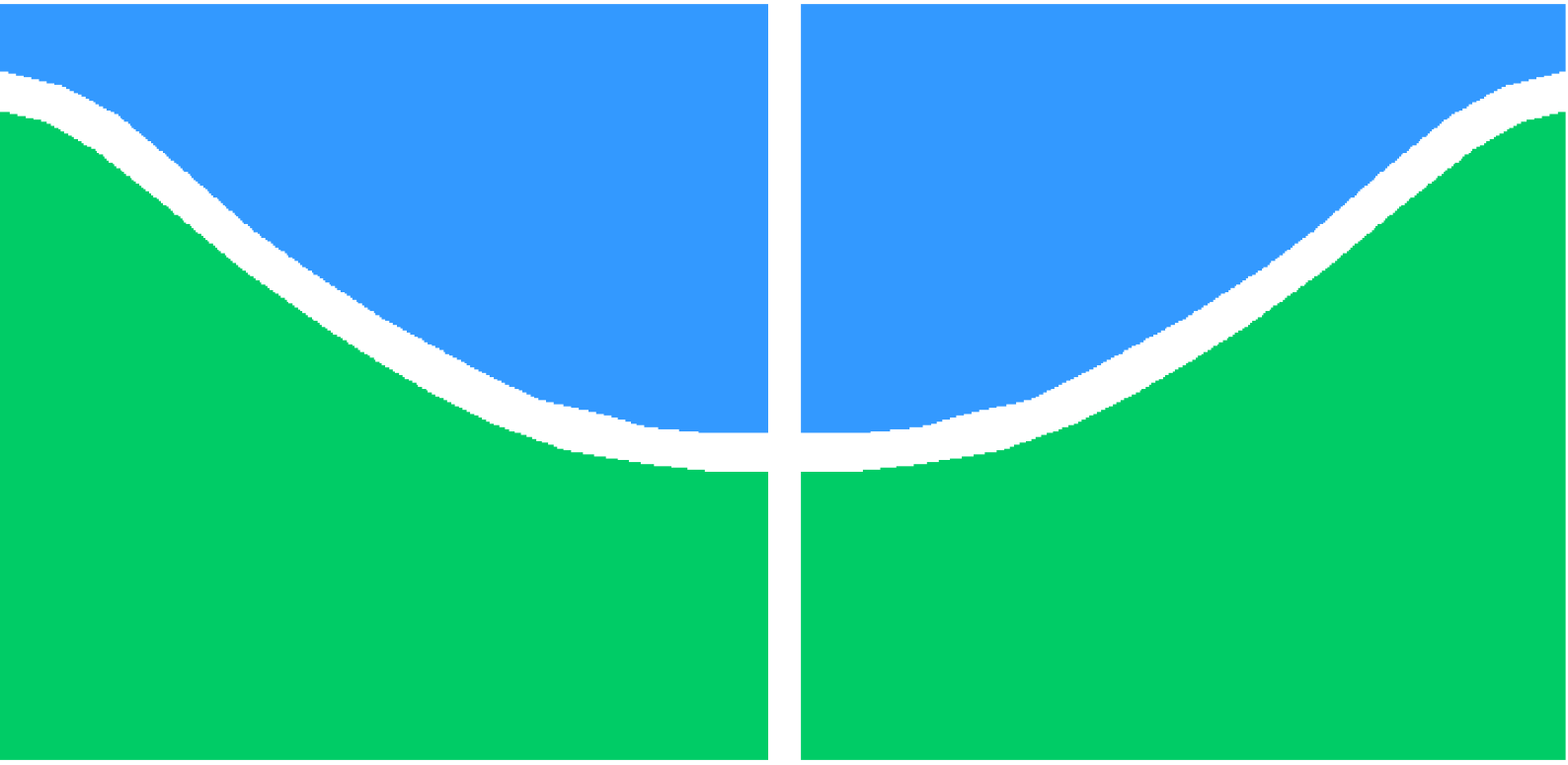

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Uma solução de apoio ao desenvolvimento EUD para um órgão público federal

Autor: Fagner Rodrigues da Silva e Filipe Borges Kelmer Condé
Orientador: Msc. Elaine Venson

Brasília, DF
2016

Fagner Rodrigues da Silva e Filipe Borges Kelmer Condé

Uma solução de apoio ao desenvolvimento EUD para um órgão público federal

Trabalho de conclusão de curso submetido ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Msc. Elaine Venson

Brasília, DF

2016

Fagner Rodrigues da Silva e Filipe Borges Kelmer Condé
Uma solução de apoio ao desenvolvimento EUD para um órgão público federal/
Fagner Rodrigues da Silva e Filipe Borges Kelmer Condé. – Brasília, DF, 2016-
143 p. : il. ; 30 cm.

Orientador: Msc. Elaine Venson

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2016.

1. End User Development. 2. Desenvolvimento Descentralizado. I. Msc. Elaine Venson. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Uma solução de apoio ao desenvolvimento EUD para um órgão público federal

CDU

004+004.4:62=811.134.3

Fagner Rodrigues da Silva e Filipe Borges Kelmer Condé

Uma solução de apoio ao desenvolvimento EUD para um órgão público federal

Trabalho de conclusão de curso submetido ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 09 de dezembro de 2016:

Msc. Elaine Venson
Orientador

Dra. Rejane Maria da Costa Figueiredo
Membro Convidado

Esp. Anderson Rodrigues Ferreira
Membro Convidado

Brasília, DF
2016

Agradecimentos

Eu, Filipe Borges, gostaria de agradecer aos meus pais (Fernando e Marlene) primeiramente, por todo o apoio e suporte que eles tem me dado durante todo o curso, sem o qual jamais iria chegar aonde cheguei.

Eu, Fagner Rodrigues, quero agradecer à minha mãe (Valdevina), por toda a dedicação, ajuda, apoio e carinho. Agradeço muito, à minha avó (Pedrelina), e fica minha dedicação a ela, em sua memória, por toda sua afetividade e amor. Também agradeço a ajuda dos meus colegas e amigos, Thiago Honorato e Laís Barreto que dedicaram tempo e compartilharam conhecimento. Agradeço também aos colegas e amigos do órgão de estudo, Benoni e Marcelo por toda ajuda que ofereceram.

Agradecemos muito a Elaine Baroni, pela sua disposição em nos ajudar na construção deste trabalho. Os constantes *feedbacks* e as ajudas foram de suma importância na escrita deste trabalho.

Agradecemos também:

À nossa orientadora Prof. Msc. Elaine Venson, pela coordialidade e empenho dedicados na construção do trabalho, fundamentais para que o mesmo viesse a ser concluído.

À Prof. Dra. Rejane Maria da Costa Figueiredo, pela ajuda na condução inicial do trabalho.

Aos caros colegas: Delvan Ferreira, Eduardo Garcia, Felipe de Deus, Geison de Souza, Jaime des Sousa, Jean Michel, Mardônio Rodrigues e Roberto de Sousa ficam nosso agradecimento. O tempo que disponibilizaram a este trabalho foi essencial para o avanço do mesmo.

Ao Esp. Anderson Rodrigues Ferreira, que forneceu os insumos necessários para a construção deste trabalho.

Termine cada dia e esteja contente com ele.
Você fez o que pôde.
Alguns enganos e tolices se infiltraram indubitavelmente;
esqueça-os tão logo você consiga.
Amanhã é um novo dia;
comece-o bem e serenamente com um espírito elevado demais
para ser incomodado pelas tolices do passado.
(Ralph Waldo Emerson)

Resumo

Nos últimos anos tem se tornado cada vez mais comum o desenvolvimento de aplicações por usuários finais, seja pelo do desenvolvimento de planilhas robustas, de *queries* em banco de dados, e até mesmo de aplicações a partir de ferramentas especializadas. Esta modalidade de desenvolvimento é denominada na literatura como *End User Development* (EUD). Na administração pública federal brasileira, um órgão tem reconhecido estes esforços dos usuários finais e adotou um modelo de desenvolvimento para aplicações feitas sob essa modalidade, resultando em um alto índice de satisfação interna. Uma das características específicas do EUD é a flexibilidade no desenvolvimento destas aplicações, o que pode levar os desenvolvedores a negligenciarem padrões e boas práticas de desenvolvimento de software. Diante deste contexto, o objetivo deste trabalho foi propor uma solução (sistema) de apoio ao desenvolvimento baseado em EUD no órgão em questão, contribuindo para a melhoria na qualidade e manutenibilidade das aplicações. A pesquisa é classificada como qualitativa, aplicada e explicativa. Como resultado houve a construção do sistema de apoio ao desenvolvimento baseado em EUD para o órgão em questão. Com o trabalho foi possível verificar que há uma necessidade de apoio ao desenvolvedor EUD dentro do órgão, visto a variedade na experiência dos mesmos e nos padrões e técnicas usados pelos diferentes departamentos da instituição.

Palavras-chaves: *end user development*. desenvolvimento descentralizado. apex. desenvolvimento por usuário final. *application express*.

Abstract

In recent years it has become increasingly common application development for end users through the development of robust spreadsheets, database queries, and even applications from specialized tools. This development method is called in the literature as End User Development (EUD). In the Brazilian federal government, a public agency has recognized these efforts by end users and adopted a development model for applications made under this modality, resulting in a high internal satisfaction. One of the specific characteristics of the EUD is flexibility in the development of these applications, which can lead developers to neglect standards and best software development practices. Given this context, the purpose of this work is to propose a development support solution (system) based on EUD within the public agency in question, contributing to improve the quality and maintainability of applications. The research is classified as qualitative, exploratory and applied. As a result there was the construction of the EUD-based development support system for the agency in question. With the work it was possible to verify that there is a need to support the EUD developer within the body, given the variety in the experience of the same and the standards and techniques used by different departments of the institution.

***Key-words**: end user development. decentralized development. apex. application express.*

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

APEX	*Application Express*
BD	Banco de Dados
DGA	Descrição Geral da Aplicação
DISIC	Diretoria de Segurança da Informação e Continuidade de Negócio
EUD	*End User Development*
EUP	*End User Programming*
EUSE	*End User Software Engineering*
MER	Modelo Entidade Relacionamento
ML	Modelo Lógico
MTCU	Museu TCU
PDESC	Processo de Desenvolvimento Descentralizado
PL/SQL	*Procedural Language/Structured Query Language*
SEADE	Serviço de Apoio ao Desenvolvimento Descentralizado
SECOF	Serviço de Programação Orçamentária e Financeira
SECOM	Secretaria de Comunicação
SEPROD	Serviço de Produção de Informações Gerenciais e Sistemas Departamentais
SQL	*Structured Query Language*
SCV	Serviço de Concessão de Vantagens
TI	Tecnologia da Informação

Sumário

1 Introdução

1.1 *Considerações Iniciais*

Neste Capítulo inicial apresenta-se o contexto no qual se insere este trabalho, as justificativas que motivaram o desenvolvimento, o problema, o objetivo, a metodologia de pesquisa adotada e a organização do trabalho nos capítulos seguintes.

1.2 *Contexto*

Atualmente vivemos em uma era onde o mundo está cada vez mais dependente de software. Ele está tão acoplado nas mais diversas áreas que seria difícil prever como o mundo seria sem o suporte e as facilidades que ele traz ao nosso dia a dia. Diante deste acoplamento tão grande e do seu enorme crescimento nos últimos anos, consequência do avanço tecnológico, se torna cada vez mais importante desenvolver software. O desenvolvimento de software pode se inserir em diferentes contextos de negócio, e com isso diferentes habilidades são exigidas dos desenvolvedores, já que cada contexto possui suas particularidades. Além disso, o ritmo acelerado do mundo moderno faz com que os contextos evoluam, e com isso as suas atividades e necessidades (requisitos) também evoluam. A mudança nos requisitos, a sua diversidade, e em alguns casos a dificuldade em definí-los, leva o desenvolvimento tradicional de software a ser lento e caro em contextos que estão sempre evoluindo (LIEBERMAN et al., 2006). Além disso, a limitação na capacidade de produção de software de uma organização pode levar à priorização das demandas, e como consequência algumas áreas de negócio podem ficar sem atendimento. Neste contexto, se torna cada vez mais comum, aplicações de software serem desenvolvidas por desenvolvedores não profissionais, pessoas que não possuem conhecimento elevado em programação, mas que possuem expertise no domínio de negócio em que estão inseridas. O desejo de suportar seus objetivos neste domínio através de uma solução computacional, os leva a serem tanto desenvolvedores quanto usuários finais do software (LIEBERMAN et al., 2006).

Um relatório divulgado pelo *Gartner* em Julho de 2011, indicou que desenvolvedores não profissionais iriam construir ao menos 25 % das novas aplicações de negócio em 2014 (PATERNÒ, 2013). Portanto um modelo de desenvolvimento de software centrado nesses usuários finais, que desenvolvem cada vez mais sistemas e aplicações, vem ganhando força. O *End User Development* - EUD tem como objetivo oferecer meios para que os usuários finais, que não são especialistas em programação, possam desenvolver aplicações de software. O EUD pode ajudar a aumentar a capacidade produtiva do departamento de TI de uma organização, bem como reduzir custos com o desenvolvimento

de aplicações de negócio.

O serviço público brasileiro, por ter natureza administrativa, apresenta grande demanda por desenvolvimento de software. Porém, a limitada capacidade produtiva da área de TI e o excesso de burocracia para o aumento da mesma, que é feita através de concursos ou de contratos terceirizados, acaba contribuindo para que as demandas das áreas de negócio consideradas menos prioritárias sejam deixadas de lado (VIDEIRA; FIGUEIREDO; VENSON, 2014). O uso de planilhas e o desenvolvimento de aplicações clandestinas sem nenhuma documentação e padrão, por essas unidades de negócio, promove o desconhecimento de soluções informatizadas por parte da alta cúpula, bem como a duplicidade de esforços pelas unidades (CARVALHO, 2011). Nesse sentido, alguns órgãos da administração pública federal vêm reconhecendo esses esforços feitos informalmente pelas áreas de negócio, e começaram a adotar o modelo EUD, com algumas adaptações, para tentar contornar os problemas relatados. Um dos órgãos pioneiros nessa adaptação, denominado de órgão público de estudo para fins de privacidade, passou a chama-la de modelo de desenvolvimento descentralizado, uma espécie de EUD onde as aplicações podem ser desenvolvidas por usuários ou por estagiários que os apóiam.

1.3 *Problema*

O principal problema identificado está relacionado à maneira em que o desenvolvimento de sistemas descentralizados é executado no órgão público de estudo. Uma vez que os sistemas são desenvolvidos de forma descentralizada, ou seja, os departamentos realizam o desenvolvimento de seus sistemas conforme suas necessidades, eles geralmente acabam negligenciando o uso de boas práticas e de padrões de desenvolvimento.

Outro problema que é inerente ao desenvolvimento descentralizado é que as entregas que contenham erros podem trazer prejuízos ao órgão. O desenvolvimento descentralizado tem particularidades que são diferentes do desenvolvimento de software tradicional, como o uso de soluções improvisadas, o usuário final não tem tempo nem conhecimento para construir soluções tão sólidas e duradouras como um desenvolvimento profissional.

Assim a questão de pesquisa abordada nesse trabalho é:

- Como apoiar o desenvolvedor na construção de sistemas descentralizados para promover a qualidade e manutenibilidade destes sistemas?

1.4 *Objetivos*

Dentro deste contexto, o objetivo geral deste trabalho é desenvolver uma solução de apoio ao desenvolvimento descentralizado no órgão público de estudo, agregando recursos

que promovam a qualidade dos sistemas desenvolvidos, a partir de princípios e conceitos de engenharia de software.

Considerando o objetivo geral, foram definidos os seguintes objetivos específicos:

1. Entender o contexto do desenvolvimento EUD no órgão público alvo;
2. Diagnosticar os problemas no desenvolvimento EUD de aplicações;
3. Obter um modelo de solução;

1.5 *Justificativa*

A falta de padrões e adesão a boas práticas no desenvolvimento de aplicações pode levar à problemas relacionados a qualidade e manutenibilidade destas aplicações. Segundo (BLACKWELL; GREEN, 2003), no caso do uso de ferramentas de programação, é muito fácil para os desenvolvedores profissionais de software assumir que qualquer pessoa irá se aproximar ao seu nível profissional de programação. A fim de evitar esta armadilha, são necessários atenção e investigação especializada para os usuários finais por parte dos desenvolvedores profissionais, durante a criação de novas tecnologias voltadas ao EUD. Baseado nesses argumentos, é necessário que os desenvolvedores EUD tenham uma atenção especial para o desenvolvimento de seus sistemas, através de um modelo que lhes dê suporte.

1.6 *Metodologia*

A escolha da estratégia de pesquisa deve satisfazer três condições: o tipo de questão de pesquisa, o controle que o pesquisador possui sobre os eventos comportamentais efetivos e o foco em fenômenos históricos, em oposição a fenômenos contemporâneos (YIN, 2001).

Levando em consideração essas condições, a metodologia de pesquisa deste trabalho foi classificada quanto à abordagem, natureza, objetivos ou tipologia, procedimentos técnicos e as técnicas de coleta de dados.

Quanto à abordagem, é classificada como qualitativa, uma vez que as pesquisas classificadas como qualitativas estão relacionadas ao levantamento de dados sobre as motivações de um grupo em compreender e interpretar determinados comportamentos, e em obter as opiniões e expectativas dos indivíduos de uma população, sem o intuito de obter dados estatísticos como resultado (GÜNTHER, 2006; MORESI et al., 2003).

Quanto à natureza, a pesquisa é classificada como aplicada, pois objetiva gerar conhecimentos para aplicação prática, que sejam dirigidos a uma solução de problemas específicos envolvendo verdades e interesses locais (GIL, 2002). Desse modo, este trabalho

se propõe a avaliar e analisar os problemas que constam no desenvolvimento descentralizado de sistemas da instituição e, a partir das análises, propor uma solução que apoie esse desenvolvimento.

As pesquisas científicas, de acordo com Gil (2002), ainda podem ser classificadas segundo seus objetivos gerais em três grandes grupos: exploratórias, descritivas e explicativas. Destes três grupos, a pesquisa explicativa será utilizada, pois a mesma registra fatos, analisa-os, interpreta-os e identifica suas causas. Essa prática visa ampliar generalizações, definir leis mais amplas, estruturar e definir modelos teóricos, relacionar hipóteses em uma visão mais unitária do universo ou âmbito produtivo em geral e gerar hipóteses ou ideias por força de dedução lógica (LAKATOS; MARCONI, 1991). A pesquisa explicativa exige maior investimento em síntese, teorização e reflexão a partir do objeto de estudo. Visa identificar os fatores que contribuem para a ocorrência dos fenômenos ou variáveis que afetam o processo. Explica o porquê das coisas. Nas áreas tecnológicas, há a necessidade da utilização de métodos experimentais de modelagem e simulação para que os fenômenos físico-químicos sejam identificados para posteriormente serem explicados.

Em relação aos procedimentos técnicos, os métodos de investigação adotados foram, pesquisa bibliográfica, pesquisa documental, e pesquisa-ação participante, pois este trabalho tem o objetivo de analisar o problema disponível, resolvendo-o e propondo uma solução prática.

- **Pesquisa-ação**: É um tipo de pesquisa com base empírica que é concebida e realizada em curta relação com uma ação ou resolução de um problema coletivo, no qual os pequisadores e participantes representativos da situação ou do problema estão envolvidos de forma cooperativa ou participativa (THIOLLENT, 2011). Assim, a pesquisa-ação é um procedimento adequado, pois pode ser realizada dentro de um contexto organizacional com o objetivo de resolver um problema prático, onde o pesquisador e os participantes colaboram para o desenvolvimento de uma solução para o problema.

- **Pesquisa participante**: É uma modalidade de pesquisa que tem como propósito "auxiliar a população envolvida a identificar por si mesma os seus problemas, a realizar a análise crítica destes e a buscar as soluções adequadas" (BOTERF, 1999).

- **Pesquisa-ação participante**: É uma pesquisa comprometida com o modelo de ação "aplicação edificante"do conhecimento científico (SANTOS, 1989), que tem, entre outros, os princípios: ter um lugar em situações concretas onde quem aplica está éticamente, existencialmente e socialmente comprometido; é um processo argumentativo entre grupos que lutam pela decisão do conflito a seu favor; envolve o cientista na luta pelo equilíbrio do poder, obrigando-o assim a tomar partido a favor daqueles que têm menos poder; aceita que os limites e deficiências dos saberes locais

não justificam a recusa desses, porque isso significa desarmar argumentativamente e socialmente seres competentes. Assim a pesquisa-ação participante pode ser considerada como o produto da influência da pesquisa participante sobre a pesquisa-ação (NOVAES; GIL, 2009).

- **Pesquisa bibliográfica**: É a pesquisa desenvolvida com base em material já elaborado, constituído principalmente de livros e artigos científicos (GIL, 2002). Dessa forma, nesse trabalho, será realizada pesquisa na principal base de dados científica, a CAPES, bem como outras bases que se julgar necessário. Os temas pesquisados incluíram os critérios de avaliação da ação a ser aplicada, a escolha de seleção de amostra, a elaboração de técnicas de coleta de dados e assuntos relacionados ao tema de desenvolvimento de software descentralizado para servir de apoio para a construção do referencial teórico.

- **Pesquisa documental**: Se difere da pesquisa bibliográfica por valer-se de materiais que não receberam, ainda, um tratamento analítico ou que ainda não podem ser reelaborados de acordo com os objetivos de pesquisa (GIL, 2002). Foi realizada a análise de documentos elaborados pelo órgão objeto de estudo, inerentes ao processo de desenvolvimento de sistemas descentralizados, para a proposição de uma solução ao problema levantado.

Quanto à classificação por técnicas utilizadas, as principais técnicas para coleta de dados quando se aborda o procedimento de pesquisa-ação participante, são a entrevista coletiva nos locais de trabalho e a entrevista individual aplicada de modo aprofundado (THIOLLENT, 2011). Ao lado dessas técnicas também podem ser utilizados questionários ou técnicas antropológicas como, por exemplo, observação participante (THIOLLENT, 2011). Desse modo, as técnicas selecionadas para este trabalho foram entrevistas semi-estruturadas, observação participante e análise documental.

- **Entrevistas semi-estruturadas**: Esse tipo de entrevista tem como característica questionamentos básicos que são apoiados em teorias e hipóteses que se relacionam ao tema da pesquisa, sendo o foco principal estabelecido pelo investigador/entrevistador, mantendo assim a presença consciente e atuante do mesmo no processo de coleta de informações. De acordo com TRIVIÑOS (2009), os questionamentos dão frutos a novas hipóteses, surgidas a partir das respostas dos informantes. Ainda segundo o autor, a entrevista semi-estruturada favorece não só a descrição dos fenômenos sociais, mas também sua aplicação e a compreensão de sua totalidade.

- **Observação participante**: Tem por definição a participação real do pesquisador com a comunidade ou grupo, ou seja, o pesquisador se incorpora ao grupo e parti-

cipa das atividades com a finalidade de obter informações (MARCONI; LAKATOS, 2010).

- **Documentos**: Se refere ao estudo de documentos publicados pelo órgão público ou instituição, como por exemplo, normas, acórdão, processos, projetos diretor, etc.

A representação da seleção dos itens da metodologia adotada é representado na Figura 1, onde são exibidos todas as opções disponíveis dentro de cada classificação, sendo as opções destacadas em preto as selecionadas para a execução do trabalho.

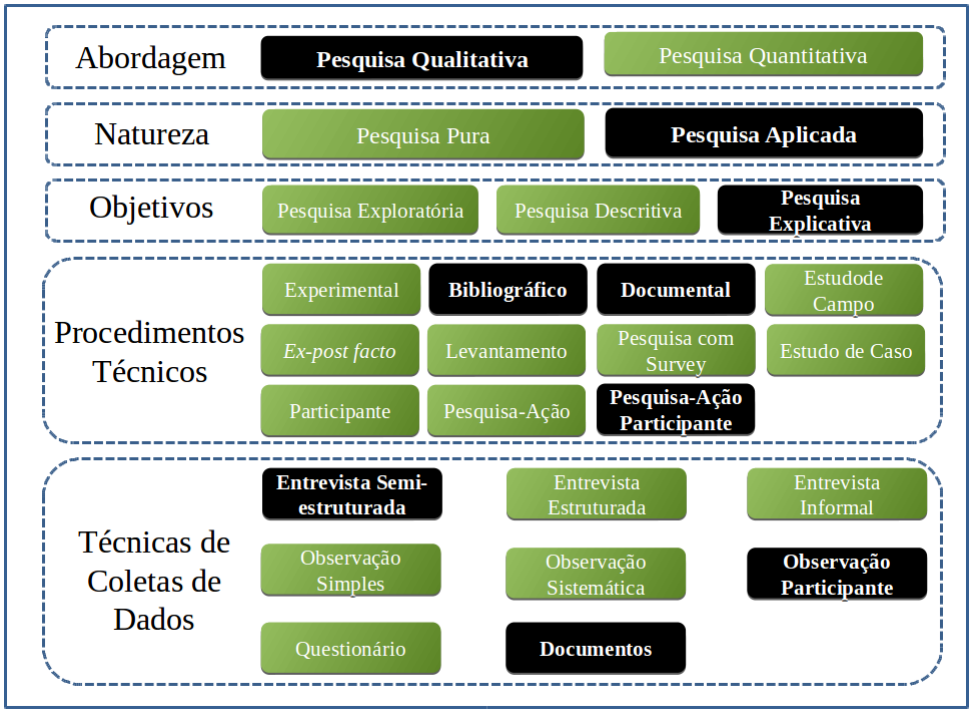

Figura 1 – Seleção da metodologia adotada da pesquisa. Os quadros com fundo preto representam as opções adotadas no presente trabalho.

Nas subseções seguintes serão detalhados a pesquisa-ação participante e a entrevista semi-estruturada, que constituem a base metodológica do presente trabalho bem como a elucidação das bases da pesquisa-ação participante que são a pesquisa-ação e a pesquisa participante.

1.6.1 Pesquisa-Ação

Com o grande uso e aumento da sua amplitude e aplicação, a pesquisa-ação tornou-se atualmente um termo aplicado de "forma vaga"a qualquer tipo de tentativa de melhora ou de investigação da prática. A pesquisa-ação pode ser definida como um termo genérico para qualquer processo que se aprimora a prática pela oscilação sistemática entre agir no campo da prática e investigar a respeito dela (TRIPP, 2005).

Essa metodologia é caracterizada como um método flexível, qualificada pelo constante ciclo das fases. Assim, o processo de execução, geralmente, não é apresentado em ordem cronológica e sim na forma de um conjunto de ações que, embora não ordenadas no tempo, podem ser consideradas como etapas da pesquisa-ação (GIL, 2002; THIOLLENT, 2011).

É necessário planejar, implementar, descrever e avaliar uma mudança para melhoria de sua prática, aprendendo mais, no decorrer do processo, tanto a respeito da prática quanto da própria investigação (TRIPP, 2005). Estas ações estão inseridas nas quatro grandes fases descritas por Thiollent (2009) para a pesquisa-ação: Fase exploratória, na qual os atores da pesquisa começam a detectar problemas a serem combatidos; Fase de pesquisa, na qual coletam-se dados com diversos instrumentos e a pesquisa é discutida pelos membros; Fase de ação, na qual definem-se objetivos, apresentam-se propostas e resultados; e por fim, a fase de avaliação, na qual se obtém conhecimento produzido pela pesquisa e análise dos resultados alcançados.

Segundo Thiollent (2009), o processo de pesquisa-ação não existe de forma padronizada, pois, dependendo da situação social ou do quadro organizacional em que se aplica, os procedimentos e a ordenação das etapas podem variar. O autor também destaca que as três primeiras fases podem ocorrer simultaneamente.

1.6.2 Pesquisa Participante

A pesquisa participante é caracterizada pela interação entre os pesquisadores e as pessoas envolvidas nas situações investigadas, possui uma diferença sutil da pesquisa-ação, pois possui um propósito de emancipação das pessoas ou das comunidades que a realizam, ou seja, ela é escolhida por quem de antemão se propõe a lutar junto a comunidades excluídas (NOVAES; GIL, 2009).

Não existe um modelo único de "pesquisa participante", pois trata-se, na verdade, de adaptar em cada caso o processo às condições particulares de cada situação concreta (os recursos, as limitações, o contexto sociopolítico, os objetivos perseguidos etc.) (BOTERF, 1999).

Suas características essenciais são:

- a realização concomitante da investigação e da ação;
- a participação conjunta de pesquisadores e de pesquisados;
- a proposta político-pedagógica em favor dos oprimidos (opção ideológica);
- o objetivo de mudança ou transformação social;

Esta técnica permite detectar conflitos, tensões e a identificação de viabilidade das mudanças. Assim, duas competências são imprescindíveis para o pesquisador:

- Ter postura política voltada às transformações sociais;
- Estar capacitado a promovê-la, como observador e como analista.

Segundo (SANTOS; COSTA; TREVISAN,) a observação participante pressupõe cinco fases:

- aproximação do grupo;
- processo de inserção;
- observação;
- análise crítica dos dados colhidos;
- retorno para discussão e avaliação dos resultados;

O pesquisador deve considerar que ele é um elemento que está no grupo, mas não é do grupo. Ele possui um papel que transcende ao grupo. Portanto, é preciso que seja aceito na função que de fato tem, ou seja, um pesquisador interessado em analisar um problema no grupo (SANTOS; COSTA; TREVISAN,).

1.6.3 Pesquisa-Ação Participante

Várias são as técnicas da pesquisa convencional que são utilizadas na Pesquisa Ação e Participante. Assim ambas distinguem uma fase de conhecimento da área, no momento que antecede o entrosamento dos pesquisadores com a população pesquisada (ou "interessada") onde aquelas lançam mão de estudos existentes e de dados secundários de várias espécies no sentido de se conscientizarem da realidade que se apresenta. Lançam mão, das técnicas da observação participante e da entrevista na coleta de dados primários (SANTOS; COSTA; TREVISAN,).

Quanto a diferenciação da pesquisa-ação e pesquisa participante, segundo Thiollent (2009) toda pesquisa-ação é do tipo participativa: a participação das pessoas envolvidas nos problemas investigados é absolutamente necessária. No entanto, tudo que é chamado de pesquisa participante não é pesquisa-ação. Isso porque pesquisa participante é em alguns casos um tipo de pesquisa baseado numa metodologia de observação participante na qual os pesquisadores estabelecem relações comunicativas com pessoas ou grupos da situação investigada, com o intuito de serem melhor aceitos. Nesse caso, a participação é sobretudo participação dos pesquisadores e consiste em aparente identificação com os

valores e os comportamentos que são necessários para a sua aceitação pelo grupo considerado. As divergências associadas à pesquisa-ação e à pesquisa participante conduziram a uma espécie de compromisso em torno da pesquisa-ação participante como uma tentativa de minimizar as diferenças e enfatizar as semelhanças entre as duas modalidades.

A proposta da pesquisa-ação participante ganhou força graças a participação de Fals Borda no Simpósio Mundial de Cartagena, que definiu a *investigacion-acción participativa* como uma metodologia inserida num processo vivencial para os grupos de base, que inclui simultaneamente educação de adultos, pesquisa científica e ação política (BORDA, 1973).

Embora vinculada originariamente a movimentos políticos e sociais latino-americanos, a pesquisa-ação participante difundiu-se e vem ganhando adeptos em outras partes do mundo. Foi concebida na América Latina e apresenta como principais influências a análise sóciotécnica norte-americana e a *work democracy research*, de origem escandinava (THORSRUD; EMERY, 1970; ELDEN, 1979).

Muitas obras tratando de pesquisa-ação participante têm sido publicadas em língua inglesa, algumas tratando de áreas específicas como saúde (KOCH; KRALIK, 2009), educação (MCINTYRE, 2007; JAMES; MILENKIEWICZ; BUCKNAM, 2007) e geografia (KINDON; PAIN; KESBY, 2007). O *Handbook of Action Research: Participative Inquiry and Practice*, (REASON; BRADBURY, 2007) uma das mais prestigiadas obras que tratam de pesquisa qualitativa, inclui em sua edição original de 2001 um capítulo de autoria de Fals Borda sobre pesquisa-ação participante (NOVAES; GIL, 2009).

Contudo, aqui poderia surgir uma questão. Qual a diferença entre pesquisa-ação participante e pesquisa participante? É possível resumir a discussão com uma formulação esquemática, mas elucidativa. Nem toda pesquisa-ação pressupõe a participação dos agentes do processo educativo em todas as suas etapas e na definição dos objetivos da pesquisa, e nem necessita que os pesquisadores assumam compromisso político com a transformação social - algo inerente à pesquisa participante. Por outro lado, nem toda pesquisa participante pressupõe ação, podendo se resumir à observação participante, em que há o envolvimento, mas não há previsão de ação planejada de intervenção direta na realidade vivenciada (TOZONI-REIS, 2007).

Assim, em síntese, podemos dizer que a pesquisa-ação participante é o modelo de pesquisa-ação que busca sintetizar ambas as tradições. Opção metodológica pela qual os envolvidos devem trabalhar como agentes sociais em igualdade de poder de decisão, mas sem com isso confundir as atribuições distintas e necessárias. Em que há compromisso político com a emancipação e com a ação reflexiva, articulando teoria e prática, para desvelar a realidade e transformá-la no sentido de fazer com que todos exerçam sua cidadania e aprendam no processo (TOZONI-REIS, 2007).

Em algumas pesquisas de outros autores, foi adotado a pesquisa-ação participante como base, fazendo-se adaptações às fases. CARNEIRO (2010) faz uma adaptação das fases da pesquisa-ação participante utilizando duas linhas de ação: o PAS (Plano de Ações Socioecológicas) e o PES (Programa de Educação Socioambiental) divididos em três fases. Floriani e Mafra (2007) objetivando identificar os problemas de comunitários para subsidiar um plano de ações futuras, adaptou as fases da pesquisa-ação participante em três etapas: sensibilização utilizando como ferramenta a entrevista semi-estruturada, diagnóstico rápido participativo e planejamento estratégico participativo. Marques e Miranda (2016) revela a pesquisa-ação participante cada vez mais eficiente no envolvimento dos grupos na busca de soluções para seus problemas propondo a articulação entre teoria e prática, assim, propõe para a pesquisa, etapas para a pesquisa-ação participante que poderão, se necessário, serem repetidas para garantir a cobertura da amplitude do fenômeno estudado, caracterizando assim cinco etapas: identificação da pergunta de pesquisa, identificação dos estudos pertinentes, seleção dos estudos, mapeamento dos dados e por fim agrupamento, resumo e relato dos resultados.

O destaque dado aqui à adequação da pesquisa-ação participante decorre fundamentalmente do fato da procura enfática por metodologias que evitem o distanciamento entre teoria e prática, obtendo resultados de curto prazo. Thiollent (2009) afirma que o planejamento da pesquisa-ação é flexível, portanto não segue fases rigidamente ordenadas. Conhecimento e ação caminham juntos em um processo dinâmico onde "trata-se de conhecer para agir, de agir para transformar".

A partir do conceito da pesquisa-ação participante, da adaptação de suas etapas realizadas nas pesquisas mencionadas anteriormente, da afirmação de Tozoni-Reis (2007) onde nem toda pesquisa participante pressupõe ação planejada de intervenção direta e nem toda pesquisa-ação pressupõe a participação dos agentes do processo em todas as suas etapas, e na definição dos objetivos da pesquisa sem que os pesquisadores assumam compromisso político com a transformação, foi adotado para o presente trabalho as seguintes etapas para pesquisa-ação participante:

- Diagnóstico da situação: Nesta etapa é realizado o diagnostico da situação vivida no órgão público de estudo, verificando como é o modelo de desenvolvimento de sistemas buscando pelos principais problemas enfrentados.

- Planejamento da ação: Nesta etapa é elaborado prováveis soluções para os problemas levantados no diagnóstico, depois é realizado uma análise documental dos modelos de desenvolvimento existentes e apoio ao desenvolvimento, posteriormente é realizado um cruzamento das informações obtidas do mapeamento das soluções proposta com a análise documental e por fim concretiza-se as soluções com a proposição de um modelo que venha a resolver os problemas levantados.

- Operação da ação: Nesta etapa são executadas subfases em ciclo buscando o aprendizado constante com a análise sempre realizada ao fim de cada ciclo e iniciando-se com uma nova observação do ambiente em que se está situado, buscando por novas soluções que resolvam os problemas vividos e que não foram totalmente diagnósticados.

Para elucidar melhor as subfases da etapa operação da ação, foi elaborado uma estratégia para a implementação da solução do modelo elaborado na fase anterior, planejamento da ação, sendo essa adoção a adaptação das fases da pesquisa participante conforme a Figura 2.

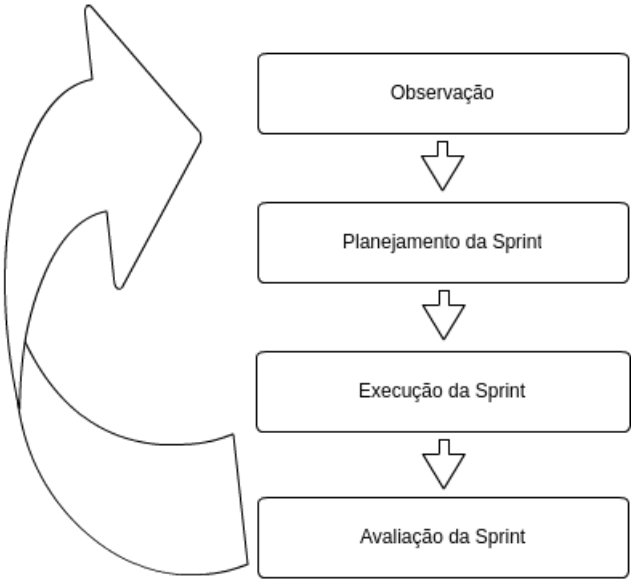

Figura 2 – Estruturação para pesquisa participante. Fonte: (SANTOS; COSTA; TREVISAN,), adaptado

Este ciclo já contempla implicitamente as fases previstas da pesquisa participante, "aproximação do grupo"e "processo de inserção", pois um dos pesquisadores já está inserido no local pesquisado fazendo parte do grupo, e o outro já fez parte do mesmo. Para as fases "observação", "análise crítica dos dados colhidos"e "retorno para discussão e avaliação dos resultados"que são previstas na pesquisa participante, foi elaborado de acordo com a Figura 2 as fases "observação", "planejamento da sprint", "executar a sprint"e "avaliação da sprint"que irão ser percorridas em ciclos iterativos, sendo n iterações a depender do andamento da implementação da solução ou sistema e de acordo com cada observação realizada em cada iteração.

Para cada fase adaptada temos o seguinte paralelo e objetivo:

- Observação: Será relizada a verificação dos dados obtidos e das necessidades observadas enquanto participante do ambiente pesquisado, no caso o órgão inserido.

- Planejamento da Sprint: Faz-se uma equivalência com a fase de análise crítica dos dados colhidos da pesquisa participante, onde nesse caso é realizada uma análise dos dados obtidos e das necessidades observadas fazendo adequações necessárias e realizando o planejamento da sprint com os objetos a serem implementados.

- Execução da Sprint: Nessa fase são implementados os objetos planejados na sprint.

- Avaliação da Sprint: Essa fase é equivalente a fase de retorno para a discussão e avaliação dos resultados da pesquisa participante, onde será avaliado e discutido os resultados da sprint, podendo caso necessite, realizar mudanças no escopo que foi planejado como alterar, incluir ou retirar dos dados obtidos os objetos a serem implementados sempre objetivando o que é melhor para o órgão, uma vez que o pesquisador é envolvido e faz parte do mesmo.

1.6.4 Entrevista Semi-Estruturada

Existem diversos métodos de coleta de dados para suportar os propósitos de um procedimento técnico, e um dos métodos comumente usados na engenharia de software é a entrevista (RUNESON et al., 2012). A entrevista é caracterizada como um método de coleta de dados onde o pesquisador está em contato direto com os entrevistados e como um método de coleta de dados em tempo real (RUNESON et al., 2012). O diálogo entre o entrevistador e o entrevistado é guiado por um conjunto de perguntas, que podem ser classificadas como abertas ou fechadas (RUNESON et al., 2012). As perguntas abertas permitem uma maior gama de respostas e relatos de problemas por parte do entrevistado. Já as perguntas fechadas oferecem alternativas mais limitadas, se comparadas as perguntas abertas.

Segundo Runeson et al. (2012), a entrevista pode ser classificada em três categorias diferentes:

1. **Não-estruturada**: As perguntas são formuladas de forma aberta e de acordo com os interesses do pesquisador. A conversa durante a entrevista irá se desenvolver de acordo com os interesses do entrevistado e do entrevistador.

2. **Semi-estruturada**: As questões planejadas não são necessariamente perguntadas na ordem em que estão listadas, tendo o desenvolvimento da conversa influência direta na ordem em que as perguntas são feitas ao entrevistado. Além disso, esse tipo de pesquisa permite o improvisamento e a exploração de problemas levantados durante a conversa. É comum em estudos de caso em engenharia de software.

3. **Completamente estruturada**: Todas as perguntas são detalhadamente planejadas de antemão e são feitas exatamente na ordem em que estão listadas. Esse tipo

de entrevista se assemelha a *surveys* baseados em questionários, pois se trata de perguntas fechadas.

Uma vez que os dados são coletados, o foco se volta para a análise e interpretação dos mesmos. O objetivo é derivar conclusões dos dados, buscando a compreensão e a formação de padrões em cima dos mesmos, por meio de uma cadeia de evidência. Uma cadeia de evidência significa que o leitor pode chegar aos mesmos resultados e conclusões em cima dos dados coletados (RUNESON et al., 2012). A análise e interpretação dos dados pode ser dividida nos seguintes procedimentos (RUNESON et al., 2012):

1. Os dados são codificados de forma que a cada pedaço do texto, que pode ser uma linha ou um parágrafo por exemplo, é atribuído um código que pode representar um certo tema, uma área, uma construção e assim por diante. Observa-se que um pedaço de texto pode possuir mais de um código e um código pode estar associado a mais de um pedaço de texto.
2. Identificar um conjunto de hipóteses em cima dos dados codificados.
3. Formar um conjunto de generalizações/conclusões.
4. Relatar os resultados.

1.6.5 Fases da Pesquisa

A partir da metodologia, foram definidas as fases e etapas do trabalho, sendo que as fases foram:

- Definição do trabalho,
- Pesquisa bibliográfica,
- Pesquisa-ação participante e documental,
- Coleta de dados

Na Figura 3 pode-se observar o plano metodológico adotado para este trabalho e em seguida uma descrição de cada uma das fases.

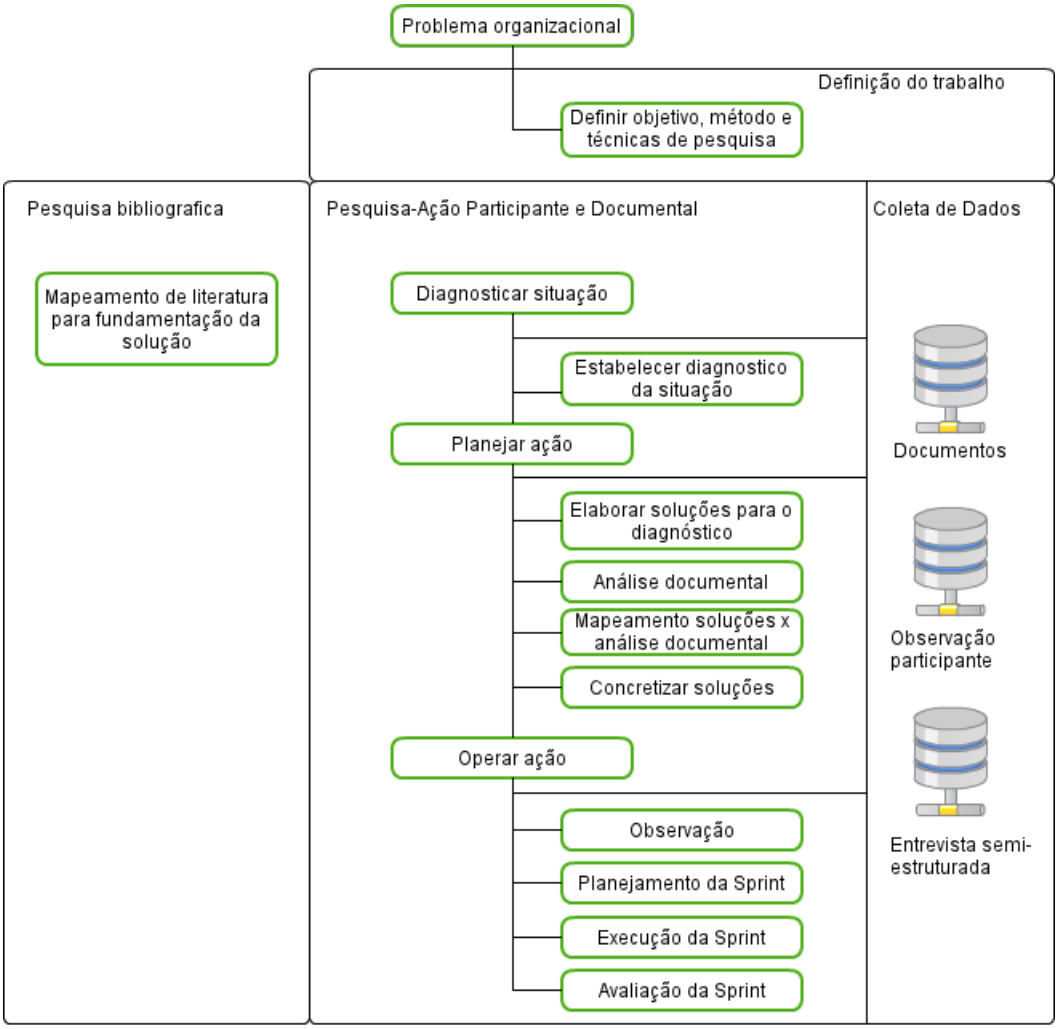

Figura 3 – Plano metodológico adotado. Fonte: autores

Definição do Trabalho

Esta etapa compreende o estabelecimento do trabalho, com base no problema organizacional levantado em alto nível. Nesta etapa foram definidos o objetivo a ser atingido, a pergunta de pesquisa, a seleção metodológica e as fases da pesquisa.

Pesquisa Bibliográfica

As pesquisas bibliográficas foram realizadas ao longo de toda a pesquisa-ação participante. Esta fase oferece apoio, principalmente, para o planejamento da ação, sendo que é nela onde ocorre o mapeamento da literatura para a construção do referencial teórico e definição da ação a ser aplicada.

Pesquisa-Ação Participante e Documental

Esta fase é a base do trabalho, onde foram realizadas as etapas definidas do processo de pesquisa-ação e participante, dentro do órgão público de estudo. A fase buscou

entender os problemas ocorridos durante a aplicação do desenvolvimento de sistemas descentralizados, através da experiência de vivência no órgão e pelos documentos inerentes ao processo, afim de encontrar uma solução de apoio. Uma vez que a solução de apoio foi encontrada, ela foi implementada.

Coletas de dados

Esta fase inclui a realização de entrevistas e análise dos documentos e experiências do órgão público de estudo referentes ao desenvolvimento descentralizado.

1.7 *Organização do Trabalho*

Este trabalho está organizado em seis capítulos. Neste capítulo 1, são apresentados: o contexto do trabalho, o problema com a questão da pesquisa, os objetivos, a justificativa e a metodologia de pesquisa adotada.

Os capítulos 2 e 3 apresentam o referencial que embasa este trabalho. O referencial teórico é o meio em que o autor do trabalho demonstra o seu conhecimento sobre uma determinada área de estudo (teorias, vocabulários, variáveis chave, fenômenos, métodos e história) (RANDOLPH, 2009).

O capítulo 4 materiais e métodos traz uma caracterização do órgão de estudo, a descrição das etapas usadas na elaboração do modelo da solução (execução das entrevistas e análise documental no órgão), e o relato do processo de implementação da solução, de acordo com a metodologia definida.

No capítulo 5 é mostrado os resultados obtidos no trabalho, onde apresentam-se as atividades e guias resultantes e complementares do diagnóstico obtido, o sistema final construído e a avaliação da conformidade do sistema com o diagnóstico obtido.

No capítulo 6 é apresentado a conclusão e os trabalhos futuros.

Encontra-se ainda o apêndice do trabalho, onde são apresentadas as perguntas das entrevistas que foram elaboradas no "Apêndice A"e as respostas das entrevistas que foram aplicadas no "Apêndice B".

2 EUD e Desenvolvimento descentralizado

2.1 *Considerações Iniciais*

Neste capítulo será apresentado um referencial teórico sobre o *End User Development*, sobre o desenvolvimento descentralizado, diferenciando este do *End User Development*, e sobre o APEX, ferramenta adotada no desenvolvimento descentralizado no órgão público de estudo.

2.2 *End User Development*

End User Development (Desenvolvimento por usuário final) é um modelo de desenvolvimento de software onde o usuário final é o principal responsável pela construção do software. Lieberman et al. (2006) define o *End User Development* (EUD) como um conjunto de métodos, técnicas e ferramentas que permite aos usuários de sistemas de software, que agem como desenvolvedores de software não profissionais, em algum ponto, criar, modificar ou estender um artefato de software.

Para que o desenvolvimento seguindo este modelo possa ocorrer é necessário que existam meios para que o usuário final possa desenvolver e adaptar o software, e para tanto a tecnologia envolvida deve diminuir o esforço cognitivo necessário para a construção do software, através da aproximação conceitual entre as ações do mundo real e as do mundo da programação (FISCHER et al., 2004). Os usuários finais geralmente não possuem habilidades de um profissional da área de software, e também não estão interessados em construir sistemas no mesmo nível que esses profissionais, por isso é necessário que a tecnologia usada no desenvolvimento alinhe a complexidade relacionada a esta atividade com as habilidades do usuário final. A motivação dos usuários finais é algo essencial para o favorecimento deste modelo, e fatores como o empoderamento, a velocidade de desenvolvimento, a flexibilidade e o controle influenciam diretamente na motivação desses usuários.

O principal objetivo do EUD é oferecer meios para que os usuários finais consigam desenvolver e adaptar software (LIEBERMAN et al., 2006). Desta maneira, as aplicações preparadas para o EUD devem ser, principalmente, flexíveis, fáceis de se entender, de se usar, e de se ensinar (LIEBERMAN et al., 2006). A preocupação com a tecnologia usada neste modelo de desenvolvimento, mais especificamente na parte das linguagens de programação e aplicações de desenvolvimento, é a relação escopo de aplicação versus esforço de aprendizagem. A Figura 4 ilustra a relação do escopo de aplicação e do custo

de aprendizagem, para diferentes linguagens de programação e aplicações. Linguagens de programação mais tradicionais como C++ e JAVA oferecem a possibilidade de construção de software de uma grande variedade de domínios, porém a um alto custo associado ao esforço de aprendizagem. Outras linguagens possuem um menor esforço de aprendizagem, ao custo de uma limitação no escopo de aplicação. As linguagens ou aplicações de desenvolvimento ideais para o EUD são as que possuem um alto escopo de aplicação e um baixo esforço de aprendizagem (FISCHER et al., 2004). As que existem atualmente só utilizam uma pequena parte do potencial do EUD, com algumas falhas (PATERNÒ, 2013).

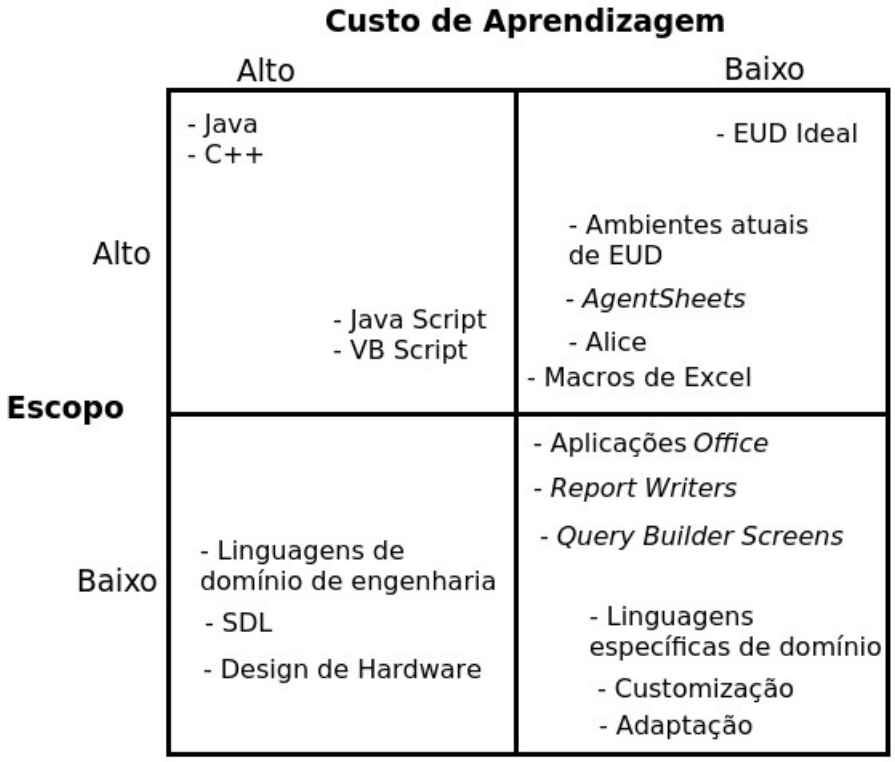

Figura 4 – Relação escopo de aplicação e custo de aprendizagem. Adaptado de: (FISCHER et al., 2004)

Lieberman et al. (2006) classifica as atividades do usuário final em dois tipos:

1. **Parametrização ou Customização:** São atividades que permitem o usuário escolher comportamentos, apresentações e mecanismos alternativos, que já existem dentro de uma aplicação.

2. **Criação e modificação de programas:** São atividades que implicam na criação ou modificação de artefatos de software. Macros e linguagens de *script* são exemplos deste tipo de atividade.

2.2.1 End User Programming

Programação do usuário final (EUP) é definida como "programação para atingir o resultado de um programa, em vez do próprio programa"(KO et al., 2011). Em EUP, o objetivo do desenvolvedor é realmente usar o programa, isso contrasta com a programação profissional, onde o objetivo é criar um programa ou sistema para que outras pessoas possam usar, na maioria das vezes em troca de valores que precificam o sistema, ou seja, na venda de seu sistema.

Os programas criados através do EUP podem ser extensões das aplicações existentes, ou podem ser novas aplicações que rodam separadamente dos que já existem. Os usuários finais podem executar EUP através de uma vasta gama de estilos de interação (NARDI, 1993).

2.2.2 End User Software Engineering

Outro conceito relacionado ao EUD é a engenharia de software do usuário final (EUSE). EUSE é relativamente um novo subconjunto de EUD que começou há cerca de uma década atrás. Sua ênfase está na qualidade do software aos usuários finais para os mesmos possam cria-lo, modifica-lo ou estende-lo; dessa forma a sua investigação centra-se sobre os métodos, técnicas e ferramentas que promovam a qualidade de tal software. Esta área tem surgido por causa da ampla evidência de que os programas criados por usuários finais são desenvolvidos com erros no final podem custar caro a uma organização (PANKO, 1998; BURNETT, 2010; KO et al., 2011).

A engenharia de software do usuário final (EUSE) é definida como "a programação do usuário final envolvendo atividades sistemáticas e disciplinadas que tratam de questões de qualidade de software"(KO et al., 2011).

A atenção à qualidade é importante para o EUP porque um software mal escrito pode causar perda de dados, falhas de segurança, perda financeira, ou até mesmo danos físicos.

As qualidades de software relevantes para EUSE são os mesmos que os de interesse para desenvolvedores profissionais que vendem seus produtos. Essas qualidades incluem confiabilidade, desempenho, facilidade de manutenção, reutilização, privacidade e segurança. Algumas qualidades, tais como manutenção e reutilização, só se tornam visíveis depois que um programa foi escrito e está em funcionamento há algum tempo. Assim, EUSE combina com o objetivo do EUP, permitindo que os usuários finais criem software, com a preocupação de qualidade desse software em toda a sua amplitude de um ciclo de vida. Este ciclo de vida inclui requisitos de criação do sistema, verificação, depuração, e a reutilização de código (KO et al., 2011).

2.2.3 Requisitos e Projeto

Requisitos descrevem o que um programa deve fazer, e o projeto refere-se à determinação de como um programa deve fazê-lo. Por exemplo, um requisito pode ser um programa que deva ser capaz de ordenar uma lista de endereços de usuários; e seu *design* ou projeto poderia ter detalhes do algoritmo de ordenação a ser usado.

Exemplos de requisitos (metas) em EUD incluem personalizar a maneira que um aplicativo ou computador se comporta, automatizando tarefas demoradas, realizando cálculos que são difíceis de fazer com precisão, ou comunicar informações (KO et al., 2011; BLACKWELL; GREEN, 2003; ROSSON; CARROLL, 2005).

Obter estes requisitos de forma correta é um aspecto crítico da perspectiva de EUSE, por causa de sua ênfase na qualidade. Espera-se que os desenvolvedores profissionais investiguem, documentem e refinam os requisitos antes de começarem a desenhar ou codificar uma aplicação. Por exemplo, eles podem tentar identificar inconsistências nos requisitos através da aplicação de uma das várias técnicas meticulosas, por exemplo, revisão das partes interessadas ou modelagem formal. Em contraposição, os usuários finais muitas vezes já tem uma idéia sobre as exigências necessárias para a aplicação, e descartam qualquer trabalho extra para alcançar esses requisitos e documentá-los.

Segundo Faily (2008), durante a preparação de casos de uso, os usuários possuem problemas com o nível de conhecimento em engenharia de software, eles acreditam que estão, implicitamente, assumindo o processo. Na sequência da elaboração inicial de casos de uso para diferentes cenários de geração, os usuários relatam problemas com a compreensão do significado das partes constituintes ao usar modelo de caso de uso.

Assim, por vezes, os usuários finais não sabem os requisitos com antecedência e não visualizam um *design*, eles esperaram que esses assuntos sejam esclarecidos durante a evolução da implementação do programa (COSTABILE et al., 2006; FISCHER; GIACCARDI, 2006; MØRCH; MEHANDJIEV, 2000). Desenvolvedores profissionais, muitas vezes, também não conhecem os requisitos com antecedência, mas eles são capazes de

tomar medidas para lidar com essa situação, como por exemplo, empregando um método iterativo que obtém os requisitos através do uso de protótipos afim de evoluí-los, ao invés de omiti-los completamente e seguir em frente. No entrando, os desenvolvedores EUD podem ir diretamente para codificação sem tomar o tempo necessário para documentar as suas necessidades ou procurar inconsistências (ROSSON; BURNETT; SCAFFIDIC, 2013).

Uma abordagem para os desenvolvedores EUD é a busca de uma revisão junto à pares mais experiente. Oa identificar listas curtas de melhores práticas e fornecer ferramentas adequadas, os pesquisadores têm tentado fazer essa prática tão eficiente quanto possível, de modo que possa ser aplicado sem abrandar consideravelmente o ciclo de vida do EUD (POWELL; BARKER, 2008; ROSSON; BURNETT; SCAFFIDIC, 2013). Tal abordagem parece ser bem sucedida em um ambiente organizacional, onde os desenvolvedores EUD tem colegas a quem recorrer e onde a hierarquisa de gestão pode ser usada, se necessário, para impor e fazer cumprir as revisões de projeto.

2.2.4 Verificação e Validação

Verificação e validação abrangem atividades que tentam certificar que um programa faz o que é pressuposto a fazer. O teste é a abordagem mais comum para a verificação e validação. Um dos primeiros trabalhos de apoio à verificação e validação em EUD, foi para ajudar os usuários finais avaliar se os seus programas continham erros, encorajando-os a testar de forma estratégica. Talvez a abordagem de teste do usuário final mais desenvolvida é o *"What You See Is What You Test"* (WYSIWYT) ou o que você vê é o que você testa, que orienta os usuários a testarem sistematicamente (II et al., 2006). O WYSIWYT emprega uma estratégia de *Surprise-Explique-Reward*, em que surpreende com bordas coloridas para atrair atenção às áreas que precisem de testes, dando dicas de ferramentas que expliquem o significado das cores aos usuários, e mostrando a potencial recompensa no uso da ferramenta (WILSON et al., 2003).

2.2.5 Depuração

Depois que um erro de programação é detectado, o próximo passo é removê-lo por depuração. Algumas das técnicas de depuração utilizadas por desenvolvedores profissionais foram adaptadas para o uso em ferramentas EUP. Além de inserir declarações que imprimem os valores das variáveis de um programa em execução, os desenvolvedores EUD podem percorrer instruções uma a uma, observando as operações que estão incorretas (LESHED et al., 2008). Afirmações, representam uma outra técnica importante que tem sido adaptada para uso em EUP. Um usuário pode inserir uma instrução condicional no código, e a execução do programa irá chamar a atenção para este ponto se a instru-

ção inserida for avaliada como falsa em tempo de execução (BURNETT; SCAFFIDI, ; KOESNANDAR et al., 2008; SCAFFIDI; MYERS; SHAW, 2008).

Várias outras ferramentas EUP que suportam técnicas de teste, podem alavancar os resultados de teste para facilitar a depuração.

2.2.6 Reuso

Quando o código é escrito, a reutilização pode acelerar a criação de programas posteriores. Apoiar a reutilização de programas de EUD é um desafio, pois os desenvolvedores EUD raramente têm a oportunidade e a formação exigida para projetar programas altamente reutilizáveis. Outro desafio é que os desenvolvedores EUD podem cometer erros ao criar programas e reutilizar estes códigos podem propagar erros nestes programas dentro de uma organização (MACKAY, 1990). Portanto, mesmo que sistemas com repositórios ou servidores de arquivos possam tornar mais fácil para os desenvolvedores EUD deixar programas ou códigos preparados para reutilização, eles podem tornar extremamente maior a dificuldade por parte de outros desenvolvedores, a avaliação do código desses programas que será reutilizado. Para ajudar a reduzir a dificuldade de reutilização de programas, modelos de programas EUP reutilizáveis estão em desenvolvimento na esperança de ajudar os usuários a procurarem repositórios de programas reutilizáveis relacionados com os seus interesses particulares (SCAFFIDI et al., 2010) (Scaffidi et al 2010). Além disso, um outro trabalho começou a explorar uma forma de ajudar os usuários a extraírem partes reutilizáveis de um programa (ONEY; MYERS, 2009).

2.2.7 Oracle Application Express - APEX

O Oracle Application Express (APEX) é uma ferramenta que permite criar, personalizar e implementar aplicações em cima do banco de dados Oracle, usando apenas um navegador de internet. O APEX é um ambiente de desenvolvimento gratuito para construir aplicações usando SQL e PL/SQL e que apresenta uma plataforma extensível, executada dentro do banco de dados Oracle (ORACLE, 2015). O APEX tem como objetivo diminuir a complexidade de desenvolvimento das aplicações, ao fornecer um ambiente interativo e de fácil manuseio para construir e executar as aplicações, garantindo que tanto desenvolvedores não profissionais quanto desenvolvedores experientes desenvolvam com agilidade. As funcionalidades da aplicação são criadas de forma fácil e rápida. As vantagens do APEX incluem (ORACLE, 2015):

- Velocidade de entrega;
- Maior controle e precisão no desenvolvimento;
- Facilidade para a criação de protótipos;

- Facilidade de implantação;
- Aplicação acessada via navegador (URL);
- Grande quantidade de usuários (desenvolvedores) ativos;

Além disso, o ambiente de desenvolvimento integrado é muito intuitivo, o que torna a introdução e a adoção do APEX uma transição fácil para desenvolvedores experientes ou usuários finais.

O APEX usa uma arquitetura simples, onde as páginas são geradas dinamicamente usando metadados armazenados no banco de dados Oracle. Não há geração de código ou compilação de arquivos. Uma vez que ele foi instalado e configurado, a URL será a via de acesso ao ambiente de desenvolvimento e às aplicações, tanto para os desenvolvedores quanto para os usuários finais. As alterações feitas durante o desenvolvimento são salvas diretamente nas tabelas de metadados, que contém as definições das aplicações. Após isso o desenvolvedor pode executar a aplicação e revisar as melhorias imediatamente. Para a implantação e execução do APEX é necessário um servidor web, um servidor de aplicação, um servidor APEX *listener* e um servidor de banco de dados Oracle, conforme a Figura 5.

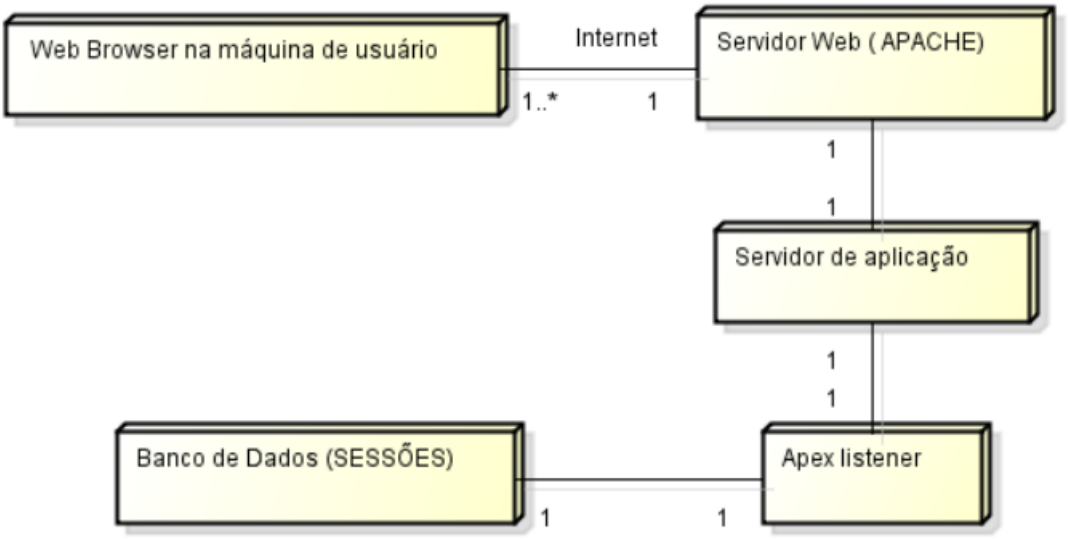

Figura 5 – Arquitetura do APEX. Fonte: (FERREIRA, 2015).

O fluxo de dados na arquitetura do APEX, é composto dos seguintes passos (FERREIRA, 2015):

1. Uma requisição HTTP é enviada, pelo navegador, quando o usuário acessa uma página da aplicação;
2. O servidor web recebe a requisição e identifica que é uma requisição do APEX, redirecionando a requisição ao servidor de aplicação correto;
3. O servidor de aplicação organiza os pedidos (filtros, *cache*, *timeouts*, filas, etc) e transfere para o APEX *listener*.
4. O APEX *listener* recebe o pedido e executa os comandos Java que irão ler os dados do banco e/ou executar códigos PL/SQL. O APEX *listener* interage com sessões de banco, onde os dados são buscados.
5. Faz-se o caminho inverso com os dados coletados, os PL/SQLs executados e os códigos HTML, *Javascript*, CSS, *Ajax* e *jQuery* gerados, que são devolvidos ao navegador que então mostra a tela da aplicação ao usuário.

2.3 Desenvolvimento descentralizado

Alguns órgãos da administração pública federal brasileira começam a ter a iniciativa de adotar este modelo. A diferença entre o EUD e o desenvolvimento descentralizado é que no desenvolvimento descentralizado não é necessariamente o usuário final do software quem irá desenvolve-lo (VIDEIRA; FIGUEIREDO; VENSON, 2014). No órgão público de estudo, que foi um dos pioneiros na implantação deste modelo, os projetos que seguem o desenvolvimento descentralizado tem sido desenvolvidos tanto por servidores, quanto por estagiários de cursos da área de TI, que possuem diferentes níveis de conhecimento sobre desenvolvimento de software, e que também não são os usuários finais do software (VIDEIRA; FIGUEIREDO; VENSON, 2014). O desenvolvedor, se for um estagiário, é alocado na área de negócio e um analista de TI é designado para dar suporte técnico, de forma a tentar garantir que o software sendo desenvolvido siga os padrões estabelecidos e garantidos pela TI corporativa (VIDEIRA; FIGUEIREDO; VENSON, 2014). Dentre as possíveis motivações que tem levado a adoção deste modelo em alguns órgãos da administração pública federal, destacam-se (CARVALHO, 2011):

- Desconhecimento das iniciativas de informatização;
- Falta de alinhamento estratégico das iniciativas;
- Duplicidade de esforços das unidades de negócio;
- Diversidade de ferramentas de desenvolvimento;
- Elevado risco de descontinuidade;
- Comprometimento da segurança da informação;

Este modelo de desenvolvimento no órgão público de estudo é viabilizado pelo uso da ferramenta Oracle APEX, que permite o desenvolvimento de aplicações seguindo a lógica do EUD.

3 Materiais e Métodos

3.1 *Considerações Iniciais*

Este capítulo trata dos procedimentos executados para se atingir o objetivo de propor a solução de apoio ao desenvolvedor EUD. Inicialmente foi realizada uma caracterização do órgão público alvo do trabalho. Em seguida são apresentadas as etapas utilizadas para a avaliação das entrevistas e dos documentos do órgão (processo de desenvolvimento descentralizado e *Wiki*), que serviram de base para a elaboração do modelo da solução. Por fim é apresentado o relato dos procedimentos realizados para a construção da solução.

3.2 Caracterização do Órgão

O presente trabalho está inserido no contexto do órgão público de estudo, o qual possui duas formas distintas de desenvolvimento de software: uma forma centralizada, que é um método tradicional de desenvolvimento de software (desenvolvimento realizado na área da TI e por profissionais de TI), e uma forma descentralizada, que é um modo menos tradicional de desenvolvimento (desenvolvimento realizado nas áreas de negócio e por pessoas com menos qualificação técnica e experiência, geralmente estagiários).

O desenvolvimento descentralizado no órgão público de estudo segue a linha do EUD (End User Development) e, devido a isso, a forma de desenvolvimento é bastante flexível, variando conforme o contexto da unidade de negócio e a experiência do desenvolvedor. A ferramenta adotada pelo órgão para viabilizar o desenvolvimento descentralizado foi o Oracle APEX, que é bastante amigável e permite que pessoas com muito pouca experiência em programação possam desenvolver aplicações. Percebendo a variação na forma de desenvolvimento dos sistemas, a unidade coordenadora do desenvolvimento descentralizado propôs:

- Uma *Wiki* com guias e tutoriais relacionados ao desenvolvimento descentralizado de aplicações, voltados à ferramenta APEX;
- Um processo de desenvolvimento descentralizado, de forma a padronizar este tipo de desenvolvimento dentro do órgão;

A *Wiki* é uma fonte importante de conhecimento ao desenvolvimento descentralizado dentro do órgão, e em alguns casos até ao desenvolvimento corporativo. Já o processo

foi elaborado em cima da ferramenta *Eclipse Process Framework*, onde ele é detalhado a nível de atividades, tarefas, papéis, guias, marcos e fases. Por falta de meios legais para a institucionalização do processo, o mesmo foi desativado.

O desenvolvimento descentralizado apresenta um bom relato de satisfação no órgão público de estudo, porém considerando a importância da aplicação de boas práticas e padrões neste contexto, existem oportunidades de melhoria.

3.3 Entrevistas Realizadas

As entrevistas tiveram como objetivo realizar um diagnóstico da situação do desenvolvimento descentralizado no órgão público de estudo, sob o ponto de vista do desenvolvedor que está inserido neste contexto. Esse diagnóstico procurou abordar os aspectos de dificuldades enfrentadas pelos desenvolvedores, de boas práticas realizadas e de sugestões de melhorias. Para que esses aspectos pudessem ser abordados na entrevista, as questões foram agrupadas por categorias, sendo estas, em sua maioria, relacionadas as fases de um ciclo de desenvolvimento de software. Cada categoria possui um objetivo, que representa qual o tipo de informação que se pretende obter. Na Tabela 1 estão expostas as informações desejadas, através dos objetivos, e suas respectivas categorias.

Tabela 1 – Categorias da entrevista e seus objetivos.

Categoria	**Objetivo**
Descrição do Desenvolvedor	1- Obter informações sobre o perfil dos desenvolvedores.
Papel do Desenvolvedor	1- Obter informações sobre o papel do desenvolvedor no órgão.
Efetividade do Treinamento	1- Obter informações sobre a preparação dos novos desenvolvedores.
Requisitos	1- Obter informações sobre como se dá o levantamento de requisitos nos departamentos da instituição. 2- Obter informações de como são armazenados e gerenciados os requisitos do sistema a ser desenvolvido. 3- Obter informações de como são englobados as mudanças de requisitos ao sistema.
Design	1- Obter informações sobre como é realizado a modelagem (arquitetura) do sistema.
Codificação	1- Obter informações sobre a depuração de erros do sistema. 2- Obter informações sobre estilos de codificação. 3- Obter informações sobre o ambiente de codificação.
Teste	1- Obter informações sobre a elaboração dos testes, bem como a forma de realizá-los.
Implantação	1- Obter informações sobre a migração do sistema para a produção.

A categoria "Descrição do Desenvolvedor"serviu unicamente para descrever o perfil dos desenvolvedores entrevistados, portanto ela não foi considerada na análise dos resultados obtidos. A partir dessas categorias, foram elaboradas as perguntas, de forma que elas atendessem os o bjetivos definidos. O *template* da entrevista pode ser visualizada no "Apêndice A".

Uma vez que a entrevista foi elaborada, o próximo passo consistiu em aplicá-la. A escolha dos desenvolvedores foi feita de forma aleatória, escolhendo-se 8 desenvolvedores diferentes que estão inseridos no contexto do desenvolvimento descentralizado do órgão público de estudo. Os 8 desenvolvedores foram escolhidos devido a objetivação pela qualidade da análise e não pela quntidade e estão divididos entre 7 estagiários e 1 servidor. A Tabela 2 contextualiza o perfil dos desenvolvedores entrevistados. Observa-se que alguns desses dados não se aplicam ao desenvolvedor 6, por este ser um servidor.

Tabela 2 – Perfil dos desenvolvedores entrevistados

Desenvolvedor	**Unidade**	**Tempo de estágio**	**Curso**	**Tempo de Curso**
Desenvolvedor 1	SECOM	1 ano	Análise e Desenvolvimento de Sistemas	2 semestres
Desenvolvedor 2	SCV	1 ano	Sistemas de Informação	8 semestres
Desenvolvedor 3	MTCU	7 meses	Sistemas de informação	6 semestres
Desenvolvedor 4	SEADE	7 meses	Análise e Desenvolvimento de Sistemas	4 semestres
Desenvolvedor 5	SECOF	1 ano	Sistemas de Informação	5 semestres
Desenvolvedor 6	SEPROD	-	-	-
Desenvolvedor 7	DISIC	1 mês	Engenharia de Software	6 semestres
Desenvolvedor 8	SEPROD	6 meses	Engenharia de Redes de Comunicação	9 semestres

Após a aplicação da entrevista, o foco voltou-se a análise dos dados da mesma. A análise feita consistiu dos seguintes passos (P1, P2, P3, P4 e P5):

P1 Atribuiu-se um código, derivado dos objetivos de cada categoria de questões, a cada unidade de resposta, que pode ser uma frase ou um conjunto de frases que tratem de um mesmo assunto;

P2 Agrupou-se as unidades de resposta de acordo com os códigos derivados que receberam;

P3 Com a finalidade de identificar as respostas semelhantes dentro dos grupos de códigos derivados, evitando redundância, rotulou-se as unidades de respostas com o uso de chaves, que representam o sentido específico de uma resposta;

P4 Agrupou-se todas as chaves únicas obtidas, de forma a construir um conjunto de elementos que representam as oportunidades de melhoria na visão dos desenvolvedores;

P5 Elaborou-se ações para esses elementos;

A Figura 6 mostra um esquema que exemplifica a execução dos passos definidos para a análise. No passo 1 (P1), observa-se o agrupamento das unidades de resposta

pelos códigos derivados. Já no passo 2 (P2), observa-se o agrupamento das respostas por códigos derivados. No passo 3 (P3), observa-se as unidades de respostas, dentro dos grupos de códigos derivados, rotuladas com as chaves. Por fim, no passo 4 observa-se os elementos obtidos que representam as oportunidades de melhoria. O passo 5 (P5) será detalhado mais adiante, já que faz parte dos resultados.

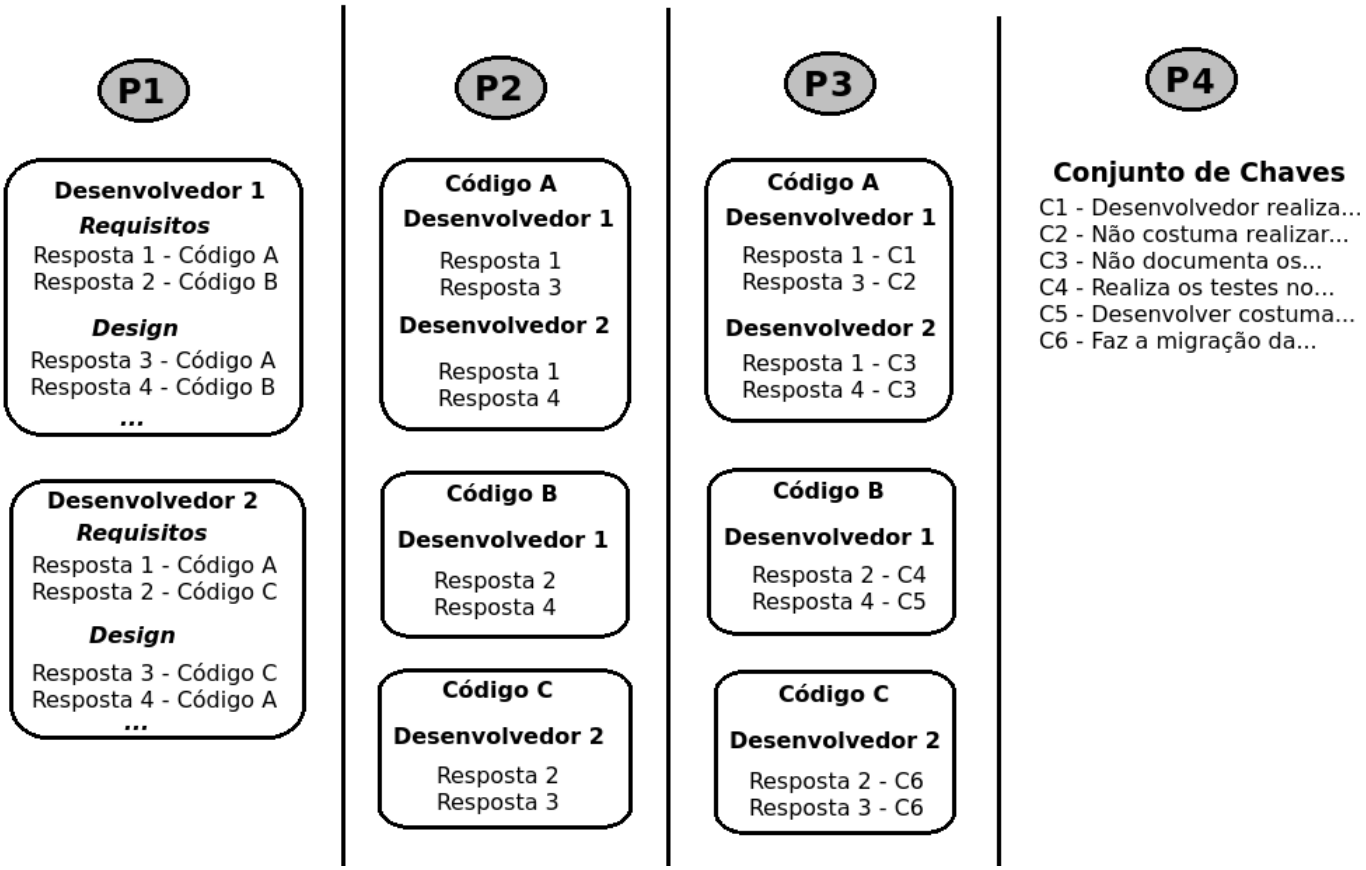

Figura 6 – Etapas realizadas na análise das respostas.

Os códigos foram derivados dos objetivos de cada categoria de perguntas do ciclo de vida. A Tabela 3 ilustra os códigos derivados de cada objetivo de categoria definido.

As chaves foram derivadas dos significados das respostas das entrevistas, de forma que se duas ou mais respostas geraram uma mesma chave, devido aos seus significados semelhantes, somente uma única chave foi considerada.

A atribuição de um código e de uma chave foram necessários, pois dentro de um determinado assunto, representado pelo código, ainda poderiam haver respostas semelhantes. Desta forma, garante-se que o conjunto final de chaves conterá somente chaves únicas, ou seja, respostas únicas, eliminando a redundância no diagnóstico.

Tabela 3 – Códigos elaborados para a análise dos resultados

Categoria	**Objetivo**	**Código Derivado**
Descrição do desenvolvedor	Obter informações sobre o perfil dos desenvolvedores	Perfil do Desenvolvedor
Papel do desenvolvedor	Obter informações sobre o papel do desenvolvedor no órgão	Papel do Desenvolvedor
Efetividade do Treinamento	Obter informações sobre a preparação dos novos desenvolvedores	Preparação
Requisitos	Obter informações sobre como se dá o levantamento de requisitos nos departamentos da instituição	Elicitação de Requisitos
	Obter informações de como é armazenado e gerenciado os requisitos do sistema a ser desenvolvido	Gerenciamento de Requisitos
	Obter informações de como é englobado as mudanças de requisitos ao sistema	
Design	Obter informações sobre como é realizado a modelagem (arquitetura) do sistema	Modelagem do sistema
Codificação	Obter informações sobre a depuração de erros do sistema	Depuração
	Obter informações sobre estilos de codificação	Estilos de codificação
	Obter informações sobre o ambiente de codificação	Ambiente de codificação
Teste	Obter informações sobre a realização dos testes, bem como a forma de realiza-los	Teste
Implantação	Obter informações sobre a migração, para a produção, do sistema	Implantação

3.4 Processo de Desenvolvimento Descentralizado

De acordo com o chefe da unidade coordenadora do desenvolvimento descentralizado, houve uma tentativa de implantação de um processo para o desenvolvimento descentralizado já foi realizada pelo órgão público de estudo, evidenciando a necessidade de um processo ou guia para o desenvolvimento descentralizado. O processo, chamado de Processo de Desenvolvimento Descentralizado do órgão público de estudo, teve a sua finalização concluída no ano de 2012, e foi elaborado pela Secretaria de Tecnologia da Informação do órgão. Apesar de o processo ter sido desenvolvido, segundo o chefe da unidade coordenadora do desenvolvimento descentralizado, ele não chegou a ser efetivamente implantado devido a questões normativas do próprio órgão.

O processo, elaborado na ferramenta *Eclipse Process Framework*, que é voltada à construção de processos, consiste de 6 fases bem definidas: Estudo Preliminar de Projeto, Elicitação de Requisitos, Projeto de Banco de Dados, Construção, Homologação e Implantação. Cada uma dessas fases são divididas em atividades e tarefas, que por sua vez são apoiadas por um conjunto de guias e modelos. Os objetivos de cada fase estão listados abaixo, e o ciclo de vida do processo, em termos de fases, é exibido na Figura 7.

1. **Estudo Preliminar de Projeto:** Fase destinada à análise da proposta de solução e, caso a proposta seja aprovada, à preparação de recursos.
2. **Elicitação de Requisitos:** Fase destinada à elicitação de requisitos e à definição da arquitetura da solução.
3. **Projeto de Banco de Dados:** Fase destinada à construção do Modelo de Dados Lógico e do Modelo de Dados Físico da solução.
4. **Construção:** Fase destinada à construção da solução.
5. **Homologação:** Fase destinada à realização dos testes da solução, à capacitação dos usuários e à divulgação da solução.
6. **Implantação:** Fase destinada à implantação da solução.

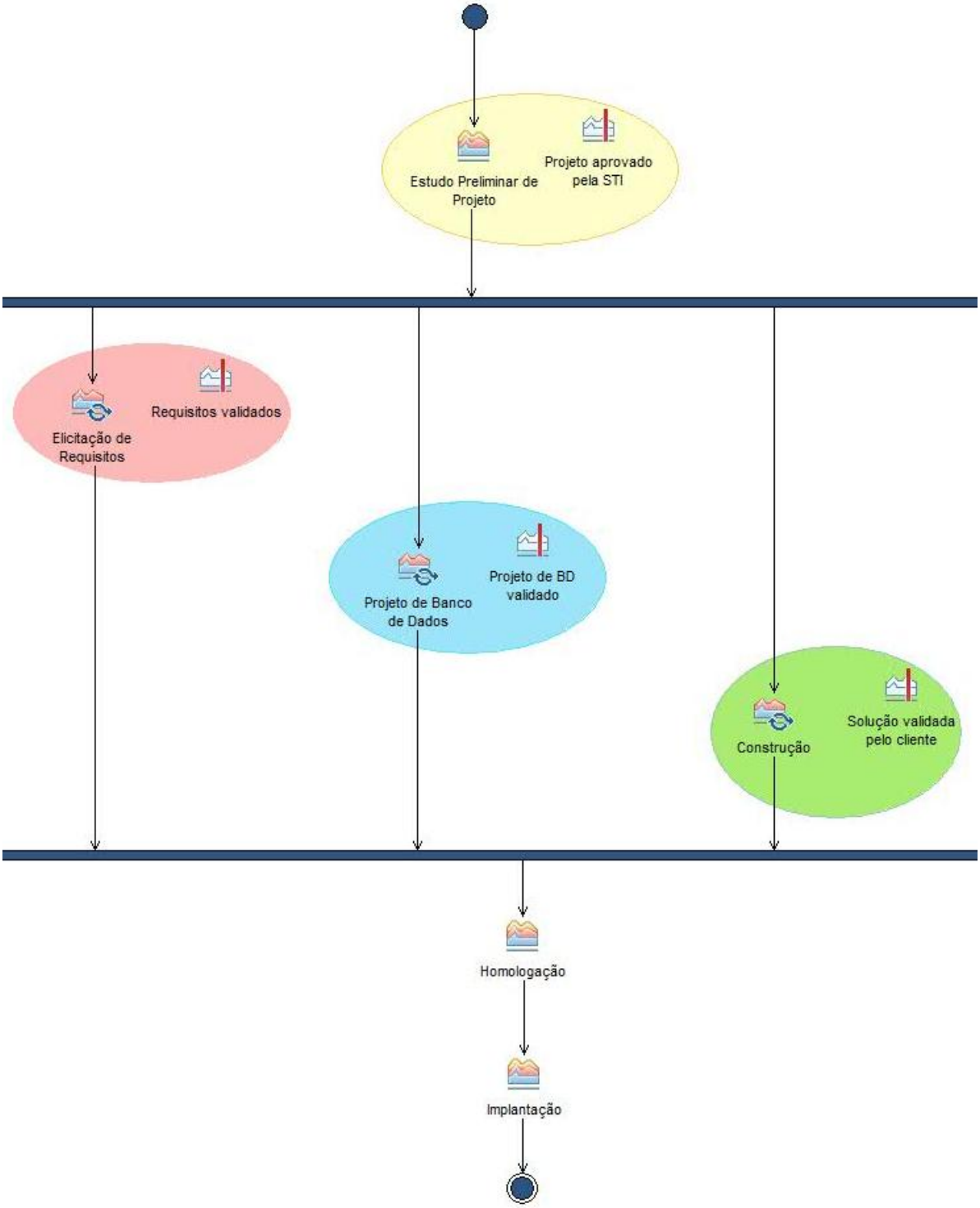

Figura 7 – Ciclo de vida do processo de desenvolvimento descentralizado.

Foram selecionados para a construção da solução de apoio, os componentes de processo (atividades, tarefas, guias e modelos) que possuem uma relação com os problemas, as sugestões e as boas práticas relatados no diagnóstico, e alguns que não estão relacionados diretamente com o diagnóstico, mas que se julgou interessante para complementar a solução.

3.5 *Wiki* do Desenvolvimento Descentralizado

O órgão público de estudo utiliza um sistema de *Wiki* como uma das formas de compartilhamento de informação entre o público interno do órgão, de forma que esse público possa usufruir e contribuir com as informações publicadas. A unidade coordenadora do desenvolvimento descentralizado também possui a sua página dentro da *Wiki*, onde são definidos os guias e as boas práticas, que contribuem para uma melhoria na qualidade das aplicações e que também possam ajudar os desenvolvedores em eventuais dúvidas. A *Wiki* do desenvolvimento descentralizado contém um conjunto de tutoriais, boas práticas e padrões, que variam em nível de complexidade e em nível de necessidade, cabendo ao desenvolvedor acessa-la e usufruir do conteúdo que se encaixa nas suas necessidades.

As boas práticas e padrões definidos na *Wiki* serão um apoio à construção da solução (funcionalidades, guias e tutoriais). Para a concepção da solução, foram selecionados somente os guias e tutoriais que tivessem alguma relação direta com alguma atividade selecionada/proposta.

3.6 Processo de Construção da Solução

O processo usado para a construção da solução foi a metodologia ágil, mais especificamente o método *Scrum*. O *Scrum* parte da premissa de que o desenvolvimento de software é muito complexo e imprevisível para ser precisamente planejado de início, e por isso um processo de controle mais empírico deve ser aplicado, objetivando garantir visibilidade, inspeção e adaptação (MAHNIC; DRNOVSCEK, 2005). Visibilidade significa dar a noção de progresso do projeto com a entrega frequente de incrementos de software executável. Inspeção significa que todos do time devem inspecionar os produtos de trabalho gerados por todos, visando a detecção de erros e o estabelecimento de um senso comum sobre o progresso do projeto. Adaptação significa aceitar as mudanças, já que o desenvolvimento de software é uma atividade complexa e imprevisível (STELLMAN; GREENE, 2014). As diferentes variáveis técnicas envolvidas em um projeto de software, como prazo, qualidade, requisitos, recursos, tecnologias de implementação e ferramentas, devem ser controladas constantemente através de um processo iterativo e incremental (MAHNIC; DRNOVSCEK, 2005).

Poranto decidiu-se pelo *Scrum* devido a sua flexibilidade e por consequência o seu menor encargo sobre a construção da solução. Duas etapas foram realizadas para o planejamento da construção:

1. Elaborar o conjunto de estórias para as atividades, guias e funcionalidades previstas na solução.
2. Definir o período de tempo de cada *Sprint*.

As estórias elaboradas seguem a hierarquia *Feature*/Estória, de forma que algumas funcionalidades e atividades viraram uma *Feature* ao invés de estória, devido à sua abrangência de conteúdo. Algumas estórias tiveram de ser criadas pois não foram previstas no diagnóstico e os autores sentiram esta necessidade, com base nas suas experiências. Além disso, certas estórias que tiveram origem no diagnóstico tiveram que ser associadas a *Features* criadas, pois o diagnóstico não permitiu derivar um conjunto de *Features* que representassem estas estórias. Por fim algumas estórias foram quebradas de forma a originar novas estórias de menor granularidade, e cada estória recebeu uma classificação em relação a sua dificuldade de implementação (Fácil, Médio, Difícil) de forma a ajudar no planejamento da *Sprint*. A Tabela 4 ilustra o conjunto de *Features*/Estórias elaborado.

Tabela 4 – *Features*/Estórias elaboradas

Feature	**Origem no Diagnóstico**	**Estória**	**Origem no Diagnóstico**
F01 - Escolher Técnica de elicitação de requisitos	Sim	H01 - Indicar técnica de elicitação com base em perguntas sobre os parâmetros do projeto (Difícil)	Não
		H02 - Visualizar guia de técnica de elicitação (Fácil)	Sim
		H03 - Cadastrar técnicas de elicitação (Fácil)	Não
		H04 - Editar técnicas de elicitação (Fácil)	Não
		H05 - Listar técnicas de elicitação (Fácil)	Não
		H06 - Cadastrar parâmetros do projeto referente as técnicas (Fácil)	Não
		H07 - Editar parâmetros do projeto referente as técnicas (Fácil)	Não

Feature	**Origem no Diagnóstico**	**Estória**	**Origem no Diagnóstico**
		H08 - Listar parâmetros do projeto referente as técnicas (Fácil)	Não
		H09 - Cadastrar pesos dos parâmetros referente as técnicas (Fácil)	Não
		H10 - Editar pesos dos parâmetros referente as técnicas (Fácil)	Não
		H11 - Listar pesos dos parâmetros referente as técnicas (Fácil)	Não
F02 - Elicitar Requisitos Junto Ao Cliente	Sim	H01 - Estruturação dos requisitos (rastreabilidade) (Médio)	Não
		H02 - Priorizar funcionalidades a serem desenvolvidas (Fácil)	Não
		H03 - Guia de prototipagem rápido (Fácil)	Sim
		H04 - Elicitar Requisitos Junto Ao Cliente (Fácil)	Sim
		H05 - Cadastrar informações levantadas na elicitação de requisitos (Fácil)	Não
		H06 - Editar informações levantadas na elicitação de requisitos (Fácil)	Não
		H07 - Listar informações levantadas na elicitação de requisitos (Fácil)	Não
		H08 - Excluir informações levantadas na elicitação de requisitos (Fácil)	Não
		H09 - Cadastrar atores no sistema a ser desenvolvido (Fácil)	Não

Feature	**Origem no Diagnóstico**	**Estória**	**Origem no Diagnóstico**
		H10 - Editar atores no sistema a ser desenvolvido (Fácil)	Não
		H11 - Listar atores no sistema a ser desenvolvido (Fácil)	Não
		H12 - Excluir atores no sistema a ser desenvolvido (Fácil)	Não
		H13 - Cadastrar *features* do projeto (Fácil)	Não
		H14 - Editar *features* do projeto (Fácil)	Não
		H15 - Listar *features* do projeto (Fácil)	Não
		H16 - Excluir *features* do projeto (Fácil)	Não
		H17 - Cadastrar funcionalidades do projeto relacionado a *feature* (Fácil)	Não
		H18 - Editar funcionalidades do projeto relacionado a *feature* (Fácil)	Não
		H19 - Listar funcionalidades do projeto relacionado a *feature* (Fácil)	Não
		H20 - Excluir funcionalidades do projeto relacionado a uma *feature* (Fácil)	Não
		H21 - Cadastrar requisitos não-funcionais do projeto (Fácil)	Não
		H22 - Editar requisitos não-funcionais do projeto (Fácil)	Não
		H23 - Listar requisitos não-funcionais do projeto (Fácil)	Não

Feature	**Origem no Diagnóstico**	**Estória**	**Origem no Diagnóstico**
		H24 - Excluir requisitos não-funcionais do projeto (Fácil)	Não
		H25 - Cadastrar regras de negócio do projeto (Fácil)	Não
		H26 - Editar regras de negócio do projeto (Fácil)	Não
		H27 - Listar regras de negócio do projeto (Fácil)	Não
		H28 - Excluir regras de negócio do projeto (Fácil)	Não
		H29 - Apresentar rastreabilidade entre projeto, *features* e funcionalidades (Médio)	Não
F03 - Construir modelo de dados	Sim	H01 - Guia de criação e padrões de BD (Médio)	Sim
		H02 - Guia de padrão de nomenclatura de objetos de dados (Fácil)	Sim
		H03 - Guia de geração de scripts sql (Fácil)	Sim
		H04 - Construir modelo de dados (Fácil)	Sim
		H05 - Validar modelo de dados (Difícil)	Sim
		H06 - Atualizar modelo de dados (Fácil)	Sim
		H07 - Cadastrar modelo de dados e informações referentes (Médio)	Não
		H08 - Excluir modelo de dados (Fácil)	Não
		H09 - Visualizar figura do modelo em *fullscreen*	Não

Feature	**Origem no Diagnóstico**	**Estória**	**Origem no Diagnóstico**
		H10 - Validação de modelo de dados por responsável técnico (Médio)	Sim
F04 - Implementação da aplicação	Sim	H01 - Guia de padrões e boas práticas Apex (Médio)	Sim
		H02 - Ajustar interface (Fácil)	Sim
		H03 - Aplicar regras de negócio (Fácil)	Sim
F05 - Elaboração de código	Sim	H01 - Guia de boas práticas plsql (Fácil)	Sim
		H02 - Guia sobre editor de código sql e plsql (Fácil)	Sim
		H03 - Cadastro de código reutilizável (Fácil)	Não
		H04 - Listar código reutilizável (Fácil)	Não
		H05 - Editar código reutilizável (Fácil)	Não
F06 - Procedimentos de teste	Sim	H01 - Preparar ambiente de testes (Fácil)	Sim
		H02 - Definir procedimento de testes (Fácil)	Sim
		H03 - Geração de casos de testes (Difícil)	Sim
		H04 - Guia de ferramenta de automação de teste (Fácil)	Sim
		H05 - Guia de teste de comportamento de código (Fácil)	Sim
		H06 - Guia de teste de página apex (Fácil)	Sim
F07 - Tratamento de erros	Sim	H01 - Guia de uso de depuração de páginas apex (Fácil)	Sim
		H02 - Cadastro de erros conhecidos (Médio)	Sim

Feature	**Origem no Diagnóstico**	**Estória**	**Origem no Diagnóstico**
F08 - Implantação	Sim	F08/H01 - Guia de Migração de Aplicações (Fácil)	Sim
		F08/H02 - Migrar Aplicação (Médio)	Sim
		F08/H03 - Definir Procedimentos de Implantação (Fácil)	Sim
F09 Projeto Kanban (Gerência de projetos)	Sim	F09/H01 - Cadastrar projeto (Fácil)	Não
		F09/H02 - Editar projeto (Fácil)	Não
		F09/H03 - Exibir/listar projetos (Fácil)	Não

Tomando como base a Figura 2, que contempla as fases da pesquisa participante adaptada ao contexto do trabalho, e conforme descrito anteriormente, as *Sprints* serão realizadas dentro dos ciclos iterativos da pesquisa participante. Um ciclo contempla a fase de observação (Verificação de dados obtidos e as necessidades observadas para a aplicação), a fase de planejamento da *Sprint* (Definição das estórias da *Sprint* com base nas necessidades observadas), a fase de execução da *Sprint* (Implementação das estórias definidas), e a fase de avaliação da *Sprint* (Avaliação e discussão dos resultados da *Sprint*).

Uma vez que as *Features* e estórias foram definidas, o início da construção da solução se inicia com a definição da primeira *Sprint*, que teve duração maior do que as outras por conta de adaptação ao processo de construção. As *Sprints* seguintes foram fixadas em períodos de 1 semana. Ao início de cada *Sprint* um conjunto de estórias foram escolhidas para serem concluídas até o final da mesma. Cada estória está associada com a sua dificuldade e com o respectivo membro responsável pela sua conclusão. As Tabelas de 5 a 14 mostram o que foi definido e concluído em cada *Sprint*.

Tabela 5 – Planejamento da *Sprint* 1

	***Features* Envolvidas**	**Planejado**	**Concluído**
Sprint 1	F01 - Escolher Técnica de elicitação de requisitos	F01/H01 - Indicar técnica de elicitação (Difícil, Fagner)	F03/H01 - Guia de criação e padrões de BD (Médio, Filipe)
		F01/H02 - Visualizar guia de técnica de elicitação (Fácil, Fagner)	
	F02 - Elicitar Requisitos Junto Ao Cliente	F02/H03 - Guia de prototipagem rápido (Fácil, Fagner)	
	F03 - Construir modelo de dados	F03/H01 - Guia de criação e padrões de BD (Médio, Filipe)	
		F03/H03 - Guia de geração de scripts sql (Fácil, Filipe)	

- Observação: Não foi observado nada de novo a acrescentar ao diagnóstico previsto.
- Planejamento: Priorizou-se que as estórias contempladas na *Sprint* 1 deveriam ser das duas etapas iniciais do fluxo da Solução (Requisitos e Banco de Dados). Dentro das estórias disponíveis nas etapas do fluxo previamente priorizadas, optou-se pela implementação das estórias descritas na tabela. Observa-se que uma das estórias foi deixada em aberto (sem nenhum membro associado), de forma que se algum dos membros terminassem suas estórias antes do término da *Sprint*, ficaria responsável pela mesma.
- Execução: Das estórias planejadas, foi concluída apenas uma única estória.
- Avaliação: O fato de ser a primeira *Sprint* (Começando a implementação do zero), acabou fazendo com que uma única estória fosse concluída, pelo fato de adequação do tempo de desenvolvimento do escopo do projeto e ambientação das informações obtidas.

Tabela 6 – Planejamento da *Sprint* 2

<table>
<tr><th></th><th>Features Envolvidas</th><th>Planejado</th><th>Concluído</th></tr>
<tr><td rowspan="7">Sprint 2</td><td rowspan="2">F01 - Escolher Técnica de elicitação de requisitos</td><td>F01/H01 - Indicar técnica de elicitação (Difícil, Fagner)</td><td rowspan="7">F03/H03 - Guia de geração de scripts sql (Fácil, Filipe)

F05/H03 - Cadastro de código reutilizável (Fácil, Filipe)

F05/H04 - Listar código reutilizável (Fácil, Filipe)</td></tr>
<tr><td>F01/H02 - Visualizar guia de técnica de elicitação (Fácil, Fagner)</td></tr>
<tr><td rowspan="3">F03 - Construir modelo de dados</td><td>F03/H02 - Guia de padrão de nomenclatura de objetos de dados (Fácil, Fagner)</td></tr>
<tr><td>F03/H03 - Guia de geração de scripts sql (Fácil, Filipe)</td></tr>
<tr><td>F03/H05 - Validar modelo de dados (Difícil, Fagner)</td></tr>
<tr><td rowspan="2">F05 - Elaboração de código</td><td>F05/H03 - Cadastro de código reutilizável (Fácil, Filipe)</td></tr>
<tr><td>F05/H04 - Listar código reutilizável (Fácil, Filipe)</td></tr>
</table>

- Observação: O diagnóstico previsto ainda continuou atendendo as necessidades da solução. Portanto nenhum acréscimo de estória precisou ser feito. Porém notou-se, de forma superficial, que a estória F01/H01 dependia de outras estórias que ainda não tinham sido previstas.

- Planejamento: Optou-se pela retirada da estória F02/H03 pois na Sprint passada nenhum dos membros conseguiu assumi-la. Optou-se também por incluir mais estórias relacionadas a *feature* F03, pois foi uma das *features* que tiveram estórias concluídas na *Sprint* passada. Além disso, resolveu-se incluir a *feature* F05 pois ela contemplava a etapa de Implementação do fluxo da solução.

- Execução: Das estórias planejadas, foram concluídas apenas 3 estórias.

- Avaliação: Houve um aumento de complexidade durante a execução de certas estórias, pois não foi possível prever esse aumento de antemão, o que acarretou na não conclusão des mesmas.

Tabela 7 – Planejamento da *Sprint* 3

	***Features* Envolvidas**	**Planejado**	**Concluído**
Sprint 3	F03 - Construir modelo de dados	F03/H02 - Guia de padrão de nomenclatura de objetos de dados (Fácil, Fagner)	F03/H02 - Guia de padrão de nomenclatura de objetos de dados (Fácil, Fagner)
		F03/H05 - Validar modelo de dados (Difícil, Fagner)	
	F04 - Implementação da aplicação	F04/H01 - Guia de padrões e boas práticas Apex (Médio, Filipe)	F05/H05 - Editar código reutilizável (Fácil, Filipe)
	F05 - Elaboração de código	F05/H05 - Editar código reutilizável (Fácil, Filipe)	

- Observação: Nenhum acréscimo de estória foi feito. Porém notou-se uma dependência da estória F03/H05 com uma estória ainda não prevista.
- Planejamento: A exclusão da estória F01/H01 foi devido a uma dependência observada na *Sprint* anterior. A estória F01/H02 também dependia da estória F01/H01, portanto foi retirada. Além disso, houveram a inclusão da estória F05/H05, relacionada com estórias concluídas na *Sprint* passada, e da estória F04/H01, considerada uma das mais importantes.
- Execução: Das estórias planejadas, foram concluídas somente duas.
- Avaliação: A dependência da estória F03/H05 fez com que não fosse possível concluí-lá, e o esforço demandado pela estória F05/H05 não permitiu a conclusão da F04/H01 na *Sprint*.

Tabela 8 – Planejamento da *Sprint* 4

	***Features* Envolvidas**	**Planejado**	**Concluído**
Sprint 4	F01 - Escolher Técnica de elicitação de requisitos	F01/H03 - Cadastrar técnicas de elicitação (Fácil, Fagner)	F01/H03 - Cadastrar técnicas de elicitação (Fácil, Fagner)
		F01/H04 - Editar técnicas de elicitação (Fácil, Fagner)	F01/H04 - Editar técnicas de elicitação (Fácil, Fagner)
		F01/H05 - Listar técnicas de elicitação (Fácil, Fagner)	F01/H05 - Listar técnicas de elicitação (Fácil, Fagner)
		F01/H06 - Cadastrar parâmetros do projeto referente as técnicas (Fácil, Fagner)	F01/H06 - Cadastrar parâmetros do projeto referente as técnicas (Fácil, Fagner)
		F01/H07 - Editar parâmetros do projeto referente as técnicas (Fácil, Fagner)	F01/H07 - Editar parâmetros do projeto referente as técnicas (Fácil, Fagner)
		F01/H08 - Listar parâmetros do projeto referente as técnicas (Fácil, Fagner)	F01/H08 - Listar parâmetros do projeto referente as técnicas (Fácil, Fagner)
		F01/H09 - Cadastrar pesos dos parâmetros referente as técnicas (Fácil, Fagner)	F01/H09 - Cadastrar pesos dos parâmetros referente as técnicas (Fácil, Fagner)
		F01/H10 - Editar pesos dos parâmetros referente as técnicas (Fácil, Fagner)	F01/H10 - Editar pesos dos parâmetros referente as técnicas (Fácil, Fagner)
		F01/H11 - Listar pesos dos parâmetros referente as técnicas (Fácil, Fagner)	F01/H11 - Listar pesos dos parâmetros referente as técnicas (Fácil, Fagner)
	F03 - Construir modelo de dados	F03/H02 - Guia de padrão de nomenclatura de objetos de dados (Fácil, Fagner)	F03/H02 - Guia de padrão de nomenclatura de objetos de dados (Fácil, Fagner)
		F03/H05 - Validar modelo de dados (Difícil, Fagner)	F04/H01 - Guia de padrões e boas práticas Apex (Médio, Filipe)
	F04 - Implementação da aplicação	F04/H01 - Guia de padrões e boas práticas Apex (Médio, Filipe)	
	F07 - Tratamento de erros	F07/H01 - Guia de uso de depuração de páginas apex (Fácil, Filipe)	

- Observação: Baseado na observação do contexto e na indicação de dependência, houve a necessidade de quebra da estória F01/H01 em outras estórias menores. Portanto a F01/H01 passou a ter outro nome, pois é uma estória de menor granularidade, acompanhada das estórias F01/H02 até a F01/H11.

- Planejamento: Foram incluídas na *Sprint* as estórias de menor granularidade derivadas da antiga F01/H01, e mantiveram-se as estórias pendentes da *Sprint* anterior.
- Execução: Das estórias planejadas, não foram concluídas somente duas.
- Avaliação: A *Sprint* teve um bom resultado, onde a quebra da antiga F01/H01 em estórias de menor granularidade contribuiu para a diminuição da complexidade de desenvolvimento.

Tabela 9 – Planejamento da *Sprint* 5

	***Features* Envolvidas**	**Planejado**	**Concluído**
Sprint 5	F01 - Escolher Técnica de elicitação de requisitos	F01/H02 - Visualizar guia de técnica de elicitação (Fácil, Fagner)	F07/H01 - Guia de uso de depuração de páginas apex (Fácil, Filipe) F01/H02 - Visualizar guia de técnica de elicitação (Fácil, Fagner)
	F03 - Construir modelo de dados	F03/H05 - Validar modelo de dados (Difícil, Fagner)	
	F07 - Tratamento de erros	F07/H01 - Guia de uso de depuração de páginas apex (Fácil, Filipe)	

- Observação: Não foi observado nada de novo a acrescentar ao diagnóstico previsto.
- Planejamento: Foi incluído na *Sprint* somente a estória F01/H02. As estórias F07/H01 e F03/H05 já haviam sido parcialmente implementadas na *Sprint* anterior, portanto foram mantidas.
- Execução: Das estórias planejadas, não foi concluída somente uma.
- Avaliação: O desempenho da *Sprint* foi bom, apesar de ter tido poucas estórias planejadas devido a problemas externos ao trabalho.

Tabela 10 – Planejamento da *Sprint* 6

	***Features* Envolvidas**	**Planejado**	**Concluído**
Sprint 6	F02 - Elicitar requisitos junto ao cliente	F02/H03 - Guia de prototipagem rápido (Fácil, Fagner)	F02/03 - Guia de prototipagem rápido (Fácil, Fagner)
	F06 - Procedimentos de teste	F06/H03 - Geração de casos de testes (Difícil, Filipe)	

- Observação: Não foi observado nada de novo a acrescentar ao diagnóstico previsto.
- Planejamento: A estória F03/H05 foi retirada da *Sprint* devido a constante dificuldade em sua conclusão por conta de dependência, e foram incluídas as estórias F02/H03 e F06/H03.
- Execução: Das estórias planejadas, não foi concluída somente uma.
- Avaliação: A estória F06/H03 apresentou uma grande dificuldade de implementação, devido a necessidade de busca na literatura sobre um método adequado de teste. Apesar disso, ela foi parcialmente concluída.

Tabela 11 – Planejamento da *Sprint* 7

	***Features* Envolvidas**	**Planejado**	**Concluído**
Sprint 7	F02 - Elicitar requisitos junto ao cliente	F02/H05 - Cadastrar informações levantadas na elicitação de requisitos (Fácil, Fagner)	F02/H05 - Cadastrar informações levantadas na elicitação de requisitos (Fácil, Fagner)
		F02/H06 - Editar informações levantadas na elicitação de requisitos (Fácil, Fagner)	F02/H06 - Editar informações levantadas na elicitação de requisitos (Fácil, Fagner)
		F02/H07 - Listar informações levantadas na elicitação de requisitos (Fácil, Fagner)	F02/H07 - Listar informações levantadas na elicitação de requisitos (Fácil, Fagner)
		F02/H08 - Excluir informações levantadas na elicitação de requisitos (Fácil, Fagner)	F02/H08 - Excluir informações levantadas na elicitação de requisitos (Fácil, Fagner)
		F02/H09 - Cadastrar atores no sistema a ser desenvolvido (Fácil, Fagner)	F02/H09 - Cadastrar atores no sistema a ser desenvolvido (Fácil, Fagner)
		F02/H10 - Editar atores no sistema a ser desenvolvido (Fácil, Fagner)	F02/H10 - Editar atores no sistema a ser desenvolvido (Fácil, Fagner)
		F02/H11 - Listar atores no sistema a ser desenvolvido (Fácil, Fagner)	F02/H11 - Listar atores no sistema a ser desenvolvido (Fácil, Fagner)
		F02/H12 - Excluir atores no sistema a ser desenvolvido (Fácil, Fagner)	F02/H12 - Excluir atores no sistema a ser desenvolvido (Fácil, Fagner)
		F02/H13 - Cadastrar features do projeto (Fácil, Fagner)	F02/H13 - Cadastrar features do projeto (Fácil, Fagner)
		F02/H14 - Editar features do projeto (Fácil, Fagner)	F02/H14 - Editar features do projeto (Fácil, Fagner)
		F02/H15 - Listar features do projeto (Fácil, Fagner)	F02/H15 - Listar features do projeto (Fácil, Fagner)
		F02/H16 - Excluir features do projeto (Fácil, Fagner)	F02/H16 - Excluir features do projeto (Fácil, Fagner)
		F02/H17 - Cadastrar funcionalidades do projeto relacionado a *feature* (Fácil, Fagner)	F02/H17 - Cadastrar funcionalidades do projeto relacionado a *feature* (Fácil, Fagner)
		F02/H18 - Editar funcionalidades do projeto relacionado a *feature* (Fácil, Fagner)	F02/H18 - Editar funcionalidades do projeto relacionado a *feature* (Fácil, Fagner)
		F02/H19 - Listar funcionalidades do projeto relacionado a *feature* (Fácil, Fagner)	F02/H19 - Listar funcionalidades do projeto relacionado a *feature* (Fácil, Fagner)
		F02/H20 - Excluir funcionalidades do projeto relacionado a uma *feature* (Fácil, Fagner)	F02/H20 - Excluir funcionalidades do projeto relacionado a uma *feature* (Fácil, Fagner)
	F06 - Procedimentos de teste	F06/H03 - Geração de casos de testes (Difícil, Filipe)	F06/H03 - Geração de casos de testes (Difícil, Filipe)

- Observação: Percebeu-se, através da avaliação de documentos do órgão, que as informações que deveriam ser estruturadas na estória F02/H01 - Estruturar informações obtidas estavam maiores do que o previsto. Portanto esta estória foi quebrada em estórias menores (F02/H05 até F02/H29), e a F02/H01 passou a se chamar Estruturação dos Requisitos (Rastreabilidade).

- Planejamento: Manteve-se a parcialmente concluída na *Sprint* passada, com a inclusão da maioria das estórias geradas pela quebra da antiga F02/H01.

- Execução: Todas as estórias planejadas foram concluídas.

- Avaliação: A *Sprint* teve um aproveitamento de 100 %, devido a quebra da antiga F02/H01 e também devido a boa maturidade no ritmo de desenvolvimento.

Tabela 12 – Planejamento da *Sprint* 8

	***Features* Envolvidas**	**Planejado**	**Concluído**
Sprint 8	F02 - Elicitar requisitos junto ao cliente	F02/H01 - Estruturação dos requisitos (rastreabilidade) (Fácil, Fagner)	F02/H01 - Estruturação dos requisitos (rastreabilidade) (Fácil, Fagner)
		F02/H21 - Cadastrar requisitos não-funcionais do projeto (Fácil, Fagner)	F02/H21 - Cadastrar requisitos não-funcionais do projeto (Fácil, Fagner)
		F02/H22 - Editar requisitos não-funcionais do projeto (Fácil, Fagner)	F02/H22 - Editar requisitos não-funcionais do projeto (Fácil, Fagner)
		F02/H23 - Listar requisitos não-funcionais do projeto (Fácil, Fagner)	F02/H23 - Listar requisitos não-funcionais do projeto (Fácil, Fagner)
		F02/H24 - Excluir requisitos não-funcionais do projeto (Fácil, Fagner)	F02/H24 - Excluir requisitos não-funcionais do projeto (Fácil, Fagner)
		F02/H25 - Cadastrar regras de negócio do projeto (Fácil, Fagner)	F02/H25 - Cadastrar regras de negócio do projeto (Fácil, Fagner)
		F02/H26 - Editar regras de negócio do projeto (Fácil, Fagner)	F02/H26 - Editar regras de negócio do projeto (Fácil, Fagner)
		F02/H27 - Listar regras de negócio do projeto (Fácil, Fagner)	F02/H27 - Listar regras de negócio do projeto (Fácil, Fagner)
		F02/H28 - Excluir regras de negócio do projeto (Fácil, Fagner)	F02/H28 - Excluir regras de negócio do projeto (Fácil, Fagner)
		F02/H29 - Apresentar rastreabilidade entre projeto, *features* e funcionalidades (Médio, Fagner)	F02/H29 - Apresentar rastreabilidade entre projeto, *features* e funcionalidades (Médio, Fagner)
	F09 Projeto Kanban (Gerência de projetos)	F09/H01 - Cadastrar projeto (Fácil, Fagner)	F09/H01 - Cadastrar projeto (Fácil, Fagner)
		F09/H02 - Editar projeto (Fácil, Fagner)	F09/H02 - Editar projeto (Fácil, Fagner)
		F09/H03 - Exibir/listar projetos (Fácil, Fagner)	F09/H03 - Exibir/listar projetos (Fácil, Fagner)

- Observação: Observou-se pela forma de desenvolvimento do órgão (por projeto) que não faria sentido existir requisitos sem estarem associados a um projeto. Portanto

foram incluídas novas estórias (F09/H01, F09/H02 e F09/H03) relacionadas ao cadastro de um projeto.

- Planejamento: Foram incluídas o restante das estórias geradas pela quebra da antiga F02/H01 na *Sprint* passada, além da inclusão das novas estórias observadas.
- Execução: Todas as estórias planejadas foram concluídas.
- Avaliação: A *Sprint* teve um aproveitamento de 100 % devido a um melhor esclarecimento da solução e a boa maturidade no ritmo de desenvolvimento.

Tabela 13 – Planejamento da *Sprint* 9

	***Features* Envolvidas**	**Planejado**	**Concluído**
Sprint 9	F01 - Escolher técnica de elicitação de requisitos	F01/H01 - Indicar técnica de elicitação com base em perguntas sobre os parâmetros do projeto (Difícil, Fagner)	F01/H01 - Indicar técnica de elicitação com base em perguntas sobre os parâmetros do projeto (Difícil, Fagner)
	F03 - Construir modelo de dados	F03/H06 - Atualizar modelo de dados (Fácil, Fagner)	F03/H06 - Atualizar modelo de dados (Fácil, Fagner)
	F03 - Construir modelo de dados	F03/H07 - Cadastrar modelo de dados e informações referentes (Fácil, Fagner)	F03/H07 - Cadastrar modelo de dados e informações referentes (Fácil, Fagner)
	F03 - Construir modelo de dados	F03/H08 - Excluir modelo de dados (Fácil, Fagner)	F03/H08 - Excluir modelo de dados (Fácil, Fagner)
	F07 - Tratamento de erros	F07/H02 - Cadastro de erros conhecidos (Médio, Filipe)	

- Observação: Foi observado que a estória F03/H05 - Validar modelo de dados dependia de um prévio cadastro do modelo. Portanto foram criadas as estórias F03/H06 até F03/H08, de forma a contemplar este cadastro.
- Planejamento: Houve a inclusão das estórias previamente criadas (F03/H06 até a F03/H08), além da estória F07/H02, que estava relacionada com uma possível base de conhecimento de erros.
- Execução: Das estórias planejadas, somente uma não foi concluída.
- Avaliação: A *Sprint* teve um bom desempenho, tendo sido apenas uma estória não concluída, devido ao seu maior grau de complexidade em comparação com as outras concluídas.

Tabela 14 – Planejamento da *Sprint* 10

Sprint 10	***Features* Envolvidas**	**Planejado**	**Concluído**
	F03 - Construir modelo de dados	F03/H05 - Validar modelo de dados (Difícil, Fagner)	F07/H02 - Cadastro de erros conhecidos (Médio, Filipe)
	F07 - Tratamento de erros	F07/H02 - Cadastro de erros conhecidos (Médio, Filipe)	

- Observação: Observou-se, após discussões entre os membros, de que a estória F03/H05 - Validar modelo de dados possuía mais dependências do que o previsto na *Sprint* passada, como por exemplo a diferenciação entre os perfis de usuário, que solicita uma avaliação, e responsável, que realiza uma avaliação solicitada.
- Planejamento: Foi incluído a estória F03/H05 devido as estórias das quais dependiam e que foram previstas na *Sprint* passada terem sido previamente concluídas.
- Execução: Das estórias planejadas, somente uma não foi concluída.
- Avaliação: A estória F03/H05 apresentou uma dificuldade maior do que a prevista, o que não permitiu que ela fosse concluída.

4 Resultados

4.1 *Considerações Iniciais*

Neste capítulo são apresentadas as atividades e os guias selecionados, resultantes tanto do diagnóstico obtido com as entrevistas quanto do processo de desenvolvimento descentralizado e da *Wiki*. Em seguida o modelo de solução gerado é apresentado. Por fim o sistema resultante é relatado através de uma breve descrição das telas mais representativas do sistema, de uma comparação da implementação realizada com o diagnóstico obtido, e de uma descrição do estado da implementação, com detalhes do que foi implementado e do que falta ser implementado para a finalização do sistema.

4.2 Resultado do Diagnóstico

O diagnóstico obtido, conforme a entrevista aplicada aos desenvolvedores que participam do desenvolvimento descentralizado (apêndice B) e a análise descrita na seção 4.3, consiste de um conjunto de problemas, sugestões e boas práticas relatados pelos desenvolvedores. A primeira parte da análise, que consistiu em aplicar os códigos às respostas obtidas, pode ser visualizada na Tabela 15. A Tabela 15 apresenta a rastreabilidade dos códigos em relação as respostas de cada desenvolvedor para cada pergunta. As respostas que faziam parte do código Perfil do Desenvolvedor, não puderam ser associadas a nenhum dos outros códigos e foram descartadas bem como o próprio código, que é o caso das perguntas P1, P2, P3, P4 e P5 que faziam parte do código perfil do desenvolvedor, posteriormente analisou-se esse código com o conjunto de perguntas e verificou-se não fazer parte da intenção da abordagem para a solução, pois trata-se apenas da contextualização do perfil dos entrevistados. Assim, os códigos nesta matriz são representados pelos símbolos Cox em que x é a numeração para contagem, e cada código ou mais é referenciado na resposta do desenvolvedor entrevistado em uma determinada pergunta.

A seguir podemos verificar os códigos que representam uma determinada categoria ou fase de desenvolvimento e a Tabela 15 de rastrebilidade dos códigos:

- Co1 - Papel do Desenvolvedor;
- Co2 - Preparação;
- Co3 - Elicitação de Requisitos;
- Co4 - Gerenciamento de Requisitos;

- Co5 - Modelagem do sistema;
- Co6 - Depuração;
- Co7 - Estilo e *Design* de Codificação;
- Co8 - Ambiente de Codificação;
- Co9 - Teste;
- Co10 - Implantação;

Tabela 15 – Matriz de rastreabilidade dos códigos

	Des.1	**Des.2**	**Des.3**	**Des.4**	**Des.5**	**Des.6**	**Des.7**	**Des.8**
P6	Co1	Co1	Co1	Co1	Co1	Co1	Co1	Co1
P7	Co1	Co1	Co1	Co1	Co1	Co1	Co1	Co1
P8	Co2	Co2	Co2	Co2	Co2	Co2	Co2	Co2
P9	Co2	Co2	Co2	Co2	Co2	Co2	Co2	Co2
P10	Co2, Co7	Co2, Co7	Co7	Co2, Co7	Co2, Co7	Co7	Co2, Co7	Co7
P11	Co7,Co8	Co7	Co7	Co7	Co7	Co7	Co7	Co7
P12	Co7	Co2	Co7	-	Co2,Co7	Co7	Co7	Co7
P13	Co3	Co3	Co3	Co3	Co3	Co3	Co3	Co3
P14	Co3	Co3	Co3	Co3	Co3	Co3	Co3,Co4	Co3,Co4
P15	Co4	Co4	Co4	Co4	Co4	Co4	Co4	Co4
P16	Co4	Co4	Co4	Co4	Co3	Co3,Co4	Co4	-
P17	Co5	Co5	Co5	Co5	Co5	Co5	Co5	Co5
P18	Co5	Co5	Co5	Co5	Co5	Co5	Co5	Co5
P19	Co7	Co7	Co7	Co7	Co7	Co7	Co7	Co7
P20	Co7	Co7	Co7	Co7	Co7	Co7	Co7	Co7
P21	Co7	Co7	Co7	Co7	Co7	Co7	Co7	Co7
P22	Co8	Co8	Co8	Co8	Co8	Co8	Co8	Co8
P23	Co6	Co6	Co6	Co6	Co6	Co6	Co6	Co6
P24	Co6	Co6	Co6	Co6	Co6	Co6	Co6	Co6
P25	Co7	Co7	Co7	Co7	Co7	Co7	Co7	Co7
P26	Co9	Co9,Co10	Co9	Co9	Co9	Co9	Co9	Co9
P27	Co9	Co9	Co9	Co9	Co9	Co9	Co9	Co9
P28	Co9	Co9	Co9	Co9	Co9	Co9	Co9	Co9
P29	Co9	Co9	Co9	Co9	Co9	Co9	Co9	Co9
P30	Co9	Co9	Co9	Co9	Co9	Co9	Co9	Co9
P31	Co10	Co10	Co10	Co9,Co10	Co10	Co10	Co10	Co10
P32	Co10	Co10	Co10	Co10	Co10	Co10	Co10	Co10
P33	Co9,Co10	Co9,Co10	Co9,Co10	Co9,Co10	Co10	Co10	Co9	Co9
P34	Co10	Co10	Co10	Co10	Co10	Co10	Co10	Co10
OBS	Co3,Co5, Co10	Co9,Co10	Co9	Co4,Co9	-	-	-	-

Uma vez que a primeira parte da análise foi aplicada, deu-se prosseguimento à próxima fase da análise. A segunda parte consistiu em rotular as respostas, que foram agrupadas pelos códigos, com as chaves. A Tabela 16 ilustra as chaves atribuídas às repostas das entrevistas e os seus significados. Estas chaves representam todos os diferentes significados de todas as respostas.

Tabela 16 – Chaves representando os diferentes significados das respostas das entrevistas

Código	**Chave**	**Descrição**
Co1 - Papel do desenvolvedor	C1	Papel claro dentro da instituição
	C2	Papel que exige grande responsabilidade dentro da instituição
	C3	Papel definido com falta de metas
	C4	Contribuição relevante para a instituição
Co2 - Preparação	C5	Curso dá somente uma introdução
	C6	O curso poderia ser melhorado
	C7	Algumas aplicações com baixa qualidade
	C8	Construção de um catálogo de informações APEX
	C9	Treinamento em boas práticas APEX
	C10	Esclarecer como funciona o sistema de desenvolvimento descentralizado
Co3 - Elicitação de Requisitos	C11	Os requisitos são levantados através de reuniões
	C12	Requisitos óbvios não anotados durante as reuniões são esquecidos
	C13	Uso de prototipagem para levantar/validar requisitos
Co4 - Gerenciamento de Requisitos	C14	Registro de requisitos em papel
	C15	Registro de requisitos em sistemas distintos em rede
	C16	Registro de requisitos inexistente
	C17	Reuniões constantes
Co5 - Modelagem do Sistema	C18	Não atualiza o modelo de dados
	C19	Não valida o modelo de dados com área técnica
	C20	Possibilidade de gerar o modelo de dados a partir das tabelas
	C21	Negligência ao modelo de dados
	C22	Valida o modelo de dados com área técnica

Código	**Chave**	**Descrição**
	C23	Elabora o modelo de dados utilizando ferramentas CASE
Co6 - Depuração	C24	Realiza depurações no sistema com o APEX
	C25	Realiza depurações no console de comandos SQL
	C26	Não realiza depurações
	C27	Não há classificação de erros
Co7 - Estilo e *Design* de Codificação	C28	Falta do uso de um padrão de boas práticas comprometeu a qualidade da aplicação
	C29	Escrita legível de código
	C30	Mais da metade da aplicação é composta de algum código
	C31	Preferência pelo uso de componentes padrões do APEX
	C32	Aplicação bem desenvolvida
	C33	Boas práticas: Nomes significativos, uso de comentários, identação, padrão de nomenclatura, desenvolver pensando nas manutenções futuras, sempre usar filtros nas consultas SQL, legibilidade do código e uso de componentes padrões do APEX sempre que possível
	C34	Falta de um padrão de boas práticas
Co8 - Ambiente de Codificação	C35	Reutilização de código
	C36	Uso de editores externos
	C37	Uso do editor do APEX
Co9 - Teste	C38	Uso de testes funcionais
	C39	Uso de teste de comandos SQL
	C40	Teste de execução de páginas
	C41	Sem armazenamento dos resultados de teste
	C42	Não há elaboração de casos de teste
	C43	Favorável à proposta de automação de testes funcionais
Co10 - Implantação	C44	Costuma verificar se os apelidos não estão duplicados
	C45	Não migra as definições de objetos
	C46	Navegação feita, antes de migrar, somente nas páginas que alterou

Código	Chave	Descrição
	C47	Navegação feita, antes de migrar, em todas as páginas da aplicação
	C48	Verifica a criação das tabelas no espaço de produção
	C49	Faz a migração da definição dos objetos de dados
	C50	Faz a homologação das aplicações antes de disponibilizar definitivamente na produção
	C51	Seria interessante o uso de uma ferramenta que mostrasse as diferenças dos objetos no desenvolvimento e na produção

4.3 Solução - Criação do Ambiente de Apoio ao Desenvolvedor EUD

Ao total foi obtido um conjunto com 51 chaves diferentes, sendo que as chaves representam o diagnóstico do desenvolvimento descentralizado do órgão público de estudo. O diagnóstico foi utilizado para a proposição de soluções que aperfeiçoassem este desenvolvimento. Para facilitar este procedimento, as chaves que tinham alguma relação direta em comum, foram agrupadas e este grupo serviu de base para as soluções propostas. Essas soluções foram definidas a partir da experiência de convívio em meio ao processo desenvolvimento descentralizado de sistemas dentro do órgão, bem como do modelo, não institucionalizado, de processo de desenvolvimento descentralizado do órgão, e da *Wiki* da unidade coordenadora. Dessa forma, foi gerado como solução um conjunto de atividades, guias de apoio e funcionalidades para apoiar o desenvolvimento de sistemas, e este constitui os requisitos de um processo/guia a ser implementado na ferramenta atualmente utilizada pelo órgão, no caso, o APEX. A Tabela 17 ilustra as soluções propostas para cada grupo de chaves. Algumas chaves não obtiveram uma solução relacionada por serem meramente uma descrição ou pela inviabilidade da solução necessária.

Tabela 17 – Soluções associadas as chaves

Chave	**Solução**
C1	Não se aplica, pois é uma descrição.
C2	Não se aplica, pois é uma descrição.
C3	Não aplicável ao escopo do trabalho.
C4	Não se aplica, pois é uma descrição.
C5, C6 e C9	1- Montar uma *Wiki* complementar, dentro da aplicação, de forma que as informações da solução possam ser consultadas de forma mais conveniente.
C7	1- Definir ou usar padrões e boas práticas existentes em APEX. 2- Definir procedimentos de teste. 3- Definir procedimentos de implantação.
C8	1- Mesma solução das chaves C5, C6 e C9.
C10	1- Elaborar um guia de como funciona desenvolvimento descentralizado no órgão.
C11, C12, C13, C14, C15, C16, C17	1- Adaptar a atividade de priorização do escopo da *release* no processo de desenvolvimento descentralizado já existente. 2- Adaptar a atividade de levantamento de requisitos no processo de desenvolvimento descentralizado já existente. 3- Usar o guia de entrevistas definidos no processo de desenvolvimento descentralizado já existente. 4- Usar a tarefa de estruturar as informações obtidas do processo de desenvolvimento descentralizado. 5- Propor um tutorial sobre prototipagem rápida. 6- Desenvolver uma funcionalidade que sugira ao usuário uma técnica de elicitação de requisitos com base no seu perfil.
C18, C19, C20, C21, C22 e C23	1- Definir uma atividade para atualizar o modelo de dados existente. 2- Adaptar a tarefa de construir o modelo lógico de dados do processo de desenvolvimento descentralizado. 3- Definir uma atividade para a validação do modelo de dados. 4- Elaborar guia de geração de scripts SQL a partir de modelo lógico implementado na ferramenta Oracle SQL Data Modeler. 5- Elaborar um tutorial de criação do banco de dados usando a ferramenta Oracle SQL Data Modeler e padrões de nomenclatura do órgão.
C24, C25, C26 e C27	1- Tutorial de uso da funcionalidade de depuração do APEX. 2- Dicas de como depurar scripts PL/SQL. 3- Formulário para cadastro de erros.

Chave	**Solução**
C28, C29, C30, C31, C32, C33, C34, C35, C36 e C37	1- Desenvolver um guia de boas práticas de desenvolvimento APEX. 2- Desenvolver um guia de boas práticas sobre PL/SQL. 3- Disponibilizar o padrão de nomenclaturas de objetos de banco definidos pelo órgão público de estudo. 4- Formulário para desenvolvedor poder salvar código que ele ache que pode ser reutilizável. 5- Guia sobre o uso de editores de código SQL e PL/SQL.
C38, C39, C40, C41, C42 e C43	1- Criar uma funcionalidade de geração de casos de teste simples. 2- Guia de como testar o comportamento dos códigos PL/SQL. 3- Guia de como testar uma página APEX. 4- Guia de utilização de ferramenta de automação de teste.
C44, C45, C46, C47, C48, C49 e C50	1- Guia de migração de aplicações. 2- Guia de homologação de aplicações. 3- Propor uma atividade de homologação da aplicação.
C51	Inviável.

Uma matriz de rastreabilidade das chaves foi elaborada, de forma a garantir a associação entre as chaves e as respostas, dadas pelos desenvolvedores, para cada pergunta. O conjunto de respostas das perguntas de 1 a 5, por se tratar meramente de uma contextualização do perfil dos entrevistados, não fez parte do escopo da análise. A Tabela 18 ilustra a rastreabilidade de chaves com as respostas das perguntas.

Tabela 18 – Matriz de rastreabilidade das chaves

	Des.1	**Des.2**	**Des.3**	**Des.4**	**Des.5**	**Des.6**	**Des.7**	**Des.8**
P6	C1	C3	C1	C1	C2	C2	C1	C2
P7	C4	C4	C4	C4	C4	C4	C4	C4
P8	C5	C5	C5	C5	C5	-	-	C6
P9	C6	C6	C6	C6	C6	-	-	C6
P10	C7,C28	C7,C32	C28	C7,C28	C9,C32	C32	C9,C29, C32	C32
P11	C33,C35	C33	C33	C33	C31	C34	C34	C34
P12	C33	C8	C33	-	C9,C10, C33	C33	C29	C31
P13	C11	C11	C11	C11	C11	C11,C13	C11	C11
P14	C11	C11	C11	C11	C11	C11	C11	C13
P15	C14,C15	C14	C15	C14,C15	C15	C15	C15	C15
P16	C17	C17	C17	C17	C11,C17	C11,C17	C17	C17
P17	C18	C18	C21	C21	C23	C23	C23	C23
P18	C19	C19	C22	C22	C19	C19	C22	C22
P19	C32	C30	C30	C30	C30	C30,C31	C30	C30
P20	C29	C29	C33	C33	C29	C29	C28	C29
P21	C33	C33	C33	C33	C34	C33	C28	C33
P22	C36	C36	C36,C37	C36	C36	C36	C37	C37
P23	C24	C24	C26	C25	C26	C24	C26	C26
P24	C27	-	C27	C27	C27	C27	C27	C27
P25	C33	C33	C33	C33	C33	C34	C29	C29
P26	C38	C38,C50	C40	C38,C39	C38	C38	C42	C38
P27	C40	C27,C40	C40	C40	C40	C40	C42	C40
P28	C41	C41	C41	C41	C41	C41	C42	C41
P29	C42	C42	C42	C42	C42	C42	C42	C42
P30	C43	C43	C43	C43	C43	-	C43	C43
P31	C45	-	C49	C49	C49	C49	-	C49
P32	C44	C44	C44	C44	C44	C44	-	C44
P33	C40,C46	C40,C46	C40,C47	C40,C47	C47	C47	C40	C40
P34	C48	C48	C48	C48	C48	C48	C48	C48
OBS	C12,C20, C51	C40,C50	C40	C16,C38, C39,C40	-	-	-	-

Uma vez que as soluções para o diagnóstico foram propostas, elas foram convertidas em atividades, em funcionalidades e em guias que compõe a solução de apoio. Estas atividades, funcionalidades e guias são divididas em três tipos conforme a sua origem:

1. Propostas;
2. Retiradas do processo de desenvolvimento descentralizado;
3. Retiradas da *Wiki*

A Tabela 19 representa as atividades e os guias gerados por todas as soluções propostas e a fonte de origem das mesmas, divididos pelas fases do ciclo de vida da solução.

Tabela 19 – Atividades e guias resultantes da solução

Solução	**Atividade, Guia, Funcionalidade**	**Fonte**	**Fase do ciclo de vida da solução**
Adaptar a atividade de priorização do escopo da *release* no processo de desenvolvimento descentralizado já existente.	Atividade: Priorizar funcionalidades a serem desenvolvidas	Solução Proposta	Requisitos
Adaptar a atividade de levantamento de requisitos no processo de desenvolvimento descentralizado já existente.	Atividade: Elicitar requisitos junto ao cliente	PDESC	
Usar o guia de entrevistas definidos no processo de desenvolvimento descentralizado já existente.	Guia: Guia de entrevistas	PDESC	
Usar a tarefa de estruturar as informações obtidas do processo de desenvolvimento descentralizado.	Atividade: Estruturar informações obtidas	PDESC	
Propor um tutorial sobre prototipagem rápida.	Guia: Guia de prototipagem rápido	Solução Proposta	

Solução	**Atividade, Guia, Funcionalidade**	**Fonte**	**Fase do ciclo de vida da solução**
Desenvolver uma funcionalidade que sugira ao usuário uma técnica de elicitação de requisitos com base no seu perfil.	Funcionalidade: Escolher técnica de elicitação	Solução Proposta	
Definir uma atividade para atualizar o modelo de dados existente.	Atividade: Atualizar modelo de dados	Solução Proposta	Banco de dados
Adaptar a tarefa de construir o modelo lógico de dados do processo de desenvolvimento descentralizado.	Atividade: Construir modelo de dados	PDESC	
Definir uma atividade para a validação do modelo de dados.	Atividade: Validar modelo de dados	Solução Proposta	
Disponibilizar padrão de nomenclaturas de objetos de banco definidos pelo órgão público de estudo.	Guia: Guia de padrão de nomenclatura de objetos de dados	*Wiki*	
Elaborar guia de geração de scripts SQL a partir de modelo lógico implementado na ferramenta Oracle SQL Data Modeler.	Guia: Guia de geração de scripts sql	Solução Proposta	
Elaborar um tutorial de criação do banco de dados usando a ferramenta Oracle SQL Data Modeler e padrões de nomenclatura do órgão.	Guia: Guia de criação e padrões de BD	Solução Proposta	
Montar uma espécie WIKI complementar, dentro da aplicação, de forma que as informações da solução possam ser consultadas de forma mais conveniente.	Funcionalidade: Consulta de informações na solução	Solução Proposta	Implementação
Desenvolver um guia de boas práticas de desenvolvimento APEX.	Guia: Guias de padrões e boas práticas APEX	Solução Proposta	

Solução	**Atividade, Guia, Funcionalidade**	**Fonte**	**Fase do ciclo de vida da solução**
Definir ou usar padrões e boas práticas existentes em APEX.			
Desenvolver um guia de boas práticas sobre PL/SQL.	Guia: Guia de boas práticas plsql	Solução Proposta	
Formulário para desenvolvedor poder salvar código que ele ache que pode ser reutilizável.	Funcionalidade: Cadastro de código reutilizável.	Solução Proposta	
Guia sobre o uso de editores de código SQL e PL/SQL.	Guia: Guia sobre editor de código sql e plsql	Solução Proposta	
Definir procedimentos de teste.	Atividade: Definir procedimentos de teste	Solução Proposta	Teste
Tutorial de uso da funcionalidade de depuração do APEX.	Guia: Guia de uso de depuração de páginas APEX	Solução Proposta	
Dicas de como depurar scripts PL/SQL.	Guia: Guia de teste de comportamento de código	Solução Proposta	
Formulário para cadastro de erros.	Funcionalidade: Cadastro de erros conhecidos	Solução Proposta	
Criar uma funcionalidade de geração de casos de teste simples.	Funcionalidade: Geração de casos de teste	Solução Proposta	
Guia de como testar o comportamento dos códigos PL/SQL.	Guia: Guia de teste de comportamento de código	Solução Proposta	

Solução	**Atividade, Guia, Funcionalidade**	**Fonte**	**Fase do ciclo de vida da solução**
Guia de como testar uma página APEX.	Guia: Guia de teste de página APEX	Solução Proposta	
Guia de utilização de ferramenta de automação de teste.	Guia: Guia de ferramenta de automação de teste	Solução Proposta	
Guia de homologação de aplicações.	Guia: Guia de homologação de aplicações	Solução Proposta	Homologação
Propor uma atividade de homologação da aplicação.	Atividade: Homologar aplicação	Solução Proposta	
Guia de migração de aplicações.	Guia: Guia de migração de aplicações	Solução Proposta	Implantação
Definir procedimentos de implantação.	Atividade: Definir procedimentos de implantação	Solução Proposta	

4.3.1 Atividades e guias complementares

Apesar de algumas atividades e guias do processo de desenvolvimento descentralizado e da *Wiki*, terem sido selecionadas para compor as soluções propostas a partir do diagnóstico, houve a necessidade de se selecionar outras atividades e guias que não foram previstas nas soluções elaboradas. As justificativas para esse complemento foram:

1. Alguns guias ficaram sem atividades para serem associadas;

2. Algumas atividades, julgadas importantes, não foram previstas nas soluções do diagnóstico;

Em relação ao processo de desenvolvimento existente, foram selecionadas, ao todo, 10 atividades e nenhum guia, de forma a complementar a solução. Já em relação a *Wiki*, nenhuma atividade e guia foi selecionados para o complemento. A Tabela 20 ilustra a

relação de atividades selecionadas do processo de desenvolvimento descentralizado, e a sua associação com as fases do ciclo de vida da solução proposta.

Tabela 20 – Atividades complementares selecionadas

Atividade	**Fonte**	**Fase do ciclo de vida da solução**
Construir modelo físico de dados -> Construir modelo de dados	PDESC	Banco de dados
Gerar esqueleto da aplicação	PDESC	Implementação
Ajustar interface do sistema	PDESC	
Aplicar regras de negócio	PDESC	
Executar testes de desenvolvedor	PDESC	Teste
Executar testes exploratórios	PDESC	
Realizar correções no sistema	PDESC	
Preparar ambiente de teste	PDESC	
Criar material de apoio ao usuário	PDESC	Homologação
Migrar aplicação	PDESC	Implantação

4.3.2 Modelo da Solução Gerada

A solução gerada consiste da junção de todas as atividades, guias e funcionalidades geradas, tanto do diagnóstico, quanto do complemento. Essas atividades, guias e funcionalidades, estão organizadas em 5 fases diferentes do ciclo de vida. Estas fases são: Requisitos, Desenvolvimento, Teste, Homologação e Implantação. A fase de Desenvolvimento, é dividida em duas sub-fases: Banco de dados e Implementação. A organização do ciclo de vida da solução é representada pela Figura 8.

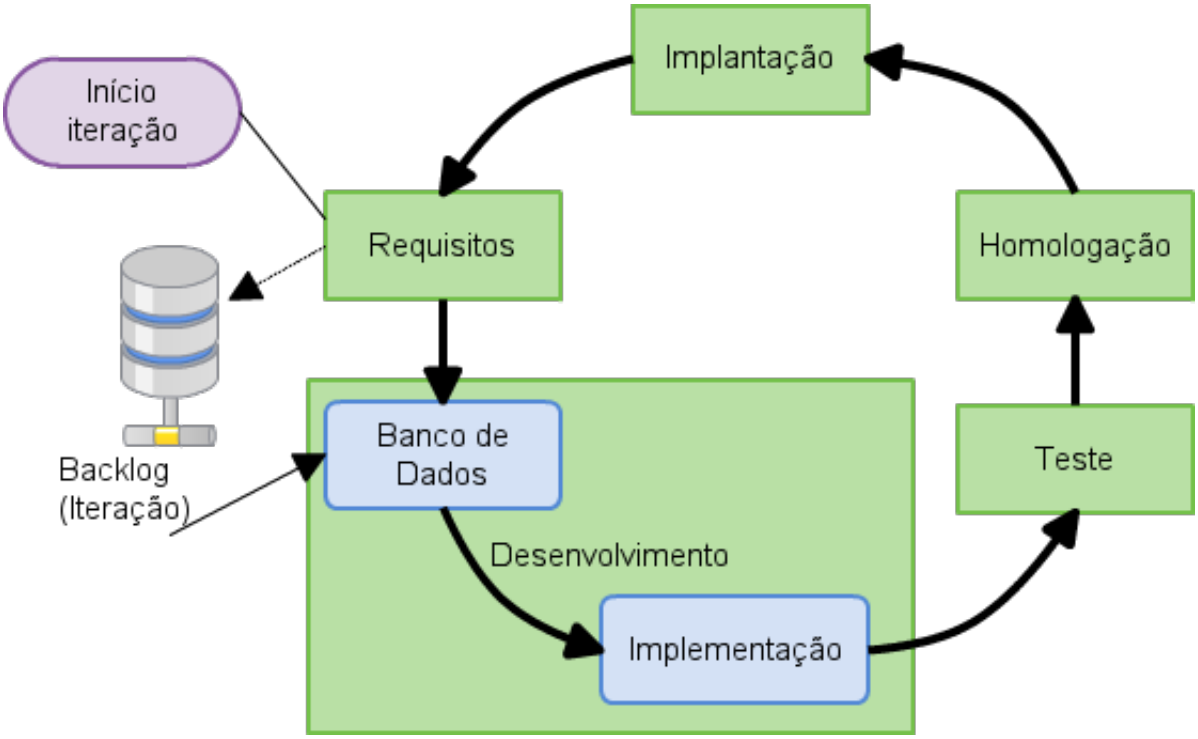

Figura 8 – Ciclo de vida da solução.

A organização do ciclo de vida foi baseada na metodologia ágil *Scrum*. A idéia básica desta organização é: definir o *backlog* de um ciclo (*backlog* da iteração) junto com o cliente, e então ir "consumindo"esse *backlog* ao longo de todo o ciclo da solução, onde cada item do *backlog* definido deverá caminhar sequencialmente pelas fases do ciclo de vida. Uma vez que todo o *backlog* foi "consumido"pelo ciclo de vida, temos o fim de um ciclo ou iteração. Ao completar uma iteração, um novo *backlog* é definido com o cliente, e então um novo ciclo se inicia. Esse fluxo se repete até que o sistema chegue ao nível desejado pelo cliente. As fases do ciclo de vida são descritas da seguinte forma:

- **Requisitos**: Priorizar as funcionalidades a serem desenvolvidas em um ciclo (*backlog* da iteração) de acordo com os requisitos levantados junto com o cliente.
- **Desenvolvimento**: Dividida pelas sub-fases Banco de dados e Implementação, esta fase tem o objetivo de realizar a construção do sistema. A sub-fase Banco de dados tem o objetivo de construir o modelo de dados, tanto lógico quanto físico. Já a sub-fase Implementação objetiva implementar de fato o sistema, baseado no modelo de dados construído.
- **Teste**: Realizar os testes, tanto a nível de páginas da aplicação, quanto a nível de código, objetivando corrigir erros encontrados.
- **Homologação**: Homologar a aplicação junto ao cliente e realizar a construção de materiais de apoio ao usuário, como guias, manuais, etc. Caso não seja homologada

pelo cliente, o fluxo do ciclo volta para a fase que necessita de correções e passa por todas as etapas subsequentes, repetindo o ciclo.

- **Implantação**: Planejar e realizar a migração do sistema para a produção.

Ao todo 18 atividades, 14 guias e 5 funcionalidades compõem a solução, onde os guias e as funcionalidades são suportes à execução de uma determinada atividade. A Figura 9 ilustra a solução elaborada, agrupando todas as atividades, as funcionalidades e os guias, pelas fases do ciclo de vida. Os retângulos azuis representam as atividades e os retângulos roxos com laterais curvas representam os guias e as funcionalidades. Um guia ou uma funcionalidade representado após uma atividade significa que ele é um suporte a atividade em questão.

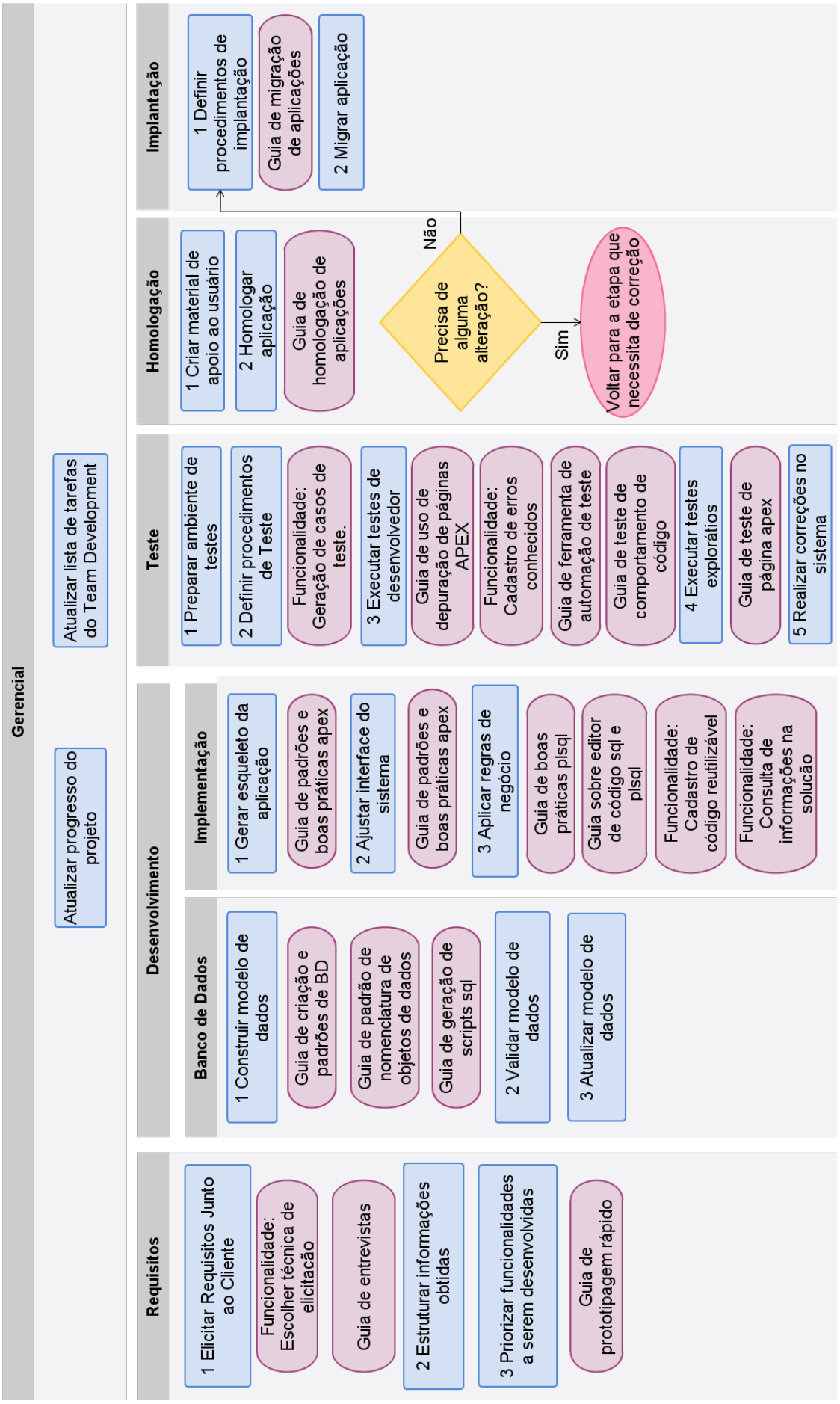

Figura 9 – Conjunto de atividades, guias e funcionalidades, por fase do ciclo de vida

Nesta solução, o ponto de decisão na fase de homologação, representado pelo losango verde, indica se a execução do fluxo irá continuar ou se irá voltar a alguma atividade dentro de uma determinada fase. Esta atividade dependerá das solicitações feitas pelo cliente durante a atividade de "Homologar aplicação".

As atividades, guias e funcionalidades são descritas brevemente da seguinte forma:

- **Requisitos**
  1. *Elicitar Requisitos Junto ao Cliente (Atividade):* Identificar os requisitos funcionais e não funcionais da aplicação junto ao cliente.
     a) *Escolher técnica de elicitação (Funcionalidade):* Funcionalidade para ajudar o desenvolvedor a escolher uma técnica de elicitação para o levantamento de requisitos junto ao cliente. Uma técnica existente pré-cadastrada será sugerida ao usuário com base nas respostas de algumas perguntas.
     b) *Guia de entrevistas (Guia):* Guia de como elaborar perguntas para uma entrevista com o cliente. Este guia será útil somente para quem for utilizar a entrevista como técnica de elicitação.
  2. *Estruturar informações obtidas (Atividade):* Transformar as informações obtidas na entrevista em requisitos, e armazená-los em documento.
  3. *Priorizar funcionalidades a serem desenvolvidas (Atividade):* Priorizar as funcionalidades a serem desenvolvidas dentro de um ciclo da solução.
     a) *Guia de prototipagem rápido (Guia):* Guia de como realizar uma prototipagem rápida das telas de uma aplicação.
- **Desenvolvimento**
  - **Banco de Dados**
    1. *Construir modelo de dados (Atividade):* Realizar a construção do modelo de dados lógico e físico.
       a) *Guia de criação e padrões de BD (Guia):* Guia com dicas de como realizar a modelagem conceitual (MER) e lógica (ML) do banco de dados da aplicação, utilizando a ferramenta Oracle Data Modeler. Utiliza o guia de padrão de nomenclatura de objetos de dados no auxílio da construção dos modelos.
       b) *Guia de padrão de nomenclatura de objetos de dados (Guia):* Guia contendo os padrões de nomenclatura usados no órgão para diversos objetos do banco de dados, como tabelas, sequências, gatilhos e *views*.
       c) *Guia de geração de scripts SQL (Guia):* Guia de como gerar os *scripts* para a construção do modelo físico a partir do modelo lógico, elaborado na ferramenta Oracle Data Modeler.

2. *Validar modelo de dados (Atividade):* Realizar a validação do modelo de dados lógico e/ou físico, conforme a necessidade, com a área técnica responsável.
3. *Atualizar modelo de dados (Atividade):* Atualizar o modelo de dados conforme as validações realizadas com a área técnica.

– **Implementação**

1. *Gerar esqueleto da aplicação (Atividade):* Gerar o esqueleto da aplicação usando a funcionalidade *defaults* de interface de usuário do APEX.
   a) *Guia de boas práticas APEX (Guia):* Guia para auxiliar a criação da aplicação usando boas práticas em APEX.
2. *Ajustar interface do sistema (Atividade):* Refinar a interface da aplicação conforme as necessidades de usabilidade da mesma.
   a) *Guia de boas práticas APEX (Guia):* Guia para auxiliar a criação da aplicação, usando boas práticas em APEX.
3. *Aplicar regras de negócio (Atividade):* Implementar as regras de negócio da aplicação.
   a) *Guia de boas práticas PL/SQL (Guia):* Guia para auxiliar a construção de código em PL/SQL.
   b) *Guia sobre editor de código SQL e PL/SQL (Guia):* Guia para auxiliar o uso do editor de código para a escrita de códigos SQL e PL/SQL.
   c) *Cadastro de código reutilizável (Funcionalidade):* Funcionalidade para permitir que o desenvolvedor possa cadastrar códigos que ele julgue ser reutilizáveis.
4. *Consulta de informações na solução (Funcionalidade):* Consulta conveniente das informações da solução, de forma a permitir um acesso mais rápido a um determinado guia ou funcionalidade. O usuário também pode cadastrar suas anotações. A idéia é semelhante a de uma *Wiki*.

• **Teste**

1. *Preparar ambiente de testes (Atividade):* Realizar os preparativos necessários ao ambiente para a realização dos testes da aplicação. Ex: Criar massa de dados, limpar dados existentes, criar um novo espaço de trabalho APEX, etc.
2. *Definir procedimentos de teste (Atividade):* Definir um conjunto de etapas a serem realizadas nos testes.
   a) *Geração de casos de teste (Funcionalidade):* Funcionalidade que auxilie os testes do desenvolvedor, oferecendo diferentes valores que o usuário pode inserir nos diferentes componentes da interface, de forma a testar várias possibilidades diferentes.

3. *Executar testes de desenvolvedor (Atividade):* Executar os testes planejados.
   a) *Guia de uso de depuração de páginas APEX (Guia):* Guia explicando como usar a funcionalidade de depurar existente no APEX, para encontrar erros na aplicação.
   b) *Cadastro de erros conhecidos (Funcionalidade):* Funcionalidade que permita que o desenvolvedor possa cadastrar os erros que for encontrando, de forma a montar uma espécie de base de conhecimento de erros.
   c) *Guia de ferramenta de automação de teste (Guia):* Guia que auxilie o desenvolvedor a criar *scripts* simples de automação de ações no navegador, de forma a criar testes automatizados.
   d) *Guia de teste de comportamento de código (Guia):* Guia que auxilie o desenvolvedor a testar, dentro do possível, o código PL/SQL e a realizar a depuração de códigos em PL/SQL.
4. *Executar testes exploratórios (Atividade):* Navegar na aplicação, sem um prévio planejamento, em busca de erros.
   a) *Guia de teste de página APEX (Guia):* Guia com dicas de como realizar os testes exploratórios.
5. *Realizar correções no sistema (Atividade):* Realizar as correções e ajustes necessários ao sistema.

- **Homologação**

1. *Criar material de apoio ao usuário (Atividade):* Criar um guia que auxilie o usuário a usar a aplicação.
2. *Homologar aplicação (Atividade):* Realizar a homologação da aplicação junto ao cliente.
   a) *Guia de homologação de aplicações (Guia):* Guia com dicas de como realizar a homologação junto ao cliente.

- **Implantação**

1. *Definir procedimentos de implantação (Atividade):* Definir as etapas necessárias para colocar a aplicação em produção.
   a) *Guia de migração de aplicações (Guia):* Guia com dicas do que se preocupar ao realizar a migração de uma aplicação APEX.
2. *Migrar aplicação (Atividade):* Executar os procedimentos de migração definidos.

- **Gerencial**

    1. *Atualizar progresso do projeto (Funcionalidade)*: Atualizar o progresso do projeto no quadro *kanban* desenvolvido como uma funcionalidade da aplicação.
    2. *Atualizar lista de tarefas do Team Development (Atividade)*: Atualizar as tarefas registradas, conforme necessidade, na funcionalidade padrão do APEX, *Team Development*.

O esboço do quadro *kanban* a ser elaborado na funcionalidade gerencial "Atualizar progresso do projeto", de forma a auxiliar o desenvolvedor na gerência do desenvolvimento, é apresentado na Figura 10.

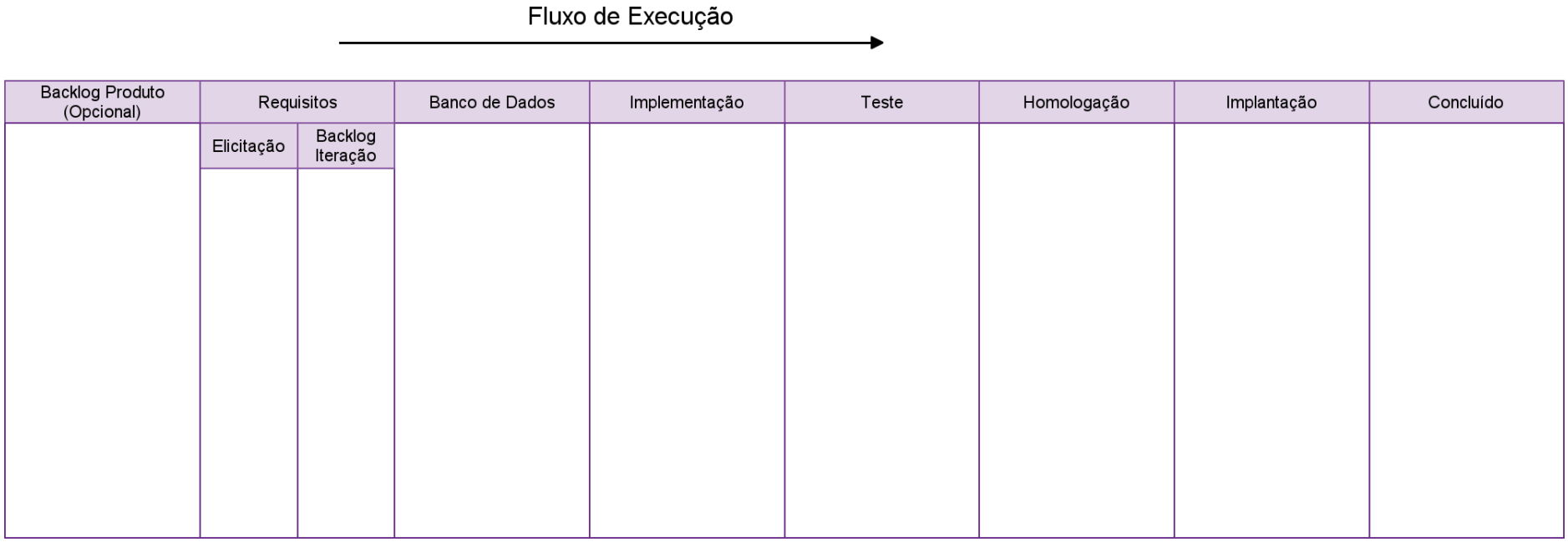

Figura 10 – Adaptação do quadro *kanban*

Este quadro *kanban* foi dividido pelas fases do ciclo de vida da solução, de forma que o desenvolvedor vai posicionando os itens do *backlog* definido de acordo com a etapa em que se encontram dentro do ciclo de vida. A ideia é que o *backlog* da iteração gerado seja em pequenas unidades de funcionalidades, de forma a facilitar o gerenciamento e acompanhamento pelo desenvolvedor. Ao terminar um conjunto destas unidades de funcionalidade, o desenvolvedor consegue gerar um software executável para o cliente. Observa-se que a divisão *Backlog* Produto é opcional, já que pode ser da preferência do cliente e do desenvolvedor levantar o *backlog* do produto. Porém, neste caso, o fluxo de execução do quadro *kanban* obriga a seleção e priorização de um *backlog* da iteração, conforme a divisão da etapa de Requisitos.

4.4 SISADD - Sistema de Apoio ao Desenvolvimento Descentralizado

O SISADD, Sistema de Apoio ao Desenvolvimento Descentralizado, é a implementação da solução gerada, desenvolvido na plataforma Oracle Application Express. As Figuras de 11 a 16 mostram as capturas das telas mais representativas do SISADD.

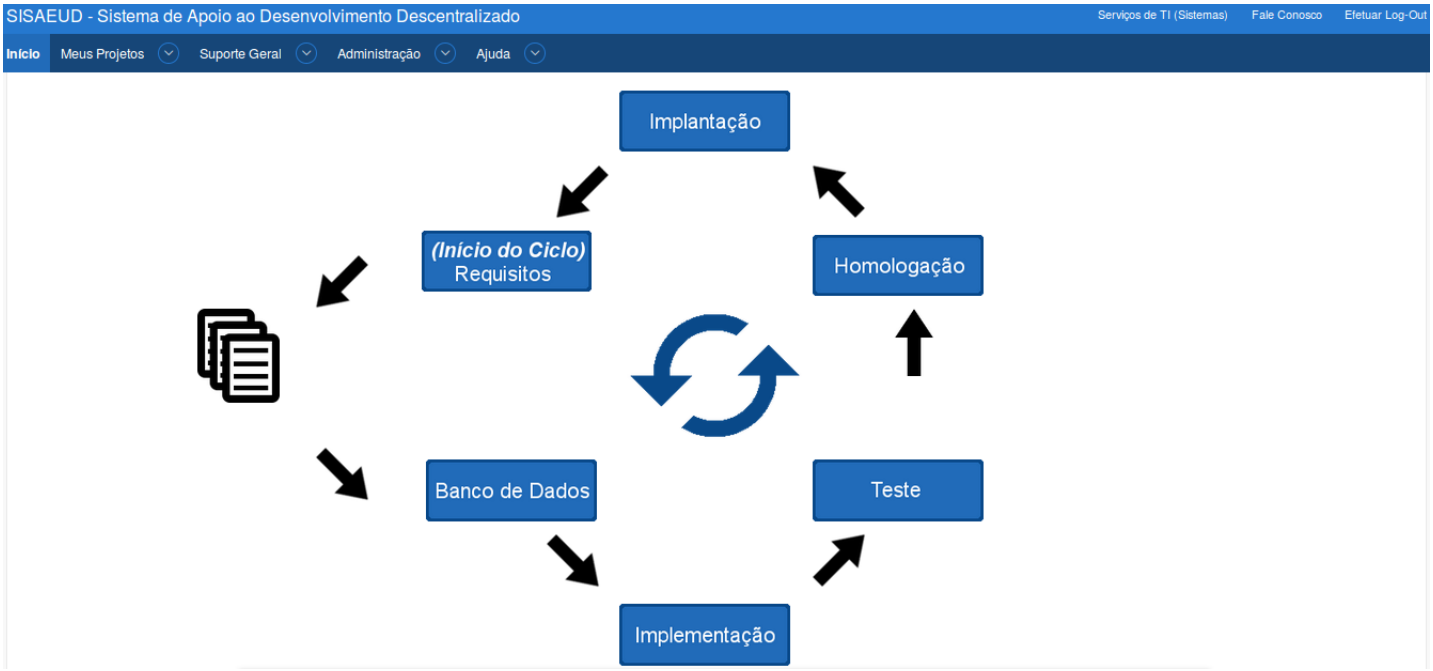

Figura 11 – Tela inicial do SISADD

Na tela inicial do SISADD é apresentado o diagrama do ciclo iterativo proposto para o desenvolvimento descentralizado, onde cada etapa deste ciclo é um botão clicável que o desenvolvedor pode usar para acessar o conjunto de atividades, guias e funcionalidades da respectiva etapa. Além disso, o menu superior oferece uma navegação mais

rápida e direta para as funcionalidades da aplicação. Observa-se que este menu superior é apresentado em todas as telas do sistema.

A Figura 12 ilustra um exemplo do detalhamento da etapa de requisitos, ou seja, é a tela apresentada caso o usuário clique na etapa de requisitos da tela da Figura 11 (tela com o ciclo de desenvolvimento).

Figura 12 – Atividades, guias e funcionalidades da etapa de requisitos

Dentro de cada etapa do ciclo temos um fluxo sequencial de atividades que são suportadas por um conjunto de guias e funcionalidades. Atividades são representadas por retângulos, guias são representados por elipses e funcionalidades são representadas por elipses com o texto sublinhado. Ainda nesta tela, a parte superior apresenta uma espécie de *wizard*, onde cada etapa anterior a etapa na qual o desenvolvedor se encontra é marcada com um ícone de verificação, dando a entender que para estar nesta etapa o desenvolvedor teria que ter passado pelas etapas anteriores.

A Figura 13 representa o cadastro de informações de um projeto, uma das mais importantes do sistema pois permite ao desenvolvedor cadastrar e editar informações a cerca do projeto no qual está trabalhando.

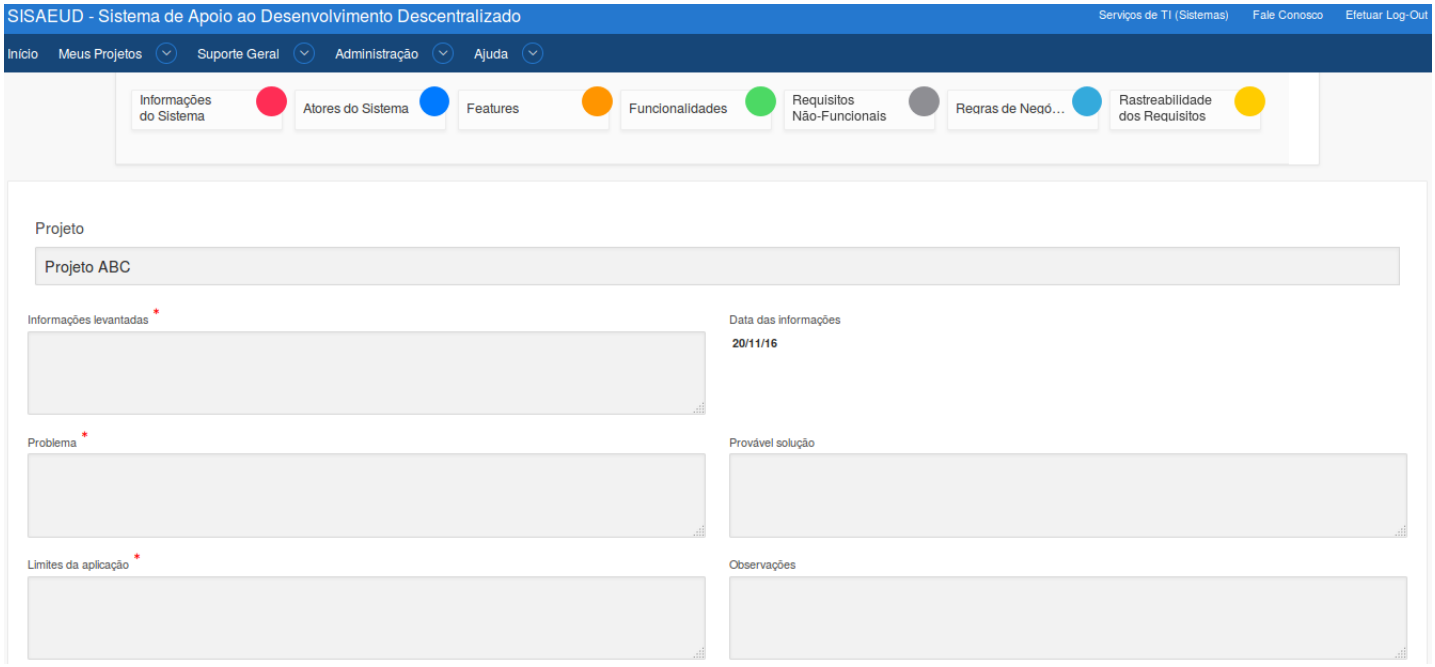

Figura 13 – Cadastro de informações de um projeto

A parte superior exibe várias abas onde o desenvolvedor pode alternar entre várias telas de cadastro/edição de informações sobre o projeto, como: Informações do Sistema, Atores do Sistema, *Features*, Funcionalidades, Requisitos Não Funcionais, Regras de Negócio e Visualizar a Rastreabilidade dos Requisitos.

A tela de geração de valores para teste do tipo caixa preta é representada pela Figura 14 e esta funcionalidade gera um conjunto de valores para teste, seguindo o método BVA (*Boundary Value Analysis*), de acordo com o componente selecionado.

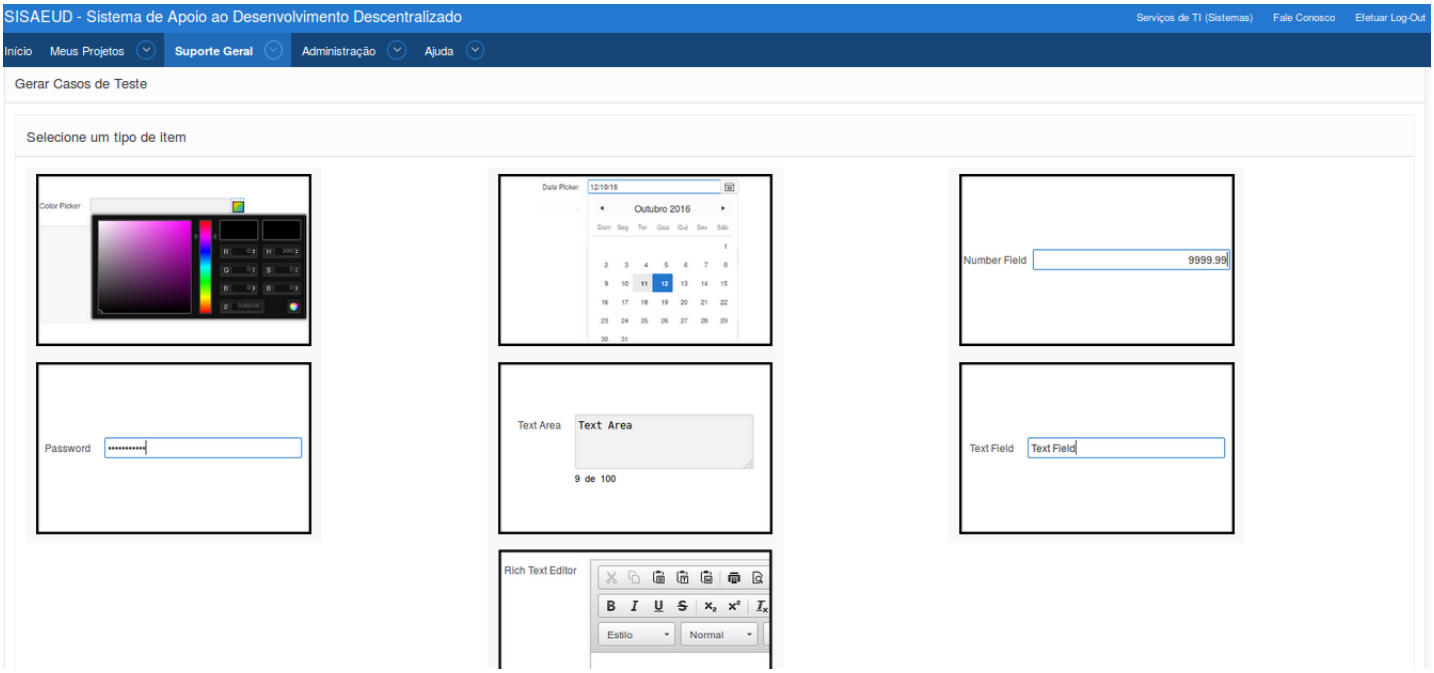

Figura 14 – Geração de valores para teste caixa preta

Os componentes apresentados para escolha são componentes de entrada de dados disponíveis na versão 5 do APEX (versão mais atual). Desta forma, permite-se que o processo de teste do tipo caixa preta das páginas desenvolvidas seja mais sistemático.

A Figura 15 exibe o guia de padrão de nomenclatura de objetos de dados.

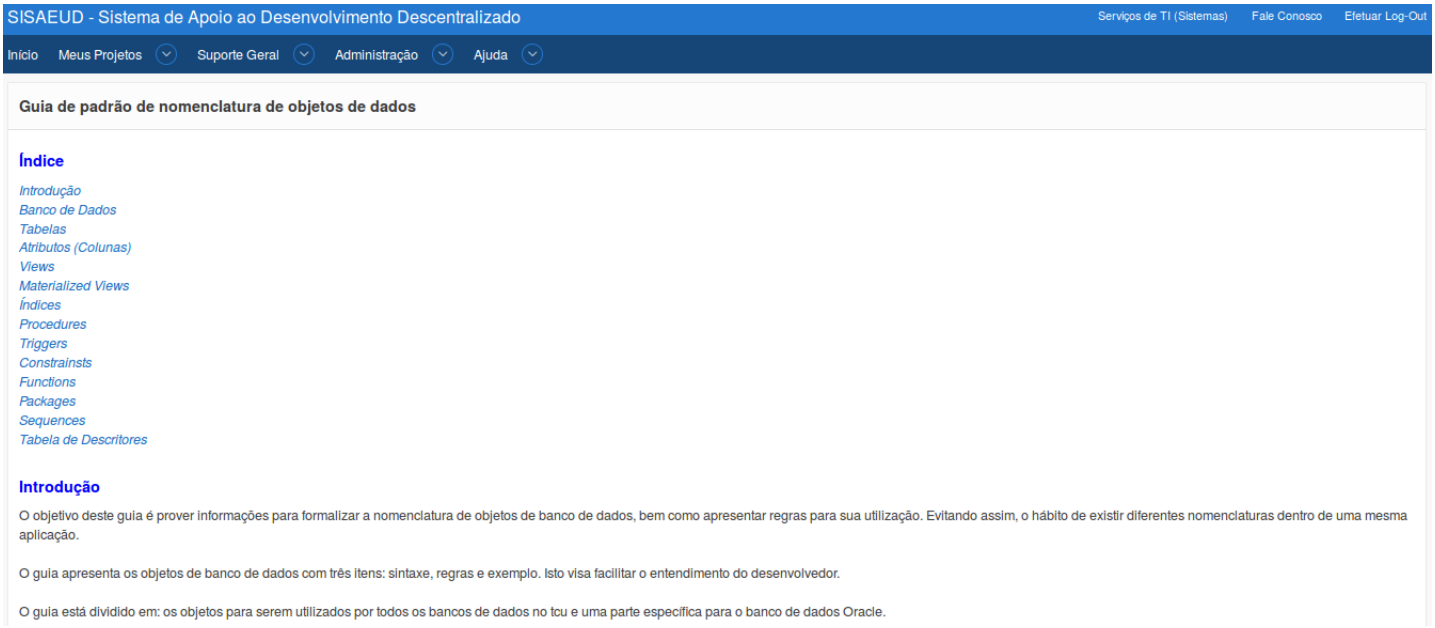

Figura 15 – Guia de padrão de nomenclatura de objetos de dados

O guia de padrão de nomenclatura de objetos de dados apresenta informações ao desenvolvedor de como nomear os objetos do banco de dados, de acordo com os padrões

definidos pelo órgão público de estudo. Em todas as telas de guias são apresentados o índice, de forma a facilitar a navegação, seguido do conteúdo das seções.

Na Figura 16 está ilustrado o *player* de vídeo usado para a reprodução de vídeos tutoriais.

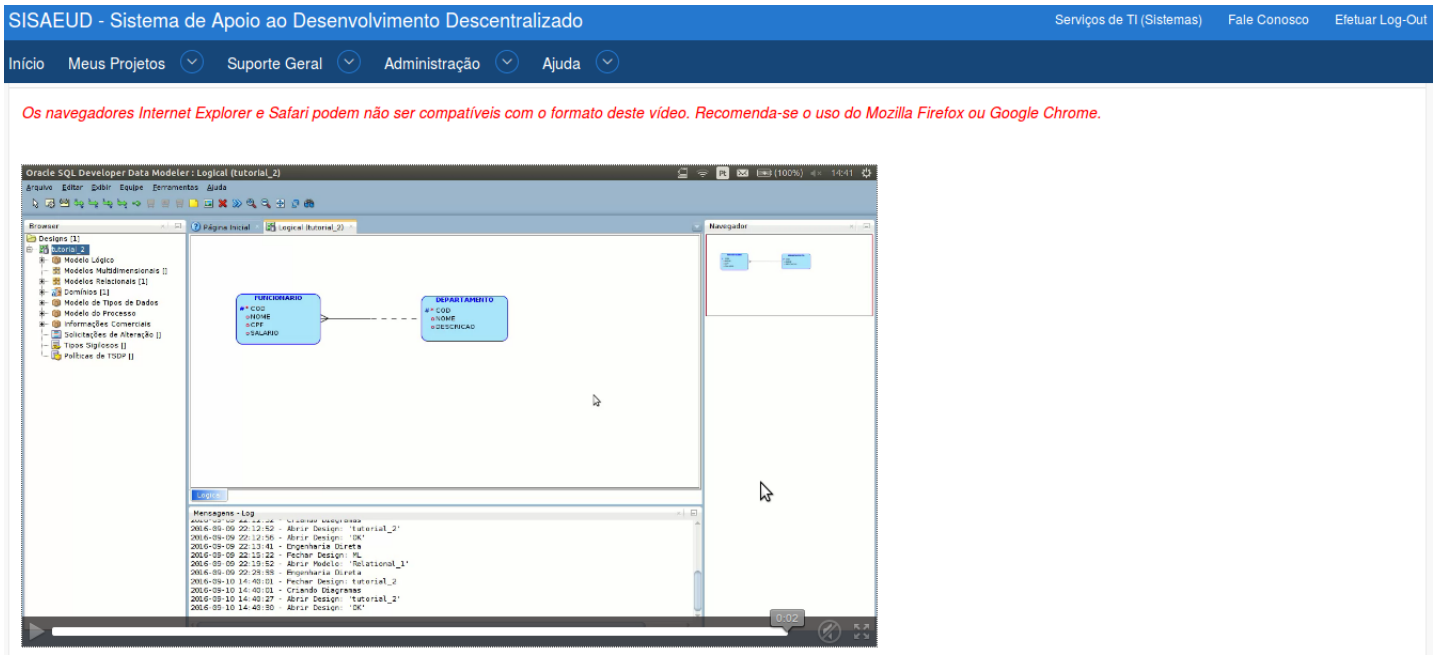

Figura 16 – Vídeo tutorial de geração de modelo lógico de banco de dados

Existem alguns guias que possuem um vídeo tutorial que foi gravado de forma a auxiliar o desenvolvedor. Para isso existe uma tela de reprodução de vídeo no formato WEBM (formato apropriado para vídeos em páginas web). O desenvolvedor pode executar todas as funções básicas de um *player* de vídeo, como avançar, aumentar a tela e silenciar o vídeo.

4.5 Análise Diagnóstico x Solução Implementada (SISADD)

Para se ter uma noção do grau de conformidade que a solução implementada possui com o diagnóstico realizado, foi elaborada uma associação entre ambos. A Tabela 21 apresenta a relação do diagnóstico obtido com o que foi implementado no SISADD. Observa-se que o diagnóstico obtido contém tanto problemas quanto algumas boas práticas e pontos positivos relatados pelos desenvolvedores entrevistados, na etapa de coleta de dados. As soluções pertencentes a coluna Implementação que contém um asterisco são soluções que até o momento não foram finalizadas ou que estão para ser iniciadas.

Tabela 21 – Diagnóstico Obtido x Solução Implementada

Diagnóstico	**Implementação**
Curso da somente uma introdução	Consulta de Informações na Solução *
O curso poderia ser melhorado	
Treinamento em boas práticas APEX	
Construção de um catálogo de informações APEX	
Algumas aplicações com baixa qualidade	Guia de Padrões e Boas Práticas APEX
	Definir Procedimentos de Teste *
	Definir Procedimentos de Implantação
Os requisitos são levantados através de reuniões.	- Priorizar Funcionalidades a Serem Desenvolvidas * - Elicitar Requisitos Junto ao Cliente - Guia de Entrevistas * - Escolher Técnica de Elicitação
Reuniões constantes.	
Requisitos óbvios não anotados durante as reuniões são esquecidos	Estruturar Informações Obtidas
Registro de requisitos em papel	
Registro de requisitos em sistemas distintos em rede	
Registro de requisitos inexistente	
Uso de prototipagem para levantar/validar requisitos	Guia de Prototipagem Rápida
Não atualiza o modelo de dados	Atualizar Modelo de Dados *
- Não valida o modelo de dados com área técnica - Valida o modelo de dados com área técnica	Validar Modelo de Dados *
Possibilidade de gerar o modelo de dados a partir das tabelas	Guia de Geração de Scripts SQL
Negligência ao modelo de dados	- Guia de Criação e Padrões, de BD - Guia de Padrão de Nomenclatura de Objetos de Dados
Elabora o modelo de dados utilizando ferramentas CASE	Guia de Criação e Padrões, de BD

Diagnóstico	**Implementação**
- Realiza depurações no sistema com o APEX - Não realiza depurações	- Guia de Uso de Depuração de Páginas APEX - Guia de Teste de Comportamento de Código *
Realiza depurações no console de comandos SQL	Guia de Teste de Comportamento de Código *
Não há classificação de erros	Cadastro de Erros Conhecidos
Falta do uso de um padrão de boas práticas comprometeu a qualidade da aplicação	- Guia de Padrões e Boas Práticas APEX - Guia de Boas Práticas PL/SQL * - Guia de Padrão de Nomenclatura de Objetos de Dados
Boas práticas realizada	
Escrita legível de código	- Guia de Boas Práticas PL/SQL * - Guia de Padrão de Nomenclatura de Objetos de Dados
Mais da metade da aplicação é composta de algum código	Guia de Boas Práticas PL/SQL *
Preferência pelo uso de componentes padrões do APEX	Guia de Padrões e Boas Práticas APEX
Aplicação bem desenvolvida	
Falta de um padrão de boas práticas	- Guia de Padrões e Boas Práticas APEX - Guia de Boas Práticas PL/SQL *
Reutilização de código	Cadastro de Código Reutilizável
Uso de editores externos	Guia Sobre Editor de Código SQL e PL/SQL *
Uso do editor do APEX	
Uso de testes funcionais	- Geração de Casos de Teste - Guia de Ferramenta de Automação de Teste *
Uso de teste de comandos SQL	Guia de Teste de Comportamento de Código *
Teste de execução de páginas	Guia de Teste de Página APEX *
Não há elaboração de casos de teste	Geração de Casos de Teste
Favorável à proposta de automação de testes funcionais	Guia de Ferramenta de Automação de Teste *
Costuma verificar se os apelidos não estão duplicados	Guia de Migração de Aplicações

Diagnóstico	**Implementação**
Não migra as definições de objetos	
Verifica a criação das tabelas no espaço de produção	
Navegação feita, antes de migrar, somente nas páginas que alterou	Definir Procedimentos de Implantação
Navegação feita, antes de migrar, em todas as páginas da aplicação	
Faz a migração da definição dos objetos de dados	
Faz a homologação das aplicações antes de disponibilizar definitivamente na produção	- Homologar Aplicação * - Guia de Homologação de Aplicações *

Em relação ao diagnóstico temos um total de 44 itens distintos que contemplam problemas, boas práticas e pontos positivos. Deste total, 19 itens estão totalmente contemplados (concluídos) na implementação atual. Do restante, 9 itens estão parcialmente contemplados (parcialmente concluídos) e 17 itens não estão contemplados (implementação pendente de iniciar). Portanto temos os seguintes dados de conformidade do SISADD ao diagnóstico obtido:

- Conformidade total ao diagnóstico: Aproximadamente 43 % (Totalmente contemplados);
- Conformidade parcial ao diagnóstico: Aproximadamente 20 % (Parcialmente contemplados);
- Não contemplados: Aproximadamente 38 %;
- Conformidade geral (Total e Parcial): Aproximadamente 64 %;

4.6 Estado da Implementação do SISADD

Durante a implementação do SISADD houveram algumas dificuldades que impediram que ele fosse totalmente finalizado. Dentre as dificuldades que impediram a sua finalização estão:

- Esforço estimado inferior ao esforço real necessário, por conta de dependências que não puderam ser previstas de antemão;
- Escopo muito grande para o tempo disponível;

A figura 17 ilustra um mapeamento das implementações que foram concluídas e das que estão pendentes.

	SISADD - Sistema de Apoio ao Desenvolvimento Descentralizado	
	Concluído	**Pendente**
Requisitos	- Elicitar Requitos Junto ao Cliente - Escolher Técnica de Elicitação - Guia de Técnica de Elicitação - Estruturar Informações Obtidas - Guia de Prototipagem Rápida	- Priorizar Funcionalidades a Serem Desenvolvidas - Escolher Ténica de Elicitação *
Banco de Dados	- Guia de Criação e Padrões, de BD - Guia de Padrão de Nomenclatura de Objetos de Dados - Guia de Geração de Scripts SQL	- Construir Modelo de Dados - Validar Modelo de Dados - Atualizar Modelo de Dados
Implementação	- Guia de Padrões e Boas Práticas APEX - Cadastro de Código Reutilizável - Consulta de Informações na Solução	- Gerar Esqueleto da Aplicação - Ajustar a Interface do Sistema - Aplicar Regras de Negócio - Guia de Boas Práticas PL/SQL - Guia Sobre Editor de Código SQL e PL/SQL
Teste	- Geração de Casos de Teste - Guia de Uso de Depuração de Páginas APEX - Cadastro de Erros Conhecidos	- Preparar Ambiente de Testes - Definir Procedimentos de Teste - Executar Testes de Desenvolvedor - Guia de Ferramenta de Automação de Teste - Guia de Teste de Comportamento de Código - Executar Testes Exploratórios - Guia de Teste de Página APEX - Realizar Correções no Sistema
Homologação		- Criar Material de Apoio ao Usuário - Homologar Aplicação - Guia de Homologação de Aplicações
Implantação	- Definir Procedimentos de Implantação - Guia de Migração de Aplicações - Migrar Aplicação	

Figura 17 – Mapeamento das Implementações Concluídas e Pendentes

Os itens apresentados no mapeamento são as atividades, guias e funcionalidades que compõem o fluxo das etapas do ciclo de vida da solução (Requisitos, Banco de Dados, Implementação, Teste, Homologação e Implantação). Portanto para a finalização do SISADD é necessário realizar a implementação dos itens de cada etapa que estão localizados na coluna Pendente.

A aplicação SISADD esta disponível na página acessada pela URL a seguir. Além dos arquivos necessários para a instalação da aplicação, a página contém instruções de instalação e a licença do software (software livre).

https://github.com/filipeborges/SISADD_APEX

5 Conclusão e Trabalhos Futuros

A falta de de padrões e adesão a boas práticas no desenvolvimento de aplicações pode levar à problemas relacionados a qualidade e manutenibilidade das mesmas. Usuários finais não tem as mesmas habilidades e experiências de um desenvolvedor profissional. Portanto é necessário que esses desenvolvedores EUD tenham uma atenção especial para o desenvolvimento de suas aplicações, através de um modelo que lhes dê suporte. No órgão público de estudo o principal problema identificado está relacionado ao modo em que o desenvolvimento de sistemas descentralizados é executado, onde os departamentos desenvolvem os sistemas conforme suas necessidades e por isso geralmente acabam negligênciando o uso de boas práticas e padrões de desenvolvimento.

Tendo em vista o problema encontrado, este trabalho se propôs a desenvolver uma solução (sistema) de apoio ao desenvolvimento descentralizado no órgão público de estudo, agregando recursos que contribuam para a melhoria na qualidade e manutenibilidade dos sistemas desenvolvidos. A questão de pesquisa abordada é como apoiar o desenvolvedor na construção de sistemas descentralizados para promover a qualidade e manutenibilidade destes sistemas. Para responder a esta questão, foi realizada uma pesquisa explicativa com o uso do procedimento técnico bibliográfico, documental e pesquisa-ação participante. Desta forma o diagnóstico que identificou problemas, boas práticas e sugestões, do ponto de vista do desenvolvedor, foi realizado.

A partir do diagnóstico obtido e da análise de documentos do órgão, foi proposto o modelo de solução que contém o conjunto de atividades, guias de apoio e funcionalidades de forma a apoiar o desenvolvimento de sistemas. Este modelo então constituiu a base de requisitos para a implementação do sistema de apoio ao desenvolvimento descentralizado (SISADD) na ferramenta *Oracle* APEX. Além da base de requisitos usada como insumo, houve o uso de observação por parte dos pesquisadores no contexto de atuação, o que permitiu uma abrangência maior na implementação do sistema.

Como trabalho futuro, é necessário a finalização da implementação da solução e a avaliação de sua efetividade em apoiar o desenvolvimento descentralizado dentro do órgão público de estudo. Os itens que faltam implementar estão ilustrados na seção 4.6, já a avaliação da solução deverá ser realizada com a escolha de um projeto piloto, a realização dos testes no mesmo e a coleta de resultados que indicam se o sistema atente ou não ao que foi proposto.

Referências

BLACKWELL, A.; GREEN, T. Notational systems–the cognitive dimensions of notations framework. *HCI Models, Theories, and Frameworks: Toward an Interdisciplinary Science. Morgan Kaufmann*, 2003. Citado 2 vezes nas páginas 23 e 40.

BORDA, O. F. Reflexiones sobre la aplicación del método de estudio-acción en colombia. *Revista Mexicana de Sociología*, JSTOR, v. 35, n. 1, p. 49–62, 1973. Citado na página 29.

BOTERF, G. L. Pesquisa participante: propostas e reflexões metodológicas. *Repensando a pesquisa participante. São Paulo: Brasiliense*, p. 51–81, 1999. Citado 2 vezes nas páginas 24 e 27.

BURNETT, C. Technology and literacy in early childhood educational settings: A review of research. *Journal of Early Childhood Literacy*, SAGE Publications, v. 10, n. 3, p. 247–270, 2010. Citado na página 39.

BURNETT, M. M.; SCAFFIDI, C. 10. end-user development. Citado na página 42.

CARNEIRO, A. Rede interativa para a gestão compartilhada da reserva extrativista marinha de arraial do cabo. In: CLADHE-II/AHM–HE. SIMPOSIO: FORMACIÓN DE REDES SOCIALES Y SU RELACIÓN CON LA ACTIVIDAD ECONÓMICA. CIUDAD DE MÉXICO. [S.l.], 2010. Citado na página 30.

CARVALHO, P. *Um modelo de desenvolvimento descentralizado de sistemas: dando poder ao usuario e ampliando a Governaca de TI*. 2011. <http://portal2.tcu.gov.br/portal/pls/portal/docs/2395121.PPT>. Citado 2 vezes nas páginas 22 e 45.

COSTABILE, M. F. et al. Supporting interaction and co-evolution of users and systems. In: ACM. *Proceedings of the working conference on Advanced visual interfaces*. [S.l.], 2006. p. 143–150. Citado na página 40.

ELDEN, M. *Three generations of worker democracy research in Norway*. [S.l.]: Associate Business Press, London, 1979. Citado na página 29.

FAILY, S. Towards Requirements Engineering Practice for Professional End User Developers: A Case Study. In: *Requirements Engineering Education and Training, 2008. REET '08*. [S.l.: s.n.], 2008. p. 38–44. Citado na página 40.

FERREIRA, G. F. Arquitetutura da informação no desenvolvimento de aplicação web. 2015. Citado 2 vezes nas páginas 13 e 44.

FISCHER, G.; GIACCARDI, E. Meta-design: A framework for the future of end-user development. In: *End user development*. [S.l.]: Springer, 2006. p. 427–457. Citado na página 40.

FISCHER, G. et al. Meta-design: a manifesto for end-user development. *Communications of the ACM*, ACM, v. 47, n. 9, p. 33–37, 2004. Citado 3 vezes nas páginas 13, 37 e 38.

FLORIANI, G. dos S.; MAFRA, S. H. Diagnótico rural participativo para gestão sócio-ambiental da araucária. *Cadernos de Agroecologia*, v. 2, n. 1, 2007. Citado na página 30.

GIL, A. *Como elaborar projetos de pesquisa*. [S.l.]: Atlas, 2002. ISBN 8522431968. Citado 4 vezes nas páginas 23, 24, 25 e 27.

GÜNTHER, H. Pesquisa qualitativa versus pesquisa quantitativa: esta é a questão. *Psicologia: teoria e pesquisa*, SciELO Brasil, v. 22, n. 2, p. 201–210, 2006. Citado na página 23.

II, M. F. et al. Integrating automated test generation into the wysiwyt spreadsheet testing methodology. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, ACM, v. 15, n. 2, p. 150–194, 2006. Citado na página 41.

JAMES, E. A.; MILENKIEWICZ, M. T.; BUCKNAM, A. *Participatory action research for educational leadership: Using data-driven decision making to improve schools*. [S.l.]: Sage Publications, 2007. Citado na página 29.

KINDON, S.; PAIN, R.; KESBY, M. *Participatory action research approaches and methods: Connecting people, participation and place*. [S.l.]: Routledge, 2007. Citado na página 29.

KO, A. J. et al. The state of the art in end-user software engineering. *ACM Computing Surveys (CSUR)*, ACM, v. 43, n. 3, p. 21, 2011. Citado 2 vezes nas páginas 39 e 40.

KOCH, T.; KRALIK, D. *Participatory action research in health care*. [S.l.]: John Wiley & Sons, 2009. Citado na página 29.

KOESNANDAR, A. et al. Using assertions to help end-user programmers create dependable web macros. In: ACM. *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*. [S.l.], 2008. p. 124–134. Citado na página 42.

LAKATOS, E. M.; MARCONI, M. de A. *Metodologia científica*. [S.l.]: Atlas São Paulo, 1991. Citado na página 24.

LESHED, G. et al. Coscripter: automating & sharing how-to knowledge in the enterprise. In: ACM. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. [S.l.], 2008. p. 1719–1728. Citado na página 41.

LIEBERMAN, H. et al. *End-user development: An emerging paradigm*. [S.l.]: Springer, 2006. Citado 3 vezes nas páginas 21, 37 e 39.

MACKAY, W. E. Patterns of sharing customizable software. In: ACM. *Proceedings of the 1990 ACM conference on Computer-supported cooperative work*. [S.l.], 1990. p. 209–221. Citado na página 42.

MAHNIC, V.; DRNOVSCEK, S. Agile software project management with scrum. In: CITESEER. *EUNIS 2005 Conference–Session papers and tutorial abstracts*. [S.l.], 2005. Citado na página 55.

MARCONI, M. d. A.; LAKATOS, E. M. Fundamentos de metodologia científica. In: *Fundamentos de metodologia científica*. [S.l.]: Atlas, 2010. Citado na página 26.

MARQUES, B.; MIRANDA, M. L. Photovoice: implicações do método colaborativo para as pesquisas em educação física e saúde. *Revista Brasileira de Atividade Física & Saúde*, v. 20, n. 6, p. 545, 2016. Citado na página 30.

MCINTYRE, A. *Participatory action research*. [S.l.]: Sage Publications, 2007. v. 52. Citado na página 29.

MØRCH, A. I.; MEHANDJIEV, N. D. Tailoring as collaboration: The mediating role of multiple representations and application units. *Computer Supported Cooperative Work (CSCW)*, Springer, v. 9, n. 1, p. 75–100, 2000. Citado na página 40.

MORESI, E. et al. Metodologia da pesquisa. *Brasília: Universidade Católica de Brasília*, v. 108, 2003. Citado na página 23.

NARDI, B. A. *A small matter of programming: perspectives on end user computing*. [S.l.]: MIT press, 1993. Citado na página 39.

NOVAES, M. B. C. de; GIL, A. C. A pesquisa-ação participante como estratégia metodológica para o estudo do empreendedorismo social em administração de empresas. *Revista de Administração Mackenzie*, v. 10, n. 1, 2009. Citado 3 vezes nas páginas 25, 27 e 29.

ONEY, S.; MYERS, B. Firecrystal: Understanding interactive behaviors in dynamic web pages. In: IEEE. *2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. [S.l.], 2009. p. 105–108. Citado na página 42.

ORACLE. *Oracle Application Express 5.0 Overview*. 2015. <http://www.oracle.com/technetwork/developer-tools/apex/learnmore/apex-50-overview-2526922.pdf>. Citado na página 42.

PANKO, R. R. What we know about spreadsheet errors. *Journal of Organizational and End User Computing (JOEUC)*, IGI Global, v. 10, n. 2, p. 15–21, 1998. Citado na página 39.

PATERNÒ, F. End user development: Survey of an emerging field for empowering people. *ISRN Software Engineering*, Hindawi Publishing Corporation, v. 2013, 2013. Citado 2 vezes nas páginas 21 e 38.

POWELL, S. G.; BARKER, K. R. *Management Science: The Art Of Modeling With Spreadsheets, (W/Cd)*. [S.l.]: John Wiley & Sons, 2008. Citado na página 41.

RANDOLPH, J. J. A guide to writing the dissertation literature review. *Practical Assessment, Research & Evaluation*, Dr. Lawrence M. Rudner, v. 14, n. 13, p. 1–13, 2009. Citado na página 35.

REASON, P.; BRADBURY, H. *The SAGE Handbook of Action Research: Participative Inquiry and Practice*. SAGE Publications, 2007. ISBN 9781446206584. Disponível em: <https://books.google.com.br/books?id=2fTlmcue2p0C>. Citado na página 29.

ROSSON, M.; BURNETT, M.; SCAFFIDIC, C. End-user development. *The Encyclopedia of Human-Computer Interaction. The Interaction Design Foundation*, Springer, 2013. Citado na página 41.

ROSSON, M. B.; CARROLL, J. M. Minimalist design for informal learning in community computing. In: *Communities and Technologies 2005*. [S.l.]: Springer, 2005. p. 75–94. Citado na página 40.

RUNESON, P. et al. *Case Study Research in Software Engineering: Guidelines and Examples*. 1st. ed. [S.l.]: Wiley Publishing, 2012. ISBN 1118104358, 9781118104354. Citado 2 vezes nas páginas 32 e 33.

SANTOS, B. de S. *Introdução a uma ciência pós-moderna*. [S.l.: s.n.], 1989. Citado na página 24.

SANTOS, L. dos; COSTA, R. R. da; TREVISAN, T. S. Pesquisa ação e participante: Suas contribuições para o conhecimento científico. Citado 3 vezes nas páginas 13, 28 e 31.

SCAFFIDI, C. et al. Using traits of web macro scripts to predict reuse. *Journal of Visual Languages & Computing*, Elsevier, v. 21, n. 5, p. 277–291, 2010. Citado na página 42.

SCAFFIDI, C.; MYERS, B.; SHAW, M. Topes: reusable abstractions for validating data. In: ACM. *Proceedings of the 30th international conference on Software engineering*. [S.l.], 2008. p. 1–10. Citado na página 42.

STELLMAN, A.; GREENE, J. *Learning Agile: Understanding Scrum, XP, Lean, and Kanban*. [S.l.]: O'Reilly Media, 2014. ISBN 9781449363857. Citado na página 55.

THIOLLENT, M. *Pesquisa-ação nas organizações*. [S.l.]: Atlas, 2009. Citado 3 vezes nas páginas 27, 28 e 30.

THIOLLENT, M. Metodologia da pesquisa-ação. In: *Metodologia da pesquisa-ação*. [S.l.]: Cortez, 2011. Citado 3 vezes nas páginas 24, 25 e 27.

THORSRUD, E.; EMERY, F. E. Industrial democracy in norway. *Industrial Relations: A Journal of Economy and Society*, Wiley Online Library, v. 9, n. 2, p. 187–196, 1970. Citado na página 29.

TOZONI-REIS, M. F. de C. *A pesquisa-ação-participativa em educação ambiental: reflexões teóricas*. [S.l.]: Annablume, 2007. Citado 2 vezes nas páginas 29 e 30.

TRIPP, D. Pesquisa-ação: uma introdução metodológica. *Educação e pesquisa*, SciELO Brasil, v. 31, n. 3, p. 443–466, 2005. Citado 2 vezes nas páginas 26 e 27.

TRIVIÑOS, A. N. S. Introdução à pesquisa em ciências sociais: a pesquisa qualitativa em educação. são paulo: Atlas, 1987. *Outros números do Informe Rural ETENE: ANO*, v. 3, 2009. Citado na página 25.

VIDEIRA, O.; FIGUEIREDO, R.; VENSON, E. It governance solution for the information technology (it) area of brazilian public federal sector. CONTECSI, 2014. Citado 2 vezes nas páginas 22 e 45.

WILSON, A. et al. Harnessing curiosity to increase correctness in end-user programming. In: ACM. *Proceedings of the SIGCHI conference on Human factors in computing systems*. [S.l.], 2003. p. 305–312. Citado na página 41.

YIN, R. *Estudo de caso: planejamento e métodos*. [S.l.]: Bookman, 2001. ISBN 9788536304625. Citado na página 23.

Apêndices

APÊNDICE A – Perguntas da entrevista

Descrição do desenvolvedor

Objetivo(s):

- Obter informações sobre o perfil dos desenvolvedores.
  - Código derivado: Perfil do Desenvolvedor;

Unidade de alocação:

1- Qual é o curso que você faz?
2- Quantos semestres são o curso?
3- Está em qual semestre do curso?
4- Qual é o nome da instituição de ensino?
5- Quanto tempo de estágio?

Papel do desenvolvedor na organização.

Objetivo(s):

- Obter informações sobre o papel do desenvolvedor no órgão.
  - Código derivado: Papel do Desenvolvedor;

6- Como você enxerga o seu papel na instituição? (A posição dentro da instituição, as responsabilidades, e etc...)
7- Você considera a sua contribuição relevante para a instituição?

Efetividade do Treinamento

Objetivo(s):

- Obter informações sobre a preparação dos novos desenvolvedores.
  - Código derivado: Preparação;

8- Você acha que o curso o preparou bem para as tarefas que realizou?
9- Qual a sua opinião com relação a efetividade do curso?

10- Como você avalia a qualidade das aplicações em que deu manutenção?

Boas práticas e padrões

Objetivo(s):

- Obter informações sobre as práticas realizadas no desenvolvimento.
  - Código derivado: Estilo e Design de codificação;

11- Você adota algum padrão ou alguma boa prática no desenvolvimento/manutenção das aplicações?
12- Teria alguma sugestão de boa prática/padrão que realiza que poderia melhorar de alguma forma a qualidade das aplicações desenvolvidas?

Requisitos

Objetivo(s):

- Obter informações sobre como se dá o levantamento de requisitos nos departamentos da instituição.
  - Código derivado: Elicitação de Requisitos;
- Obter informações de como é armazenado e gerenciado os requisitos do sistema a ser desenvolvido.
  - Código derivado: Gerenciamento de Requisitos;
- Obter informações de como é englobado as mudanças de requisitos ao sistema.
  - Código derivado: Gerenciamento de Requisitos;

13- Como você faz para levantar os requisitos do sistema que foi solicitado por alguma unidade?
14- Como você aborda o cliente para levantar os requisitos do sistema?
15- Como você armazena ou documenta o que foi passado pelo cliente (requisitos)? Existe algum meio de armazenar eles diretamente no sistema?
16- Você costuma fazer reunião com o cliente em outras etapas do desenvolvimento, para levantar outros requisitos e/ou esclarecer os que já existem?

Design

Objetivo(s):

- Obter informações sobre como é realizado a modelagem do sistema.
  - Código derivado: Modelagem do sistema;

17- Como você elabora o modelo de dados do sistema (ferramentas, MER, ML)? Se nunca elaborou, considera-o importante?
18- Você valida tecnicamente este modelo com área responsável ou com algum padrão disponível (nomenclatura e formas normais)? Se nunca validou, considera-o importante?

Codificação

Objetivo(s):

- Obter informações sobre a depuração de erros do sistema.
  - Código derivado: Depuração;
- Obter informações sobre estilos de codificação.
  - Código derivado: Estilo e *Design* de codificação;
- Obter informações sobre o ambiente de codificação.
  - Código derivado: Ambiente de codificação;

19- Você costuma utilizar código PL/SQL, HTML e Javascript em mais da metade da aplicação?
20- Você procurar escrever os seus códigos PL/SQL, SQL, HTML e Javascript de forma legível?
21- Você segue algum padrão de codificação ou boas práticas (constantes, nomes significativos) para PL/SQL, SQL, HTML e Javascript?
22- Você costuma usar algum editor de código ou o próprio APEX, para escrever seus códigos?
23- Você costuma depurar o código?
24- Você costuma classificar os erros encontrados (falha de execução ou comportamento inesperado), de forma que possa ajudá-lo na abordagem de resolução?
25- Você costuma comentar o código que faz?

Teste

Objetivo(s):

- Obter informações sobre a realização dos testes, bem como a forma de realizá-los.
  - Código derivado: Teste;

26- Como você testa as suas aplicações?
27- Você costuma testar todas as páginas de aplicação que desenvolve?
28- Você costuma armazenar os resultados dos testes que realiza?
29- Você costuma a elaborar casos de testes para sua aplicação?
30- Seria bom testes automatizados de interface para sua aplicação (selenium)?

Implantação

Objetivo(s):

- Obter informações sobre a migração, para a produção, do sistema.
  - Código derivado: Implantação;

31- Você costuma migrar as definições dos objetos de banco (tabelas, sequences, views e etc) junto com a aplicação?
32- Você costuma verificar se os apelidos da aplicação de desenvolvimento e produção estão iguais?
33- Realiza uma navegação por todas as páginas da aplicação?
34- Verifica se todas as tabelas foram criadas?

APÊNDICE B – Respostas das Entrevistas

- **Desenvolvedor 1**

Descrição do Desenvolvedor

1. Qual é o curso que você faz?
   - Análise e Desenvolvimento de Sistemas.

2. Quantos semestres são o curso?
   - 5 semestres.

3. Está em qual semestre do curso?
   - Terceiro semestre.

4. Qual é o nome da instituição de ensino?
   - Faculdade Senac.

5. Quanto tempo de estágio?
   - 1 ano.

Papel do desenvolvedor na organização

6. Como você enxerga o seu papel na instituição? (A posição dentro da instituição, as responsabilidades, e etc. . .)
   - O papel está claro. Foi contratado para dar manutenção em uma aplicação da Secretária de Comunicação (SECOM), porém esta alocado fisicamente no SEADE, e também realiza alguns trabalhos para o mesmo, apesar de sua função principal ser atender a Secretária de Comunicação.

7. Você considera a sua contribuição relevante para a instituição?
   - Sente que sua contribuição é relevante.

Efetividade do Treinamento

8. Você acha que o curso o preparou bem para as tarefas que realizou?
   - Não. O curso somente introduz o APEX, muitas das coisas são vistas somente na prática.

9. Qual a sua opinião com relação a efetividade do curso?
   - Considera-o importante para apresentar a ferramenta, porém muitas coisas não são abordadas.

10. Como você avalia a qualidade das aplicações em que deu manutenção?
    - A aplicação em que deu manutenção algumas partes estavam muito bem detalhadas e com bastante comentários, porém outras partes estavam com coisas que não estavam mais sendo utilizadas, que deveriam ser removidas. Segundo o entrevistado, parece ser consequência da aplicação ter passado por vários desenvolvedores diferentes.

Boas práticas e padrões

11. Você adota algum padrão ou alguma boa prática no desenvolvimento/manutenção das aplicações?
    - Mantem códigos que não tem costume de desenvolver com frequência em um arquivo de texto separado, para que possa ser reutilizado em futuros códigos seguindo a mesma lógica. Também costuma comentar o código, apesar de ter adotado essa prática somente depois de um certo tempo de desenvolvimento.

12. Teria alguma sugestão de boa prática/padrão que realiza que poderia melhorar de alguma forma a qualidade das aplicações desenvolvidas?
    - Usar o campo de comentário nas páginas das aplicações APEX, de forma a facilitar o entendimento de outros desenvolvedores.

Requisitos

13. Como você faz para levantar os requisitos do sistema que foi solicitado por alguma unidade?
    - A maioria das reuniões a unidade é quem marca com o desenvolvedor para levantar os requisitos junto ao usuário.

14. Como você aborda o cliente para levantar os requisitos do sistema?
    - Via reuniões.

15. Como você armazena ou documenta o que foi passado pelo cliente (requisitos)? Existe algum meio de armazenar eles diretamente no sistema?
    - Armazena inicialmente em um papel e depois passa para o sistema.

16. Você costuma fazer reunião com o cliente em outras etapas do desenvolvimento, para levantar outros requisitos e/ou esclarecer os que já existem?
    - Sim, realiza reuniões durante o desenvolvimento.

Design

17. Como você elabora o modelo de dados do sistema (ferramentas, MER, ML)? Se nunca elaborou, considera-o importante?
    - Não atualiza o modelo de dados. Considera-o importante.

18. Você valida tecnicamente este modelo com área responsável ou com algum padrão disponível (nomenclatura e formas normais)? Se nunca validou, considera-o importante?
    - Não valida o modelo de dados. Considera-o importante.

Codificação

19. Você costuma utilizar código PL/SQL, HTML e Javascript em mais da metade da aplicação?
    - Menos da metade da aplicação que dava manutenção tinha código, só em ocasiões especiais.

20. Você procurar escrever os seus códigos PL/SQL, SQL, HTML e Javascript de forma legível?
    - Costuma, as vezes, escrever de forma legível.

21. Você segue algum padrão de codificação ou boas práticas (constantes, nomes significativos) para PL/SQL, SQL, HTML e Javascript?
    - Sim, usa nomes significativos.

22. Você costuma usar algum editor de código ou o próprio APEX, para escrever seus códigos?
    - Usa o editor notepad++, porque os arquivos podem ficar salvos também em outro lugar.

23. Você costuma depurar o código?
    - Costuma depurar a aplicação, usando a funcionalidade de depurar.

24. Você costuma classificar os erros encontrados (falha de execução ou comportamento inesperado), de forma que possa ajudá-lo na abordagem de resolução?
    - A classificação de erros é somente de forma mental.

25. Você costuma comentar o código que faz?
    - Comenta as vezes.

Teste

26. Como você testa as suas aplicações?
    - Testa as aplicações usando testes funcionais.

27. Você costuma testar todas as páginas de aplicação que desenvolve?
    - Sim, testa todas as páginas que desenvolve.

28. Você costuma armazenar os resultados dos testes que realiza?
    - Não armazena os resultados dos testes.

29. Você costuma a elaborar casos de testes para sua aplicação?
    - Não elabora casos de teste, o faz de forma mais intuitiva.

30. Seria bom testes automatizados de interface para sua aplicação (selenium)?
    - Acha importante teste automatizado de interface, principalmente nos casos com muitos dados a serem setados que tem chances de erro.

Implantação

31. Você costuma migrar as definições dos objetos de banco (tabelas, sequences, views e etc) junto com a aplicação?
    - Não. Costuma anotar as alterações feitas e realiza essas alterações no espaço de produção.

32. Você costuma verificar se os apelidos da aplicação de desenvolvimento e produção estão iguais?
    - No começo não. Agora costuma pensar mais na troca de apelidos, após os avisos dos alertas.

33. Realiza uma navegação por todas as páginas da aplicação?
    - Navegação feita geralmente mais nas páginas que altera.

34. Verifica se todas as tabelas foram criadas?
    - Verifica a criação das tabelas.

Observações

- Se tivesse uma ferramente que mostrasse as diferenças entre os modelos de dados dos espaços de desenvolvimento e de produção seria interessante.
- Requisitos muito óbvios o desenvolvedor não anotava na reunião, e depois acabava esquecendo eles.
- Existe a possibilidade de gerar o modelo de dados a partir das tabelas construídas.

- **Desenvolvedor 2**

Descrição do Desenvolvedor

1. Qual é o curso que você faz?
   - Sistemas de informação.

2. Quantos semestres são o curso?
   - 8 semestres.

3. Está em qual semestre do curso?
   - Nono semestre, porém em termos de conteúdo esta no sexto.

4. Qual é o nome da instituição de ensino?
   - Universidade Católica de Brasília.

5. Quanto tempo de estágio?
   - 1 ano.

Papel do desenvolvedor na organização

6. Como você enxerga o seu papel na instituição? (A posição dentro da instituição, as responsabilidades, e etc. . .)
   - Entende o seu papel no sistema, porém ao terminar suas tarefas no sistema fica perdido em o que fazer em seguida.

7. Você considera a sua contribuição relevante para a instituição?
   - Sim. Da idéias e sugestões aos servidores, e algumas são escritas como tutoriais na *Wiki*.

Efetividade do Treinamento

8. Você acha que o curso o preparou bem para as tarefas que realizou?
   - Não. O curso somente da uma introdução, porém necessita de mais aprofundamento.

9. Qual a sua opinião com relação a efetividade do curso?
   - Poderia melhorar o curso um pouco mais.

10. Como você avalia a qualidade das aplicações em que deu manutenção?
    - Entre regular e bom, pois algumas coisas são feitas sem se preocupar com manutenções futuras.

Boas práticas e padrões

11. Você adota algum padrão ou alguma boa prática no desenvolvimento/manutenção das aplicações?
    - Documentar o código.

12. Teria alguma sugestão de boa prática/padrão que realiza que poderia melhorar de alguma forma a qualidade das aplicações desenvolvidas?
    - Construir uma API APEX, com informações que podem ser utilizadas no órgão, como: procedimento para obter o código de uma unidade, o e-mail do chefe de serviço, como implementar funcionalidades fora do comum, e etc. A idéia é divulgar o conhecimento. Atualmente esta em processo de definição.

Requisitos

13. Como você faz para levantar os requisitos do sistema que foi solicitado por alguma unidade?
    - Reunião junto com o consultor técnico no início. Após um tempo, o desenvolvedor começou a marcar as reuniões diretamente com os interessados.

14. Como você aborda o cliente para levantar os requisitos do sistema?
    - Através de reuniões.

15. Como você armazena ou documenta o que foi passado pelo cliente (requisitos)? Existe algum meio de armazenar eles diretamente no sistema?
    - Anota em papel quando vai nas reuniões. Após implementar os requisitos, passa eles para o DGA.

16. Você costuma fazer reunião com o cliente em outras etapas do desenvolvimento, para levantar outros requisitos e/ou esclarecer os que já existem?
    - Não, pois realiza primeiro a implementação no sistema e logo após é que confirma se os requisitos estão corretos, junto ao cliente.

Design

17. Como você elabora o modelo de dados do sistema (ferramentas, MER, ML)? Se nunca elaborou, considera-o importante?
    - Não atualiza o modelo de dados, pois ele esta muito desatualizado e necessitaria de um esforço grande pra deixar ele refletindo o estado atual do sistema.

18. Você valida tecnicamente este modelo com área responsável ou com algum padrão disponível (nomenclatura e formas normais)? Se nunca validou, considera-o importante?
    - Nunca validou o modelo de dados. Considera-o importante.

Codificação

19. Você costuma utilizar código PL/SQL, HTML e Javascript em mais da metade da aplicação?
    - Mais da metade da aplicação que trabalhou possui código.

20. Você procurar escrever os seus códigos PL/SQL, SQL, HTML e Javascript de forma legível?
    - Sim, procura escrever os códigos de forma legível.

21. Você segue algum padrão de codificação ou boas práticas (constantes, nomes significativos) para PL/SQL, SQL, HTML e Javascript?
    - Sim, identação e padrões de nomenclatura.

22. Você costuma usar algum editor de código ou o próprio APEX, para escrever seus códigos?
    - Usa o editor notepad++.

23. Você costuma depurar o código?
    - Costuma depurar a aplicação, usando a funcionalidade de depurar.

24. Você costuma classificar os erros encontrados (falha de execução ou comportamento inesperado), de forma que possa ajudá-lo na abordagem de resolução?
    - Não classifica os erros encontrados.

25. Você costuma comentar o código que faz?
    - Sim, costuma comentar bastante o código.

Teste

26. Como você testa as suas aplicações?
    - Testa a aplicação no ambiente de teste. Após os testes passarem, leva para o ambiente de produção. No ambiente de produção, faz uma homologação com o usuário antes de liberar de forma definitiva para a produção. Desta forma, a aplicação atual continua funcionando, e a nova vai sendo validada pelo usuário. Qualquer ajuste que precisa ser feito não interfere na aplicação atual. Após a validação do usuário, disponibiliza definitivamente para o uso.

27. Você costuma testar todas as páginas de aplicação que desenvolve?
    - Sim, testa todas as páginas que desenvolve.

28. Você costuma armazenar os resultados dos testes que realiza?
    - Não armazena os resultados dos testes.

29. Você costuma a elaborar casos de testes para sua aplicação?
    - Tenta prever de forma mais intuitiva os casos de teste.

30. Seria bom testes automatizados de interface para sua aplicação (selenium)?
    - Acha que seria benéfico teste automatizado de interface.

Implantação

31. Você costuma migrar as definições dos objetos de banco (tabelas, sequences, views e etc) junto com a aplicação?
    - Nunca precisou fazer migração de objetos de banco.

32. Você costuma verificar se os apelidos da aplicação de desenvolvimento e produção estão iguais?
    - Sim, costuma verificar os apelidos. O alerta ajudou a se preocupar com isso.

33. Realiza uma navegação por todas as páginas da aplicação?
    - Navega somente nas páginas que fazem parte do fluxo relacionado com a funcionalidade.

34. Verifica se todas as tabelas foram criadas?
    - Verifica se as tabelas foram criadas.

Observação

- Por conta de muita demanda, alguns pontos podem acabar sendo esquecidos de testar.
- Faz homologação das aplicações junto ao cliente.

- **Desenvolvedor 3**

Descrição do Desenvolvedor

1. Qual é o curso que você faz?
    - Sistemas de Informação.

2. Quantos semestres são o curso?
   - 8 semestres.

3. Está em qual semestre do curso?
   - Sétimo semestre.

4. Qual é o nome da instituição de ensino?
   - Universidade Católica de Brasília.

5. Quanto tempo de estágio?
   - 7 meses.

Papel do desenvolvedor na organização

6. Como você enxerga o seu papel na instituição? (A posição dentro da instituição, as responsabilidades, e etc. . .)
   - O papel esta claro na visão do desenvolvedor, suas responsabilidades estão claras desde quando o mesmo foi contratado.

7. Você considera a sua contribuição relevante para a instituição?
   - Sim, pois entende que esta contribuindo para uma aplicação da qual a instituição depende.

Efetividade do Treinamento

8. Você acha que o curso o preparou bem para as tarefas que realizou?
   - Não, o curso aborda de forma muito superficial o desenvolvimento APEX, ainda mais se o desenvolvedor nunca trabalhou ou ouviu falar do APEX, principalmente com a questão do uso de código na aplicação.

9. Qual a sua opinião com relação a efetividade do curso?
   - Efetividade fraca.

10. Como você avalia a qualidade das aplicações em que deu manutenção?
    - Qualidade baixa, aplicação desenvolvida de uma forma não sustentável, dando a impressão de que ela nem chegaria a ser concluída da forma que estava sendo feita, pois usava muito código e pouco componente padrão do APEX.

Boas práticas e padrões

11. Você adota algum padrão ou alguma boa prática no desenvolvimento/manutenção das aplicações?
    - Sim, usa comentários.

12. Teria alguma sugestão de boa prática/padrão que realiza que poderia melhorar de alguma forma a qualidade das aplicações desenvolvidas?
    - Comentar as aplicações e procurar usar os componentes padrões do APEX ao máximo, evitando o uso de código sempre que possível de forma a evitar problemas a futuros desenvolvedores inexperientes.

Requisitos

13. Como você faz para levantar os requisitos do sistema que foi solicitado por alguma unidade?
    - Procura reunir todos os interessados, de forma a obter os requisitos. Dúvidas pontuais que surgem, são tiradas com os interessados através de trocas de e-mail.

14. Como você aborda o cliente para levantar os requisitos do sistema?
    - Através de reuniões e trocas de e-mail.

15. Como você armazena ou documenta o que foi passado pelo cliente (requisitos)? Existe algum meio de armazenar eles diretamente no sistema?
    - Armazena os requisitos levantados no próprio e-mail do desenvolvedor.

16. Você costuma fazer reunião com o cliente em outras etapas do desenvolvimento, para levantar outros requisitos e/ou esclarecer os que já existem?
    - Sim, costuma reunir novamente com o cliente.

Design

17. Como você elabora o modelo de dados do sistema (ferramentas, MER, ML)? Se nunca elaborou, considera-o importante?
    - Não chegou a elaborar o modelo de dados, mas o considera importante.

18. Você valida tecnicamente este modelo com área responsável ou com algum padrão disponível (nomenclatura e formas normais)? Se nunca validou, considera-o importante?
    - Sim, procura validar as alterações feitas no modelo de dados junto a área técnica.

Codificação

19. Você costuma utilizar código PL/SQL, HTML e Javascript em mais da metade da aplicação?
    - Na maior parte das páginas que desenvolveu, usou algum tipo de código.

20. Você procurar escrever os seus códigos PL/SQL, SQL, HTML e Javascript de forma legível?
    - Sim, sempre pensando no próximo que irá mexer na aplicação.

21. Você segue algum padrão de codificação ou boas práticas (constantes, nomes significativos) para PL/SQL, SQL, HTML e Javascript?
    - Sim, costuma usar nomes significativos para variáveis.

22. Você costuma usar algum editor de código ou o próprio APEX, para escrever seus códigos?
    - Para códigos grandes costuma usar um editor de código. Para códigos pequenos, costuma escrever/editar no próprio APEX.

23. Você costuma depurar o código?
    - Não, pois não conhecia muito bem esta funcionalidade.

24. Você costuma classificar os erros encontrados (falha de execução ou comportamento inesperado), de forma que possa ajudá-lo na abordagem de resolução?
    - Não classifica os erros encontrados.

25. Você costuma comentar o código que faz?
    - Sim, comenta o código.

Teste

26. Como você testa as suas aplicações?
    - Solicita ao cliente para testar a aplicação, de forma a coletar *feedback* da experiência de uso (erros e sugestão de melhorias).

27. Você costuma testar todas as páginas de aplicação que desenvolve?
    - Sim, testa todas as páginas desenvolvidas.

28. Você costuma armazenar os resultados dos testes que realiza?
    - Não armazena os resultados dos testes.

29. Você costuma a elaborar casos de testes para sua aplicação?
    - Elabora de forma mental.

30. Seria bom testes automatizados de interface para sua aplicação (selenium)?
    - Seria interessante dependendo do tamanho da aplicação. O esforço para codificar os testes não seria interessante pra uma aplicação muito pequena. Já para uma aplicação média/grande, seria interessante.

Implantação

31. Você costuma migrar as definições dos objetos de banco (tabelas, sequences, views e etc) junto com a aplicação?
    - Algumas vezes migra as definições dos objetos.

32. Você costuma verificar se os apelidos da aplicação de desenvolvimento e produção estão iguais?
    - Sim, verifica os apelidos.

33. Realiza uma navegação por todas as páginas da aplicação?
    - Sim, navega por todas as páginas.

34. Verifica se todas as tabelas foram criadas?
    - Somente as tabelas que foram alteradas ou criadas, de forma que as que existem acabam dependendo da execução da aplicação para saber se continuam corretas ou não (se a aplicação não quebrou).

Observações

- Costuma, de forma proposital, deixar um erro óbvio para verificar se o cliente de fato esta testando a aplicação.

- **Desenvolvedor 4**

Descrição do Desenvolvedor

1. Qual é o curso que você faz?
   - Análise e Desenvolvimento de Sistemas.

2. Quantos semestres são o curso?
   - 5 semestres.

3. Está em qual semestre do curso?
   - Quinto semestre.

4. Qual é o nome da instituição de ensino?
   - Faculdade Projeção.

5. Quanto tempo de estágio?
   - 7 meses.

Papel do desenvolvedor na organização

6. Como você enxerga o seu papel na instituição? (A posição dentro da instituição, as responsabilidades, e etc. . .)
   - Enxerga seu papel de forma clara.

7. Você considera a sua contribuição relevante para a instituição?
   - Sim, considera sua contribuição relevante.

Efetividade do Treinamento

8. Você acha que o curso o preparou bem para as tarefas que realizou?
   - Não, o curso é bem básico.

9. Qual a sua opinião com relação a efetividade do curso?
   - Poderia melhorar, a demanda das tarefas vai além do que o curso apresenta.

10. Como você avalia a qualidade das aplicações em que deu manutenção?
    - A aplicação não foi bem construída, pois passou pelas mãos de várias pessoas, o que acabou deixando gambiarras/improvisos no sistema.

Boas práticas e padrões

11. Você adota algum padrão ou alguma boa prática no desenvolvimento/manutenção das aplicações?
    - Sim, procura construir consultas SQL com filtros, para melhor performance.

12. Teria alguma sugestão de boa prática/padrão que realiza que poderia melhorar de alguma forma a qualidade das aplicações desenvolvidas?
    - Não.

Requisitos

13. Como você faz para levantar os requisitos do sistema que foi solicitado por alguma unidade?
    - O responsável pelo desenvolvedor marca uma reunião com o cliente e desenvolvedor vai junto para a reunião, onde vai anotando os pontos levantados pelo cliente.

14. Como você aborda o cliente para levantar os requisitos do sistema?
    - Através de reuniões.

15. Como você armazena ou documenta o que foi passado pelo cliente (requisitos)? Existe algum meio de armazenar eles diretamente no sistema?
    - Primeiramente anota em um papel durante a reunião, e então passa para o documento de Descrição Geral de Aplicação, e armazena o documento no repositório SVN.

16. Você costuma fazer reunião com o cliente em outras etapas do desenvolvimento, para levantar outros requisitos e/ou esclarecer os que já existem?
    - Sim, costuma reunir várias vezes com o cliente.

Design

17. Como você elabora o modelo de dados do sistema (ferramentas, MER, ML)? Se nunca elaborou, considera-o importante?
    - Não chegou a elaborar o modelo de dados, mas o considera importante.

18. Você valida tecnicamente este modelo com área responsável ou com algum padrão disponível (nomenclatura e formas normais)? Se nunca validou, considera-o importante?
    - Sim, algumas vezes valida o modelo de dados.

Codificação

19. Você costuma utilizar código PL/SQL, HTML e Javascript em mais da metade da aplicação?
    - Mais da metade da aplicação que trabalhou possui código.

20. Você procurar escrever os seus códigos PL/SQL, SQL, HTML e Javascript de forma legível?
    - Sim, procura usar comentários e identação no código.

21. Você segue algum padrão de codificação ou boas práticas (constantes, nomes significativos) para PL/SQL, SQL, HTML e Javascript?
    - Sim, procura usar comentários, identação, nomes significativos, otimizar as consultas com filtro, e evita usar ao máximo usar funções para não pesar tanto a aplicação.

22. Você costuma usar algum editor de código ou o próprio APEX, para escrever seus códigos?
    - Costuma usar um editor de código.

23. Você costuma depurar o código?
    - Sim, costuma testar no console de comandos SQL para verificar se o código se comporta como o esperado.

24. Você costuma classificar os erros encontrados (falha de execução ou comportamento inesperado), de forma que possa ajudá-lo na abordagem de resolução?
    - Não classifica os erros.

25. Você costuma comentar o código que faz?
    - Sim, de forma que ajude também o desenvolvedor futuramente a se lembrar do comportamento do código.

Teste

26. Como você testa as suas aplicações?
    - Testa os códigos desenvolvidos no console de comandos SQL, e também testa manualmente a funcionalidade da aplicação.

27. Você costuma testar todas as páginas de aplicação que desenvolve?
    - Sim, testa todas as páginas desenvolvidas.

28. Você costuma armazenar os resultados dos testes que realiza?
    - Não, porém alguns erros dependendo da situação podem ser documentados.

29. Você costuma a elaborar casos de testes para sua aplicação?
    - Elabora os casos de teste mentalmente.

30. Seria bom testes automatizados de interface para sua aplicação (selenium)?
    - Seria interessante teste automatizado de interface.

Implantação

31. Você costuma migrar as definições dos objetos de banco (tabelas, sequences, views e etc) junto com a aplicação?
    - Sim, costuma sempre migrar as definições de objeto.

32. Você costuma verificar se os apelidos da aplicação de desenvolvimento e produção estão iguais?
    - Sim, verifica se os apelidos são iguais.

33. Realiza uma navegação por todas as páginas da aplicação?
    - Sim, realiza uma navegação por todas as páginas.

34. Verifica se todas as tabelas foram criadas?
    - Sim, verifica todas as tabelas.

Observações

- Existir alguma ferramenta que permitisse testar os códigos sem depender do APEX, de forma que o trabalho não seja interrompido caso o APEX venha a cair.

- Sempre testar o código desenvolvido no console de comandos SQL, antes de passar para a aplicação APEX.
- Usar uma pessoa que não seja o desenvolvedor para testar a aplicação, de forma a ter um teste menos viciado e mais completo.
- Maior problema que teve foi com relação a falta de uma documentação detalhada do sistema, devido ao sistema ser grande e complexo. Documentação foi negligenciada em favor do desenvolvimento do sistema.

- **Desenvolvedor 5**

Descrição do Desenvolvedor

1. Qual é o curso que você faz?
   - Sistemas de informação.
2. Quantos semestres são o curso?
   - 8 semestres.
3. Está em qual semestre do curso?
   - Sexto semestre.
4. Qual é o nome da instituição de ensino?
   - Faculdade Projeção.
5. Quanto tempo de estágio?
   - 1 ano.

Papel do desenvolvedor na organização

6. Como você enxerga o seu papel na instituição? (A posição dentro da instituição, as responsabilidades, e etc. . .)
   - Bastante responsabilidade, pois desenvolve alguns sistemas sozinho e é responsável desde o levantamento dos requisitos até a implantação, foi responsável pelo desenvolvimento do tema de interface que todos os departamentos utilizam em seus sistemas, fatos esses que corroboram para uma responsabilidade grande dentro da instituição.
7. Você considera a sua contribuição relevante para a instituição?
   - As matérias básicas vista na faculdade são importantes mas boa parte não se aplica para a instituição com exceção de levantamento de requisitos e lógica de programação que se faz relevante a contribuição para a instituição.

Efetividade do Treinamento

8. Você acha que o curso o preparou bem para as tarefas que realizou?
   - Não, o curso é fraco.

9. Qual a sua opinião com relação a efetividade do curso?
   - A efetividade do curso é fraca.

10. Como você avalia a qualidade das aplicações em que deu manutenção?
    - As aplicações em que deu manutenção estavam bem desenvolvidas não precisando esforço para entendimento do código, pois foram aplicações desenvolvidas por um servidor que já tem conhecimento sobre desenvolvimento e sabe da importância de um sistema ser bem desenvolvido para diminuir a complexidade de futuras manutenções.

Boas práticas e padrões

11. Você adota algum padrão ou alguma boa prática no desenvolvimento/manutenção das aplicações?
    - Procura adotar os padrões do apex, tentando não inventar novas formas de implementar.

12. Teria alguma sugestão de boa prática/padrão que realiza que poderia melhorar de alguma forma a qualidade das aplicações desenvolvidas?
    - Treinamento de boas práticas voltadas ao APEX e o sistema de desenvolvimento da instituição. Há muitos recursos do APEX que não são passados.

Requisitos

13. Como você faz para levantar os requisitos do sistema que foi solicitado por alguma unidade?
    - Como ao desenvolver um sistema normalmente a unidade se utiliza de muitas regras de negócio, então utilizava-se de gravações com o usuário durante as entrevistas, para ter os registros de detalhes importantes que normalmente não se conseguem ver de início e também não esquecer os detalhes. Também faz um esboço do que entendeu e tenta se imergir no ambiente, vivendo o negócio para o entender bem.

14. Como você aborda o cliente para levantar os requisitos do sistema?
    - Com entrevistas.

15. Como você armazena ou documenta o que foi passado pelo cliente (requisitos)? Existe algum meio de armazenar eles diretamente no sistema?
    - Utiliza a ferramenta SVN para guardar os áudios e documentos.

16. Você costuma fazer reunião com o cliente em outras etapas do desenvolvimento, para levantar outros requisitos e/ou esclarecer os que já existem?
    - Sim, realiza reuniões semanalmente. Porém normalmente já está na mesma sala que o cliente, mantendo sempre contato para sanar dúvidas, alterar requisitos, incluir novos requisitos e assim por diante.

Design

17. Como você elabora o modelo de dados do sistema (ferramentas, MER, ML)? Se nunca elaborou, considera-o importante?
    - Elabora primeiramente desenhando na mão, depois utiliza a ferramenta *Workbench* para a modelagem de dados.

18. Você valida tecnicamente este modelo com área responsável ou com algum padrão disponível (nomenclatura e formas normais)? Se nunca validou, considera-o importante?
    - Não valida o modelo de dados.

Codificação

19. Você costuma utilizar código PL/SQL, HTML e Javascript em mais da metade da aplicação?
    - Usa em mais da metade da aplicação, mas tem ciência de que para o apex isso é ruim, no entanto como precisa entregar o sistema num dado prazo acaba fazendo dessa maneira e corrigindo posteriormente numa manutenção.

20. Você procurar escrever os seus códigos PL/SQL, SQL, HTML e Javascript de forma legível?
    - Sim, pois sabe da importância de ter legibilidade de código.

21. Você segue algum padrão de codificação ou boas práticas (constantes, nomes significativos) para PL/SQL, SQL, HTML e Javascript?
    - Não, pois não tem um padrão específico da instituição a se seguir, utiliza padrões e boas práticas que julga ser correta para o desenvolvimento.

22. Você costuma usar algum editor de código ou o próprio APEX, para escrever seus códigos?
    - Usa o *sublime* para funções grandes e o próprio apex para funções que não são grandes.

23. Você costuma depurar o código?
    - Não usa porque não faz parte do escopo.

24. Você costuma classificar os erros encontrados (falha de execução ou comportamento inesperado), de forma que possa ajudá-lo na abordagem de resolução?
    - Não costuma classificar.

25. Você costuma comentar o código que faz?
    - Pouco, pois procura deixar o código legível, com exceção de códigos *jquery* e *javascript* que costuma comentar bastante.

Teste

26. Como você testa as suas aplicações?
    - Testa manualmente inserido dados de entrada e verificando se as saídas são o que se esperava ou comportamento esperado.

27. Você costuma testar todas as páginas de aplicação que desenvolve?
    - Sim, todas as páginas são testadas várias vezes.

28. Você costuma armazenar os resultados dos testes que realiza?
    - Não armazena os resultados dos testes.

29. Você costuma a elaborar casos de testes para sua aplicação?
    - Não elabora casos de teste.

30. Seria bom testes automatizados de interface para sua aplicação (selenium)?
    - Seria bom.

Implantação

31. Você costuma migrar as definições dos objetos de banco (tabelas, sequences, views e etc) junto com a aplicação?
    - Realizou o processo de migração de objetos apenas uma vez.

32. Você costuma verificar se os apelidos da aplicação de desenvolvimento e produção estão iguais?
    - Sim, costuma verificar os apelidos.

33. Realiza uma navegação por todas as páginas da aplicação?
    - Sim, navega por todas as páginas.

34. Verifica se todas as tabelas foram criadas?
    - Sim verifica as tabelas.

- **Desenvolvedor 6**

Descrição do Desenvolvedor

1. Qual é o curso que você faz?
   - Não se aplica.

2. Quantos semestres são o curso?
   - Não se aplica.

3. Está em qual semestre do curso?
   - Não se aplica.

4. Qual é o nome da instituição de ensino?
   - Não se aplica.

5. Quanto tempo de estágio?
   - Não se aplica.

Papel do desenvolvedor na organização

6. Como você enxerga o seu papel na instituição? (A posição dentro da instituição, as responsabilidades, e etc. . .)
   - Grande responsabilidade, pois é o chefe do setor de um dos 3 núcleos centrais de desenvolvimento descentralizado.

7. Você considera a sua contribuição relevante para a instituição?
   - Muito importante pois vários setores sempre solicitam o trabalho e ocorre bastante demanda para desenvolvimento de sistemas.

Efetividade do Treinamento

8. Você acha que o curso o preparou bem para as tarefas que realizou?
   - Não se aplica.

9. Qual a sua opinião com relação a efetividade do curso?
   - Não se aplica.

10. Como você avalia a qualidade das aplicações em que deu manutenção?
    - As aplicações foram bem desenvolvidas recebendo normalmente vários elogios, as manutenções que ocorrem são pontuais alterando algumas partes a pedido do usuário.

Boas práticas e padrões

11. Você adota algum padrão ou alguma boa prática no desenvolvimento/manutenção das aplicações?
    - Adota padrões próprios e alguns padrões sugeridos pela informática.

12. Teria alguma sugestão de boa prática/padrão que realiza que poderia melhorar de alguma forma a qualidade das aplicações desenvolvidas?
    - Padrões de nomenclatura de tabelas e nomes de colunas.

Requisitos

13. Como você faz para levantar os requisitos do sistema que foi solicitado por alguma unidade?
    - Faz uma reunião com o cliente e já vai desenvolvendo os requisitos obtidos como um protótipo. O protótipo desenvolvido é testado com o cliente, o que diminui muito as chances de desenvolver o sistema de forma errada.

14. Como você aborda o cliente para levantar os requisitos do sistema?
    - Sempre marca hora, agendando entrevistas com o cliente.

15. Como você armazena ou documenta o que foi passado pelo cliente (requisitos)? Existe algum meio de armazenar eles diretamente no sistema?
    - Utiliza um sistema chamado documentador para armazenamento das informações.

16. Você costuma fazer reunião com o cliente em outras etapas do desenvolvimento, para levantar outros requisitos e/ou esclarecer os que já existem?
    - Sim, realiza várias outras reuniões para ir mostrando como o sistema está ficando e trabalhando em cima do que já foi feito para chegar ao ponto que o cliente deseja.

Design

17. Como você elabora o modelo de dados do sistema (ferramentas, MER, ML)? Se nunca elaborou, considera-o importante?
    - Elabora, utilizando a ferramenta *DataModeler* para criar o modelo e posteriormente gerar os *scripts* para criação do banco de dados.

18. Você valida tecnicamente este modelo com área responsável ou com algum padrão disponível (nomenclatura e formas normais)? Se nunca validou, considera-o importante?
    - Não precisou validar o modelo de dados.

Codificação

19. Você costuma utilizar código PL/SQL, HTML e Javascript em mais da metade da aplicação?
    - Para javascript é muito pouco, pois procura seguir o que o apex já proporciona, até por questão de manutenção posterior, já os outros PLSQL e HTML usa em mais da metade.

20. Você procurar escrever os seus códigos PL/SQL, SQL, HTML e Javascript de forma legível?
    - Sim, respeitando o padrão da informática de nomenclatura.

21. Você segue algum padrão de codificação ou boas práticas (constantes, nomes significativos) para PL/SQL, SQL, HTML e Javascript?
    - Sim, segue o padrão da informática.

22. Você costuma usar algum editor de código ou o próprio APEX, para escrever seus códigos?
    - Usa o *sqldeveloper* para códigos grandes para apontar possíveis erros.

23. Você costuma depurar o código?
    - Depura mas com pouca frequência.

24. Você costuma classificar os erros encontrados (falha de execução ou comportamento inesperado), de forma que possa ajudá-lo na abordagem de resolução?
    - Não classifica.

25. Você costuma comentar o código que faz?
    - Não costuma comentar.

Teste

26. Como você testa as suas aplicações?
    - Testa manualmente navegando nas páginas.

27. Você costuma testar todas as páginas de aplicação que desenvolve?
    - Sim, testa todas as páginas.

28. Você costuma armazenar os resultados dos testes que realiza?
    - Não.

29. Você costuma a elaborar casos de testes para sua aplicação?
    - Não elabora casos de teste.

30. Seria bom testes automatizados de interface para sua aplicação (selenium)?
    - Não soube opinar.

Implantação

31. Você costuma migrar as definições dos objetos de banco (tabelas, sequences, views e etc) junto com a aplicação?
    - Sim, faz a migração dos objetos.

32. Você costuma verificar se os apelidos da aplicação de desenvolvimento e produção estão iguais?
    - Sim, verifica os apelidos.

33. Realiza uma navegação por todas as páginas da aplicação?
    - Sim, navega por todas as páginas.

34. Verifica se todas as tabelas foram criadas?
    - Sim verifica a criação das tabelas.

- **Desenvolvedor 7**

Descrição do Desenvolvedor

1. Qual é o curso que você faz?
   -Engenharia de Software.

2. Quantos semestres são o curso?
   - 10 semestres.

3. Está em qual semestre do curso?
   - Sétimo semestre.

4. Qual é o nome da instituição de ensino?
   - Universidade de Brasília.

5. Quanto tempo de estágio?
   - 1 mês.

Papel do desenvolvedor na organização

6. Como você enxerga o seu papel na instituição? (A posição dentro da instituição, as responsabilidades, e etc. . .)
   - Vejo como um prestador de serviços na área de tecnologia.

7. Você considera a sua contribuição relevante para a instituição?
   - Sim, considera importante.

Efetividade do Treinamento

8. Você acha que o curso o preparou bem para as tarefas que realizou?
   - Sim, porque o curso aborda as principais funcionalidades do apex.

9. Qual a sua opinião com relação a efetividade do curso?
   - Foi efetivo para criar e dar manutenção nos sistemas feitos em apex.

10. Como você avalia a qualidade das aplicações em que deu manutenção?
    - Considera de boa qualidade, somente quando continha códigos sql, era complicado dar manutenção.

Boas práticas e padrões

11. Você adota algum padrão ou alguma boa prática no desenvolvimento/manutenção das aplicações?
    - Não.

12. Teria alguma sugestão de boa prática/padrão que realiza que poderia melhorar de alguma forma a qualidade das aplicações desenvolvidas?
    - Comentários nos códigos sql, plsql e javascript.

Requisitos

13. Como você faz para levantar os requisitos do sistema que foi solicitado por alguma unidade?
    - Entrevista com o cliente.

14. Como você aborda o cliente para levantar os requisitos do sistema?
    - Geralmente conversamos sobre o sistema e documentamos as ideias no Descrição Geral da Aplicação (DGA).

15. Como você armazena ou documenta o que foi passado pelo cliente (requisitos)? Existe algum meio de armazenar eles diretamente no sistema?
    - Criamos um documento no Microsoft Word e compartilhamos na rede interna.

16. Você costuma fazer reunião com o cliente em outras etapas do desenvolvimento, para levantar outros requisitos e/ou esclarecer os que já existem?
    - Sim.

Design

17. Como você elabora o modelo de dados do sistema (ferramentas, MER, ML)? Se nunca elaborou, considera-o importante?
    - Fazemos o modelo de dados por meio de softwares disponíveis no órgão, ex: *Data modeler*, *mysql workbench*, *astah* e etc.

18. Você valida tecnicamente este modelo com área responsável ou com algum padrão disponível (nomenclatura e formas normais)? Se nunca validou, considera-o importante?
    - Sim, todos os documentos são validados pelo servidor responsável.

Codificação

19. Você costuma utilizar código PL/SQL, HTML e Javascript em mais da metade da aplicação?
    - Sim, em mais da metade.

20. Você procurar escrever os seus códigos PL/SQL, SQL, HTML e Javascript de forma legível?
    - Sim, tentando aplicar boas práticas que julga como corretas.

21. Você segue algum padrão de codificação ou boas práticas (constantes, nomes significativos) para PL/SQL, SQL, HTML e Javascript?
    - Sim, o padrão que julga o correto.

22. Você costuma usar algum editor de código ou o próprio APEX, para escrever seus códigos?
    - O próprio APEX.

23. Você costuma depurar o código?
    - Raramente.

24. Você costuma classificar os erros encontrados (falha de execução ou comportamento inesperado), de forma que possa ajudá-lo na abordagem de resolução?
    - Não realiza depuração.

25. Você costuma comentar o código que faz?
    - Não utiliza comentários.

Teste

26. Como você testa as suas aplicações?
    - Não chegou ainda a etapa de testes.

27. Você costuma testar todas as páginas de aplicação que desenvolve?
    - Pretende testar todas.

28. Você costuma armazenar os resultados dos testes que realiza?
    - Não chegou ainda a etapa de testes.

29. Você costuma a elaborar casos de testes para sua aplicação?
    - Não chegou ainda a etapa de testes.

30. Seria bom testes automatizados de interface para sua aplicação (selenium)?
    - Sim, seria de grande ajuda.

Implantação

31. Você costuma migrar as definições dos objetos de banco (tabelas, sequences, views e etc) junto com a aplicação?
    - Ainda não finalizou nenhuma aplicação, mas pretende.

32. Você costuma verificar se os apelidos da aplicação de desenvolvimento e produção estão iguais?
    - Não realizou migração ainda.

33. Realiza uma navegação por todas as páginas da aplicação?
    - Sim, realiza navegação por todas as páginas.

34. Verifica se todas as tabelas foram criadas?
    - Sim, verifica todas as tabelas.

- **Desenvolvedor 8**

Descrição do Desenvolvedor

1. Qual é o curso que você faz?
   -Engenharia de Redes de comunicação.

2. Quantos semestres são o curso?
   - 10 semestres.

3. Está em qual semestre do curso?
   - Décimo semestre.

4. Qual é o nome da instituição de ensino?
   - Universidade de Brasília.

5. Quanto tempo de estágio?
   - 6 meses.

Papel do desenvolvedor na organização

6. Como você enxerga o seu papel na instituição? (A posição dentro da instituição, as responsabilidades, e etc. . .)
   - Vejo como profissional atuante na parte técnica do tribunal, e o trabalho que eu faço julgo como de suma importância na automatização dos trabalhos críticos da instituição.

7. Você considera a sua contribuição relevante para a instituição?
   - Sim, pois o trabalho facilita e acelera a produtividade de outros servidores.

Efetividade do Treinamento

8. Você acha que o curso o preparou bem para as tarefas que realizou?
   -Não, porque o curso não condiz com os desafios do trabalho.

9. Qual a sua opinião com relação a efetividade do curso?
   - Considera fraco, a maior parte se aprende com a prática durante o desenvolvimento.

10. Como você avalia a qualidade das aplicações em que deu manutenção?
    - Acha bem estruturadas, pois foram desenvolvidas por um profissional já conhecedor de boas práticas.

Boas práticas e padrões

11. Você adota algum padrão ou alguma boa prática no desenvolvimento/manutenção das aplicações?
    - Usa alguns padrões próprios de codificação.

12. Teria alguma sugestão de boa prática/padrão que realiza que poderia melhorar de alguma forma a qualidade das aplicações desenvolvidas?
    - Sim, usar mais as funções pré-estabelecidas do apex.

Requisitos

13. Como você faz para levantar os requisitos do sistema que foi solicitado por alguma unidade?
    - Reunião com o cliente.

14. Como você aborda o cliente para levantar os requisitos do sistema?
    - Tenta ver como está a organização dos dados atualmente, tenta discutir melhoras da organização dos dados com o próprio cliente.

15. Como você armazena ou documenta o que foi passado pelo cliente (requisitos)? Existe algum meio de armazenar eles diretamente no sistema?
    - Armazena no próprio computador.

16. Você costuma fazer reunião com o cliente em outras etapas do desenvolvimento, para levantar outros requisitos e/ou esclarecer os que já existem?
    - Sim, com mais reuniões.

Design

17. Como você elabora o modelo de dados do sistema (ferramentas, MER, ML)? Se nunca elaborou, considera-o importante?
    - Através da ferramenta *datamodeler*.

18. Você valida tecnicamente este modelo com área responsável ou com algum padrão disponível (nomenclatura e formas normais)? Se nunca validou, considera-o importante?
    - Sim, com o supervisor.

Codificação

19. Você costuma utilizar código PL/SQL, HTML e Javascript em mais da metade da aplicação?
    - Sim, em mais da metade.

20. Você procurar escrever os seus códigos PL/SQL, SQL, HTML e Javascript de forma legível?
    - Sim, considera bastante legível.

21. Você segue algum padrão de codificação ou boas práticas (constantes, nomes significativos) para PL/SQL, SQL, HTML e Javascript?
    - Sim, padrão passado pelo setor.

22. Você costuma usar algum editor de código ou o próprio APEX, para escrever seus códigos?
    - O próprio APEX.

23. Você costuma depurar o código?
    - Não, não utiliza depuração.

24. Você costuma classificar os erros encontrados (falha de execução ou comportamento inesperado), de forma que possa ajudá-lo na abordagem de resolução?
    - Não classifica.

25. Você costuma comentar o código que faz?
    - Sim, realiza comentários.

Teste

26. Como você testa as suas aplicações?
    - Não costuma testar as aplicações com teste funcional.

27. Você costuma testar todas as páginas de aplicação que desenvolve?
    - Sim, fazendo navegação.

28. Você costuma armazenar os resultados dos testes que realiza?
    - Não realiza armazenamento de testes.

29. Você costuma a elaborar casos de testes para sua aplicação?
    - Sim, de forma mental.

30. Seria bom testes automatizados de interface para sua aplicação (selenium)?
    - Seria bom, pois melhoraria a qualidade.

Implantação

31. Você costuma migrar as definições dos objetos de banco (tabelas, sequences, views e etc) junto com a aplicação?
    - Sim, normalmente faz-se a migração de definições.

32. Você costuma verificar se os apelidos da aplicação de desenvolvimento e produção estão iguais?
    - Nunca verificou.

33. Realiza uma navegação por todas as páginas da aplicação?
    - Sim.

34. Verifica se todas as tabelas foram criadas?
    - Sim, verifica todas as tabelas.