UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CONSISTÊNCIA DE INTERFACES EM SOFTWARE LIVRE: PROCESSO E GUIA DE RECOMENDAÇÕES PARA O LMS AMADEUS

Paulo Henrique da Cruz Pereira

Trabalho de Graduação

Recife
FEVEREIRO DE 2015

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA

Paulo Henrique da Cruz Pereira

CONSISTÊNCIA DE INTERFACES EM SOFTWARE LIVRE: PROCESSO E GUIA DE RECOMENDAÇÕES PARA O LMS AMADEUS

Trabalho apresentado ao Programa de GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO do CENTRO DE INFORMÁTICA da UNIVERSIDADE FEDERAL DE PERNAMBUCO como requisito parcial para obtenção do grau de CIÊNCIA DA COMPUTAÇÃO.

Orientador(a): *Prof Dr Alex Sandro Gomes*

Recife
FEVEREIRO DE 2015

À minha família.

AGRADECIMENTOS

Meu primeiro agradecimento é a Deus, por ter, em meio a tantas pressões e incertezas, me dado possibilidades de chegar na reta final da graduação, e com forças suficientes para executar bons trabalhos.

Também agradeço à minha família, a quem devo minha vida, pois sempre possibilitaram seguir com minhas escolhas e nunca deixaram de me apoiar, mesmo nos momentos mais críticos. À minha mãe Ecilda, que com seus abraços me dá a confiança da qual preciso para seguir em frente e alcançar meus objetivos. À minha avó Inês, que junto ao meu avô José fez e faz o impossível para criar essa bela família. Ao meu irmão Júnior, que com suas curiosidades científicas me fez despertar um novo mundo nas exatas. Ao meu tio Elcy, que desde sempre é uma grande influência e me fez perceber que os estudos podem levar a um bom caminho. Ao meu pai Paulo e a minha avó Creuza, que também são muito importantes pra mim.

Agradeço à Cynthia, minha namorada, por todo seu apoio e companheirismo, por levantar minha cabeça e me fazer persistir, por ter paciência e me dar os conselhos certos nas horas certas, e por estar esse tempo todo ao meu lado, sem ela toda essa jornada teria sido mais difícil. Te amo mô!

Aos amigos de colégio, Daniel, Diogo, Felipe, Hugo, Inaldo e Luciano, os quais tive a oportunidade de passar bons momentos juntos, e apesar da vida nos distanciar tenho a certeza que nos reencontros a amizade é sempre a mesma. Aos amigos da universidade, Alberto, Bruna, Filipe, Luiz, Marcos, Jonatas e Tomaz, com quem pude compartilhar os bons momentos e os desafios que esse curso nos propôs.

Agradeço ao meu orientador Alex Sandro, que me guiou de forma atenciosa e esteve sempre disposto a ajudar no desenvolvimento desse trabalho.

E a outros tantos amigos e familiares que de alguma forma me influenciaram, me ajudaram ou passaram pela minha vida de forma positiva, sou grato.

RESUMO

Neste trabalho iremos entender fenômenos relativos à consistência de interfaces em contribuições às comunidades de *software* livre, como também estudar gargalos no processo de integração no ambiente de desenvolvimento do LMS AMADEUS. Por meio de um estudo de caso, no qual será realizada a integração de duas contribuições ao projeto em um ambiente colaborativo, iremos propor um processo de integração e um documento de diretrizes de interface. O processo será capaz de guiar as integrações realizadas por meio de um ambiente de desenvolvimento colaborativo, favorecendo a consistência de interfaces ao final da execução. Já o documento de diretrizes será necessário para padronizar a criação de novas telas no LMS AMADEUS, definindo pontos como princípios de usabilidade, posicionamento na tela e paleta de cores. Por fim faremos uma análise heurística nessas contribuições, com foco na revisão de *guidelines* e na inspeção de consistência, e com esse resultado revisaremos o código para corrigir os problemas encontrados.

Palavras-chave: floss, diretrizes de usabilidade, processo de integração, análise heurística, revisão de *guidelines*, inspeção de consistência.

ABSTRACT

In this work, we will understand phenomena related to the consistency of interfaces in contributions to free software communities, as well as studying bottlenecks in the integration process in the development environment of LMS AMADEUS. Through a case study, where we will integrate two contributions to the project in a collaborative environment, we will propose a process of integration and interface guidelines document. The process will be able to guide the integrations performed through a collaborative development environment, favouring the consistency of interfaces at the end of the process. The guidelines document will be necessary to standardize the creation of new screens in LMS AMADEUS, defining points as usability principles, positioning on the screen and colour palette. Finally, we will perform a heuristic analysis in such contributions, focusing on the revision of guidelines and inspection of consistency, and with this result, we will review the code to fix any problems found.

Keywords: consistency of interfaces, free software, floss, usability guidelines, integration process, heuristic analysis, guidelines revision, consistency inspection.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Em *software* livre, o desenvolvimento distribuído traz dificuldades no que diz respeito à manutenção de interfaces consistentes [1], acarretando na criação de problemas de usabilidade ao usuário comum. Este trabalho visa apreciar meios de evitar inconsistências de interfaces, bem como propor um processo de integração e um documento de diretrizes que corroborem com o objetivo.

1.1 MOTIVAÇÃO

O *software* livre e de código aberto (F/LOSS) unindo-se ao desenvolvimento colaborativo ganha mais adeptos a cada dia, sendo crescente o número de projetos passando a usar esse modelo [2]. Dessa forma alguns problemas se tornam mais evidentes, como o de manter a organização dos códigos e arquiteturas pré-definidas, ou mesmo o de manter a consistência de novas interfaces que serão desenvolvidas pelos colaboradores. Tudo isso se dá por, comumente, não existir um processo de integração definido dentro dos projetos de *software* livre [3].

A ausência de diretrizes que definam padrões visuais e de comportamento da interface é responsável pela dificuldade em manter a consistência das novas telas em relação ao que era esperado, e isso ocorre apenas durante a integração das colaborações realizadas por terceiros.

Esse problema se agrava principalmente quando lidamos com sistemas de uso amplo, nos quais o usuário espera ações comuns do sistema, mas uma colaboração pode não responder da forma esperada. Para reduzi-lo, as dez (10) Heurísticas de Nielsen [4] podem ser tomadas como referência, ou mesmo podem ser usadas diretrizes de mais alto nível como as definidas pelo Departamento de Saúde e Serviços Humanos (HHS) do governo dos Estados Unidos, o Usability.gov [5].

1.2 CONTEXTO

O Amadeus é um ambiente de ensino a distância inicialmente desenvolvido pelo grupo de Ciências Cognitivas e Tecnologias Educacionais (CCTE), do Centro de Informática da Universidade Federal de Pernambuco (CIn-UFPE), e posteriormente se tornou um *software* livre e de código aberto [6].

Nesse sistema os problemas citados anteriormente são realidade, pois não existe um processo de integração nem diretrizes definidas para o desenvolvimento de interfaces. Dessa forma os desenvolvedores podem tomar decisões erradas, fazendo com que as interfaces sejam

inconsistentes com o todo, principalmente em pontos específicos relacionados ao Amadeus que não estão contemplados em diretrizes mais gerais.

1.3 OBJETIVOS

Este trabalho tem como principal objetivo sugerir um processo e artefatos para contribuir com a manutenção da consistência de interfaces que serão desenvolvidas pela comunidade Amadeus.

Os objetivos específicos da proposta são:

- Estudar inconsistências de interface nas colaborações mais recentes ao Amadeus;
- Aplicar o novo ambiente de desenvolvimento colaborativo, GitHub, para gerenciamento do código do LMS AMADEUS;
- Identificar necessidades de processo para manutenção da consistência de interfaces do projeto;
- Definir um processo de manutenção da consistência de interfaces do LMS AMADEUS a partir da identificação das necessidades;
- Desenvolver um documento diretrizes para consistência de interfaces do LMS AMADEUS;
- Analisar contribuições mais recentes para o LMS AMADEUS por meio de avaliação heurística, com foco na revisão de *guidelines* e inspeção de consistência.

O resultado esperado é um documento de diretrizes para formalizar um padrão às novas interfaces e um processo para guiar as integrações realizadas por meio de um ambiente de desenvolvimento colaborativo. A partir disso faremos a avaliação em duas colaborações ao LMS AMADEUS para identificar problemas na consistência de interfaces.

2 CONSISTÊNCIA DE INTERFACES EM SOFTWARE LIVRE

Neste capítulo apresentaremos o estado da arte sobre o problema da manutenção da consistência em produtos de código aberto. A revisão da literatura tem como objetivos compreender o contexto no qual está inserido o LMS AMADEUS e explorar conceitos de usabilidade, como análise heurística e consistência de interfaces, dentro das comunidades de *software* livre.

2.1 PROJETOS DE *SOFTWARE* LIVRE

Segundo a *Free Software Foundation*, *software* livre é definido como um "*software* que respeita a liberdade e senso de comunidade dos usuários" [7]. Dessa forma é possível executar, estudar, copiar, distribuir e melhorar os códigos livremente. Essa filosofia torna a comunidade capaz de controlar os programas e o que é feito por eles, diferente de *software* proprietário, no qual uma corporação controla o programa e esse controla o usuário.

O acesso ao código-fonte é pré-requisito para um programa ser *software* livre, mas além disso é necessário permitir outras ações, como:

- Executar o programa como desejar, para qualquer propósito;
- Estudar como o programa funciona, e adaptá-lo às necessidades particulares;
- Redistribuir cópias de modo que seja possível ajudar ao próximo; e
- Distribuir cópias de suas versões modificadas a outros.

Esses projetos de *software* livre são movimentados por comunidades de pessoas que se unem com o objetivo de evoluir a solução. Geralmente as interações entre indivíduos dessas comunidades ocorre por meio de fóruns ou diretamente em ferramentas de gerenciamento de projeto e controle de código.

2.2 OS PROBLEMAS EM PROJETOS DE *SOFTWARE* LIVRE

Colaboradores de projetos de *software* livre lidam diariamente com uma série de problemas, principalmente quando são recém chegados à comunidade. REDMILES [1] estudou as dificuldades desse perfil de colaborador, no qual ele destaca as barreiras que são comumente enfrentadas, mapeadas em de uma pesquisa com desenvolvedores novos nas comunidades de *software* livre.

As barreiras encontradas foram divididas em categorias, e a principal categoria encontrada na pesquisa [1] foi a de problemas com documentação, já que novos colaboradores necessitam aprender aspectos sociais e técnicos do projeto para contribuir. Especificamente os

problemas relatados foram relacionados à ‘ausência de documentação’ sobre: a estrutura do projeto; a configuração do ambiente de trabalho; o processo de colaboração; e os padrões de *design*. Ainda assim houve relatos sobre dificuldades com a falta de comentários em código e documentação pouco clara ou defasada.

A categoria com segunda maior ocorrência de barreiras foi relacionada ao ‘conhecimento técnico’ dos novos colaboradores [1]. E dentre as barreiras relatadas estão: conhecimento prévio sobre ferramentas usadas no projeto; conhecimento prévio em sistemas de controle de versão; falta de conhecimento nas tecnologias utilizadas; curva de aprendizado nas ferramentas do projeto; e o uso correto da linguagem de programação usada.

Em seguida foram encontrados problemas na ‘interação social’ desses novos colaboradores com a comunidade já estabelecida [1]. Atraso nas respostas, respostas indelicadas, dificuldade de encontrar alguém para ajudar, uso de termos intimidadores e problemas de comunicação são as barreiras relatadas nessa categoria.

Nesse trabalho [1] houve relatos de problemas relacionados à ‘configuração do ambiente de trabalho’, assim como à dependência de plataformas específicas ou bibliotecas e à dificuldade em encontrar o código correto. Também foram percebidos ‘problemas de código’, relacionados à má qualidade dos códigos existentes, ao tamanho da base de códigos, à existência de códigos defasados, aos problemas no entendimento dos códigos e à ausência de padrões de código.

Ainda houve uma categoria sobre a dificuldade dos novos colaboradores em ‘encontrar uma forma de começar a contribuir’ [1]. Nela as barreiras relatadas foram: encontrar o melhor trecho de código para trabalhar; lista de *bugs* defasada; e encontrar uma tarefa para começar. E por fim, na categoria de ‘comportamento do colaborador’ foram encontradas duas barreiras: ausência de *commits* e subestimar o desafio.

2.3 CONSISTÊNCIA DE INTERFACES EM PROJETOS DE *SOFTWARE* LIVRE

A manutenção da consistência de interfaces significa que novas telas estão de acordo com o todo do projeto, tanto visualmente quanto em termos de fluxo de telas e usabilidade [8]. Por exemplo: botões que são esperados em um lado da tela devem ser mantidos do mesmo lado em novas telas onde se faça necessário; e cores usadas representando sucesso em uma tela devem ser usadas como sucesso em todas as telas onde também seja preciso.

Apesar de ser crescente o pensamento de que a usabilidade tem papel fundamental na concepção e desenvolvimento de projetos de TI, as comunidades de *software* livre tendem a ter maior resistência na adoção desse tipo de filosofia. RAJANEN [9] enumerou as características

da filosofia do *software* livre, das quais "*Talk is cheap, show me the code*" ("Falar é fácil, mostre-me o código", em português) é um indicativo de que o desenvolvimento centrado no usuário e preocupado com a consistência de interfaces não é prioridade.

A afirmativa de que sistemas de código aberto tem interfaces pouco refinadas, em partes é verdade, pois geralmente esses projetos são feitos de engenheiros para engenheiros [10]. Existem poucos profissionais especialistas em usabilidade dentro desse ecossistema, por isso um número reduzido de práticas de usabilidade é aplicado.

Ainda assim há exemplos de projetos de *software* livre que elaboram ações de modo a mitigar problemas relacionados à usabilidade e mais especificamente à consistência de interfaces. Na seção 2.4 serão descritas algumas dessas tentativas em projetos e pesquisas relacionados.

2.4 TENTATIVAS DE SOLUÇÃO: USABILIDADE EM PROJETOS DE *SOFTWARE* LIVRE

Como garantir a qualidade no desenvolvimento de interfaces de colaborações da comunidade? Existem tentativas de solucionar ou ao menos reduzir esse problema, tais quais diretrizes de produto, análise heurística e outras soluções, que serão detalhadas a seguir.

2.4.1 Diretrizes de usabilidade

Documentos de diretrizes de usabilidade, também chamados de *guidelines* em inglês, têm como principal objetivo melhorar a usabilidade na interação humano-computador [11]. *Guidelines* são baseadas em teorias, dados empíricos e experiências práticas, que unidos são capazes de aconselhar as pessoas de como algo deve ser feito ou como deve ser [12].

Em *software* livre há alguns projetos que já utilizam esse tipo de documentação. Este é o caso do Openredu [13], plataforma educacional que possui no documento de *guidelines* os princípios nos quais a interface se baseou, especificações de cores, modelo de corpo, tipografia, tabelas, listas, formulários, botões, navegação, alertas, janelas e ícones.

No caso do FirefoxOS [14], sistema operacional para dispositivos móveis, há *guidelines* bem definidas para o desenvolvimento de aplicações que rodem nessa plataforma, como informações sobre paleta de cores, tipografia, cabeçalhos, planos de fundo, listas, botões, barras de ferramenta, campos de texto, seletores e ícones.

Já para o Android Lollipop [15], outro sistema operacional móvel, existe uma vasta documentação sobre o estilo de interfaces *Material Design*. Por meio do site oficial [15], é possível entender quais os objetivos e princípios desse novo *design*, bem como conhecer estilos

de cor e fontes pré-definidos, padrões de interação por meio da tela de toque e componentes disponíveis na plataforma.

Com isso fica entendido que diretrizes de usabilidade são um caminho coerente e aceito por grandes plataformas como meio para a manutenção da consistência de interfaces. Portanto as diretrizes devem ser consideradas como parte de uma solução para problemas de usabilidade em *software* livre.

2.4.2 Análise heurística

A análise heurística é uma técnica de avaliação de usabilidade desenvolvida por NIELSEN [16]. Nela os especialistas são orientados a partir de uma série de princípios de usabilidade chamados de heurísticas, de modo a avaliarem se os componentes de interface estão de acordo com esses princípios. Essas heurísticas muito se assemelham às recomendações ou aos princípios de *design* de alto nível e o conjunto original surgiu a partir da análise de duzentos e quarenta e nove (249) problemas de usabilidade [8].

PREECE [8], destacou dez (10) dessas heurísticas, são elas: visibilidade de status do sistema; compatibilidade do sistema com o mundo real; controle do usuário e liberdade; consistência e padrões; ajudar os usuários a reconhecer, diagnosticar e corrigir erros; prevenção de erros; reconhecer, em vez de relembrar; flexibilidade e eficiência no uso; estética e *design* minimalista; e ajuda e documentação.

Porém a especificidade de alguns produtos torna necessário a escolha de um outro subconjunto de heurísticas ou a elaboração de novas heurísticas. É possível adaptar as heurísticas de Nielsen, ou mesmo usar documentos de requisitos, pesquisas de mercado e recomendações de *design* como base na criação de heurísticas [8]. Em seguida é reunido um conjunto de heurísticas que se encaixa no contexto do produto e com a finalidade de obter um melhor resultado na análise.

Definido o conjunto de heurísticas, os avaliadores passam a utilizar o produto se apropriando da visão de um usuário comum, e anotam todos os problemas encontrados. Durante esse processo as heurísticas têm como finalidade focar a atenção dos especialistas em determinados pontos, por isso a escolha de heurísticas apropriadas é fundamental na análise.

É possível usar uma infinidade de avaliadores nessa técnica, porém existem evidências empíricas indicando que cinco avaliadores detectam 75% dos erros de usabilidade existentes no produto [8]. Os restantes dos erros são tidos como os mais críticos e geralmente só são identificados pelo usuário, por isso é importante o uso de outras técnicas que o incluam. A

principal vantagem desse método é o baixo custo, pois não é necessário nenhum contato direto com o usuário, reduzindo assim questões éticas e logísticas.

A análise heurística possui três estágios [8]. No primeiro, os especialistas são orientados sobre o que fazer. Nessa fase é importante garantir que todos tenham as mesmas informações. Na segunda etapa a avaliação ocorre. Nela cada especialista inspeciona independentemente o produto durante uma ou duas horas, usando as heurísticas como guia. É importante que os testes sejam realizados ao menos duas vezes: a primeira para entender o fluxo do produto e a segunda com um olhar mais atento aos componentes de interface e aos possíveis problemas de usabilidade. Tudo o que for encontrado deve ser anotado detalhadamente, pois na terceira etapa os especialistas discutem esses problemas, priorizam e sugerem soluções.

RAJANEN [17], propôs a introdução de técnicas de usabilidade em projetos de *software* livre. Nele executou um experimento com um jogo desenvolvido em código aberto. Nesse experimento os desenvolvedores puderam executar uma análise heurística usando um formato pré-definido, resultando em 30 erros de usabilidade encontrados. Isso aconteceu sem passar por uma validação com usuários reais, ou seja, em casos similares a esse, quando o usuário real fosse acionado para realizar testes o produto conteria menos erros comuns e o foco estaria no refinamento da usabilidade.

A análise heurística se mostra um método barato e eficaz no que diz respeito a identificação de erros mapeados previstos por princípios e diretrizes de usabilidade. Mesmo não sendo uma solução completa para problemas em *software* livre, esse método se mostra viável para entender se novas interfaces estão consistentes com os padrões pré-definidos.

2.4.3 Outras soluções

Em um projeto de *software* a documentação tem papel fundamental para uma evolução coerente, e ainda mais quando falamos em desenvolvimento distribuído, porém a ausência de documentação é uma barreira nesse processo, sendo esse um dos principais problemas, conforme citado no seção 2.2. Existem propostas que objetivam melhorar o processo da elaboração de documentos nesse tipo de projeto, como no trabalho de MEDEIROS [18], no qual é analisado o impacto do uso de *storyboards*, técnica de prototipação que descreve de forma visual e narrativa o fluxo de ações, como meio de melhorar a qualidade das atividades de documentação e validação de requisitos.

Nessa pesquisa o *storyboard* funcionou como guia durante a etapa de elicitação de requisitos. A partir do ponto que ele não sofreu mais modificações, essa etapa foi dada como concluída. Tendo esse *storyboard* como entrada para a fase de documentação foi possível

compartilhar o entendimento de forma mais efetiva entre equipes que trabalham a distância [18]. Porém, apesar de ser uma boa solução para projetos distribuídos de corporações, no ambiente do *software* livre há dificuldades de aplicação, pois não há um cliente específico e o grau de distribuição dos desenvolvedores é elevado [19].

Ainda há soluções como o *Bootstrap* [20], um *framework* criado pelo Twitter[1] que facilita a implementação de interfaces mais amigáveis e modernas, inclusive adaptáveis para *smartphones* e *tablets*, por meio do recurso de responsividade. Nele há uma documentação extensa sobre componentes e classes CSS disponíveis para utilização, também há exemplos do uso desses componentes tanto de forma visual como a implementação em código HTML.

Dessa forma o *framework* permite a manutenção da consistência de interfaces pelo padrão visual pré-determinado ou por meio de temas adicionados posteriormente. Mas mesmo com a padronização há projetos que não podem usar o *Bootstrap* e portanto têm mais dificuldade em desenvolver interfaces consistentes. São exemplos projetos que necessitam da criação de novos componentes, possuem *design* muito particular, possuem *design* legado ou têm conflitos técnicos que dificultam a aplicação do *framework*.

2.5 TENTATIVAS DE SOLUÇÃO: FLUXO DE COLABORAÇÃO EM PROJETOS DE *SOFTWARE* LIVRE

Em *software* livre o processo de colaboração é descentralizado, ocorrendo por meio das comunidades, e por isso existe uma grande dificuldade no controle do que está sendo feito. Porém existem formas de mitigar esse problema, podendo ser pelo uso de sistemas de controle de versão e/ou de ferramentas que integram o código de forma social. A seguir detalhamos algumas formas.

2.5.1 Versionamento com Git

Controle de versão é um sistema que grava mudanças feitas em arquivos durante o tempo, dessa forma é possível recuperar versões anteriores de um momento específico [21]. Há vários métodos para realizar esse controle, podendo ser feito localmente (*Local Version Control Systems*), numa base centralizada em um servidor (*Centralized Version Control Systems*) ou de forma distribuída (*Distributed Version Control Systems*). Essa última abordagem foi a escolhida para o Git e permite uma série de fluxos impossíveis em outros métodos, pois o usuário mantém

[1] Disponível em <https://twitter.com/ >. Acesso em 7 de novembro de 2014.

uma base local com suas mudanças e pode sincronizá-la com a base do servidor a qualquer momento.

Depois de instalado, o Git passa a funcionar na linha de comando do sistema operacional por meio do `git <command> [<args>]`. O primeiro passo com o Git é iniciar um repositório, com o comando `git init`, ou clonar um repositório existente com o `git clone <repo>`, no qual também passamos a *url* do repositório como parâmetro do comando [21].

Com isso as contribuições já podem ser desenvolvidas no ambiente de trabalho do desenvolvedor, que deve enviar as modificações para a base local à medida que houver progressos funcionais. Isso deve ser feito por meio do comando `git commit [options] <pathspec>`, sendo mais comum da seguinte forma `git commit -am <message>` no qual todas as modificações são enviadas (`-a`) e uma mensagem (`-m`) descreve o *commit* [21].

Para adicionar novos arquivos ao controle de versão é necessário executar o comando `git add [options] <pathspec>`, no qual para um arquivo específico é necessário passar o caminho após o *add*. Já para adicionar todos os novos arquivos encontrados no repositório deve ser passada apenas opção `–A` ou `--all` [21]. Também é possível conhecer o status da base local, obtendo um relatório de arquivos modificados, adicionados ou removidos, e isso pode ser feito por meio do `git status`.

Para enviar os *commits* ao repositório remoto é necessário executar o comando `git push origin master`, no qual *'origin'* é a *branch* local atual e '*master'* é a *branch* principal no servidor. Quando se deseja obter novos arquivos do servidor deve ser executado o `git pull origin master`. Já para manter o repositório local atualizado com arquivos de um repositório acima deve ser feito o *rebase*, geralmente por meio de `git pull --rebase upstream master`.

Nesse contexto ainda há o conceito de *branch*, que são caminhos nos quais o desenvolvimento pode se dar de forma paralela, sendo um deles o principal, a *branch master*. Essas *branches* podem ser unidas (*merge*) ou substituir outras a qualquer momento. Para criá-las é necessário o comando `git checkout –b <name>` indicando o nome da nova *branch* como argumento.

Todos esses recursos objetivam um melhor controle de código, facilitando a integração e a evolução do projeto. Porém não existe uma garantia que novas interfaces integradas sejam consistentes com as anteriores. Portanto, o Git não pode ser considerado uma solução completa, mesmo sendo uma importante ferramenta no desenvolvimento de aplicações.

2.5.2 Colaboração por meio do GitHub

GitHub [22] é um serviço *web* que permite o armazenamento de repositórios de projetos com o controle de versão Git. Nele também existe uma grande interação entre os membros das comunidades, pois são disponibilizados fóruns, *wiki* e gerenciamento de falhas e melhorias em uma área chamada de *issues*.

No serviço há uma série de recursos que facilitam o uso do Git, como a criação de *branches*, *pull requests* e *merges* automáticos, além de ser possível editar arquivos diretamente pelo site, no qual já é efetuado um *commit* com as mudanças realizadas [22]. Além disso, quando *commits* são realizados o GitHub cria *threads*, ou conversas, no qual é possível comentar as modificações do código e interagir com outros desenvolvedores.

Ao acessar repositórios públicos no *site* do serviço é possível visualizar a opção *'fork'*, que ao ser acionada cria uma versão paralela do projeto administrada pelo próprio usuário [22]. Dessa forma melhorias podem ser realizadas mesmo sem o acesso de edição ao repositório original, com isso as contribuições podem ser submetidas posteriormente por meio do *'pull request'*, notificando ao administrador do repositório original que existem novas contribuições aguardando integração.

Sozinho o GitHub não garante que todos os problemas de *software* livre sejam resolvidos, principalmente os relacionados à usabilidade, que dependem muito de padrões bem documentados e análise das contribuições por especialistas. Portanto a adoção desses padrões de diretrizes e processos é uma decisão de projeto, cabendo aos administradores do *software* definir ou não de que forma os colaboradores devem contribuir.

3 MÉTODO

O desenvolvimento deste trabalho ocorreu em diversas etapas. O primeiro passo foi realizar uma revisão de literatura sobre os tópicos trabalhados, em seguida foi feito um estudo de caso de duas contribuições ao LMS AMADEUS. A partir disso foram elaborados um processo, representando de forma estruturada o que deve ocorrer durante a integração das contribuições, e um documento de diretrizes, detalhando os padrões de interface seguidos pelo *software*.

3.1 OBJETIVOS

Cada um dos métodos descritos as seções 3.2 e 3.3 foi importante para atingir os objetivos dessa etapa:

- Entender fenômenos relativos à proposição de contribuições da comunidade e perceber gargalos no processo de integração por meio do ambiente de desenvolvimento escolhido;
- Elaborar e avaliar um processo gestão de configuração que corresponda à estrutura de práticas em comunidade de *software* livre; e
- Conceber um documento de diretrizes para o produto LMS Amadeus de modo a permitir a realização da análise heurística, com foco na revisão de *guidelines* e inspeção de consistência.

3.2 REVISÃO DA LITERATURA

A revisão de literatura deste trabalho foi feita de três formas: a partir da busca, usando palavras-chave relacionadas ao tema, em páginas que reúnem publicações acadêmicas; por meio publicações aceitas em conferências de *software* livre e usabilidade; e por meio de livros relacionados à usabilidade, análise heurística, *software* livre e gerência de código.

A principal base de publicações usada na revisão de literatura foi a biblioteca digital do portal ACM, que se autodescreve como a maior sociedade informática educativa e científica do mundo. Por meio dele foi possível encontrar publicações feitas em *workshops*, conferências e revistas em todo o mundo, e neste trabalho mais de 70% da revisão de literatura foi encontrada nele.

Dentro do tema abordado alguns tópicos são os considerados mais relevantes: análise heurística, diretrizes de usabilidade, fluxo de contribuição do Git, processo de integração e *software* livre. O quadro 1 relaciona cada tópico com palavras-chave em inglês.

Quadro 1. Tópicos e palavras-chave usados na revisão de literatura

Tópico	Palavras-chave
Análise heurística	*heuristic analysis;* *heuristic evaluation;*
Diretrizes de usabilidade	*usability guidelines;* *product guidelines;* *user interface guidelines;*
Fluxo de contribuição do Git	*git contributions workflow;* *git workflow;*
Processo de integração	*integration process;*
Software livre	*free software;* *open source;* *floss;*

Fonte: O autor.

Todas as palavras-chave foram pesquisadas na biblioteca digital do ACM. De forma isolada os tópicos de análise heurística, diretrizes de usabilidade e processo de integração trouxeram dezenas de milhares de opções. Dessa forma foi feito uma restrição adicionando palavras-chave relacionadas ao tópico de *software* livre, como por exemplo '*heuristic analysis floss*' que retorna 80 artigos mais bem relacionados ao tema deste trabalho.

Ainda assim, artigos relacionados ao fluxo de contribuição do Git em geral não possuíam informações relevantes. Dessa forma a pesquisa foi estendida para a *web*, através de buscadores, onde foi possível encontrar conteúdo detalhado sobre o tópico na documentação oficial do Git [21].

De forma complementar foram utilizados dois livros clássicos relacionados à Engenharia de Software e Usabilidade. O primeiro deles foi '*Software Engineering: A Practitioner's Approach*' de PRESSMAN (2009) [23], e o segundo foi '*Design* de Interação - Além da Interação Homem-computador' de PREECE (2005) [8].

Ainda na revisão de literatura as conferências se mostraram úteis como forma de encontrar trabalhos relacionados de forma mais prática. Há exemplos como a 'AcademicMindTrek '13' no qual RAJANEN (2013) [9] publicou um artigo discutindo as práticas de usabilidade em projetos de *software* livre, sendo esse um trabalho fundamental para aprofundar conhecimentos no tema.

3.3 ESTUDOS DE CASO

Para realizar o estudo de caso foram analisadas contribuições dos colaboradores MEDEIROS [24] e PERRIS [25], ambas integradas à versão 0.9.x do AMADEUS, disponível

no portal do *Software* Público Brasileiro [26], possibilitando vivenciar atividades realizadas por um gerente de configuração durante a integração de contribuições.

Com o objetivo de aprofundar o entendimento nas interações sociais, o trabalho de MEDEIROS propõe formas de monitoramento das redes sociais dentro do ambiente educacional [24]. Nesse contexto foram desenvolvidas três ferramentas: fórum de discussão, mensagens e integração às redes sociais (Facebook[2] e Twitter[3]). Essas ferramentas foram desenvolvidas com base nos requisitos funcionais listados abaixo:

- Fórum de discussão:
    - Somente o professor ou tutor cria um tópico, inicialmente. Os aprendizes comentam o tópico ou os comentários dos outros aprendizes, dependendo do seu objetivo;
    - Estruturar o fórum através do aninhamento das mensagens comentadas, de forma a ficar claro para o usuário quem respondeu quem. Possibilitar que somente o professor ou tutor criem o primeiro tópico e todos os envolvidos no processo de ensino aprendizagem possam comentar de forma irrestrita qualquer um dos comentários postados naquele fórum, promovendo um aumentodas interações sociais;
    - Apresentar na descrição do fórum a quantidade de mensagens ainda não lidas naquele fórum;
    - Possibilitar que o aprendiz ou o professor possam transportar, por meio de um simples clique, uma postagem do fórum para a rede social Twitter e/ou Facebook. Haverá a necessidade de uma configuração das associações do *login* na plataforma, com o *login* do Twitter ou Facebook;
- Mensagens:
    - Percepção em tempo real dos usuários *online* e dos usuários que não estão *online*. O primeiro nome que aparece é ‘Todos’. A lista não deve aparecer totalmente na tela. Deverá ser reservado um espaço para uma quantidade de nomes e o restante seria alcançados por meio de *scrollbars*;
    - Ao clicar em um dos usuários da lista, esteja *online* ou não, uma caixa de mensagem é aberta abaixo do nome do usuário para que seja possível enviar uma mensagem. Há um espaço para colocar o título e outro para o corpo da

[2] Disponível em <https://www.facebook.com/>. Acesso em 30 de janeiro de 2015.
[3] Disponível em <https://twitter.com/ >. Acesso em 30 de janeiro de 2015.

mensagem. Deverá haver um botão ‘Enviar’ abaixo da caixa de mensagem e também um ‘Cancelar’;

- Possibilidade de enviar uma mensagem para todos os usuários do curso. Basta clicar no nome ‘Todos’ e enviar a mensagem como se estivesse enviando para somente um usuário;
- As mensagens recebidas estarão disponibilizadas em uma caixa abaixo dos participantes do curso. Em cada mensagem aparece quem a enviou, a data e a hora. Nesta caixa aparecem somente as 10 últimas mensagens. Abaixo das mensagens, há um *link* para visualização de todas as mensagens, que serão abertas em outra tela. Quando uma mensagem for clicada, o corpo da mensagem aparece. Abaixo da mensagem aberta há botões para ‘Excluir’ e ‘Responder’ ao emissor;

- Integração às redes sociais:
  - Integrar o AMADEUS com ferramentas de redes sociais *online*, oferecendo meios de associar o *login* do usuário do AMADEUS com os *logins* das redes sociais *online*;
  - Devido à integração, ao entrar no AMADEUS, o usuário automaticamente e individualmente visualiza os posts relacionados à rede social cadastrada. A atualização da visualização dentro do AMADEUS deve ser automática.

Já no outro trabalho, como forma de criar novos meios de interação no ambiente de ensino a distância, PERRIS contribuiu com um módulo de grupos para o LMS AMADEUS [25]. Dessa forma foram definidos uma série de requisitos funcionais e não-funcionais com os quais o desenvolvimento da colaboração foi pautado:

- Requisitos não-funcionais:
  - Indicar integrantes nos grupos; e
  - Status da atividade (grupo ou usuário);
- Requisitos funcionais:
  - Habilitar a opção de criar grupos para o discente;
  - Criar grupos;
  - Habilitar tutor/moderador aos grupos;
  - Modificar/editar grupo;
  - Deletar grupo;
  - Dar percepção ao desempenho ou participação do grupo;

- Comparação de desempenho entre grupos;
- Listar usuários cadastrados no curso;
- Log de acessos;
- Andamento das atividades;
- *Timeline* (linha do tempo); e
- Relatório das atividades.

Comparando os códigos dessas colaborações com o código da versão 0.9.x do AMADEUS foi possível identificar, em termos de código, o tamanho das contribuições, o que nos ajudou a decidir qual seria a primeira a ser integrada. Dessa forma vimos que o código de PERRIS modificou cinquenta e sete (57) arquivos no projeto. Já o de MEDEIROS modificou mais de cem (100) arquivos de código.

Também foi analisada a consistência das contribuições entre si e com a última versão de modo a perceber se havia um padrão de interfaces coerente. Esse tipo de informação contribuiu para a elaboração de um processo de integração e de um documento de diretrizes para ser usado como base de uma análise heurística executada dentro da integração.

4 RESULTADOS

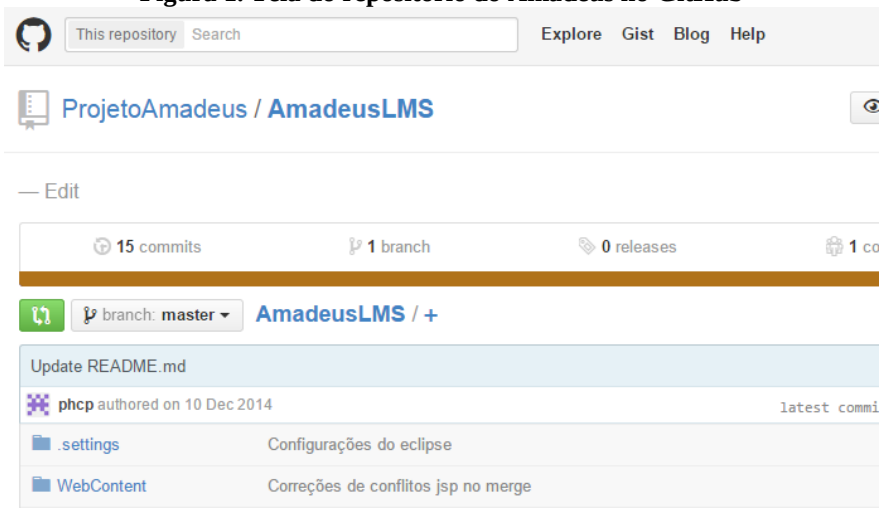
Neste capítulo serão apresentados os resultados obtidos na pesquisa, sendo eles o estudo de caso de duas contribuições recentes da comunidade, um processo para manutenção da consistência de interfaces e um documento de diretrizes para o LMS AMADEUS.

4.1 ANÁLISE DAS CONTRIBUIÇÕES

De modo a aprofundar o entendimento do processo de colaboração e integração do LMS AMADEUS, realizamos um estudo de caso com duas contribuições recentes da comunidade, baseadas na versão 0.9.x disponível no Portal SPB [26]. Todo o fluxo ocorreu com o uso do Git e da plataforma GitHub, sugeridos como novas ferramentas para a evolução e integração de códigos.

No estudo de caso foram executados todos os passos necessários para realizar a integração de códigos por meio do ambiente escolhido. Primeiro foi criada a organização 'ProjetoAmadeus' para abrigar o repositório 'AmadeusLMS', como pode ser visto na Figura 1, onde foi colocada a última versão estável do AMADEUS, disponível no portal do *Software* Público Brasileiro.

Figura 1. Tela do repositório do Amadeus no GitHub

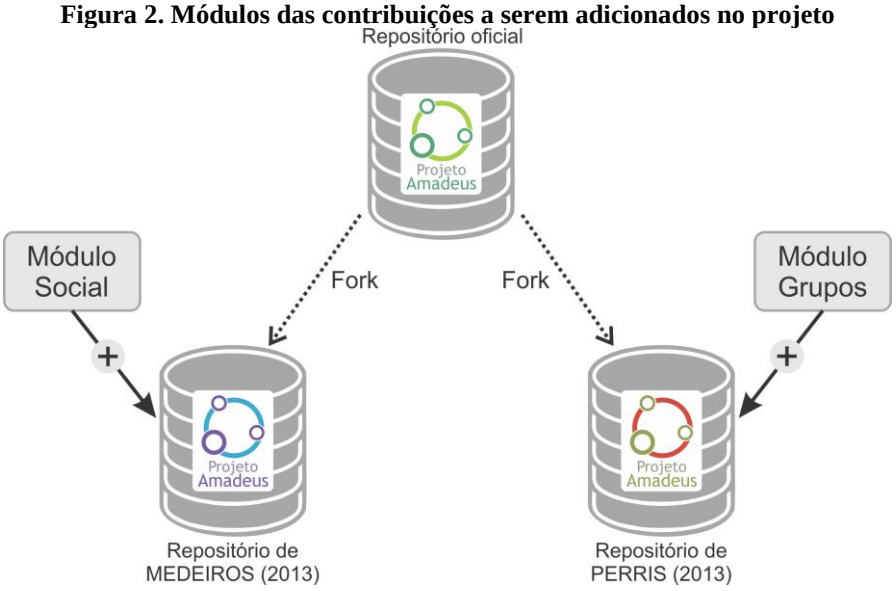


Fonte: Repositório do LMS AMADEUS no GitHub[4].

Como as contribuições eram baseadas na mesma versão foi necessário criarmos um *fork* para cada uma delas e adicionar as contribuições (ver Figura 2). Após isso executamos um *pull request* que foi integrado ao repositório principal após a resolução de uma série de conflitos.

[4] Disponível em <https://github.com/ProjetoAmadeus/AmadeusLMS>. Acesso em 17 de outubro de 2014.

Figura 2. Módulos das contribuições a serem adicionados no projeto



Fonte: O autor.

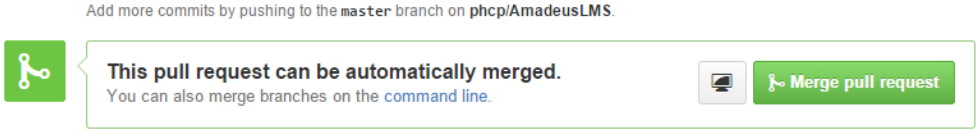
Todos os passos citados anteriormente foram executados com base nas práticas de uso do GitHub e na documentação do Git. Dessa forma foi possível vivenciar a integração de um projeto sem a análise da consistência, perceber de que forma isso se encaixa no fluxo e propor um processo para a manutenção da consistência dessas interfaces. A seguir detalhamos os resultados do estudo de caso de cada contribuição.

4.1.1 Integração da contribuição de MEDEIROS (2013)

Para realizar o estudo de caso da contribuição de MEDEIROS [24], foi necessário criar um *fork* do repositório oficial na conta temporária ‘franciscotemp’. A partir disso realizamos um *clone* para a máquina local de modo a possibilitar a manipulação do projeto. Em seguida todos os arquivos do projeto foram substituídos pela versão da contribuição, o projeto foi testado e um *commit* foi realizado localmente. Após isso houve a sincronização com o repositório *master* e foi realizado um pedido de integração por meio da função *pull request* no GitHub.

A partir disso nos colocamos na posição de gerente de configuração e começamos a analisar a solicitação por meio do relatório de diferenças fornecido pelo serviço. Essa primeira análise foi bastante confusa, já que houve apenas um *commit* com todas as mudanças e muitos dos arquivos não tiveram as diferenças identificadas, tendo sido dados como totalmente modificados.

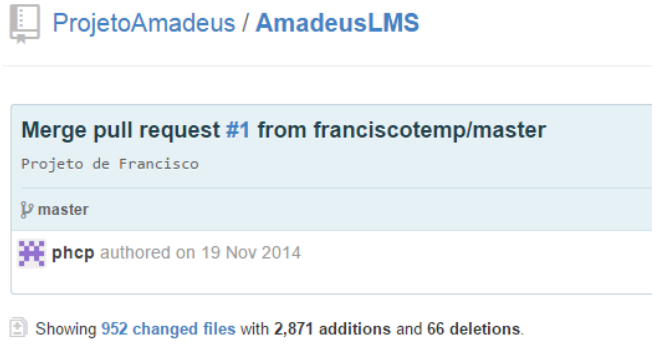
Figura 3. Opção de *merge* automático através do GitHub



Fonte: Repositório do LMS AMADEUS no GitHub.

Como a versão do repositório oficial não tinha modificações, o pedido de integração da contribuição de MEDEIROS pôde ser integrado automaticamente por meio do GitHub (ver Figura 3). Dessa forma a *branch* principal do repositório passou a conter uma nova versão, com um *merge* contendo o módulo social (ver Figura 4). Nesse ponto percebemos que poderia ser importante criar uma *branch* para realizar a integração antes de colocar na *master*, já que em uma situação real a contribuição poderia não ter sido devidamente testada.

Figura 4. *Merge* da contribuição de MEDEIROS concluído



Fonte: Repositório do LMS AMADEUS no GitHub.

4.1.2 Integração da contribuição de PERRIS (2013)

No segundo caso realizamos o estudo da contribuição de PERRIS, no qual também foi preciso criar um *fork* do repositório oficial (ainda sem a integração da versão de MEDEIROS) com a conta temporária ‘perristemp’, em seguida fizemos um clone na máquina local para adicionar as contribuições.

A partir disso todos os arquivos do projeto foram trocados pelos novos arquivos e testes locais foram realizados. Nesse caso foi preciso alterar informações de conexão com o banco de dados, já que não estavam no padrão do projeto original. Por fim, fizemos um *commit* com toda a contribuição, sincronizando com a *master* e realizando o pedido de integração.

Passando para o papel do gerente de configuração foi preciso ter maior cautela, pois a integração anterior já havia sido feita, acarretando a mudança de versão, ou seja, a versão base da contribuição de PERRIS estava defasada. Nesse caso haviam duas soluções possíveis: solicitar ao colaborador que realizasse o *rebase*, atualizando seu projeto com o que há de novo

no repositório oficial, ou realizar um *merge* e corrigir possíveis problemas numa *branch* alternativa.

A escolha nesse caso foi criar uma nova *branch* para realizar o *merge*, pois os problemas de *merge* seriam idênticos nos dois casos, a única mudança é em quem realizaria o procedimento. O próprio GitHub identificou essa necessidade por conta da versão defasada e sugeriu executar os comandos da Figura 5, primeiro realizando o download da versão *master* (com as contribuições de MEDEIROS) para a *branch* alternativa 'perristemp-master' e depois executando um *merge* (*git pull*) com a versão do repositório de PERRIS.

Figura 5. Comandos do Git para realizar merge com a contribuição de PERRIS

```
git checkout -b perristemp-master master
git pull https://github.com/perristemp/AmadeusLMS.git master
```

Fonte: O autor.

Ao final desse processo, executado por meio do terminal, o Git informou quantos conflitos ocorreram, no caso oitenta e três (83). Dessa forma iniciamos as correções dos conflitos para obter uma versão estável do projeto com as duas contribuições, o que foi alcançado após alguns dias. Por fim essa versão integrada foi colocada na *branch master*, gerando assim uma nova versão, com as duas contribuições.

4.2 PROCESSO DE INTEGRAÇÃO DE CONTRIBUIÇÕES DA COMUNIDADE AMADEUS

A integração das contribuições com foco em manter consistências de interfaces é extensa, o que torna importante a elaboração de um processo para destacar as atividades dos envolvidos nas colaborações e do gerente de configuração.

O processo de integração detalhado abaixo explicita o fluxo de ações para que uma contribuição seja desenvolvida e integrada com o repositório principal do LMS AMADEUS. Também são destacados os atores que participam desse processo, o ambiente utilizado e os documentos e ferramentas relevantes para a execução do mesmo. Esta elaboração foi realizada com base no fluxo tradicional de colaborações do ambiente previamente definido para ser usado no projeto.

4.2.1 Atores

Dentro desse processo de integração há dois atores principais, mas um terceiro pode ser considerado: o gerente de configuração, o colaborador externo e o colaborador interno.

O gerente de configuração detém o controle sob o repositório oficial e normalmente é um grande entendedor do código e da arquitetura do sistema. Porém, no caso do AMADEUS, ele também precisa estar atento a detalhes relacionados à interface, com o auxílio de uma documentação específica. A principal função desse ator é integrar códigos das contribuições de terceiros, garantindo a qualidade e a integridade do sistema, tanto em termos funcionais quanto em relação à interface.

Pressman, em *Software Engineering: A Practitioner's Approach* [23], define de forma mais abrangente o conceito de gerência de configuração como:

> *"...conjunto de atividades projetadas para controlar as mudanças pela identificação dos produtos do trabalho que serão alterados, estabelecendo um relacionamento entre eles, definindo o mecanismo para o gerenciamento de diferentes versões destes produtos, controlando as mudanças impostas, e auditando e relatando as mudanças realizadas."*

O colaborador externo é aquele que cria uma versão particular do repositório oficial, por meio do *fork*, no qual desenvolve todas as alterações que achar necessário para corrigir um *bug* ou acrescentar uma nova funcionalidade. A principal função dele é evoluir o código tendo em vista os interesses globais do projeto, de tal forma que essas mudanças podem eventualmente ser incluídas na versão oficial após uma avaliação positiva do gerente de configuração.

Já o colaborador interno está ligado diretamente ao repositório oficial. Assim, ele é capaz de publicar alterações sem a necessidade de passar pelo crivo do gerente de configuração. Nesse caso é de extrema importância que o colaborador tenha amplo conhecimento sobre o sistema, desde a arquitetura até a padrões de desenvolvimento e de interfaces.

4.2.2 Ambiente

O ambiente de colaboração usado nesse processo é o GitHub [22], um serviço *web* que permite o armazenamento de repositórios de projetos, *open source* ou proprietários, com o controle de versão Git. O serviço também possibilita grande interação entre os membros das comunidades, pois são disponibilizados fóruns, *wiki* e gerenciamento de *issues*.

Sendo assim os atores podem se comunicar em caso de dúvidas ou correções, como por exemplo: um colaborador realiza alterações indevidas e as submete ao repositório oficial, consequentemente o gerente de configuração não aceita a colaboração e tem a possibilidade de informar ao colaborador que alterações precisam ser realizadas por meio de um fórum criado apenas para essa contribuição.

Outra possibilidade é de um colaborador obter acesso aos objetivos, a visão e às necessidades do projeto para manutenção e evolução, isso por meio da *wiki* que possui

documentos criados pelos colaboradores internos. Para um acompanhamento detalhado o gerente de configuração pode obter acesso a todas as operações já realizadas dentro do repositório e a relatórios sobre a atividade dos colaboradores.

4.2.3 Documentos

Durante o processo, os colaboradores irão se confrontar com um conjunto de *issues* no GitHub, que são erros e pontos de melhorias do sistema. Em cada *issue* registrada há uma conversa no qual é possível entrar em contato com os desenvolvedores oficiais e a comunidade, tirar dúvidas e sugerir soluções. A análise dessa documentação e do histórico de conversas com outros colaboradores, principalmente sobre as *issues* das quais deseja tratar, são de grande importância para gerar colaborações pertinentes.

Os códigos também podem ser considerados documentos e são fundamentais para o entendimento do projeto em termos técnicos. Dessa forma todos os códigos estão disponíveis para acesso por meio do ambiente de colaboração, assim como o histórico de modificações realizadas em cada um deles.

Além disso estará disponível um documento de diretrizes do AMADEUS, necessário para que os colaboradores entendam de que forma devem usar determinados componentes e qual o padrão visual adotado pelo sistema. O intuito desse documento, que será detalhado mais adiante, na seção 4.3, é minimizar problemas com a inconsistência das novas interfaces a serem desenvolvidas.

4.2.4 Ferramentas

Nesse cenário as ferramentas são usadas como facilitadores para a execução do processo, auxiliando o desenvolvimento, os testes e a integração. Porém, antes de definir as ferramentas é necessário citar as tecnologias que o AMADEUS usa, são elas:

- Java 7[5]: linguagem de programação usada na camada de negócio;
- JSP[6]: tecnologia para criação de *sites* dinâmicos com Java;
- Hibernate[7]: tecnologia usada para fazer conexões e executar comandos no banco de dados; e
- PostgreSQL[8]: sistema de banco de dados usado pelo projeto.

[5] Disponível em <http://www.oracle.com/technetwork/pt/java/index.html>. Acesso em 18 de novembro de 2014.
[6] Disponível em <http://www.oracle.com/technetwork/java/javaee/jsp/index.html>. Acesso em 18 de novembro de 2014.
[7] Disponível em <http://hibernate.org/orm/>. Acesso em 18 de novembro de 2014.
[8] Disponível em <http://www.postgresql.org/>. Acesso em 18 de novembro de 2014.

Como ferramenta de desenvolvimento, é recomendado o uso do Eclipse[9], por haver integração com todas as tecnologias usadas no projeto. Também é primordial ter Java Development Kit 7 (JDK 7) completo e instalado no computador. Como servidor local é interessante o uso do TomCat 6[10], pois o projeto já está configurado prevendo o uso do mesmo.

Em termos de sistema de banco de dados é importante ter disponível o PostgreSQL na versão 4 ou mais recente e ter a ferramenta PGAdmin 3[11] para gerenciar. Além disso é necessário configurar uma base de dados chamada 'amadeus_web', funcionando na porta 5432 e com administrador padrão 'postgres' com senha 'postgres'. Em caso de divergências é necessário alterar os arquivos de configuração localmente.

4.2.5 Etapas

O primeiro passo para se tornar um colaborador é o entendimento das reais necessidades do sistema, dos objetivos e da visão do projeto. A partir disso o desenvolvedor cria um *fork* do repositório oficial para trabalhar de forma isolada. Assim a cada progresso *commits* devem ser realizados nesse repositório local, de modo a manter um histórico da evolução do código.

Com o desenvolvimento de melhorias ou correções em fase de conclusão, o colaborador deve executar testes exploratórios, verificando os fluxos criados ou modificados para reduzir as possibilidades de erro. Caso encontre problemas será necessário corrigi-los até que os fluxos estejam funcionando dentro do esperado, mas ainda assim podem ocorrer erros que serão tratados mais adiante. Ainda antes de submeter as contribuições, o desenvolvedor deve atualizar a base do repositório para a última versão oficial (*rebase*) e corrigir possíveis erros.

Após concluídas as alterações, o colaborador as submete para o repositório oficial por meio de um *pull request*, no qual terá espaço para descrever o que foi realizado. Em seguida o gerente de configuração é notificado sobre a submissão e inicia a análise da colaboração, tanto observando os códigos quanto os executando. A cada problema encontrado ele informa ao desenvolvedor por meio do fórum dessa contribuição. O colaborador pode aceitar prontamente a análise ou prosseguir com a discussão até que cheguem numa conclusão sobre o que deve ser feito com a modificação.

A partir do momento que todas as modificações passem por essa primeira análise com sucesso, o gerente de configuração inicia a etapa de integrar os códigos com o projeto oficial. O primeiro passo para isso é criar uma *branch* secundária para realizar a operação, em seguida

[9] Disponível em <https://eclipse.org/>. Acesso em 19 de novembro de 2014.
[10] Disponível em <http://tomcat.apache.org/download-60.cgi>. Acesso em 19 de novembro de 2014.
[11] Disponível em <http://www.pgadmin.org/>. Acesso em 19 de novembro de 2014.

realizar o *merge* automático da versão oficial com a nova contribuição, por meio do ambiente, e verificar se existem conflitos possíveis de serem resolvidos localmente ou que precisam voltar para o desenvolvedor. No segundo caso o uso do fórum ocorre novamente, no qual é discutida a solução do impasse para uma integração ser bem sucedida.

Também existe a possibilidade de ocorrer de várias submissões com base na versão atual, e dessa forma apenas a primeira é integrada de forma automática, tendo poucos ou nenhum problema. Porém a partir daí todas as outras contribuições não têm a mesma versão base, já que é gerada uma nova após a integração. Dessa forma o ambiente sugere a realização do *merge* via linha de comando. Nesse caso também é necessário criar uma *branch* para executar a integração. Depois disso realizar um *merge* da versão oficial com a contribuição através do comando *pull*. Ao fim do *merge* será informado quantos conflitos ocorreram. Todos devem ser corrigidos diretamente nos arquivos.

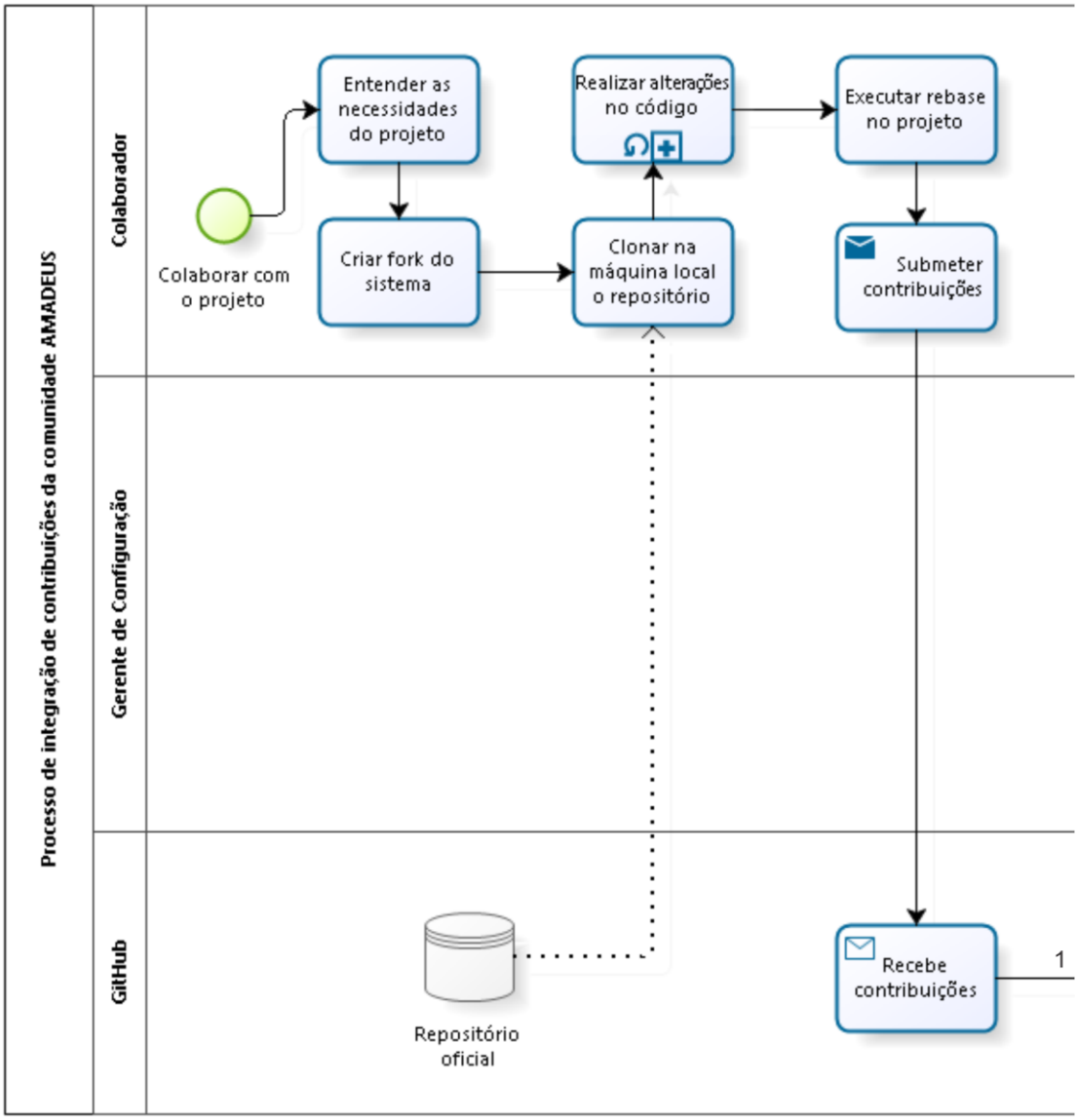
Em seguida a integração deve estar funcional. Assim é possível executar a análise heurística para manutenção da consistência de interfaces do AMADEUS e verificar se as novas interfaces estão dentro do esperado para o projeto. Se encontrar falhas, o gerente de configuração sugere melhorias para o desenvolvedor por meio do fórum e aguarda as modificações. Assim que a contribuição estiver apta, ou seja, sem falhas, sem conflitos de integração e com a interface consistente, o gerente de configuração a transfere para a *branch* principal (*master*).

4.2.6 Modelo

Para sintetizar o processo descrito anteriormente foi utilizada a notação BPMN (Business Process Model and Notation) [27] com a ferramenta Bizagi Modeler[12]. Dessa forma o fluxo é exibido em eventos (círculos), tarefas (retângulos) e decisores (losangos) [27]. Por conta da extensão da imagem, a mesma foi dividida em seis partes e os fluxos entre as partes foi numerado.

[12] Disponível em <http://www.bizagi.com/en/bpm-suite/bpm-products/modeler>. Acesso em 10 de dezembro de 2014.

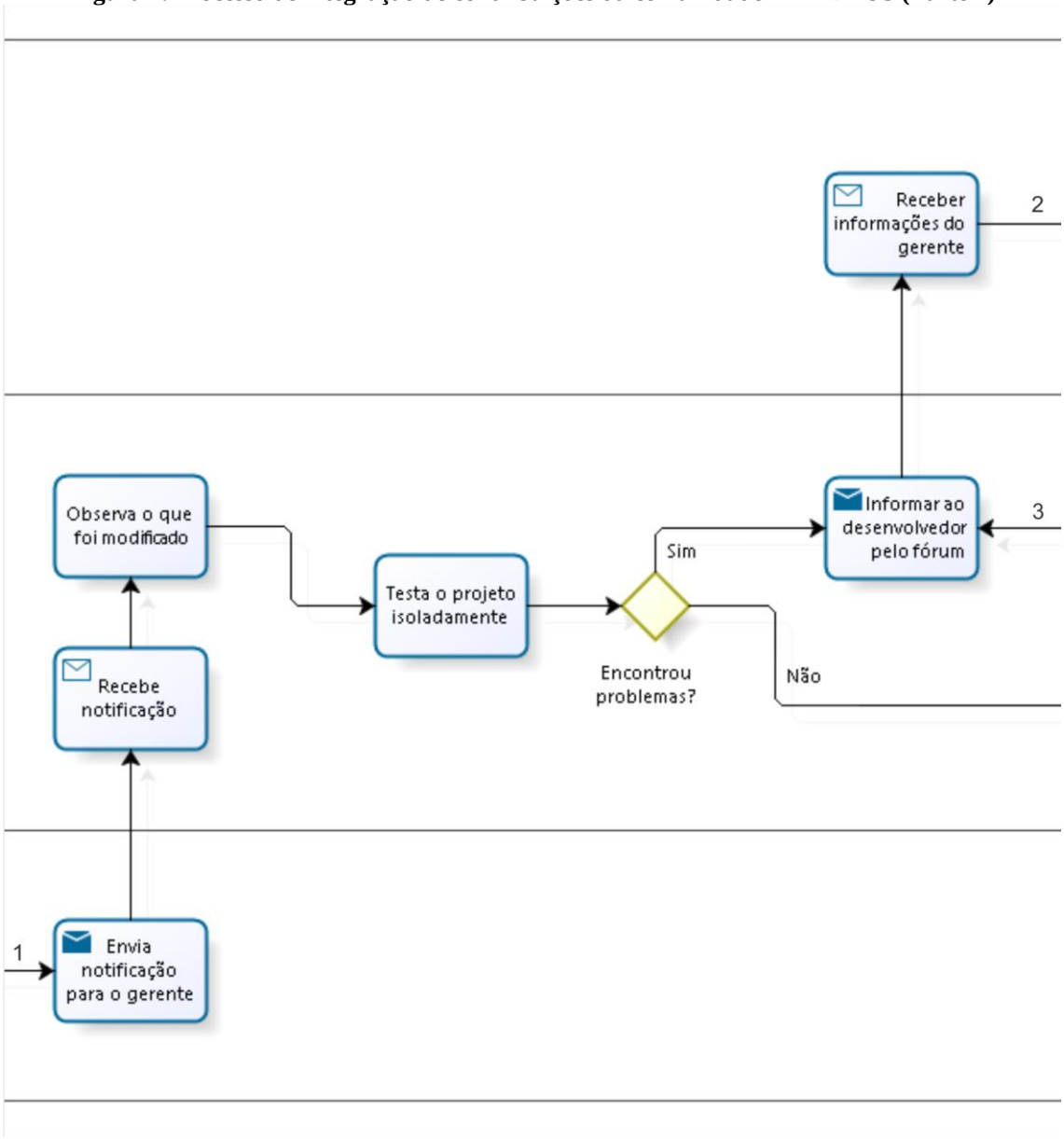
Figura 6. Processo de integração de contribuições da comunidade AMADEUS (Parte 1)



Fonte: O autor.

A primeira parte (ver Figura 6) destaca as tarefas do colaborador de entendimento das necessidades do projeto, criação do *fork* e da versão local do repositório. Após isso um sub-processo de desenvolver alteração (detalhado na Figura 12) é executado, além de um *rebase* e da submissão das contribuições e recebimento pelo GitHub.

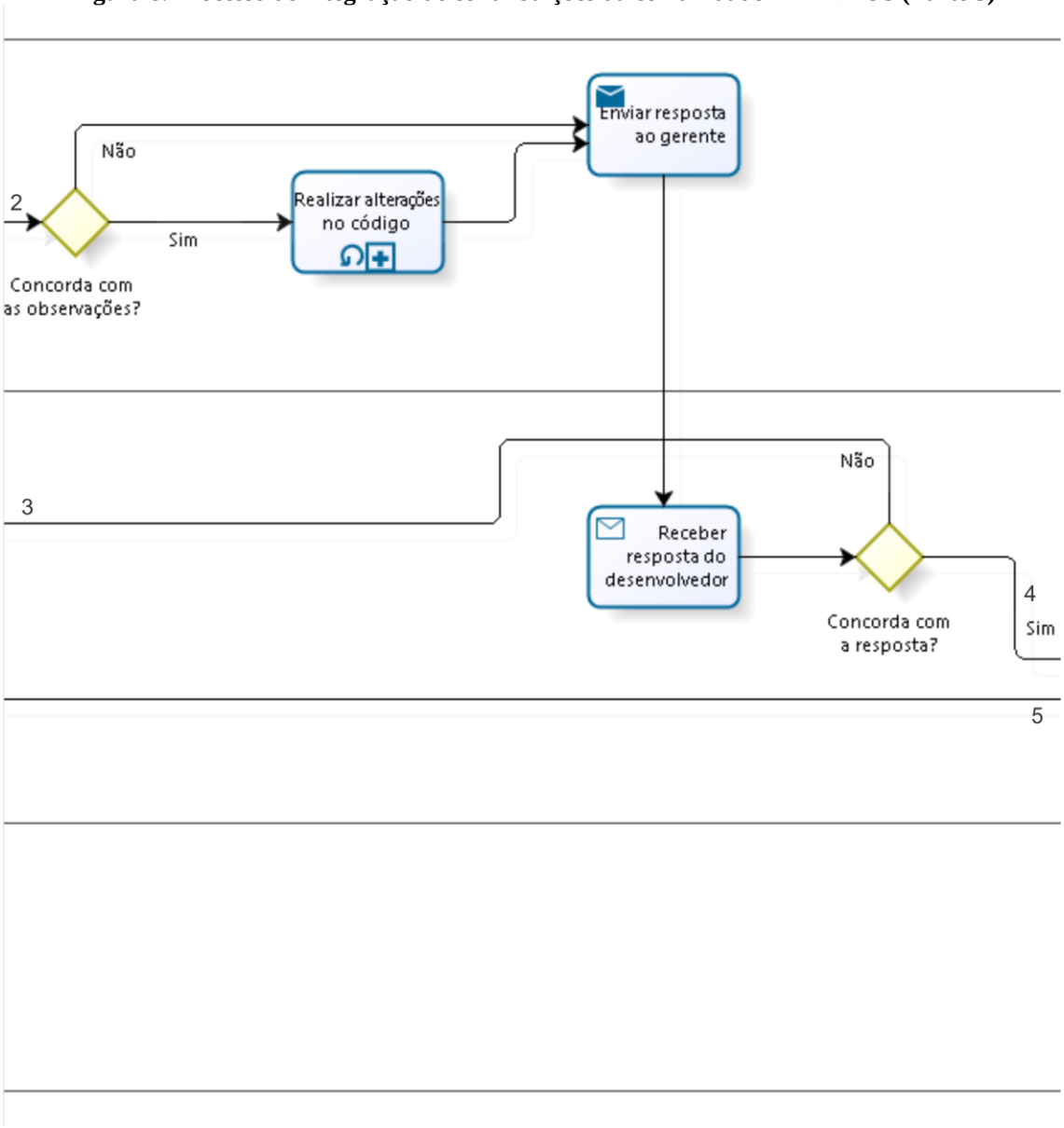
Figura 7. Processo de integração de contribuições da comunidade AMADEUS (Parte 2)



Fonte: O autor.

Na parte seguinte o GitHub (ver Figura 7) envia a notificação para o gerente do repositório, esse recebe a notificação, observa as modificações e testa o projeto. Em caso de problemas o colaborador é contatado, recebendo informações por meio do fórum.

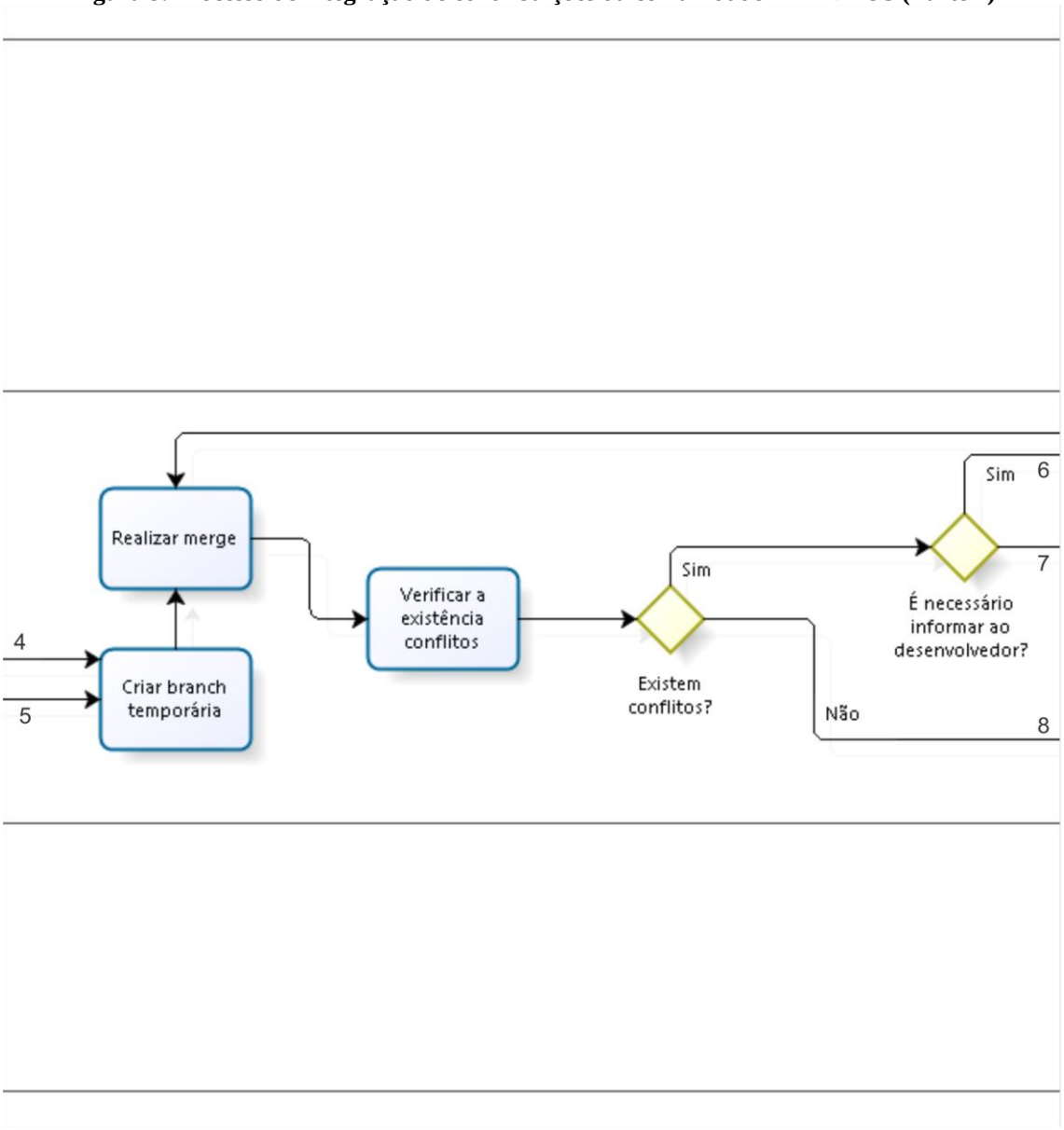
Figura 8. Processo de integração de contribuições da comunidade AMADEUS (Parte 3)



Fonte: O autor.

Na terceira parte (ver Figura 8) há um decisor, pois se o colaborador não concordar é possível entrar em contato com o gerente, que recebe a reposta e pode não concordar, realizando um ciclo de comunicação, até chegarem a uma decisão final com todas as alterações realizadas.

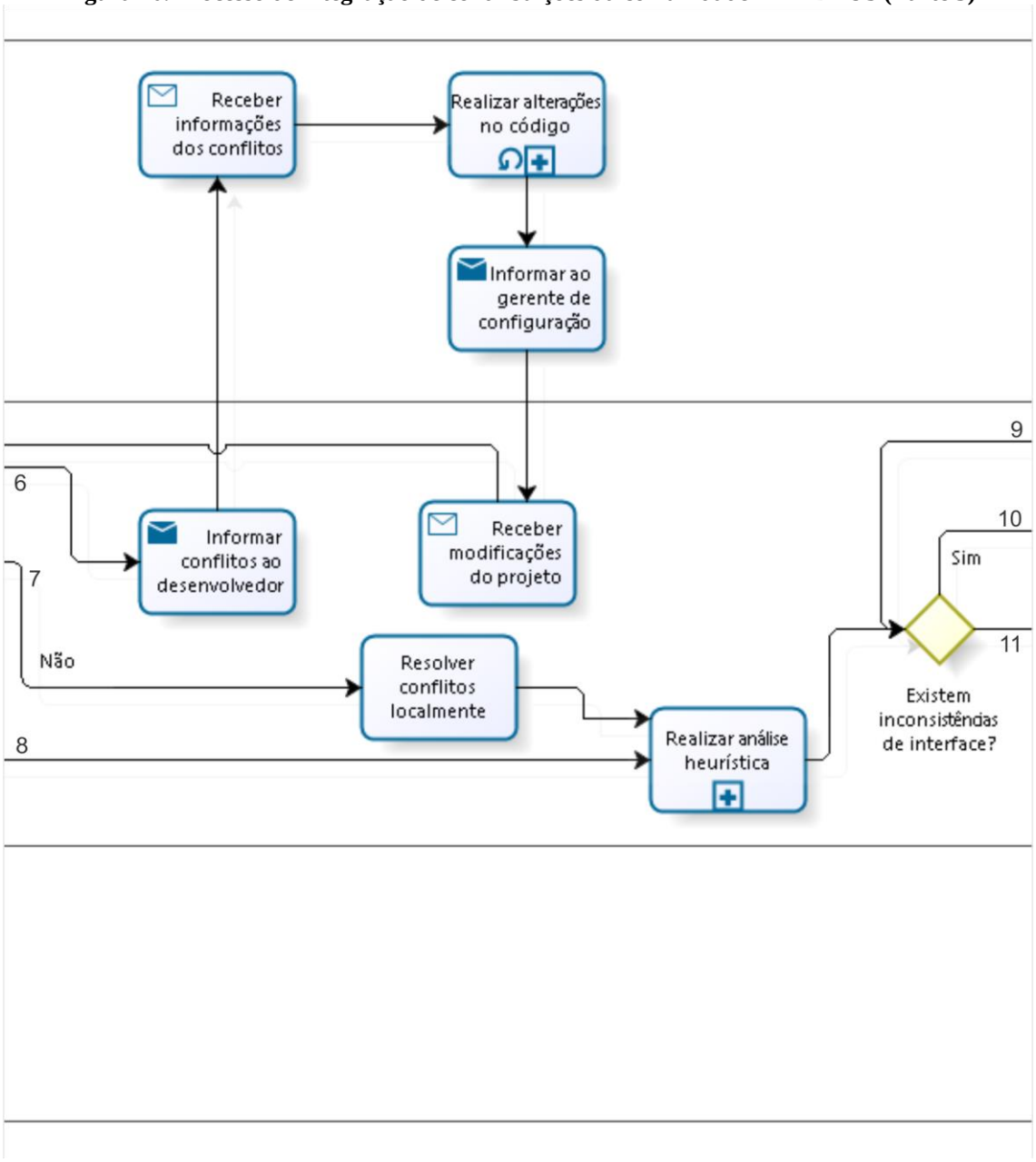
Figura 9. Processo de integração de contribuições da comunidade AMADEUS (Parte 4)



Fonte: O autor.

Na Figura 9 o gerente de configuração cria uma *branch* temporária para executar a integração, e após isso realiza o merge e verifica se há conflitos. Em caso positivo vê o nível de criticidade para informar ou não ao desenvolvedor.

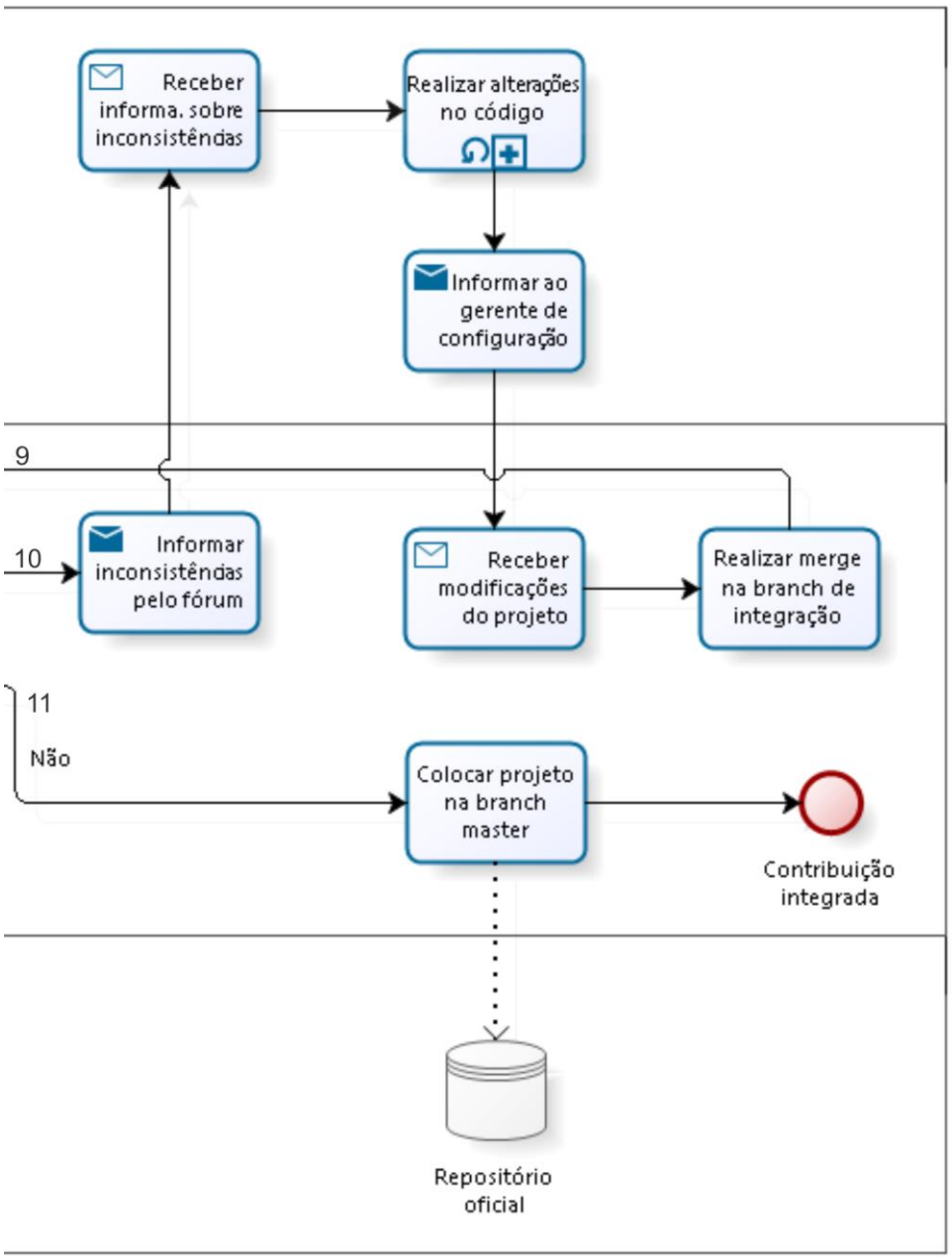
Figura 10. Processo de integração de contribuições da comunidade AMADEUS (Parte 5)



Fonte: O autor.

Na Figura 10 o gerente pode informar os conflitos ao desenvolvedor e aguardar a solução ou pode resolver os conflitos localmente. Caso não existam conflitos a análise heurística será a subtarefa a ser realizada (melhor descrita na figura 13), e por fim é verificado se essa análise encontra inconsistências.

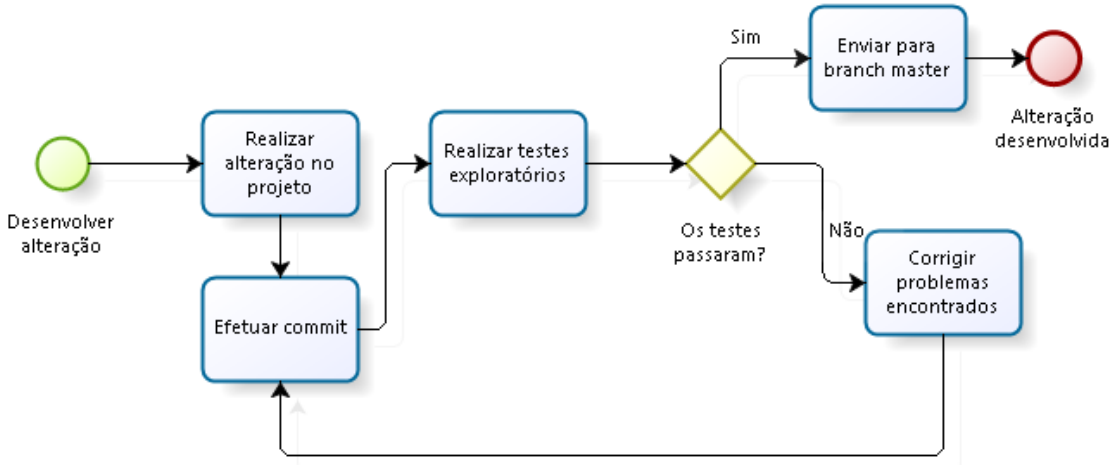
Figura 11. Processo de integração de contribuições da comunidade AMADEUS (Parte 6)



Fonte: O autor.

Na última parte (ver Figura 11) o gerente pode informar as inconsistências ao desenvolvedor ou se não houver nenhuma então irá mover o projeto para o repositório oficial e terá a contribuição integrada.

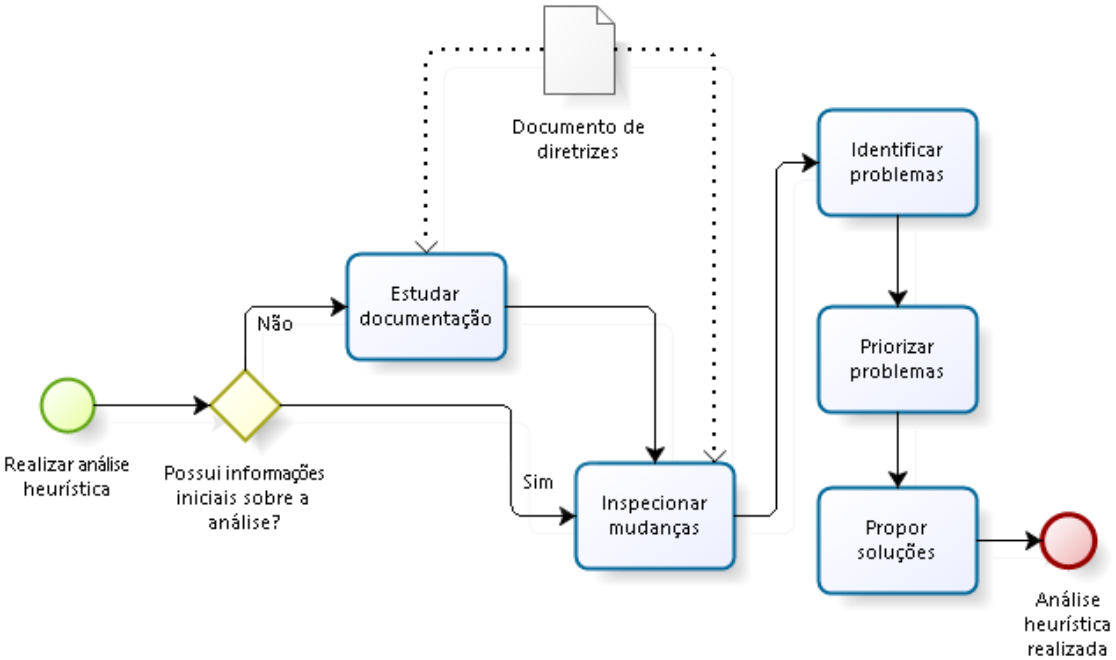
Figura 12. Subprocesso de desenvolver alteração no código



Fonte: O autor.

Na Figura 12 é exibido o fluxo de tarefas realizadas no subprocesso de desenvolver alteração no código. Primeiro as alterações são realizadas e os *commits* executados (no repositório local), após isso são realizados testes exploratórios a fim de encontrar problemas. Caso passe nos testes a alteração será enviada para a *branch master*. Caso contrário os problemas serão corrigidos até que tudo esteja funcionando.

Figura 13. Subprocesso de realizar análise heurística



Fonte: O autor.

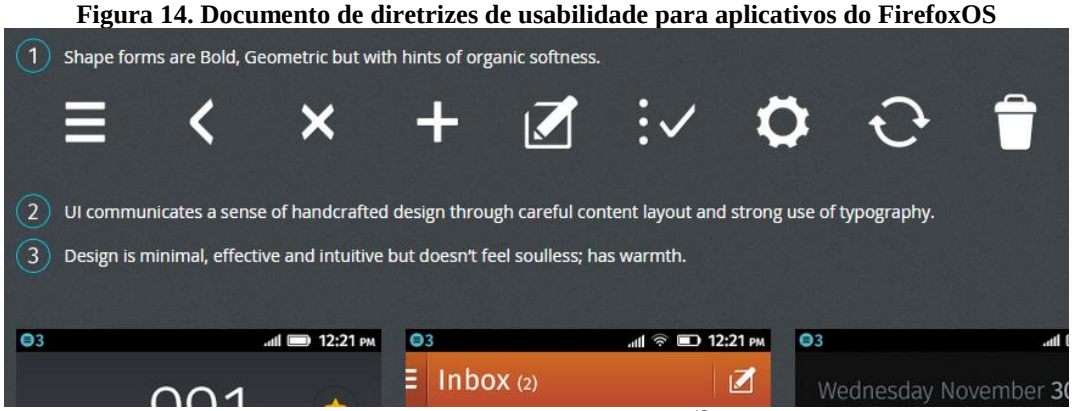
Já na Figura 13 está modelado o subprocesso da análise heurística, que se inicia com um decisor, pois no caso do especialista não possuir conhecimento sobre a documentação então ele deverá estudá-la. Após isso será realizada a inspeção na interface, tendo como escopo as

mudanças realizadas na colaboração a ser integrada. Em seguida os problemas são identificados, priorizados e soluções são propostas, concluindo assim a análise.

4.3 DOCUMENTO DE DIRETRIZES DO AMADEUS

O ponto de partida para a elaboração do documento se deu com uma análise em documentos de diretrizes de outros projetos de *software* livre, dos quais o principal foi o Openredu [13], uma plataforma educacional desenvolvida na UFPE. Na documentação constam princípios nos quais a interface se baseou e especificações de cores, modelo de corpo, tipografia, tabelas, listas, formulários, botões, navegação, alertas, janelas e ícones.

Figura 14. Documento de diretrizes de usabilidade para aplicativos do FirefoxOS



Fonte: Firefox OS Guidelines[13].

Ainda assim foi analisada a documentação do Firefox OS (ver Figura 14) [14], um sistema operacional para dispositivos móveis que permite o desenvolvimento de aplicações por meio de um padrão visual muito bem definido. Nessa especificação há seções sobre paleta de cores, tipografia, cabeçalhos, planos de fundo, listas, botões, barras de ferramenta, campos de texto, seletores e ícones.

Com base nisso foi percebido uma estrutura que se repete nas duas documentações. Sendo assim esse formato foi parcialmente adotado na elaboração da primeira versão do documento de diretrizes do LMS Amadeus. Nesse caso alguns tópicos não eram necessários e outros foram unidos. No fim, os tópicos definidos foram: princípios, *grid* (modelo de corpo), cores, tipografia, formulários e navegação.

O desenvolvimento do conteúdo da documentação se deu com a análise de folhas de estilo do projeto para cada componente necessário e com o estudo de documentação legado do projeto. Nela há *wireframes* (protótipos) para todas as telas desenvolvidas no sistema, implementando casos de uso também definidos em documentação.

Nesse documento de diretrizes são destacados os princípios utilizados na elaboração de interfaces do projeto, o esquema de cores e fontes, como os módulos estão organizados na tela e como se estruturam formulários, botões e a navegação.

---

[13] Disponível em < https://www.mozilla.org/en-US/styleguide/products/firefox-os/>. Acesso em: 13 de novembro de 2014.

4.3.1 Princípios

Considerando o contexto da elaboração de interfaces para *web* há dez (10) princípios básicos, definidos por NIELSEN [16] e selecionados por PREECE [8], que podem ser seguidos na elaboração de uma interface *web*:

1. Visibilidade de status do sistema - o sistema deve manter o usuário informado sobre o que está acontecendo, incluindo *feedbacks* para as ações do usuário em tempo hábil. Na prática podemos citar o uso de mensagens de sucesso e erro como recursos-chave para seguir esse princípio;
2. Compatibilidade do sistema com o mundo real - as páginas devem possuir linguagem simples e acessível para o público alvo. Por isso é importante usar um vocabulário adequado ao contexto;
3. Controle do usuário e liberdade - é necessário permitir que o usuário possa voltar de páginas indesejadas. Como forma de atingir esse princípio se faz útil o uso de botões para voltar à página anterior e caminhos de página (*breadcrumbs*), nos quais o usuário tem a noção do que percorreu para chegar àquela tela;
4. Consistência e padrões - também é importante manter as ações consistentes dentro do sistema, de modo que ações similares sempre ocorram de modo esperado, com um retorno visual coerente. Para isso é importante o uso de documentação de padrões visuais, como por exemplo o guia de diretrizes para *web* do governo americano, *Usability.gov* [5], ou documentos específicos para um sistema, como este em desenvolvimento;
5. Ajudar os usuários a reconhecer, diagnosticar e corrigir erros - é necessário que o sistema exiba mensagens claras em casos de erro, e que dê possibilidades ao usuário para solucioná-lo de forma adequada;
6. Prevenção de erros - na elaboração das telas é necessário analisar a facilidade do usuário cometer erros e entender o porquê. Geralmente caixas de confirmação e botões de cancelar são boas saídas para que menos erros ocorram;
7. Reconhecer, em vez de relembrar - significa reduzir a necessidade da memória permanente do usuário guardar informações sobre caminhos complexos ou comandos para acessar determinada página ou funcionalidade. Portanto, é necessário estimular o reconhecimento, de modo que o usuário realize atividades de forma intuitiva;
8. Flexibilidade e eficiência no uso - é importante que o sistema disponha de atalhos para que usuários experientes realizem tarefas de forma mais ágil. De modo a atingir esse

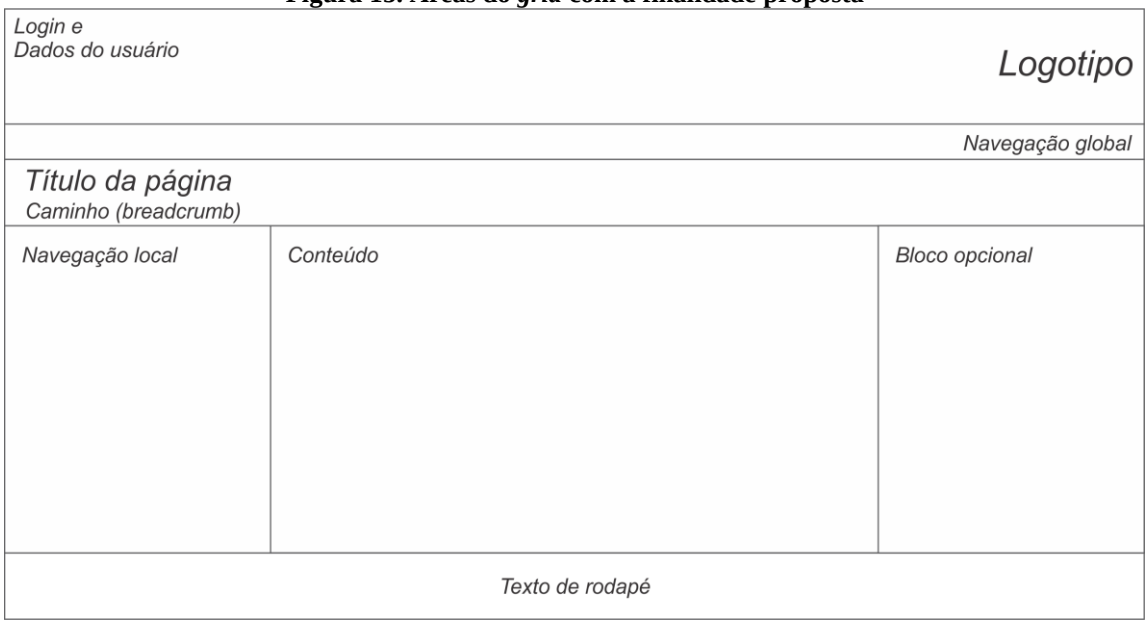
objetivo, o uso de *menus* de contexto ou de sugestões a itens relacionados são possibilidades;

9. Estética e *design* minimalista - nas telas do sistema é necessário manter apenas informações relevantes, de modo que o usuário não encontre barreiras por conta de uma infinidade informações desnecessárias; e
10. Ajuda e documentação - em todos os pontos do *site* devemos manter acesso às páginas de ajuda ou às dicas que auxiliem o usuário na execução de uma tarefa. Dessa forma é possível usar páginas de *F.A.Q.* (perguntas frequentes), para esclarecer as dúvidas mais comuns sobre a página, e *tooltips* (dicas flutuantes), como forma de explicar o que deve ser preenchido em um campo, por exemplo.

4.3.2 Grid

A partir dos *wireframes* do projeto foi construído o *grid* que representa a estrutura geral das páginas. Também foram calculados os tamanhos de largura, altura e espaçamento das áreas. Esse tipo de informação orienta o desenvolvimento de novas funcionalidades que exijam o uso de uma ou mais regiões do *site* dentro do *layout* padrão. Na Figura 15 é destacada a finalidade de cada área do *grid* cuja funcionalidade será descrita adiante.

Figura 15. Áreas do *grid* com a finalidade proposta



Fonte: O autor.

Login e dados do usuário: nessa região estão contidas informações referentes ao usuário, como o nome, além de prover acesso ao perfil e ter uma opção para sair do sistema, porém no

caso de não haver sessão ativa será exibido formulário de *login*. Portanto essa área contempla funcionalidades diretamente associadas ao usuário e não deve ser usada para outro fim.

Logotipo: essa área contém o logotipo da versão atual do sistema e deve ser alterada a cada novo *release*.

Navegação global: é um menu horizontal que contém opções relacionadas ao projeto, como por exemplo um *link* de acesso a uma página sobre a comunidade do AMADEUS. Esse menu também pode conter opções de configuração geral do sistema e deve ser fixo para todas as páginas. Portanto, não deve ter informações contextualizadas com o estado atual.

Título da página: contém o título da página que está sendo exibida e não deve ser exibido caso esteja na página inicial.

Caminho (*breadcrumb*): exibe uma lista horizontal das páginas percorridas para chegar na página atual, no qual todas as páginas anteriores devem conter um *link* de acesso direto.

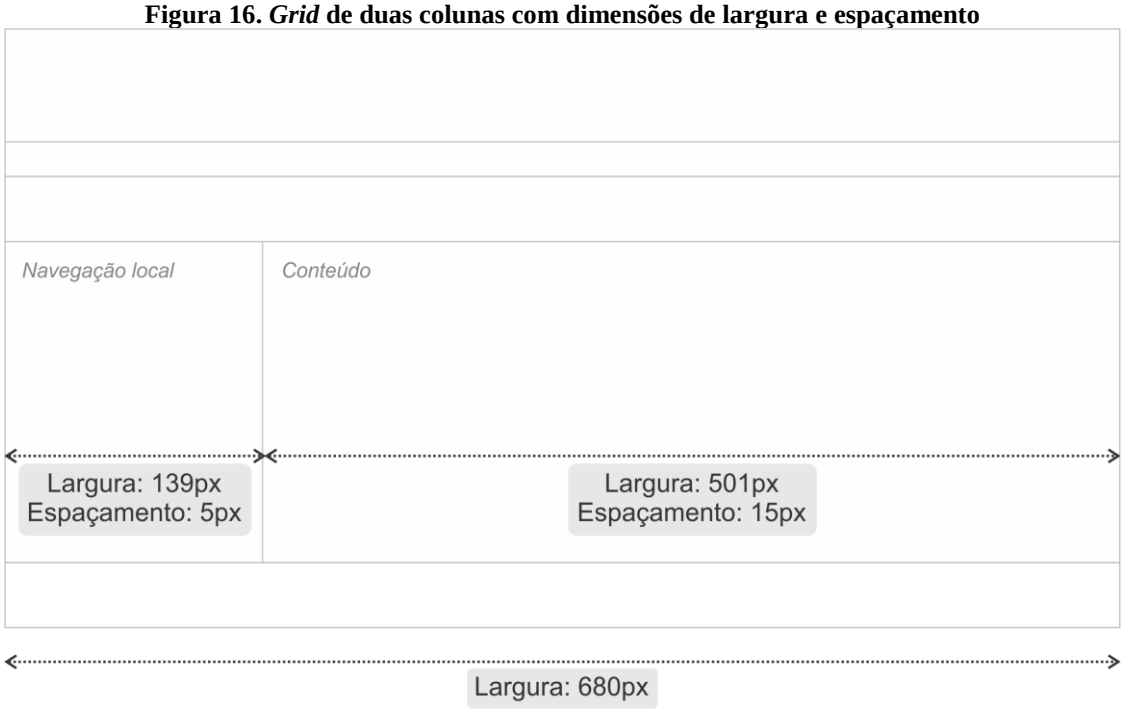
Navegação local: essa coluna deve conter informações e/ou *menus* verticais referentes à página exibida no momento. O intuito dessa região é prover opções contextualizadas. Portanto, não devem haver funções gerais do sistema.

Conteúdo: é a principal região da página, no qual é exibido o conteúdo escolhido pelo usuário. Tem fins gerais e deve ser usada na criação de novas funcionalidades que exijam uma grande área de tela.

Bloco opcional: essa região não possui opções previstas e pode ser usada para fins quaisquer em uma melhoria ou um novo recurso. Portanto essa área pode não ser exibida em uma página, permitindo assim mais espaço para o conteúdo.

Texto de rodapé: contém informações sobre licença de uso e versão do *software*. Dessa forma, não deve ser usada em outra situação, e sim atualizada quando necessário.

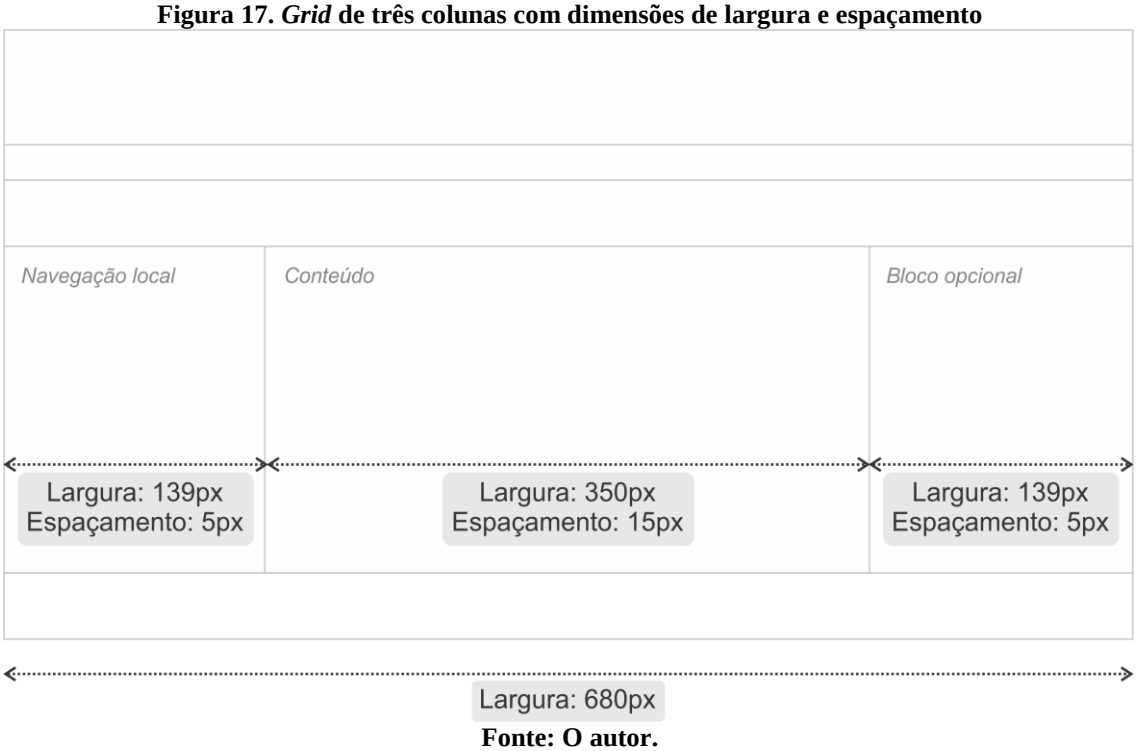
Figura 16. *Grid* de duas colunas com dimensões de largura e espaçamento



Fonte: O autor.

Além disso há componentes que possuem largura fixa, conforme pode ser observado na Figura 16. Sendo px o símbolo da unidade pixel, navegação local possui 139px de largura e espaçamento 5px e conteúdo possui largura de 501px e 15px de espaçamento. Já a largura total da página é de 680px.

Figura 17. *Grid* de três colunas com dimensões de largura e espaçamento



Fonte: O autor.

Também existe a possibilidade de haver um bloco opcional, com largura e espaçamento idênticos a navegação local, dessa forma a área de conteúdo é reduzida para 350px de largura, como ilustrado na Figura 17.

Figura 18. *Grid* com dimensões de altura e espaçamento

Fonte: O autor.

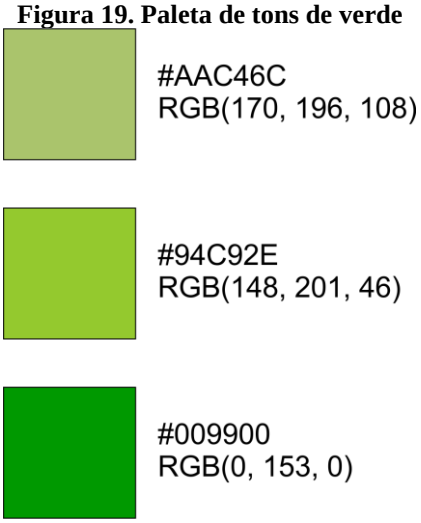
Do mesmo modo a altura é fixada em algumas regiões da página, conforme visto na Figura 18. A região de logotipo possui altura 66px e espaçamento 2px, já a navegação global possui 13px de altura e espaçamento de 1px, a área de título da página e caminho possui 36px de altura e 2px de espaçamento e o texto de rodapé possui 40px de altura e 8px de espaçamento.

4.3.3 Cores

A paleta de cores foi definida principalmente como forma de manter a identidade visual do projeto, mas também para destacar informações relevantes com base no senso comum, como por exemplo em mensagens de sucesso e erro. Quatro conjuntos de cores foram usados, duas escalas de verde, uma escala de cinza e uma de vermelho.

Nas imagens das Figura 19, Figura 20, Figura 21 e Figura 22 estão representadas as cores definidas na paleta, onde o quadrado contém a cor de fato. Ao lado de cada um há o valor hexadecimal e os valores no modo RGB (vermelho, verde, azul). Antes, porém, é necessário definir que textos pequenos são compostos por até trinta (30) palavras e textos médios são compostos por até cem (100) palavras. Além disso as áreas pequenas representam até 10% da tela e as áreas médias representam até 30% da tela.

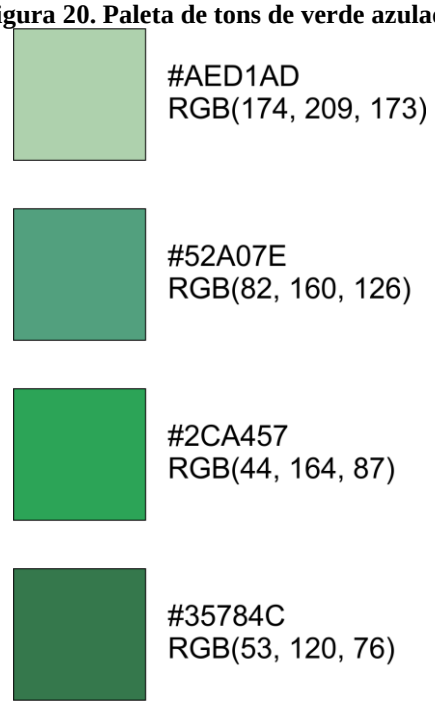
Figura 19. Paleta de tons de verde



Fonte: O autor.

Os tons de verde da Figura 19 têm uso genérico no sistema. São recomendados quando se deseja destacar áreas e textos menores. O verde médio dessaturado (primeira cor) é mais recomendado na definição de planos de fundo. Já o verde médio (segunda cor) pode ser usado para definir bordas e o verde-escuro (terceira cor) pode ser usado em outras situações que seja necessário destacar uma palavra ou texto curto.

Figura 20. Paleta de tons de verde azulado

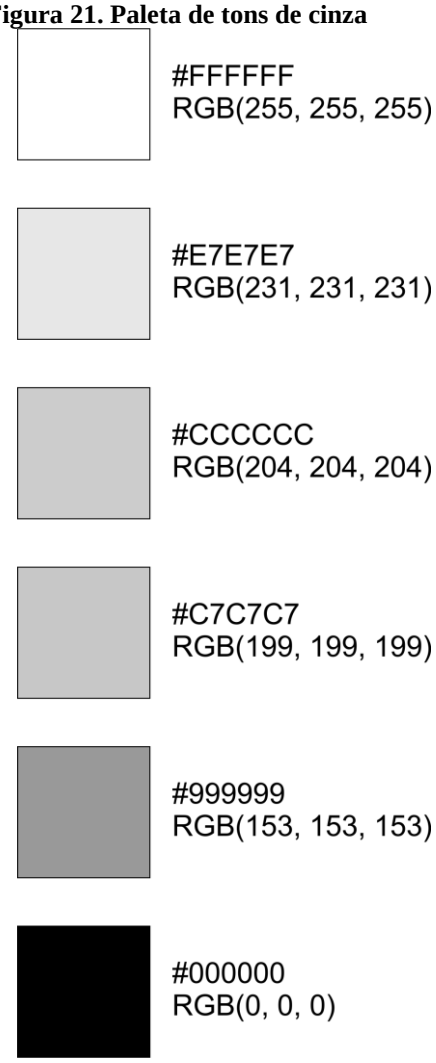


Fonte: O autor.

Os tons de verde azulado da Figura 20 podem ser usados para definir detalhes, como bordas, e alterações de estado em botões, por exemplo. A cor verde-clara (primeira cor) pode ser usada como plano de fundo de uma região clara, possibilitando mais destaque ou maior

sensação de organização na página. O verde médio (segunda cor) pode ser usado em textos, destacando informações relevantes ou trechos de tamanho médio e a cor verde-escura (quarta cor) é usada no texto de mensagens de sucesso. No entanto o verde médio mais saturado (terceira cor) possui uso padrão no sistema, sendo usado nos *links* no estado em que o *mouse* está sobre o texto.

Figura 21. Paleta de tons de cinza



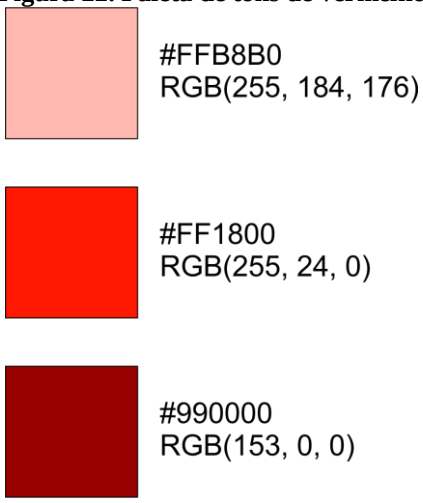
Fonte: O autor.

Os tons de cinza da Figura 21 são usados de forma menos restrita no sistema. Basicamente, o conteúdo de todas as interfaces possui fundo base branco (primeira cor) e a maior parte dos textos está em cor preta (sexta cor), inclusive os *links* em estado normal (quando o *mouse* não está sobre a região).

Já os tons intermediários têm uso mais específico. O cinza mais claro (segunda cor) é usado como plano de fundo em partes do conteúdo, quando se deseja dar uma noção de divisão ou de continuidade em uma linha. O cinza médio (terceira cor) é usado no plano de fundo externo do site e de bordas em geral. Da mesma forma, o cinza escuro (quinta cor) é usado em

bordas, mais especificamente em bordas de janelas sobrepostas e notificações gerais. O outro cinza médio (quarta cor) é usado em planos de fundo de formulários, preferencialmente nas linhas das etiquetas.

Figura 22. Paleta de tons de vermelho



Fonte: O autor.

Os tons de vermelho da Figura 22 indicam situações que podem ter um resultado indesejado, como por exemplo em ações de exclusão ou de cancelamento. Também é recomendado que notificações de erro usem esse conjunto cores. O vermelho mais claro (primeira cor) deve ser usado como plano de fundo e o vermelho escuro (terceira cor) como cor do texto regiões de alerta. Caso o plano de fundo dessas regiões necessite ser branco, então o vermelho forte (segunda cor) é usado como cor do texto.

Ao definir uma cor é importante pensar que o texto deve contrastar com o plano de fundo definido para não dificultar a leitura. Dessa forma caso o plano de fundo seja claro então o texto deve ser escuro. Além disso o uso das cores deve ser feito com cautela, sempre buscando manter a finalidade para que foram definidas e evitando exageros. Como forma de consulta o quadro 2 resume a finalidade de cada cor, junto aos valores hexadecimal e RGB.

Quadro 2. Paleta de cores para o AMADEUS

Cor	Hexadecimal	RGB	Finalidade
	#AAC46C	(170, 196, 108)	Planos de fundo em áreas pequenas ou médias de conteúdo.
	#94C92E	(148, 201, 46)	Bordas em áreas pequenas ou médias de conteúdo.
	#009900	(0, 153, 0)	Textos curtos ou palavras em destaque.
	#AED1AD	(174, 209, 173)	Plano de fundo em linhas de tabelas, etiquetas de formulários, títulos de conteúdo ou em outras regiões de destaque.

	#52A07E	(82, 160, 126)	Textos médios em destaque.
	#2CA457	(44, 164, 87)	*Links* em estado ativo ou com o mouse sobreposto.
	#35784C	(53, 120, 76)	Texto de mensagens de sucesso.
	#FFFFFF	(255, 255, 255)	Plano de fundo geral de conteúdo.
	#E7E7E7	(231, 231, 231)	Planos de fundo em áreas pequenas ou médias de conteúdo.
	#CCCCCC	(204, 204, 204)	Plano de fundo externo e bordas de *layout* do site.
	#C7C7C7	(199, 199, 199)	Plano de fundo em formulários, preferencialmente nas linhas das etiquetas.
	#999999	(153, 153, 153)	Bordas de janelas sobrepostas e bordas de notificações genéricas.
	#000000	(0, 0, 0)	Texto padrão e de *links* em estado normal.
	#FFB8B0	(255, 184, 176)	Plano de fundo em regiões de alerta e erro.
	#FF1800	(255, 24, 0)	Cor do texto em regiões de alerta caso o plano de fundo seja branco, possivelmente em áreas de conteúdo.
	#990000	(153, 0, 0)	Cor do texto em regiões de alerta e erro quando o plano de fundo for vermelho claro.

Fonte: O autor.

4.3.4 Tipografia

A tipografia é o processo de organizar e definir estilos em fontes de texto, e tem papel importante dentro do *design* do projeto. Através dela é possível estabelecer hierarquias e adicionar uma ordem estrutural dentro da página, os artifícios mais comuns para isso são a variação de tamanhos, pesos e tipos de fonte.

Figura 23. Exemplo de fonte sem serifa

AaBbCc

Fonte: DaFont[14].

No projeto AMADEUS a família de fontes escolhida foi *Sans Serif* (do latim, sem serifa), que possui traços mais simples como a da Figura 23. Na folha de estilos foi definida uma ordem de prioridade em que a primeira é a fonte Arial, disponível em computadores com o sistema operacional Windows. Em segundo a fonte Helvetica, disponível em computadores

[14] Disponível em < http://www.dafont.com/coolvetica.font?text=AaBbCc> Acesso em 29 de dezembro de 2014.

com o MacOS, só será aplicada se não for possível a opção anterior. Ainda assim, caso nenhuma das duas esteja presente no dispositivo, outra fonte da mesma família (escolhida pelo navegador) será encarregada de compor a página.

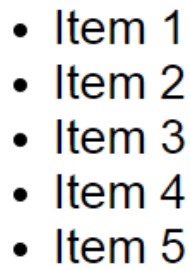
Em toda a folha de estilos as fontes estão personalizadas com tamanhos na unidade *em*, muito aplicada na área de tipografia e teve o uso considerado uma boa prática no CSS2 (*Cascading Style Sheets Level 2*)[15]. Como principal vantagem essa unidade possui dimensões relativas (por vezes expressas em notação decimal), tornando as fontes adaptáveis ao tamanho da tela. O tamanho relativo é calculado com base em um tamanho raiz definido em pixels, geralmente no seletor *body* da folha de estilos da página. Caso não seja definido, o navegador definirá qual o melhor tamanho de referência.

Figura 24. Formatação dos títulos em textos

Fonte: O autor.

Os títulos das páginas devem estar em proporção a Figura 24, ‘título 1’ refere-se à *tag* HTML `<h1>`, ‘título 2’ à `<h2>` e ‘título 3’ à `<h3>`. Os títulos 1 e 2 possuem tamanhos relativos de 1,4em e 1,3em, respectivamente, peso negrito e não possuem espaçamento nem margem. Já o título 3 possui tamanho relativo 1em, peso negrito, margem inferior de 1px e margem superior de 2px.

Figura 25. Formatação das listas em textos

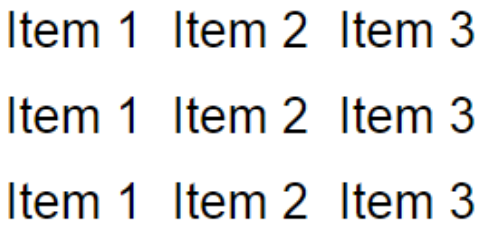


Fonte: O autor.

[15] Disponível em < http://www.w3.org/TR/CSS21/>. Acesso em 15 de dezembro de 2014.

As listas referentes à *tag* `<ul>` devem estar conforme a Figura 25, na qual a lista possui um marcador do tipo disco e as fontes têm tamanho 0,7em. O espaçamento à esquerda é de 40px e as margens superior e inferior possuem 12px.

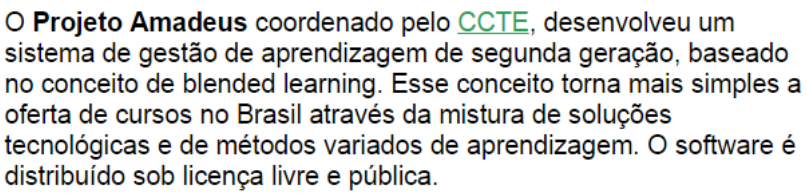
Figura 26. Formatação das tabelas de conteúdo



Fonte: O autor.

Já as tabelas não possuem espaçamento nem margem no *container* `<table>`, mas as células `<td>` possuem espaçamento de 1px e margem de 2px. Além disso, o tamanho de fonte usado nas tabelas é de 0,9em. O estilo padrão proposto para o projeto não define borda nem plano de fundo para as tabelas, mas isso pode ser feito por meio de classes customizadas seguindo o padrão de cores definido anteriormente.

Figura 27. Formatação de parágrafos e *links* de texto



Fonte: O autor.

Os parágrafos possuem margem superior e inferior de tamanho 12px, mas não possuem espaçamento. O tamanho de fonte dessas áreas é de 0,7em com peso normal, mas caso haja necessidade de destacar alguma palavra, como na Figura 27, o recurso de negrito pode ser usado por meio da *tag* `<strong>`, alterando a palavra para deixa-la em evidência. Ainda na Figura 27 é exibido um *link* (na palavra CCTE) em estado de mouse sobreposto, identificado como *hover* no CSS, no qual aparece sublinhado e com cor verde de hexadecimal #2CA457. Já o estado normal dos *links* deve manter o padrão do texto, com fontes de tamanho 0,7em, cor preta e sem sublinhado.

No quadro 3 são apresentadas as especificações referentes a cada item de forma resumida:

Quadro 3. Especificações de formatação para componentes HTML no AMADEUS

Item	***Tag* HTML**	**Especificações**
Título 1	`<h1>`	Tamanho 1,4em, peso negrito, sem margem e sem espaçamento.

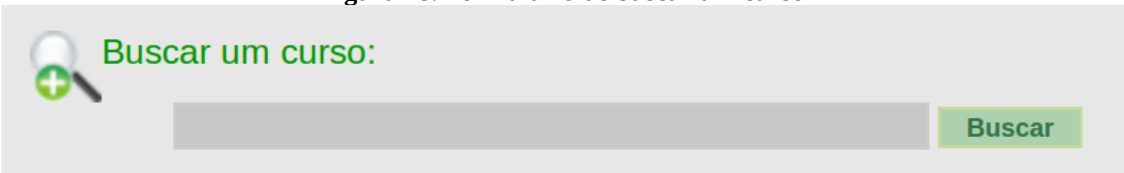
Título 2	`<h2>`	Tamanho 1,3em, peso negrito, sem margem e sem espaçamento.
Título 3	`<h3>`	Tamanho 1em, peso negrito, margem inferior de 1px, margem superior de 2px e sem espaçamento.
Lista	`<ul>`	Marcador tipo disco, fonte de tamanho 0,7em, espaçamento à esquerda de 40px e margens superior e inferior de 12px.
Tabela	`<table>`	Sem margem, sem espaçamento e sem bordas.
Célula da tabela	`<td>`	Fonte de tamanho 0,9em, espaçamento de 1px e margem de 2px.
Parágrafo	`<p>`	Tamanho 0,7em, peso normal, margem superior e inferior de tamanho 12px e sem espaçamento.
Link normal	`<a>`	Preserva as características do texto onde está inserido.
Link sobreposto	`<a>`	Sublinhado e de cor verde (#2CA457).

Fonte: O autor.

4.3.5 Formulários

Nas páginas do projeto são encontrados três estilos de formulários, que na maioria dos casos são usados em cadastros e buscas. Nesses formulários é importante atentar a necessidade de tudo estar dentro de uma *tag* `<form>` com atributos como o método de envio e a ação a ser executada.

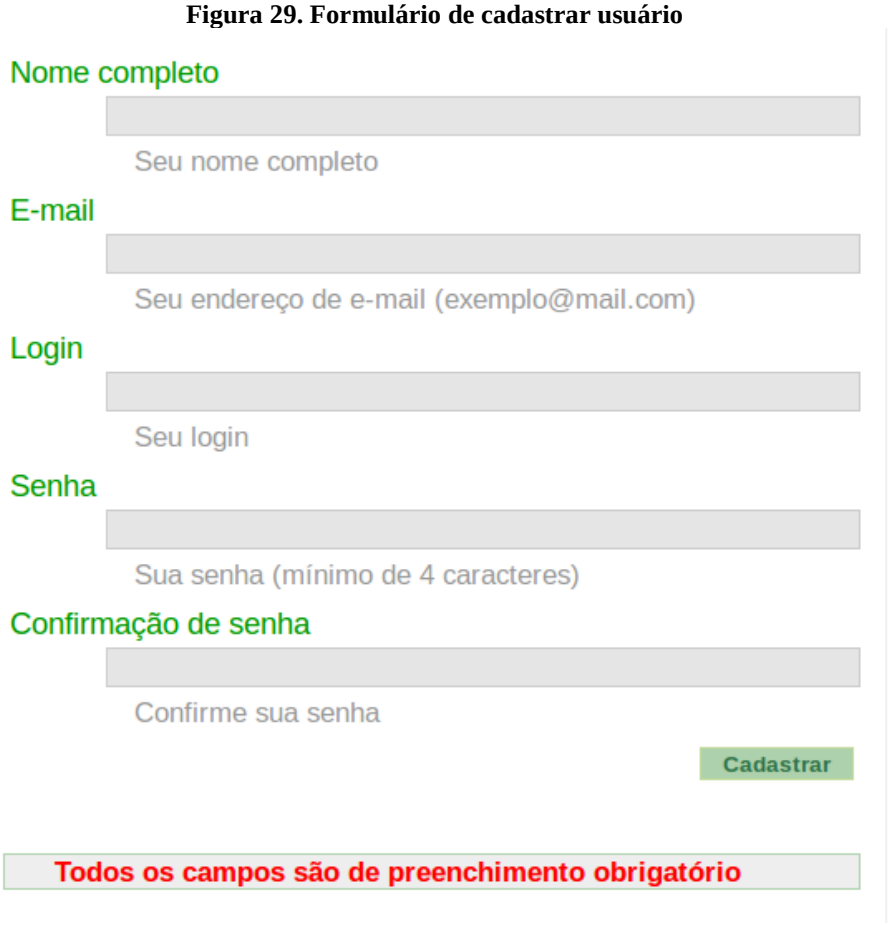
Figura 28. Formulário de buscar um curso



Fonte: Foto da tela do LMS AMADEUS.

O primeiro deles é o formulário horizontal, atualmente usado apenas na busca simplificada e é indicado para ações similares, com apenas um campo e um botão. Na Figura 28 pode ser visto um exemplo. Para aplicar esse estilo é necessário colocar o campo do botão dentro de um mesmo *container* (por exemplo, uma `<div>`), o campo de texto deve conter a classe ‘formfield2’ e o botão a classe ‘button’.

Figura 29. Formulário de cadastrar usuário



Fonte: Foto da tela do LMS AMADEUS.

Os formulários extensos, como na Figura 29, são uteis para criação de cadastros. Nele cada campo deve vir com uma etiqueta e opcionalmente com uma descrição. Na Figura 30 pode ser vista a estrutura HTML necessária para criar um formulário como esse:

Figura 30. Exemplo de HTML para formulário consistente

```
<form name="formName" method="post" action="#">
    <label class="youCanTitle">Nome completo</label>
    <div class="field">
        <input name="name" class="formfield" type="text" …/>
    </div>
    <div class="description">Digite seu nome completo</div>
    <div class="field">
        <input value="Cadastrar" class="button" type="submit" …/>
    </div>
</form>
```

Fonte: Foto da tela do LMS AMADEUS.

A primeira *tag* é um `<form>` que possui atributos específicos para o envio do formulário. Em seguida é a etiqueta do campo, disposta em um `<label>` com a classe ‘youCanTitle’. Já o campo de texto `<input>` deve ter a classe ‘formfield’ e ficar dentro de uma `<div>` com a classe ‘field’. A descrição do campo deve estar em uma `<div>` com a classe

‘description’ e o botão `<input>` deve possuir a classe ‘button’ e estar dentro de uma `<div>` com a classe ‘field’.

Figura 31. Formulário simplificado

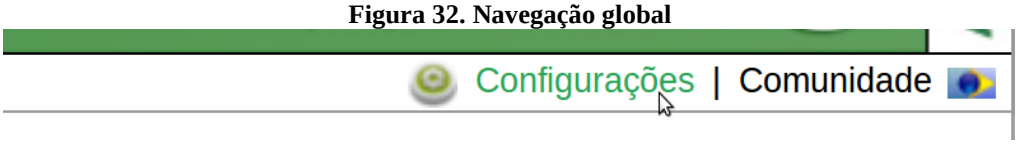
Fonte: Foto da tela do LMS AMADEUS.

Ainda há o formulário simplificado (Figura 31), feito de forma tabular, no qual as etiquetas estão em células da coluna esquerda e os respectivos campos estão em células da coluna direita. A primeira coluna possui alinhamento à direita, já a segunda está alinhada à esquerda, exceto no caso do botão pesquisar, que está na última linha alinhado à direita.

4.3.6 Navegação

No AMADEUS toda a navegação está compartimentada em três (3) áreas distintas, sendo uma delas referente à navegação geral, outra com o caminho percorrido e a última com um menu contextualizado.

Figura 32. Navegação global

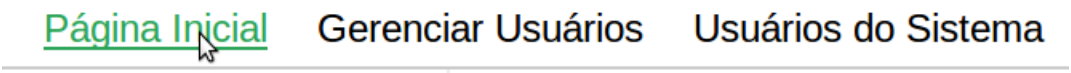


Fonte: Foto da tela do LMS AMADEUS.

A navegação global está localizada na barra horizontal acima do conteúdo, e é composta por um menu horizontal e separadores de texto. As opções desse menu estão ligadas a fatores mais gerais como configurações e informações do produto e devem ser mantidas em todas as páginas, não tendo necessariamente uma ligação direta com o conteúdo exibido na página.

Para adicionar opções adequadas a esse menu, é necessário inserir uma *tag* `<span>` dentro da `<div>` identificada como ‘institutional_menu’ com o *link* direcionando para a página desejada e um separador de textos do tipo barra vertical, conforme visto na Figura 32. É importante destacar a necessidade de avaliar o alvo do *link*, pois caso direcione para fora do sistema deve ser aberta uma nova janela.

Figura 33. *Breadcrumb*

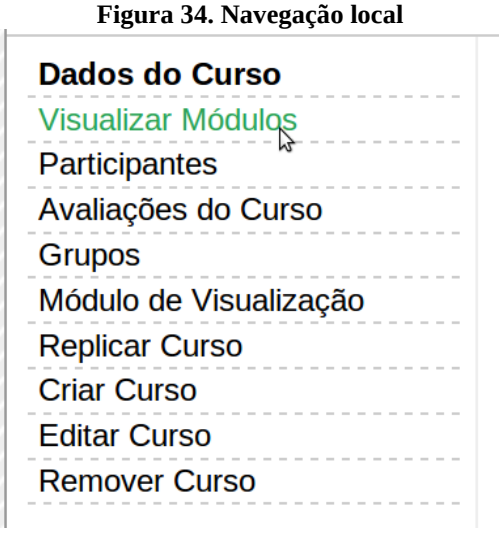


Fonte: Foto da tela do LMS AMADEUS.

O caminho, comumente chamado de *breadcrumb*, representa as páginas percorridas até chegar no conteúdo atual. Isso possibilita um maior senso de localização dentro do site e permite que o usuário volte para as outras páginas, já que todas as páginas anteriores estão representadas na forma de *link*.

Esse componente precisa ser criado dinamicamente em cada página. Para isso é preciso ter uma `<div>` com a classe ‘pBreadCrumbs’ e dentro dessa área ter uma lista `<ul>` com o id ‘breadcrumb’. Com exceção do último item, cada item `<li>` da lista tem um *link* para a página referente, e como pode ser observado na Figura 33, na qual a página inicial está sobreposto pelo mouse, os *links* possuem o mesmo estilo dos parágrafos.

Figura 34. Navegação local



Fonte: Foto da tela do LMS AMADEUS.

A navegação local ou contextualizada é composta de um menu vertical, localizado na coluna esquerda da página. Na Figura 34 é exibido o menu de ‘Dados do Curso’. Este só fica disponível para usuários com sessão ativa que estão na página de um curso. Nesse exemplo as opções exibidas são referentes ao perfil do usuário ativo, no caso um administrador.

Para construir menus desse tipo é necessário apenas criar uma lista desordenada `<ul>` com um id ‘menu_sessoes’, e cada item dessa lista será um item do menu. Dessa forma o *layout* definido por padrão na folha de estilos do AMADEUS se mantém, com uma borda inferior cinza e pontilhada em cada item. Já os *links* em estado normal possuem cor preta e quando estão com o mouse sobreposto passam a ter cor verde, como nos *links* de parágrafos, mas nesse caso

sem sublinhado. Se for preciso um título para o menu, este deve vir como primeiro item da lista e em negrito.

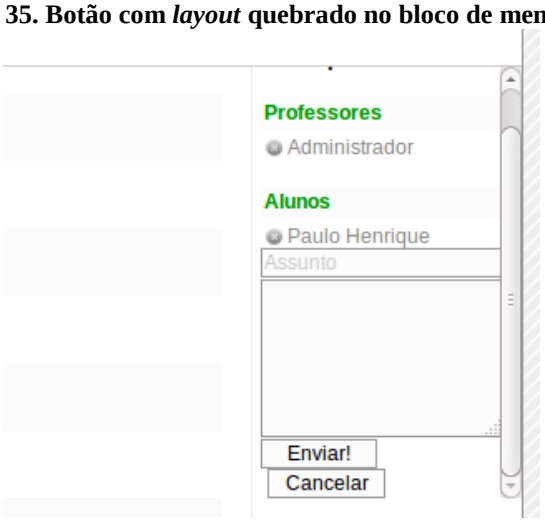
4.4 REVISÃO DE *GUIDELINES* E INSPEÇÃO DE CONSISTÊNCIA

Com o documento de diretrizes em mãos foi possível realizar a análise heurística, com destaque a dois pontos principais: a revisão de *guidelines*, de modo a verificar a conformidade das interfaces com a documentação proposta; e a inspeção de consistência, possibilitando identificar falhas mais gerais, relacionadas aos princípios básicos, *layouts* incomuns, formatos de entrada e saída inesperados [28].

4.4.1 Análise da contribuição de MEDEIROS (2013)

A primeira contribuição analisada foi a de MEDEIROS, na qual os módulos sociais foram analisados. Em termos de posicionamento o módulo de mensagens foi colocado na coluna direita, correspondendo a uma decisão correta, já que essa região do *grid* foi definida para módulos adicionais (ver Figura 35). A escolha de cores também está dentro do esperado na paleta de cores definida anteriormente, na qual o verde é usado para destacar um texto curto e o cinza como plano de fundo desses textos.

Figura 35. Botão com *layout* quebrado no bloco de mensagens



Fonte: Foto da tela do LMS AMADEUS.

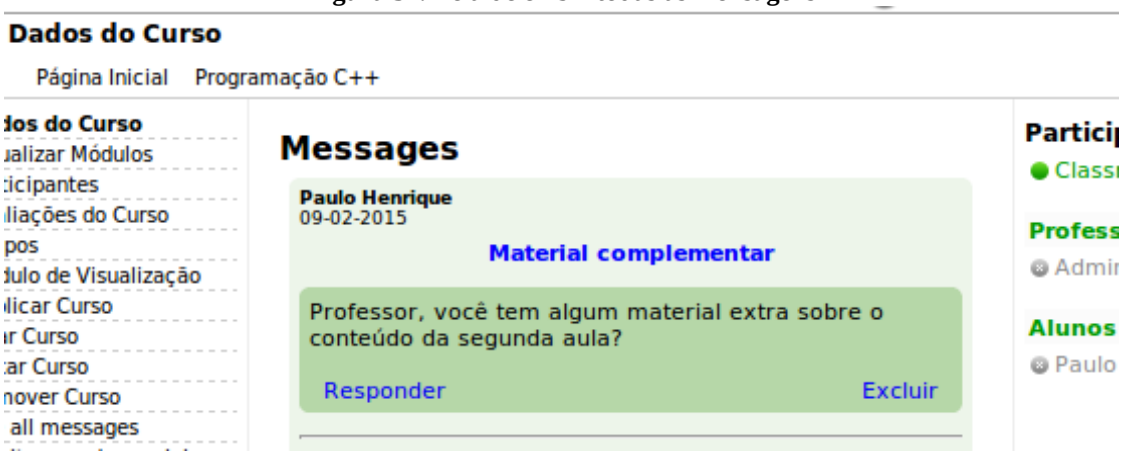
Porém quando a altura a área alocada para enviar mensagens é estourada há uma quebra na posição do botão ‘Cancelar’, Isso acontece porque a barra de rolagem passa a ser exibida e reduz a largura disponível para esses botões. Além disso o campo ‘Assunto’ possui uma cor muito clara para texto escrito pelo usuário, podendo causar dificuldades aos usuários que possuem monitor com um brilho alto ou baixo contraste.

Figura 36. Feedback de mensagem enviada sendo exibido

Fonte: Foto da tela do LMS AMADEUS.

Ao enviar mensagens um retorno é exibido na tela, respeitando o princípio que indica a necessidade de fornecer *feedback* ao usuário para as ações executadas (ver Figura 36). Da mesma forma, ao tentar enviar mensagens com campos vazios um alerta é exibido no mesmo local, porém com cores diferentes.

Figura 37. Tela de exibir todas as mensagens



Fonte: Foto da tela do LMS AMADEUS.

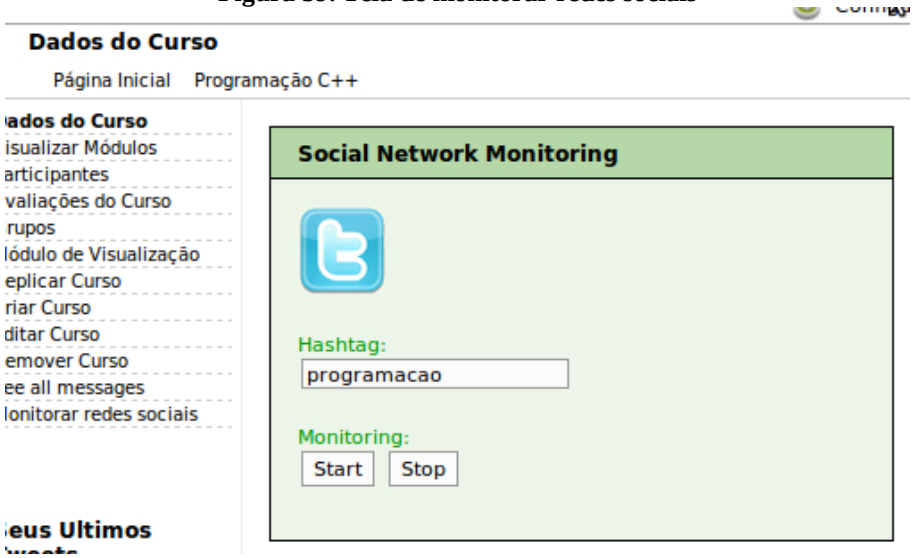
Na página de visualização de todas as mensagens (Figura 37) a área de conteúdo da tela é utilizada de forma correta, exibindo as mensagens correspondentes. Nessa área de conteúdo as cores também foram usadas corretamente, o verde claro para plano de fundo e o verde médio para destacar uma região específica. Porém, no caso dos *links* a cor usada, azul, não está presente na paleta de cores e também não encaixa na identidade visual do *site*. Também foram usadas bordas arredondadas as áreas em verde, algo que poderia ter sido evitado pois em todas as outras telas não há nenhuma região que use esse tipo de borda, fazendo com que a consistência seja comprometida.

Figura 38. Tela de integrar redes sociais

Fonte: Foto da tela do LMS AMADEUS.

Na tela de integração com redes sociais há poucos problemas (ver Figura 38). O formulário foi feito de forma correta e o botão ‘Salvar’ manteve o padrão usado nos outros formulários do sistema e definido na paleta de cores. Porém os ícones usados para ilustrar o Twitter e o Facebook estão em desacordo entre si. O primeiro possui borda quadrada e cor de fundo chapada, já o segundo possui borda arredondada e cor de fundo com degradê. Além disso os ícones estão desalinhados, quando deveriam estar alinhados à esquerda.

Figura 39. Tela de monitorar redes sociais



Fonte: Foto da tela do LMS AMADEUS.

Já na parte de monitoramento de redes sociais (ver Figura 39) foi usado um formato de telas totalmente diferente dos anteriores, com bordas pretas em uma tabela, quando a estrutura usada deveria ser similar à de um formulário. O ícone no Twitter está em um tamanho desproporcional além de ser desnecessariamente diferente do ícone usado na tela de integração das redes sociais.

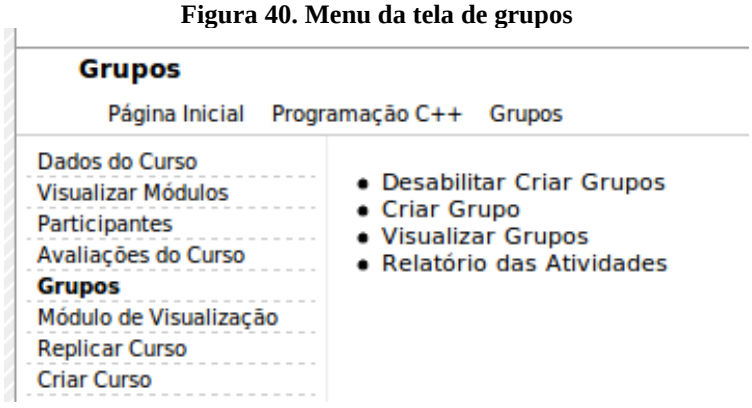
No quesito navegação houveram algumas ocorrências de erro, pois os títulos nas telas de exibir todas as mensagens, integração de redes sociais e monitoramento de redes eram ‘Dados do curso’, ‘Trocar senha’ e ‘Dados do curso’, respectivamente. A contribuição também não previne erros em algumas situações, como no caso de enviar mensagens para toda a sala, quando apenas pressionando o botão enviar toda a sala é notificada, mas nesse caso seria importante uma mensagem de confirmação questionando se o professor realmente deseja entrar em contato com toda a turma.

De uma forma geral a contribuição de MEDEIROS está compatível com o esperado por usuários do sistema: permite que o usuário tenha liberdade de cancelar, retornar ou iniciar ações quando desejado e mantém um padrão coerente com as diretrizes e a identidade visual. Os problemas estão concentrados em dois princípios: ‘prevenção de erros’ e ‘consistência e padrões’, esse último com poucos erros referentes a cor e navegação.

4.4.2 Análise da contribuição de PERRIS (2013)

A contribuição de PERRIS foi analisada em seguida, contemplando o módulo de grupos, no qual é possível criar, visualizar e obter relatórios de grupos. Na primeira tela de grupos é exibida uma lista com marcadores simbolizando o *submenu* de opções de grupo (ver Figura 40). Nesse caso não há uma especificação no documento para tal. Porém, uma lista usada como menu é contraintuitivo, ferindo o princípio ‘reconhecer, em vez de relembrar’, já que o usuário precisa identificar que a lista é um menu, o que geralmente só acontece por tentativa e erro.

Figura 40. Menu da tela de grupos

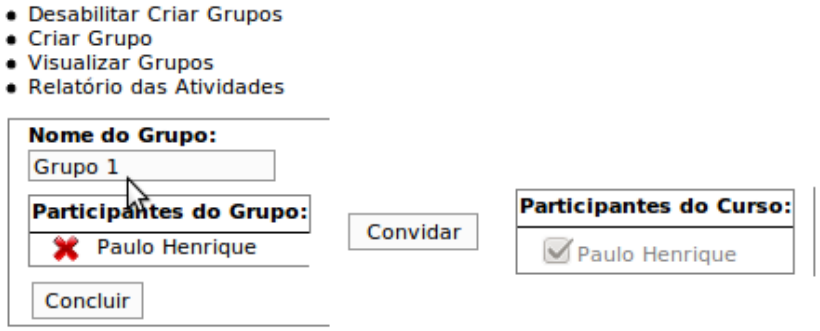


Fonte: Foto da tela do LMS AMADEUS.

Nessa tela também são exibidos os resultados das opções de ‘criar grupo’ e ‘visualizar grupos’, o que acontece sem o recarregamento da página, permitindo uma maior eficiência no uso. Porém como a navegação não é atualizada para a nova tela, isso pode gerar confusão por parte do usuário, que deixa de entender onde está localizado. Já ao acionar a opção ‘desabilitar

criar grupos’ uma janela de confirmação é exibida, questionando se o usuário realmente deseja executar aquela ação, estando de acordo com o princípio da ‘prevenção de erros’.

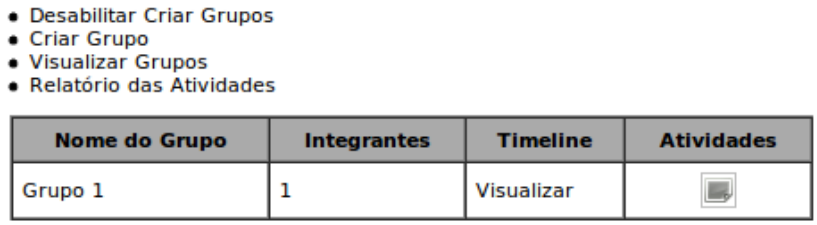
Figura 41. Tela de criar grupo



Fonte: Foto da tela do LMS AMADEUS.

A tela de formulários também está fora do padrão, pois os nomes dos campos estão em cor preta quando deveriam ser verdes. Ainda, há uma borda preta em parte da região do formulário. Essa falta de continuidade cria a impressão de falha no carregamento. Além disso a ação de convidar está semanticamente incorreta, pois ao selecionar um usuário na tabela à direita e clicar em ‘convidar’ ele é adicionado ao grupo automaticamente, quebrando o princípio de ‘compatibilidade do sistema com o mundo real’. Ainda assim é notada a ausência de um botão cancelar, indo contra o princípio ‘controle do usuário e liberdade’, e faltam mensagens de sucesso e erro, impedindo a ‘visibilidade de status do sistema’.

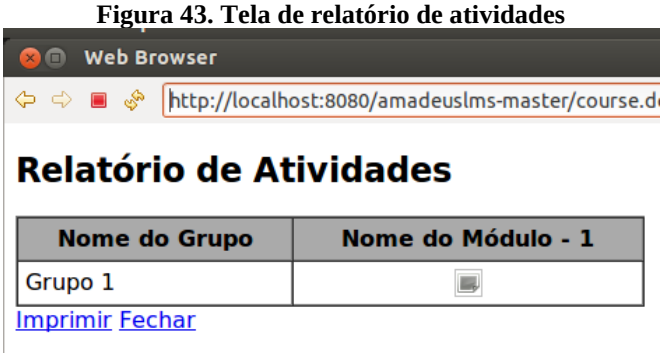
Figura 42. Tela de visualizar grupos



Fonte: Foto da tela do LMS AMADEUS.

Na tela de visualizar grupos é exibida uma tabela, que usa cores da paleta de forma incorreta, como no cinza aplicado como plano de fundo das células de título, quando no caso deveria ser verde claro. Além disso o preto deve ser usado apenas para textos, e não para definir bordas em tabelas, como foi feito nessa página. No entanto a tela é bastante objetiva, permitindo que o usuário acesse as opções praticamente sem errar.

Figura 43. Tela de relatório de atividades



Fonte: Foto da tela do LMS AMADEUS.

Já o relatório de atividades é exibido em uma nova tela, pois o objetivo é a impressão. Dessa forma, o *design* escolhido foi minimalista, sem exageros (ver Figura 43). Ainda assim o cinza médio escolhido para ser a cor das células de título pode gerar problemas em algumas impressoras, para isso um verde ou cinza mais claros seriam uma boa alternativa.

Dessa forma concluímos que a contribuição de PERRIS está em desacordo com o esperado por um usuário do LMS AMADEUS. As interfaces não estão consistentes com o padrão estabelecido pelo *software* e ferindo vários dos princípios definidos. Portanto, em uma situação real essa contribuição necessitaria de uma série de ajustes para ser integrada à versão oficial do projeto.

4.4.3 Avaliação dos resultados

Como resultado da inspeção de consistência, que avalia as interfaces com base nos princípios pré-definidos, podemos concluir que a avaliação de MEDEIROS está mais próxima do ideal, pois houveram apenas duas quebras de princípios, já a contribuição de PERRIS teve seis quebras de princípios. O Quadro 4 resume os resultados obtidos, no qual cada contribuição está relacionada aos princípios por meio de um '✓' indicando aprovação ou um '✗' indicando reprovação, ou seja, ao menos uma ocorrência de falha relacionada.

Quadro 4. Resultado da inspeção de consistência

Princípio	Contribuição	
	MEDEIROS	PERRIS
Visibilidade de status do sistema	✓	✗
Compatibilidade do sistema com o mundo real	✓	✗
Controle do usuário e liberdade	✓	✗
Consistência e padrões	✗	✗
Ajudar os usuários a reconhecer, diagnosticar e corrigir erros	✓	✓
Prevenção de erros	✗	✗

Reconhecer, em vez de relembrar	✓	✗
Flexibilidade e eficiência no uso	✓	✓
Estética e *design* minimalista	✓	✓
Ajuda e documentação	✓	✓

Fonte: O autor.

Já na revisão de *guidelines*, que avalia as interfaces com base no documento de diretrizes do produto, teve como conclusão que a contribuição de MEDEIROS foi um pouco melhor, com falhas em duas diretrizes, e a de PERRIS teve falhas em três diretrizes. O reúne todas os resultados dessa revisão, relacionando as contribuições com as diretrizes e sinalizados no mesmo formato do Quadro 4.

Quadro 5. Resultado da revisão de *guidelines*

Diretrizes	**Contribuição**	
	MEDEIROS	**PERRIS**
Grid	✓	✓
Cores	✗	✗
Tipografia	✓	✓
Formulários	✓	✗
Navegação	✗	✗

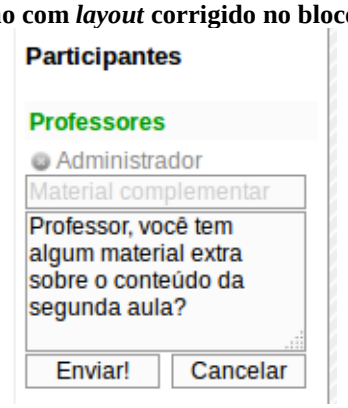
Fonte: O autor.

A quantidade de erros identificados evidencia a necessidade de uma documentação mais clara no que diz respeito a padrões de interface, pois dessa forma os colaboradores teriam condições de desenvolver interfaces consistentes com o projeto. Ainda assim, caso inconsistências fossem submetidas, essas seriam identificadas pelo gerente de configuração após realizar a análise heurística, que em seguida solicitaria correções ao colaborador, conforme descrito no processo sugerido neste trabalho.

4.5. REVISÃO DO CÓDIGO PARA CONSISTÊNCIA E PUBLICAÇÃO NO PORTAL SPB

Com os erros delimitados no capítulo anterior foi possível revisar o código para executar correções com o objetivo de tornar as interfaces mais consistentes. Todas as correções seguiram as observações delimitadas na seção 4.4.

Figura 44. Botão com *layout* corrigido no bloco de mensagens



Fonte: Foto da tela do LMS AMADEUS.

A primeira tela a ser corrigida na contribuição de MEDEIROS foi a de mensagens, na qual antes era exibido um botão com o posicionamento quebrado e que agora está correto (ver Figura 44). Também foi corrigida a ocorrência de bordas arredondadas em todas as telas dessa contribuição, deixando-as da forma vista na Figura 45.

Os *links* azuis foram corrigidos em todas as telas nas quais era exibido, passando a ter um estilo padrão do sistema, com cor preta e cor verde e sublinhado ao passar o mouse (ver Figura 45). Além disso, a tela de visualizar todas as mensagens teve o título alterado de ‘Dados do Curso’ para ‘Mensagens’, como pode ser visto na Figura 45, pois era uma falha capaz de gerar problemas de navegação ao usuário.

Figura 45. Tela de exibir todas as mensagens corrigida

Fonte: Foto da tela do LMS AMADEUS.

A tela de monitorar redes sociais (ver Figura 46) também teve correção no título, mas além disso todo o alinhamento foi refeito, colocando o ícone do Twitter à esquerda e o formulário com uma margem de 10 pixels à direita. Além disso a borda usada na tela para

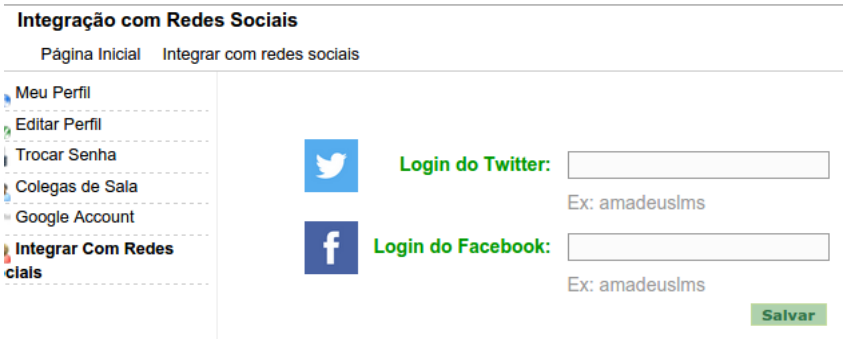
delimitar as áreas foi removida por não estar no padrão do sistema, tornando a tela mais limpa e confortável.

Figura 46. Tela de monitorar redes sociais corrigida

Fonte: Foto da tela do LMS AMADEUS.

Já a tela de integração com redes sociais teve os ícones substituídos, que no caso do Twitter é o mesmo da tela da Figura 47, mantendo-a coerente entre as interfaces. A escolha foi por ícones sem bordas arredondadas e sem variações de degradê ao fundo, seguindo o padrão visual adotado em todo o *site*. Também foi corrigido o alinhamento entre os ícones, antes desalinhados e, agora, alinhados à direita.

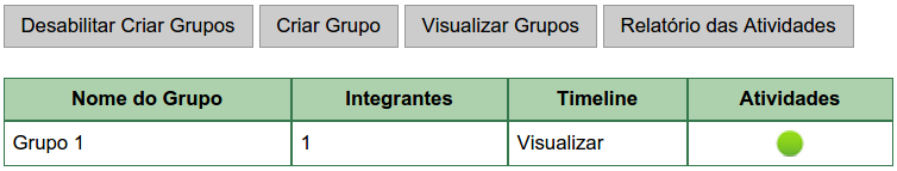
Figura 47. Tela de integrar redes sociais corrigida



Fonte: Foto da tela do LMS AMADEUS.

Na contribuição de PERRIS a correção iniciou pelo *menu* da tela de grupos, que antes era exibido como uma lista não ordenada e agora está em formato de *menu* horizontal, como visto na Figura 48. Dessa forma, ficando mais familiar ao esperado pelos usuários. Além disso a tabela exibida ao clicar em visualizar grupos também foi modificada para aderir ao estilo adotado pelo *site*, tendo as cores de plano de fundo e bordas sido alteradas conforme a definição do documento de diretrizes (ver Figura 48).

Figura 48. Menu e lista de grupos corrigidos



Fonte: Foto da tela do LMS AMADEUS.

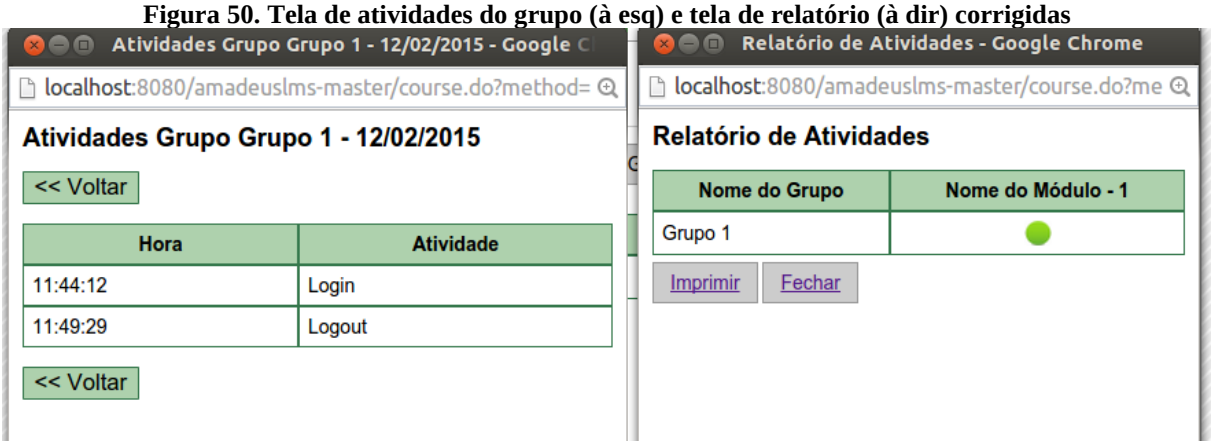
Na tela de criar grupo foram removidas as bordas irregulares e o alinhamento dos campos foi melhorado (ver Figura 49). Além disso foi corrigido o nome do botão ‘Adicionar’, que representava uma adição de um participante ao grupo, que era nomeado como ‘Convidar’. Nesse botão também foi adicionada uma seta para a esquerda, indicando para o usuário o sentido da ação.

Figura 49. Tela de criar grupo corrigida

Fonte: Foto da tela do LMS AMADEUS.

As telas de *timeline* e relatório foram mantidas como *pop-up*, porém agora estão com o padrão de interfaces estabelecido no sistema, tanto em termos de tipografia quanto no esquema de cores. Como pode ser visto na Figura 50, na qual estão dispostas lado a lado. Ainda assim as telas se mantiveram simples o suficiente para o caso de ser necessário imprimi-las.

Figura 50. Tela de atividades do grupo (à esq) e tela de relatório (à dir) corrigidas



Fonte: Foto da tela do LMS AMADEUS.

Com todas as modificações e *commits* realizados, o projeto foi sincronizado com o repositório no GitHub. A partir desse ponto nós tínhamos uma versão com as duas contribuições integradas e estável. Portanto, a disponibilizamos no Portal do *Software* Público Brasileiro. Essa nova versão foi publicada junto a um registro de alterações (*changelog*) que disponibilizamos abaixo:

- Integração da contribuição de MEDEIROS (2013), contemplando os módulos de redes sociais, mensagens e fórum;
- Integração da contribuição de PERRIS (2013), contemplando o módulo de grupos;
- Correções de consistência de interfaces nas duas contribuições;
- Refatoração de códigos do projeto; e
- Correções de *bugs* e pequenas melhorias.

5 DISCUSSÃO

A manutenção de projetos de *software* livre é de fato um grande desafio, tanto para os administradores e gerentes de configuração quanto para os novos colaboradores das comunidades. Para que isso aconteça com eficácia é preciso criar documentações como as de arquitetura, configuração, interfaces e visão do projeto, o que gera uma carga de trabalho normalmente não aceita pelos desenvolvedores, que querem codificar. Como mapeado por RAJANEN [9], a filosofia mais seguida em projetos de *software* livre é "falar é fácil, mostre-me o código", e esse tipo de pensamento é um dos grandes entraves para a adoção de documentações necessárias a uma evolução consistente.

No entendimento de fenômenos relativos à proposição de contribuições das comunidades de *software* livre percebemos que a ausência de documentação é um dos fatores para a geração de inconsistências. E essa ausência é tida como o principal problema encontrado por novos colaboradores das comunidades de *software* livre, como identificado por REDMILES [1], criando dificuldades até mesmo para encontrar uma forma de começar a contribuir. Esse problema pode se desmembrar de diversas formas, mas nesse trabalho, tivemos como objeto a consistência de interfaces, assim passamos a estudar soluções propostas por comunidades preocupadas em evitá-lo ou reduzi-lo.

Dentre as tentativas de soluções listadas estava usar o GitHub como ambiente de desenvolvimento distribuído, melhorando fluxo de colaboração e atraindo mais desenvolvedores. Em seguida percebemos a necessidade de formalizar o processo a ser executado nesse ambiente, o que fizemos por meio do estudo de caso de duas contribuições ao projeto. Numa tarefa desse processo, antes da tarefa de integração final, pudemos incluir a realização da análise heurística, que é uma técnica clássica e barata de usabilidade, sendo capaz de definir se uma interface está ou não consistente com o padrão definido nas diretrizes e princípios.

Com o processo em mãos, necessitávamos de um documento de diretrizes para executá-lo, o que foi feito por meio da análise do estilo do próprio sistema e de documentos de outros projetos. Dessa forma foi desenvolvida a primeira versão do documento de diretrizes de interface do LMS AMADEUS, permitindo a todos os colaboradores elaborarem novas telas mais próximas aos padrões usados no sistema. Para esta documentação ainda cabe uma clara evolução, como a previsão de novos formatos de tela. Porém, isso deve acontecer ao passo que o código das telas também evolua, mantendo sempre o documento atualizado e consistente com o que já está incorporado ao sistema.

A partir da documentação fomos capazes de analisar as duas contribuições do ponto de vista da consistência de interfaces, e com isso o que já era claro na observação ficou claro também em números. As duas contribuições tiveram diversas falhas, considerando dez (10) princípios e cinco (5) diretrizes a de MEDEIROS teve dois (2) problemas em cada. Já a de PERRIS teve seis (6) e três (3) problemas, respectivamente. Porém as colaborações não devem ser desmerecidas, pois isso é reflexo do principal problema citado neste trabalho, a ausência de documentação, fazendo com que os desenvolvedores não tenham material suficiente para entender o que e como deve ser feito. Documentações como essas resultam em desenvolvedores mais engajados e dispostos a contribuir para a evolução dos projetos em comunidades de *software* livre.

6 CONCLUSÃO

Em projetos de *software* livre os colaboradores têm dificuldades de encontrar uma forma de começar a contribuir, muito pela falta de entendimento das necessidades do projeto, pois é comum não haver documentação suficiente para tal. Também é raro encontrar especificações do *design* usado no projeto, com isso, quando desenvolvedores decidem contribuir, não o fazem mantendo as interfaces consistentes com o todo.

A elaboração de um processo de integração, no qual é previsto a necessidade de analisar a consistência de novas interfaces integradas ao projeto, significa um passo adiante no desafio de manter um projeto de *software* livre. Dessa forma foi realizado um estudo de caso com a integração de duas contribuições ao LMS AMADEUS, a fim de obter um processo capaz de representar de fato o novo ambiente de desenvolvimento colaborativo.

No processo de integração há a tarefa de realizar análise heurística, necessária para perceber se uma contribuição está consistente ou não, tanto por meio de princípios como através de diretrizes especificadas. Assim analisamos folhas de estilo do LMS AMADEUS e documentos de diretrizes de outros projetos de *software* livre, e com isso desenvolvemos um guia para interfaces de novas contribuições ao projeto.

Por fim avaliamos as duas contribuições do estudo de caso por meio do novo documento de diretrizes, como uma forma de validá-lo dentro do processo. Como resultado da avaliação ficou claro que a ausência de documentação gerou uma série de erros facilmente evitáveis, na qual foram mais críticos os encontrados na contribuição de PERRIS [25]. Porém em ambos os casos, o desenvolvedor não foi instruído de forma correta em como se apropriar dos padrões, amplamente usados nas telas já presentes no sistema.

Entendemos que, com um processo bem definido e uma documentação de interfaces disponível para os colaboradores, é possível manter projetos de *software* livre com interfaces consistentes em novas contribuições. E dessa forma o projeto pode se tornar mais atrativo a novos colaboradores, fomentando a comunidade e permitindo uma melhor evolução, tanto de códigos quanto de interfaces.

6.1 LIMITAÇÕES

A principal limitação deste trabalho é que o documento de diretrizes não relata todos os aspectos de interfaces que são possíveis em páginas *web*. Além disso o documento não possui uma versão reduzida, no formato de especificação técnica, similar as do Bootstrap [20]. Já na

revisão do código para consistência não pudemos corrigir alguns dos problemas encontrados, devido à grande profundidade desses.

6.2 TRABALHOS FUTUROS

Além de melhorar pontos definidos nas limitações, podemos citar como trabalho futuro um estudo de caso no qual sejam analisados dois projetos de *software* livre similares. Um deles tendo um processo de integração bem definido, e o outro sem processo de integração estabelecido. A partir desse estudo pode ser analisada na prática a eficiência real de ter ou não um processo definido.

Também podemos sugerir um estudo da comunidade do AMADEUS para a melhoria do documento de diretrizes. Nesse estudo seria analisada a necessidade de incluir novos pontos no documento, baseando-se nas dúvidas da comunidade, em entrevistas com colaboradores e na análise dos problemas existentes nas contribuições.

REFERÊNCIAS BIBLIOGRÁFICAS

1. REDMILES, D. et al. The Hard Life of Open Source Software Project Newcomers. **CHASE 2014 Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering**, Hyderabad, India, 2-3 jun. 2014. 72-78.
2. MURILLO, L. F. **Tecnologia, Política e Cultura na Comunidade Brasileira de Software Livre e de Código Aberto**. 2009. 172 f. Dissertação (Mestrado) - Curso de Antropologia Social, Instituto de Filosofia e Ciências Humanas, Ufrgs. Porto Alegre, RS. 2009.
3. REIS, C. R. **Caracterização de um Processo de Software para Projetos de Software Livre**. 2003. 158 f. Dissertação (Mestrado) - Curso de Ciências da Computação e Matemática Computacional, Departamento de Instituto de Ciências Matemáticas e de Computação, USP. São Paulo, SP. 2003.
4. PEREIRA JR, C.; CAPETO, R. Indicadores para avaliação de websites. **Anais do III Workshop sobre Fatores Humanos em Sistemas Computacionais, IHC 2000**, Gramado, RS, 18-20 out. 2000. 75-80.
5. U.S. DEPT. OF HEALTH AND HUMAN SERVICES. Research-Based Web Design and Usability Guidelines. **Usability.gov**. Disponivel em: <http://guidelines.usability.gov/>. Acesso em: 9 out. 2014.
6. GOMES, A. S. et al. Amadeus: Novo modelo de sistema de gestão de aprendizagem. **Revista Brasileira de Aprendizagem Aberta e A Distância**, Recife, PE, v. 8, p. 10-27, 2009.
7. FREE SOFTWARE FOUNDATION. O que é software livre? **Free Software Foundation**. Disponivel em: <http://www.gnu.org/philosophy/free-sw.html>. Acesso em: 29 out. 2014.
8. PREECE, J.; ROGERS, Y.; SHARP, H. **Design de Interação:** Além da Interação Homem-computador. 2. ed. São Paulo, SP: Bookman, 2005. 548 p.
9. RAJANEN, M.; IIVARI, N. Open Source and Human Computer Interaction Philosophies in Open Source Projects – Incompatible or Co-Existent? **AcademicMindTrek'13**, Tampere, Finland, 04 jan. 2013.
10. BENSON, C.; MÜLLER-PROVE, M.; MZOUREK, J. Professional Usability in Open Source Projects: GNOME, OpenOffice.org, NetBeans. **CHI 2004**, Vienna, Austria, 24-29 abr. 2004.
11. BURMESTER, M.; MACHATE, J. Creative Design of Interactive Products and Use of Usability Guidelines – a Contradiction? **Proceedings of HCI international 2003. Mahwah: Lawrence Erlbaum**, Mahwah, USA, 2003.
12. CRONHOLM, S. The Usability of Usability Guidelines - a Proposal for Meta-Guidelines. **Proceedings of the 21st Australasian Computer-Human Interaction Conference (OZCHI)**, Melbourne, Australia, 2009.
13. OPENREDU. Características. **Openredu**. Disponivel em: <http://openredu.cin.ufpe.br/?page_id=700&lang=pt>. Acesso em: 06 jan. 2015.
14. MOZILLA. Firefox OS Guidelines. **Mozilla Style Guide**. ISSN https://www.mozilla.org/en-US/styleguide/products/firefox-os/. Acesso em: 13 nov. 2014.

15. GOOGLE. Introduction. **Material design**. Disponivel em: <http://www.google.com/design/spec/material-design/introduction.html>. Acesso em: 13 jan. 2015.

16. NIELSEN, J.; MACK, R. L. **Usability Inspection Methods**. New York, USA: John Wiley & Sons, 1994.

17. RAJANEN, M.; IIVARI, N.; KESKITALO, E. Introducing Usability Activities into Open Source Software Development Projects – a Participative Approach. **Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design**, New York, USA, 2012.

18. MEDEIROS, L. et al. Uso de StoryBoards para a Documentação dos Requisitos no Desenvolvimento Distribuído de Software. **I Workshop de Desenvolvimento Distribuído de Software (WDDS) - SBES 2007**, João Pessoa, PB, 2007.

19. MEDEIROS, L. M. **Proposta de Processo de Requisitos para Equipes de Desenvolvimento Distribuídas Visando Melhorar a Documentação e Validação com Uso de Protótipos**. UFPE. Recife, PE, p. 165. 2007.

20. TWITTER. About. **Bootstrap**. Disponivel em: <http://getbootstrap.com/about/>. Acesso em: 26 jan. 2015.

21. CHACON, S.; STRAUB, B. **ProGit**. 2. ed. San Francisco, USA: Apress, 2014.

22. GITHUB. About. **GitHub**. Disponivel em: <https://github.com/about>. Acesso em: 14 nov. 2014.

23. PRESSMAN, R. **Software Engineering:** A Practitioner's Approach. 7. ed. New York, USA: Mcgraw-hill, 2009. 976 p.

24. MEDEIROS, F. P. **Uma abordagem de monitoramento abrangente da experiência do usuário em ambientes colaborativos virtuais de aprendizagem como suporte a presença docente**. 2013. 205 f. Tese (Doutorado) - Curso de Ciência da Computação, Centro de Informática, Ufpe. Recife, PE. 2013.

25. PERRIS, P. A. **Colaboração e coordenação de grupos em Ambiente LMS**. 2013. 125 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Centro de Informática, Ufpe. Recife, PE. 2013.

26. GOVERNO FEDERAL. Amadeus. **Portal do Software Público Brasileiro**. Disponivel em: <http://www.softwarepublico.gov.br/ver-comunidade?community_id=9677539>. Acesso em: 13 out. 2014.

27. SHAPIRO, R. et al. **BPMN 2.0 Handbook**. 2. ed. [S.l.]: [s.n.], 2012.

28. ROCHA, H.; BARANAUSKAS, M. **Design e avaliação de interfaces humano-computador**. São Paulo, SP: IME-USP, 2000. 242 p.