UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

DESENVOLVIMENTO DE SISTEMA DE INFORMAÇÃO PARA TROCA DE DADOS BASEADO NO PADRÃO XML UTILIZANDO FILEMAKER

HENRIQUE M. F. BILBAO

BLUMENAU
2012

2012/2-14

HENRIQUE M. F. BILBAO

DESENVOLVIMENTO DE UM SISTEMA DE INFORMAÇÃO PARA TROCA DE DADOS BASEADO NO PADRÃO XML UTILIZANDO FILEMAKER

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Oscar Dalfovo, Dr.

BLUMENAU
2012

2012/2-14

DESENVOLVIMENTO DE UM SISTEMA DE INFORMAÇÃO PARA TROCA DE DADOS BASEADO NO PADRÃO XML UTILIZANDO FILEMAKER

Por

HENRIQUE M. F. BILBAO

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Oscar Dalfovo, Doutor – FURB
Membro:	Prof. Jacques Robert Heckmann
Membro:	Prof. Cláudio Ratke

Blumenau, 15 de Dezembro de 2012

Dedico este trabalho a minha esposa, que sempre esteve ao meu lado durante o desenvolvimento deste trabalho e foi a maior incentivadora para conclusão do projeto, também aos meus pais, incentivadores da iniciação da graduação e amigos que me auxiliaram na elaboração do trabalho.

AGRADECIMENTOS

Agradeço inicialmente a DEUS, a quem devo tudo o que sou.

À minha esposa, Alessandra Neves Hoffmann Bilbao, pelo incentivo e apoio para conclusão deste projeto, por todo amor, carinho e compreensão.

À minha Família, pelo exemplo e amor, que ajudaram muito a definir o meu caráter, e a escolha sábia de cursar algo que realmente amo.

Ao meu orientador, Professor Oscar Dalfovo, pela paciência, pelas sugestões, por ter acreditado na realização desta pesquisa e confiado em meus ideais.

Ao coordenador do Curso de Bacharelado em Ciências da Computação, Professor Alexander Roberto Valdameri, sempre solícito e compreensivo às nossas dificuldades.

Aos professores, colegas e todos os integrantes do curso de graduação, que direta ou indiretamente contribuíram para a conclusão desse trabalho.

Aos colaboradores do fórum da *FileMaker Business Alliance* – FBA, pela oportunidade de realização da pesquisa e pela colaboração na coleta de informações.

Aos professores Dalton Solano dos Reis e Paulo Rodacki pelos incentivos mac-maníacos concedidos durante toda graduação para que pudesse do começo ao fim deste curso utilizar recursos Apple.

Aos meus grandes amigos, Josimar Zimmermann e Daniel Batiston, pela amizade e ajuda nas pesquisas que pareciam malucas, mas foram tomando rumo e empolgando cada vez mais.

“Muitas pessoas na nossa indústria não tiveram experiências diversas. Então não possuem pontos para conectar e acabam chegando a soluções lineares, sem uma perspectiva ampla do problema. Quanto mais ampla for a visão do entendimento da experiência, melhor o projeto será”

Steve P. Jobs

RESUMO

Neste trabalho é apresentado o desenvolvimento de um aplicativo para acesso a informações de pedidos de um restaurante, armazenadas em um banco de dados FileMaker através de um padrão XML. O acesso é feito por um dispositivo iPad, o qual conecta em um servidor que mantém o banco de dados e os códigos desenvolvidos para trazer as informações de forma organizada. Além de promover essa estrutura, ao receber as informações, o iPad faz um processo de entendimento dos dados importados para apresentar de maneira gráfica os dados do restaurante, como pedidos em andamento, itens de maior consumo e algumas estatísticas. Para o desenvolvimento deste aplicativo foram utilizados um iPad, um servidor Apple e o FileMaker instalado em ambos, focando a melhor experiência no conceito da troca de informações entre eles de forma diferenciada, onde até então o padrão era via conexões da própria FileMaker, limitando o uso a somente programas desenvolvidos em FileMaker. Utilizou-se a linguagem FileMaker para desenvolvimento da camada de visão e XML e PHP como linguagem de servidor. O banco de dados utilizado foi o FileMaker. Com a ferramenta pronta, aplicou-se um questionário de qualidade de software a um grupo de usuários para avaliar a qualidade, eficiência e usabilidade da ferramenta elaborada. Como resultado o aplicativo mostrou-se funcional, confiável, usável e eficiente para os usuários.

Palavras-chave: Apple. iPad. XML. FileMaker.

ABSTRACT

This project presents the development of an application that is able to access a restaurant's FileMaker database, by a XML standard. Access is through an iPad device, connecting to the server that host the database and developed codes that brings the information organized. Besides promoting this structure, upon receiving the information, the iPad make a process to understand the imported data to graphically present the data of the restaurant, as pendencies, top consumed items and some statistics. To develop this application an iPad, Apple Server and FileMaker licenses were utilized, where the concept of exchanging different information between the devices was focused. Nowadays, the standard of exchange information using FileMaker tools is limited only for programs or databases developed by FileMaker, this project represents a new way to exchange information between an FileMaker layer and database language with a XML and PHP server language. With the project ready, a quality questionnaire was applied with a group of users to evaluate the quality, efficiency and usability of the project. The result of the application showed that the application seems to be functional, reliable, usable and efficient for users.

Key-words: Apple. iPad. XML. FileMaker.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1. **INTRODUÇÃO**

A tendência tecnológica está caminhando cada vez mais para a produtividade e a facilidade de obter informações. Na área de gestão de empresas isso cresce gradualmente, com a necessidade de velocidade e qualidade, principalmente no acesso aos resultados obtidos no decorrer dos meses, dias e, em alguns casos, até horas. Com o aumento do uso da informática por administradores de empresas e gerentes de negócios, a oportunidade de obter informações ampliou-se e, unida com a possibilidade de acesso remoto através de dispositivos móveis, tornou-se grande aliada no aprimoramento da gestão dos negócios e identificação de custos, permitindo maior rapidez na tomada de decisão (DALFOVO; TAMBORLIN, 2010, p. 1).

A redução de custos é uma busca constante das organizações. No passado as empresas tinham como meta, nesse sentido, a eliminação de desperdício e redução do quadro funcional. Atualmente, como as estruturas organizacionais já estão bastante enxutas, a melhoria do desempenho dos processos e o aumento da produtividade é o caminho natural para a redução de custos e aumento nos resultados. Algumas empresas conseguiram implementar ações, com base em uma mudança de cultura, passando a buscar uma gestão baseada em informações rápidas e geradas por ferramentas tecnológicas (BILBAO; FRANZ, 2009, p. 41).

Sendo a alta competitividade uma tendência crescente, é necessário que o planejamento estratégico esteja presente, tanto nas organizações como na própria vida pessoal. O planejamento estratégico, a bem da verdade, está atento para ações que podem ser tomadas para enfrentar ameaças e aproveitar as oportunidades encontradas no ambiente competitivo. E vale ressaltar que os ambientes mudam com extrema rapidez (BILBAO; FRANZ, 2009, p. 86).

A gestão de uma empresa deve respeitar normas legais quanto à apresentação de seus números. Os balanços seguem padrões definidos por lei, e a entrada e saída de mercadorias e a prestação de serviços são monitoradas pelo governo através de sistemas de controle de recolhimento de impostos. A fiscalização das empresas baseia-se em documentos e ferramentas padronizadas e homologadas pelo governo. Por exemplo, o emissor de cupom fiscal de uma empresa de varejo deve ser homologado e lacrado conforme a legislação (NUNES; BARBOSA; LUZIRENE, 2011, p. 36)

Com o avanço da tecnologia e da informática, os profissionais das áreas de gestão das empresas já podem usufruir de diversas ferramentas, tais como *notebooks* e dispositivos

móveis, como o iPhone e o iPad para atender as necessidades de visualização de dados, trazendo assim, a possibilidade dos profissionais destas áreas terem acesso a informações que ajudam a tomada de decisão (LIMA, 2012, p.1).

O comércio é uma das atividades mais dinâmicas, mudando a todo momento em resposta às preferências dos consumidores e à oferta de novos produtos. Acompanhar sistematicamente o que se passa no mercado (lançamentos, produtos substitutos, novos concorrentes, novas tecnologias, alternativas de financiamento e possibilidade de crédito) é antes de tudo, uma questão de sobrevivência para quem está e pretende continuar no comércio (RUAS; PINHEIRO, 1996, p. 9).

É inevitável que o empresário queira acompanhar, mesmo que a distância, o que está acontecendo em sua empresa. O protótipo que será desenvolvido neste trabalho demonstrará uma das maneiras deste acesso ser feito de forma ágil e eficiente, sem a necessidade de uso de sistemas complexos. O protótipo deverá ser rápido na troca de informações entre a empresa e seu gestor, mesmo se for acessado fora da sua rede local. O sistema que terá seu banco de dados aberto para integração é o software chamado RestauranteHi (sistema desenvolvido para substituir os cardápios por iPads, concentrando dados em um servidor com banco de dados em FileMaker).

O sistema de troca de informações desenvolvido neste trabalho permitirá que o gestor tenha acesso a informações pré-determinadas e exportadas do sistema de cardápios em iPad desenvolvido pela HiMaker Sistemas chamado RestauranteHi. Será utilizado o padrão *eXtensible Markup Language* (XML) como linguagem para troca de dados entre o RestauranteHi e o gestor, já que a linguagem permite descrever, armazenar, intercambiar e manipular dados. De acordo com Heitlinger (2001, p. 9), o XML é o formato universal para compartilhamento de dados entre aplicações e suas possibilidades são inúmeras, podendo conter bases de dados, transações comerciais, catálogos de produtos, relatórios e mais uma infinidade de informações.

Com o avanço da tecnologia móvel e com a necessidade de acesso rápido às informações, justifica-se a criação de uma ferramenta de troca de informações entre o sistema de gestão da organização (RestauranteHi) e dispositivos móveis (no caso o iPad), permitindo ao usuário analisar informações, como por exemplo, gráficos de desempenho, escolhas mais frequentes de pratos e sobremesas servidos em determinados dias.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é o desenvolvimento de um sistema para troca eletrônica de dados baseada no padrão XML aplicada a restaurantes, concentrando informações em um servidor com banco de dados FileMaker instalado.

Os objetivos específicos do trabalho são:

a) disponibilizar informações do restaurante em um servidor com banco de dados em FileMaker remotamente, sobre vendas até a hora atual, metas de vendas estabelecidas e itens de maior consumo;
b) Exportar dados do sistema aplicado chamado RestauranteHi, exportando-o para o formato XML;
c) disponibilizar informações importadas a partir de um servidor com banco de dados em FileMaker para dentro do sistema instalado no iPad;
d) disponibilizar telas e relatórios para visualizar graficamente as metas e comparações de indicadores de desempenho, através de dispositivos móveis (iPad).

1.2 ESTRUTURA DO TRABALHO

No primeiro capítulo apresenta-se a introdução, mostrando a tendência do uso dos dispositivos móveis para produtividade, além de alinhar onde o sistema proposto se insere e objetivos do trabalho.

No segundo capítulo expõe-se a fundamentação teórica sobre o tema estudado, elucidando os conhecimentos necessários para compreensão do trabalho. A fundamentação consiste no dispositivo móvel iPad, Sistemas de Informação, XML, XMLMind, PHP, FileMaker e trabalhos correlatos.

No terceiro capítulo apresentam-se os detalhes inerentes ao desenvolvimento e programação do aplicativo, detalhando as especificações e implementações aplicadas durante o processo de elaboração do trabalho. Também são expostos os detalhes da operacionalidade, bem como os resultados alcançados com o trabalho.

No quarto capítulo estão as conclusões e ideias para extensões futuras ao trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os conceitos relativos a Sistemas de Informação, iPad, XML, XMLMind, PHP, FileMaker, trabalhos correlatos e RestauranteHi.

2.1 SISTEMAS DE INFORMAÇÃO

Os canais de informação e as redes de comunicação são elementos fundamentais para a tomada de decisões. Com os canais de informação, as empresas definem onde serão adquiridos os dados e com as redes de comunicação é possível definir para onde os dados serão direcionados (BATISTA, 2004, p. 20).

Para a realização bem-sucedida dos trabalhos ligados ao conhecimento, Davenport e Prusak (1999, p. 12) debatem que é essencial que as organizações saibam definir o que são dados, informações e conhecimento. O sucesso ou o fracasso organizacional muitas vezes pode depender da aplicação destes elementos para solução de problemas e tomada de decisões.

Padoveze (1997, p. 43) evidencia que a informação é o dado que foi processado de forma compreensível ao receptor e que apresenta valor real para basear suas decisões correntes ou prospectivas. Desta forma, compreende-se que a exportação de dados contribuirá para que o protótipo corresponda ao seu objetivo.

Stair (1998, p. 5) conceitua sistemas de informação como sendo um conjunto de dados, regras, procedimentos e relações que devem ser seguidos para se atingir o valor informacional ou resultado adequado do processo que está contido na base do conhecimento.

O protótipo a ser desenvolvido é baseado em um Sistema de Processamento de Transações (SPT), que consiste na troca de valores que afetam a lucratividade ou o ganho global de uma organização. O SPT pode ser considerado o centro do sistema da empresa em nível transacional, apoiando a realização e monitorando as negociações (DALFOVO; TAMBORLIN, 2010, p. 26).

2.2 IPAD

O iPad é um dispositivo móvel que possibilita a execução de vários aplicativos, além de ter tamanho que permite uma visualização de qualidade juntamente com a facilidade de mobilidade (APPLE, 2010a, p. 9).

O iPad apresentado na figura 1 é um dispositivo com várias características empregadas pela Apple Incorporation, tais como resolução de 1024x728 *pixels* em uma tela de 10 polegadas, orientação da tela de acordo com o movimento do usuário, opções de uso de teclado externo e padrão de cores em 24 bits, com mais 8 bits para uso de camadas alpha (que indica a transparência da imagem). A própria Apple recomenda o uso do *Picture Network Graphics* (PNG) como formato de imagem (APPLE, 2010a. p. 9).

Fonte: Apple (2010b, p.7).
Figura 1 – Dispositivo móvel com recursos computacionais da Apple

No dispositivo iPad está uma versão do protótipo que será aberta por meio do software FileMaker Go, o qual servirá como meio de comunicação para receber as informações em XML.

2.3 XML

XML é o formato universal para partilha de dados entre aplicações. O conceito XML é simples e as possibilidade são inúmeras. Documentos em formato XML podem conter: base de dados, transações comerciais, catálogos de produtos, gráficos vetoriais, equações matemáticas, fórmulas químicas, relatórios financeiros, dados bibliográficos, anúncios publicitários, enfim, quase todos os dados estruturados, em documento de texto (HEITLINGER, 2001 p. 3).

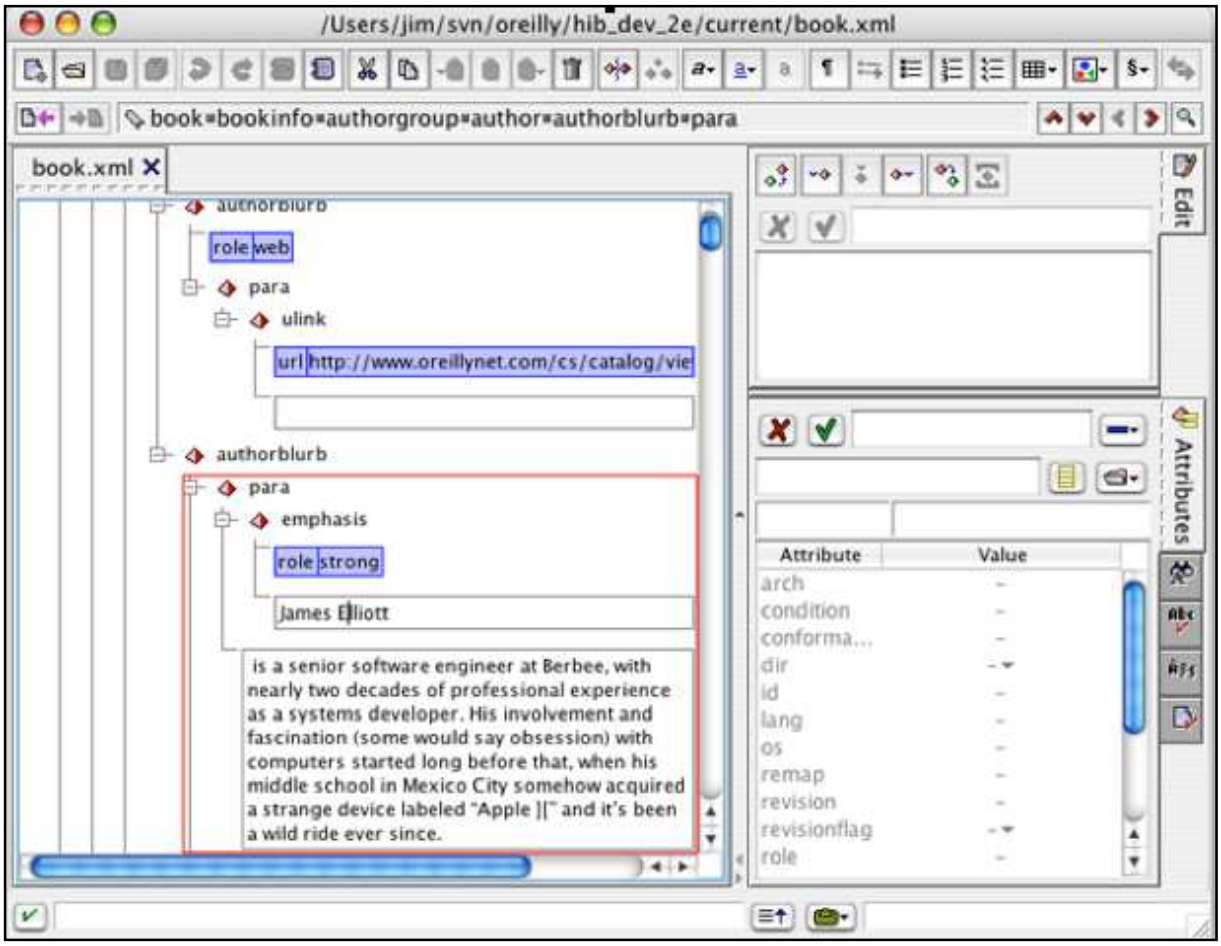
A flexibilidade do XML provém da facilidade de transportar variados tipos de dados e mantê-los estruturalmente coesos. Heitlinger (2001, p. 9) comenta que a XML serve muito bem para a estruturação de alguns tipos de dados e para descrevê-los sem dúvidas ou ambiguidades em formato de texto. O autor já defendia em 2001 que a XML iria fomentar o surgimento de uma nova geração de aplicações de manipulação e visualização de dados.

McLaughlin (2001, p. 301) afirma que a XML resolve o problema de transferência de dados e é aceita em quase todas as linguagens de programação, como C, Perl, Python, LISP, Java, Pascal, C#, Visual Basic, etc. No entanto, as estruturas de serviços *online* vão um pouco mais longe. A chave para a interoperabilidade não são apenas dados, mas que tipos de dados, além de saber como obter e fornecer informações de um sistema. O autor afirma que deve haver um meio de difusão que faz serviços de um aplicativo para outro.

2.4 XMLMIND

O XMLMind é um software validador e editor de arquivos XML com recursos semelhantes a um processador de textos, com base na tecnologia Java. Ele torna a autoria dos conteúdos mais produtiva, possui uma *engine* de planilha eletrônica integrada que a torna adequada a uma grande quantidade de aplicações. Além disso, é altamente extensível e suas extensões são fáceis de ser desenvolvidas. Roda em ambiente gráfico X11, Mac Os, Linux e Windows XP. O XMLMind permite a conversão de documentos XML em arquivos de ajuda de HTML, PDF, entre outros (XMLMIND, 2012).

Na figura 2 é possível visualizar a programação exportada com base no editor de XML.



Fonte: Elliot e Loy (2007).

Figura 2 – Visualização em árvore do software XMLMind

Na figura 3, exibida abaixo, é possível visualizar os nós do XML gerado para leitura de dados da localização de um restaurante.

```
Line
 1 <?xml version="1.0" encoding="UTF-8" ?>
 2 <restaurant>
 3     <info>
 4         <name>Diner</name>
 5         <logo>content/logo.png</logo>
 6         <address>
 7             <street>1567 Broadway</street>
 8             <city>New York</city>
 9             <country>United States</country>
10             <coordinates>
11                 <latitude>40.75930175423635</latitude>
12                 <longitude>-73.98525953292847</longitude>
13             </coordinates>
14         </address>
15         <map>
16             <minZoomLevel>12</minZoomLevel>
17             <maxZoomLevel>17</maxZoomLevel>
18         </map>
19         <telephone>+2129189999</telephone>
20         <url>http://www.diner.com/</url>
```

Fonte: Jarragh (2011).

Figura 3 – Configuração em XML de um menu de restaurante

2.5 *HYPERTEXT PREPROCESSOR*

Significa um acrônimo recursivo para PHP, é uma linguagem de *script open source* de uso geral, muito utilizada e especialmente guarnecida para o desenvolvimento de aplicações Web embutível dentro do HTML (PHP.NET, 2012).

Ao invés de muitos comandos para mostrar HTML, páginas PHP contém HTML juntamente com códigos que fazem algo na página. O código PHP é delimitado por *tags* iniciais e finais `<?php` e `?>` que lhe permitem pular pra dentro e pra fora do "modo PHP" (PHP.NET, 2012).

2.6 FILEMAKER

O FileMaker é uma linguagem avançada de desenvolvimento e customização de banco de dados nativo. A linguagem possui suporte a *Open Data Base Connectivity* (ODBC), utilizando drivers que permitem o compartilhamento de dados e a interação com dados de outros aplicativos. Ele também serve para acessos via *Hipertext PreProcessor* (PHP) e demais linguagens que disponibilizam compatibilidade de conexão (HENRY, 2010, p. 11).

A empresa FileMaker, Inc., responsável pela linguagem FileMaker, é subsidiária da Apple desde 1998 e, portanto, segue algumas regras internas que beneficiam desenvolvedores, com necessidade de credenciamento para atuar no mercado de desenvolvimento Apple. Os produtos desenvolvidos pela FileMaker já foram vendidos para clientes em todo o mundo, como pequenas e médias empresas, organizações governamentais, universidades, fundições e outras que o utilizam para administrar projetos, recursos e pessoas de forma simples.

Enquanto outras linguagens usam jargões como consulta, *join* (a combinação de registros de duas ou mais tabelas em um banco de dados) e *alias* (comando definido pelo usuário para automatizar tarefas), o FileMaker utiliza conceitos como encontrar, classificar e conectar, além de trazer a conexão entre o sistema operacional MacOS e Microsoft Windows (PROSSER, 2010, p. 3). É necessário a instalação do FileMaker na máquina que pretende-se abrir o banco de dados. FileMaker sendo passível de instalação em ambos os sistemas operacionais, pode-se abrir a mesma base de dados através de uma rede local e até uma rede remota, deixando a conexão entre computadores de diferentes plataformas funcional.

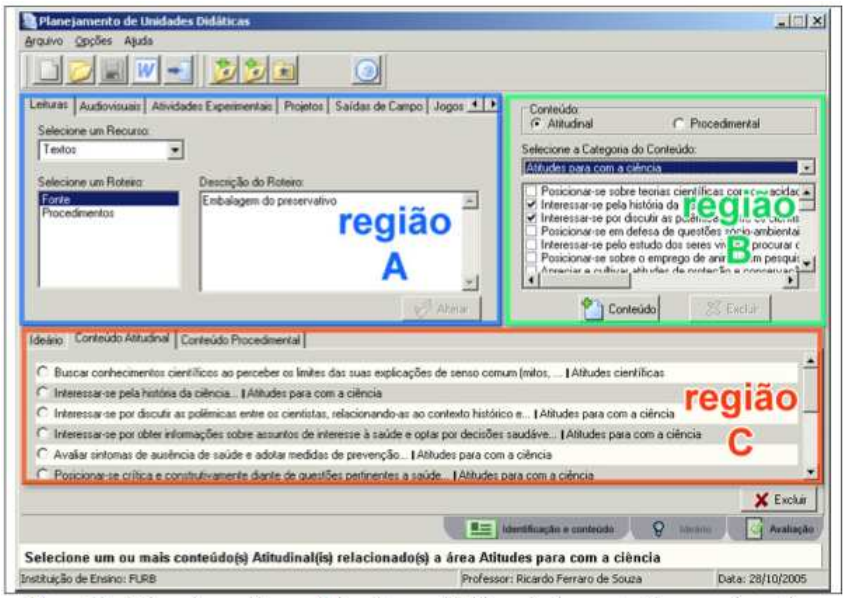
A característica principal do FileMaker é que o motor de banco de dados é integrado com as formas (telas, layouts, relatórios, etc) utilizadas para acessá-lo. Ele é projetado para ser fácil de desenvolver e fazer alterações em tempo real.

Outra característica interessante é que o FileMaker é multi-plataforma e sua interface de desenvolvimento é baseada em *Graphical User Interface* (GUI), permitindo aos usuários modificarem o banco de dados arrastando elementos novos em *layouts*, telas ou formulários e ver em tempo real as modificações dos mesmos.

Para ter acesso aos dados em FileMaker através do PHP, é necessário usar de uma tecnologia que a FileMaker adotou chamada *Custom Web Publishing* (CWP), que interpreta dados FileMaker de uma forma personalizada para Web. É disponibilizado uma API FileMaker para PHP que é uma classe PHP criada pela FileMaker que acessa banco de dados hospedados em um servidor FileMaker (BOWERS, 2011, p. 1).

2.7 TRABALHOS CORRELATOS

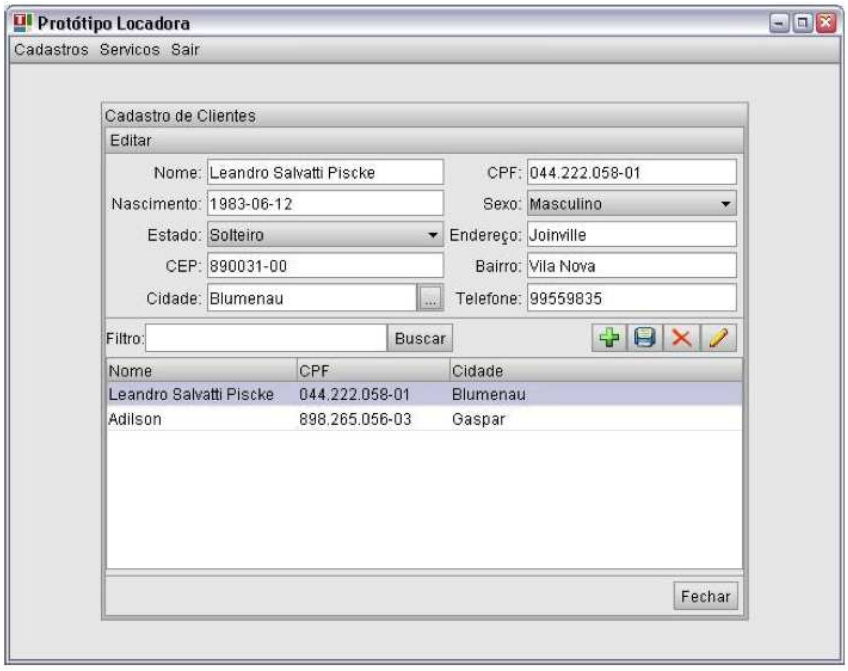
Para auxiliar os professores no planejamento de unidades didáticas no aspecto dos conteúdos abordados e estratégias utilizadas, Souza (2005, p. 1), Figura 4, desenvolveu uma ferramenta utilizando o XML para facilitar a elaboração de unidades didáticas a partir de uma situação contexto e em consonância com conteúdos das áreas de biologia, física e química do ensino médio. Este trabalho especificou a estrutura de uma ferramenta que utiliza a tecnologia *extensible Markup Language Schema* (XML *Schema*), a qual permite definir a estrutura utilizada, bem como possibilita a validação dos documentos XML gerados, viabilizando a integração com outros ambientes. Segundo a entidade mantenedora dos padrões utilizados pela *web*, (WORLD WIDE WEB CONSORTIUM, 2003, p. 1), “o uso desta tecnologia [XML *Schema*] vem ganhando aceitação como principal forma de especificar documentos XML”. O trabalho atingiu o objetivo principal.



Fonte: SOUZA (2005, p.43).

Figura 4 – Tela onde o professor adiciona itens no ideário

Já o trabalho de Piscke (2007, p. 1), Figura 5, usou XML apenas para fazer a ligação com o *Data Transfer Object* (DTO), pois seu objetivo era desenvolver um *framework* em Java para a geração automática de telas no modelo *Create, Read, Update and Delete* (CRUD). Segundo Alur, Crupi e Malks (2004, p. 374), um DTO serve para a transferência de dados entre camadas. Ele contém todos os elementos de dados em uma única estrutura tanto para solicitações quanto para respostas.



Fonte: PISCKE (2007, p.58).

Figura 5 – Tela do cadastro de clientes do trabalho de Piscke

Yoder, Johnson e Wilson (1998, p. 9) defendem que objetos persistentes precisam de operações de leitura e escrita em um banco de dados, pois às vezes eles precisam ser excluídos e é importante fornecer um mínimo de operações, para criar, ler, atualizar e excluir os dados usando uma arquitetura no padrão *Model View Controller* (MVC). Como forma de complemento, Gamma et al. (2000, p. 20) diz que, antes do MVC, os projetos de interface para o usuário tendiam a agrupar os objetos de modelo, visão e controle.

O principal objetivo do trabalho de Piscke é separar dados lógicos de negócios da interface do usuário e o fluxo da aplicação. Os resultados apresentados pelo *framework* desenvolvido comprovam que sistemas de informação podem ser simplificados e desenvolvidos com o foco na implementação da lógica de negócio da aplicação (PISCKE, 2007, p. 64).

2.7.1 Semelhanças e diferenças

O Quadro 1 faz o comparativo entre os dois trabalhos correlatos acima citados.

Trabalho correlato	Uso de PHP	Uso de XML	Construção de aplicativo
Piscke		X	
Souza		X	

Quadro 1 – Principais características implementados pelos sistemas

2.8 RESTAURANTEHI

O RestauranteHi é um sistema que auxilia o menu convencional, onde este é utilizado para enviar os pedidos selecionados diretamente para o servidor do restaurante. Neste sistema os pedidos são enviados diretamente para impressoras não fiscais intaladas na cozinha e no bar do restaurante, assim, o controlador dessas impressoras fica no mesmo local onde está o servidor FileMaker.

2.8.1 Funcionamento do RestauranteHi

A Figura 6 mostra o fluxo de uso do RestauranteHi, onde a Tela A tem como objetivo fazer a escolha da mesa para quem o garçom levará o iPad. Desta maneira os pedidos são agrupados por mesa para facilitar a entrega dos itens solicitados pelo cliente.

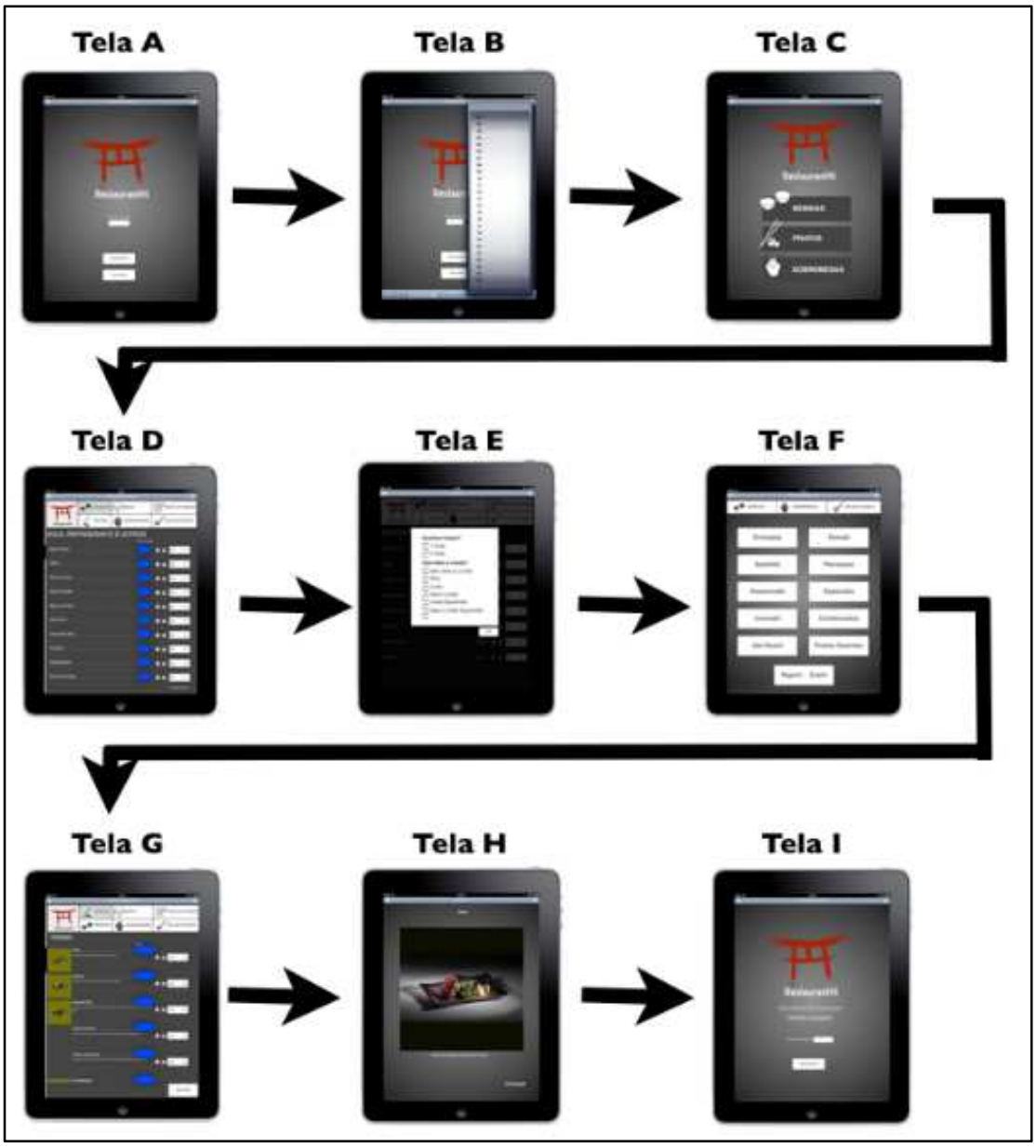


Figura 6 – Fluxograma de sequencia de telas para pedidos no RestauranteHi

Após, conforme Tela B da Figura 6, a lista apresentada para escolha vem do limite de mesas inseridos pelo administrador do restaurante.

O cliente recebe o iPad em mãos diretamente na tela do menu principal, Tela C, onde se pode escolher o sub-menu que desejar.

Caso o cliente clique em Bebidas, é aberta a tela com a lista de todas as bebidas em estoque, Tela D, separadas por tipos (água, cervejas, vinhos, etc). O cliente pode clicar nos botões de `adicionar` para solicitar um item ou aumentar a quantidade como também pode clicar em `X`, que significa diminuir a quantidade pedida.

No caso da requisição de uma bebida, dependendo do seu tipo, é apresentado um pop-up de opções, Tela E, com quantidade de copos e se deseja-se gelo ou limão. Desta maneira ao ser impresso nas impressoras não fiscais que estão localizadas no bar é colocado uma observação ao pedido com a opção escolhida pelo cliente.

Na seção de Pratos, Tela F, para simplesmente não ter uma lista, o restaurante optou por uma tela com os grupos em formas de botões para o cliente ter uma melhor forma de escolha. Assim pode-se filtrar de acordo com o clique que o cliente fizer nesta tela os pratos, conforme Tela G.

Após selecionar uma opção do sub-menu de pratos, são listados os pratos, preços, fotos e quantidade para novamente o cliente fazer a escolha do que deseja e sua respectiva quantidade.

Caso o cliente tenha dúvida do que trata-se determinado prato, esse pode clicar sobre a imagem que fará o tamanho da imagem aumentar, apresentando também uma breve descrição sobre o prato, Tela H.

Ao finalizar os pedidos, o cliente clica sobre o botão de `Finalizar Pedido` que o levará a tela de `Chame o garçom`, Tela I, pois o iPad pode ser retirado da mesa ou o garçom pode conferir o pedido.

3 DESENVOLVIMENTO

O desenvolvimento do protótipo inicia-se com a percepção da necessidade de uma tecnologia para gerenciar a troca de informações além dos recursos já oferecidos pela própria FileMaker, pois estes recursos se limitam aos dispositivos que a Apple vende como iPad, iPhones e iPod *Touchs*. Realiza-se o levantamento e análise dos requisitos do sistema, os quais serão expostos neste capítulo, juntamente com detalhes de especificação e implementação do sistema. Por fim, são apresentados os resultados obtidos com o trabalho.

3.1 REQUISITOS DO SISTEMA

No Quadro 2 e no Quadro 3 são apresentados, respectivamente, os requisitos funcionais (RF) e os requisitos não funcionais (RNF) implementados no protótipo.

REQUISITOS FUNCIONAIS
RF01: O sistema cliente deverá permitir exportação de dados de determinadas tabelas.
RF02: O sistema cliente deverá permitir que o usuário visualize gráficos de metas de vendas estabelecidas e itens de maior consumo.
RF03: O sistema cliente deverá permitir que o usuário visualize gráficos de vendas.
RF04: O sistema cliente deverá permitir que o usuário visualize itens de maior consumo.
RF05: O sistema cliente deverá controlar identificação e autenticação.

Quadro 2 – Requisitos funcionais

REQUISITOS NÃO FUNCIONAIS
RNF01: O sistema servidor deverá utilizar FileMaker como banco de dados.
RNF02: O sistema servidor deverá ser executado em computadores Apple com sistema operacional Mac OS 10.5 ou superior e iPad com sistema iOS 4.2.1 ou superior.
RNF03: O sistema servidor deverá ser desenvolvido em FileMaker.

Quadro 3 – Requisitos não funcionais

3.2 ESPECIFICAÇÃO

Neste capítulo é apresentada a especificação do protótipo, a qual foi modelada utilizando-se a ferramenta FileMaker. O sistema foi desenvolvido seguindo a análise orientada a objetos. Utiliza-se a notação *Unified Modeling Language* (UML) para a criação do

diagrama de casos de uso, atividades e sequência.

3.2.1 Modelo de entidades e relacionamentos do sistema RestauranteHi

A Figura 7 mostra como encontra-se o modelo de entidade e relacionamento da parte de pedidos no sistema RestauranteHi. Há uma divisão onde a primeira parte, intitulada ESS, significa, de acordo com o desenvolvedor do projeto RestauranteHi *External SQL Source*, pois é nesta área que se tem uma tabela de integração para que o protótipo possa consultar/coletar dados.

A segunda divisão traz as tabelas que são necessárias para constituir um pedido feito pelos clientes do RestauranteHi, onde é necessário gravar desde o *login* do garçom até que tipo de produto.

Para finalizar as tabelas necessárias que constituem o cardápio do RestauranteHi, existe uma área somente para cadastros, onde tem–se o cadastro de usuários, os tipos de produtos e uma tabela de tipos que armazena dados para observação apenas, como quantidade de copos ou se terá gelo e/ou limão na hora de pedir determinado produto.

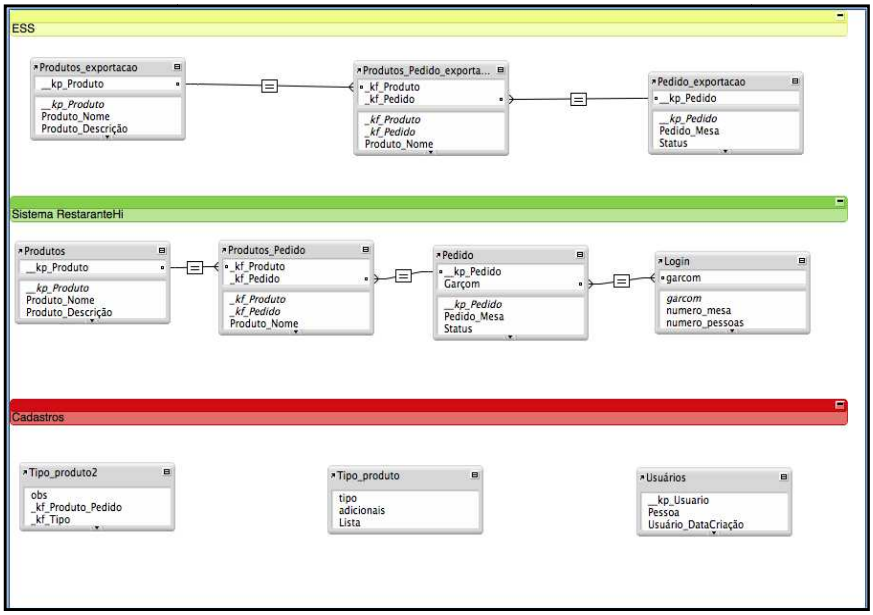


Figura 7 – Modelo de entidades e relacionamentos do sistema RestauranteHi

3.2.2 Modelo de Entidades e Relacionamentos do Protótipo

Para exemplificar o banco de dados do Protótipo, a Figura 8 retrata seu modelo de entidade e relacionamento, onde quatro módulo foram criados.

O módulo de Menu serve apenas para trazer uma tabela referenciada ao layout de menu, visto que FileMaker tem a necessidade de vincular sempre um layout a uma tabela. Assim um link com a tabela de preferências foi criado. O módulo chamado Protótipo é onde as tabelas de pedidos e metas se encaixam, onde os *scripts* realizam as pesquisas de metas e armazenam os dados importados. O módulo Importação é usado para intermediar a coleta de dados do sistema RestauranteHi. Assim os dados importados ficam armazenados nele até finalizar a importação, para depois serem transferidos para a tabela Pedidos. E por último o módulo de preferências, onde o caminho do servidor é criado para em qualquer parte do sistema ser usado para fazer uma nova consulta.

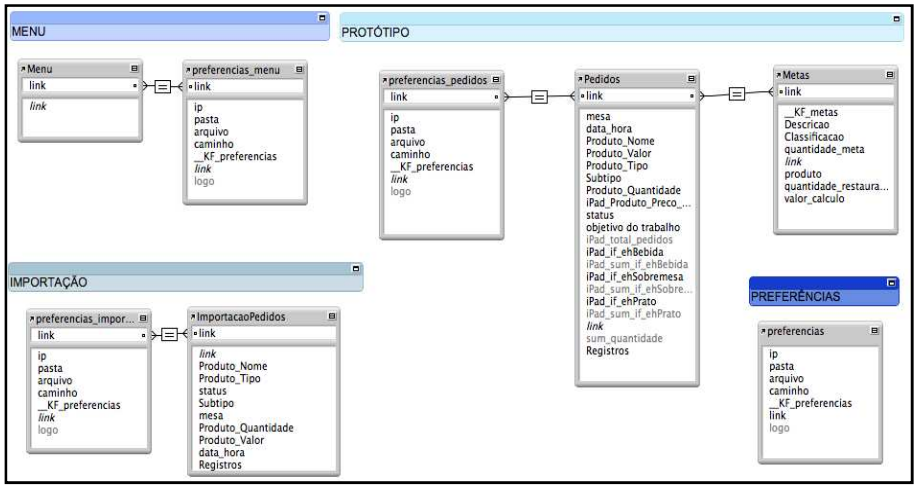


Figura 8 – Modelo de entidades e relacionamentos do protótipo desenvolvido

3.2.3 Diagrama de casos de uso

A seguir é apresentado o diagrama de casos de uso modelado na etapa de especificação do sistema.

O ator no sistema é o usuário no iPad. A Figura 9 mostra as principais funcionalidades pertinentes ao usuário do sistema.

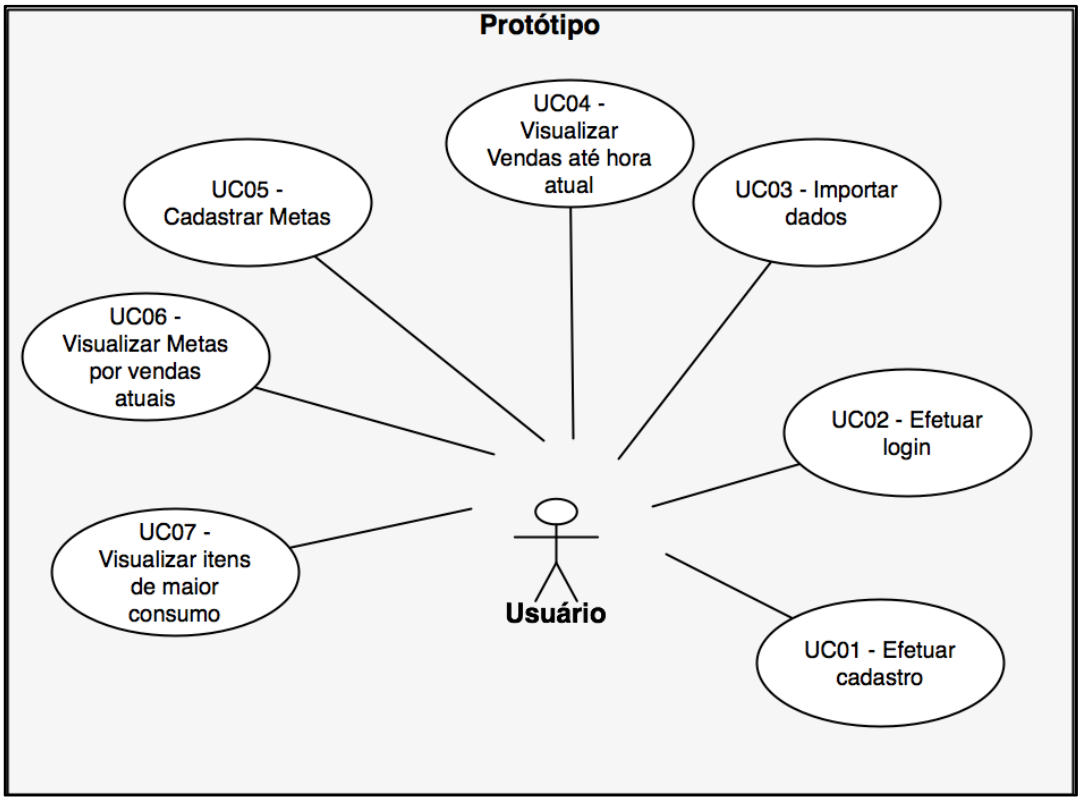


Figura 9 – Diagrama de casos de uso do usuário

Os casos de uso da Figura 9 são:

a) `UC01 – Efetuar cadastro`: requisita as informações em uma tela de preferências para efetuar cadastro;

b) `UC02 – Efetuar login`: permite acessar as funcionalidades internas e restritas do sistema, foram criados apenas usuários administradores para acessar qualquer tabela;

c) `UC03 – Importar dados`: faz uso da importação através de parâmetros para conectar-se ao servidor;

d) `UC04 – Visualizar vendas até hora atual`: faz-se uso de uma tabela intermediária para receber as informações, mas após importar, todos os dados são adicionados na tabela que mostrará as vendas;

e) `UC05 – Cadastrar Metas`: em forma de lista, escolhe-se as metas a atingir; cada meta nova é adicionada ao gráfico principal que compara com itens pedidos no restaurante;

f) `UC06 – Visualizar Metas por Vendas atuais`: na forma de gráfico em coluna são apresentados todas as vendas atuais por tipo e por meta adicionada;

g) `UC07 – Visualizar itens de maior consumo`: na forma de gráfico pizza são apresentados itens de maior consumo com a porcentagem sobre todos os pedidos feitos.

3.2.4 Diagrama de atividades

Após entrar no sistema, é possível ter acesso ao menu que ressalta os objetivos deste trabalho. No diagrama de atividade, representado na Figura 10, pode-se ter a visão ampla do processo de uso, desde a efetuação do acesso, passando pela validação e fazendo a tentativa de importação de acordo com as preferências.

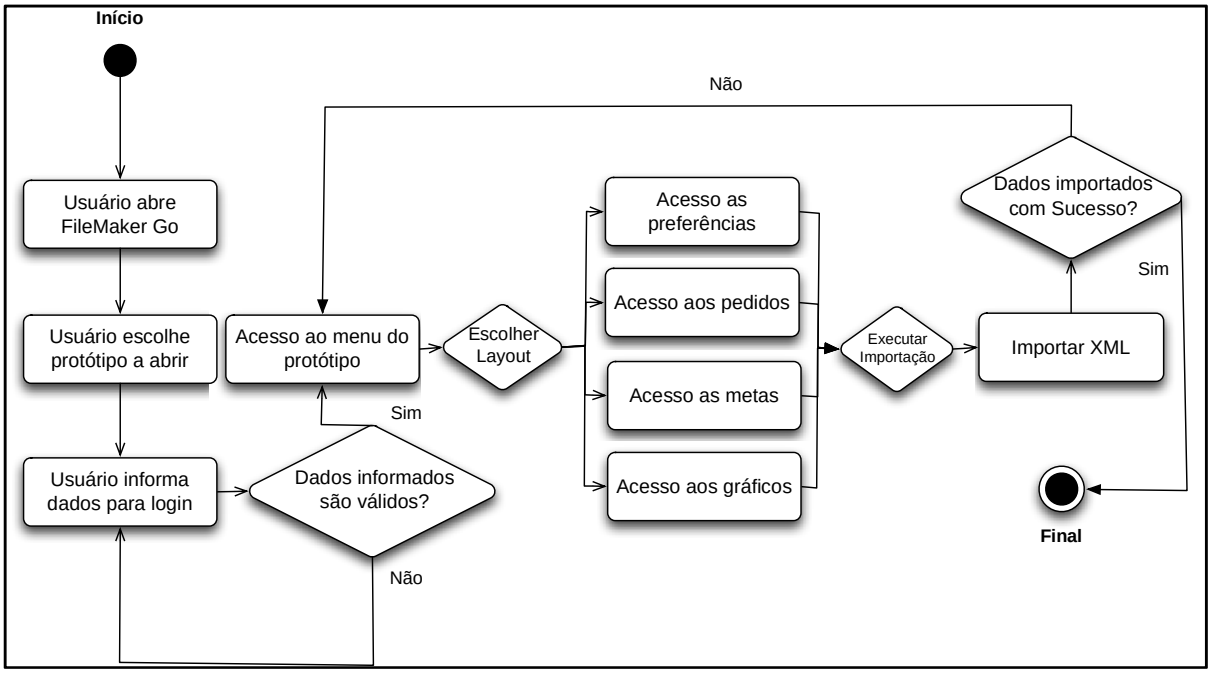


Figura 10 – Diagrama de atividades no uso do protótipo

3.2.5 Diagrama de classes: representação o conjunto de classes PHP

A Figura 11 apresenta as principais classes utilizadas da API disponibilizada pela FileMaker, objetivando representar a estrutura principal focada no planejamento da montagem do XML.

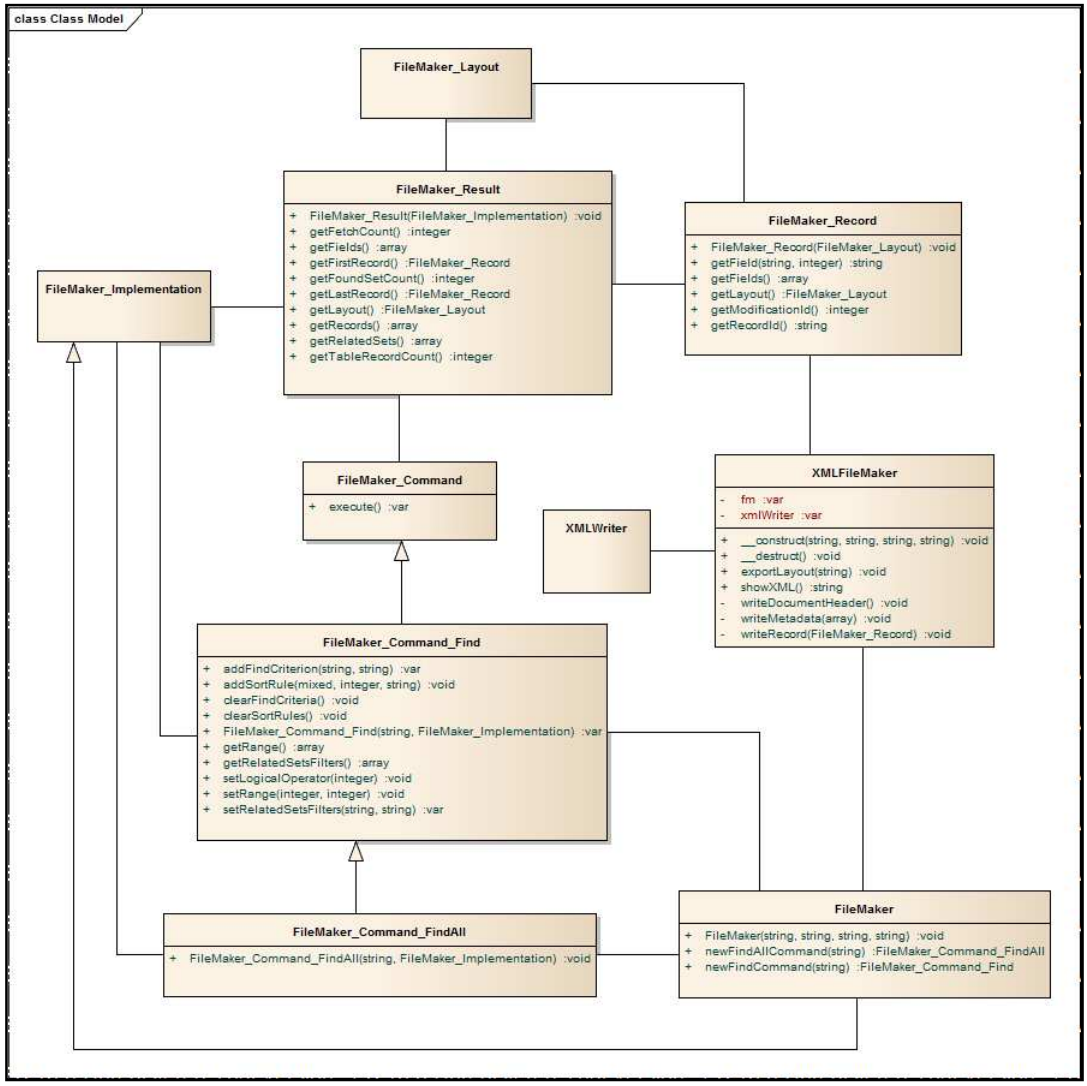


Figura 11 – Diagrama de classes

A classe `XMLFileMaker`, cujo funcionamento é detalhado nos parágrafos posteriores, utiliza várias classes da API do FileMaker escrita para a linguagem PHP. Dentre as classes que compõem essa API, algumas são utilizadas de forma explícita pela classe `XMLFileMaker`, ao passo que outras são utilizadas internamente pela própria API em auxílio à outras classes. Os parágrafos subsequentes descrevem brevemente as classes apresentadas na Figura 11 e que fazem parte da API PHP para FileMaker.

A classe `FileMaker`, que estende a classe `FileMaker_Implementation`, é responsável por estabelecer e persistir conexão com um banco de dados FileMaker, com base nos parâmetros de usuário, senha, endereço e nome do banco de dados. Também

fornece métodos para instanciar objetos que auxiliam na consulta, inclusão, atualização e exclusão de registros, entre outras tarefas.

A API fornece classes para consultas no banco de dados FileMaker. As classes de consulta extendem a classe `FileMaker_Command`. A classe `FileMaker_Command_Find` permite recuperar registros de um *layout* específico utilizando critérios de busca, ao passo que a classe `FileMaker_Command_FindAll` recupera todos os registros presentes em um *layout* FileMaker.

O resultado de uma consulta realizada por um objeto instanciado de uma classe que implementa a interface `FileMaker_Command` é um novo objeto da classe `FileMaker_Result`. Um objeto desta classe armazena todos os registros recuperados na consulta em uma estrutura do tipo `array`, sendo que cada elemento desse `array` é um objeto do tipo `FileMaker_Record`. A classe `FileMaker_Result` também fornece informação sobre o *layout* consultado por intermédio de um objeto do tipo `FileMaker_Layout`.

Cada registro recuperado do banco de dados FileMaker ficará armazenado em um objeto do tipo `FileMaker_Record`. Através desta classe é possível obter os valores dos campos presentes no registro utilizando o método `getField`. E para descobrir quais campos compõem o registro, utiliza-se o método `getFields`, que retorna uma estrutura do tipo `array`, onde cada elemento é o nome de um campo.

3.2.6 Diagrama de sequência: acesso restrito no sistema e importação XML

Para uso dos recursos desenvolvidos, tendo em vista que armazenamento na internet tem custos, foi utilizado um FileMaker Server e um servidor emprestado da empresa HiMaker

para armazenar ambos os sistemas. O protótipo é uma copia do RestauranteHi. Conforme o diagrama de sequência representado na Figura 12, ao efetuar a autenticação, o usuário precisa clicar em um botão que executa a requisição apenas das informações pertinentes à necessidade do protótipo, para posteriormente fazer cálculos e apresentar gráficos.

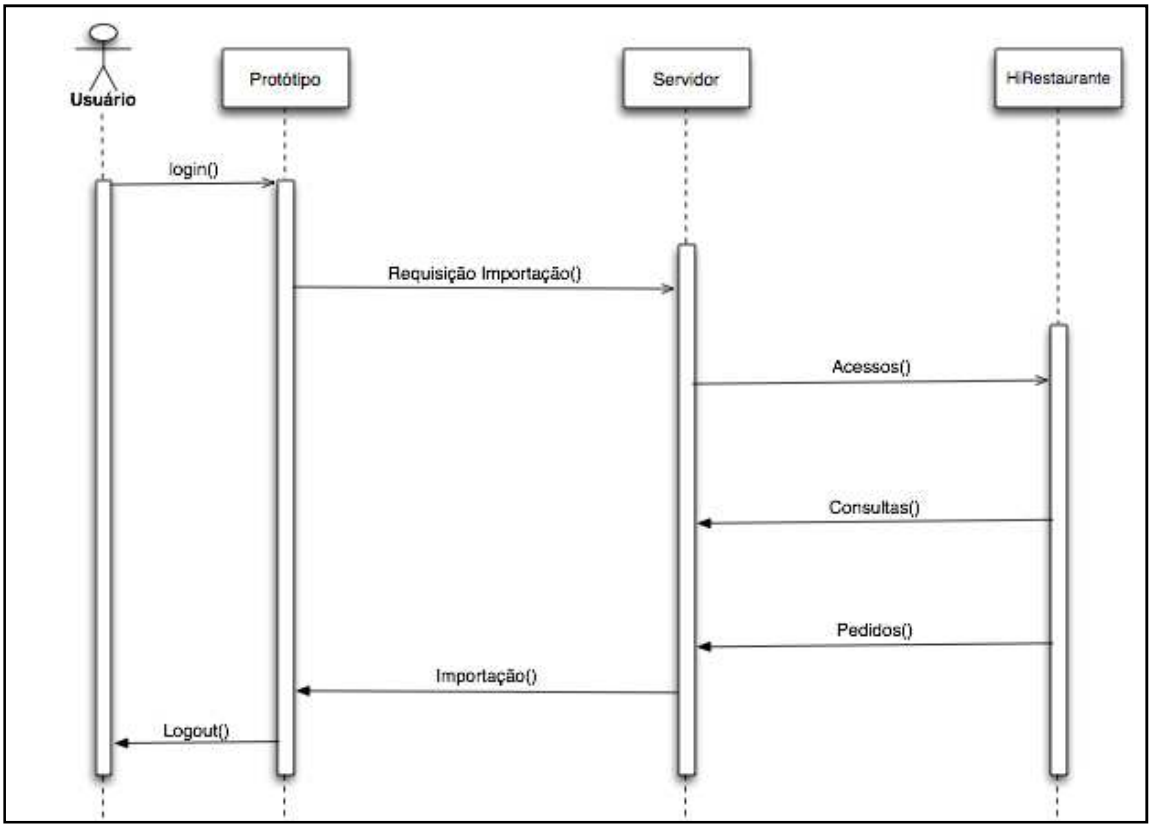


Figura 12 – Diagrama de sequência do uso do protótipo

3.2.7 Diagrama de implantação: etapas para acesso aos dados

Como representado na Figura 13, tem-se o protótipo instalado em um iPad, o qual acessa uma base em um servidor remoto para coletar informações.

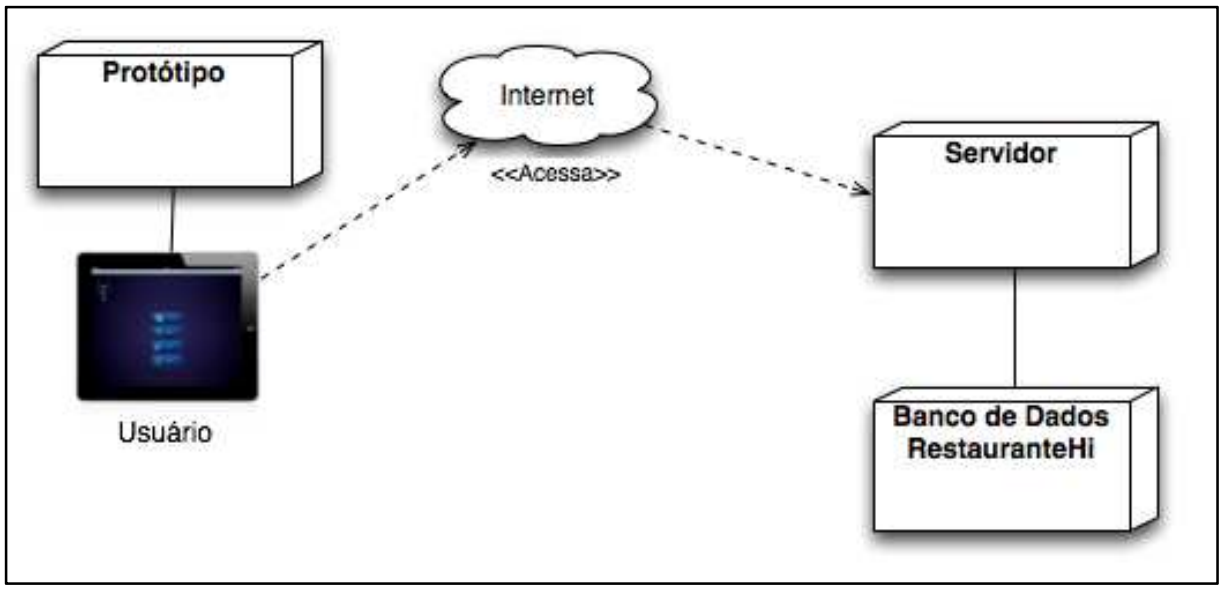


Figura 13 – Diagrama de implantação do protótipo

3.3 IMPLEMENTAÇÃO

A seguir são apresentadas e detalhadas as ferramentas e técnicas utilizadas para chegar ao resultado final do desenvolvimento do protótipo. Posteriormente a isto, é apresentada a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

Para o desenvolvimento do protótipo foi utilizado o ambiente de desenvolvimento chamado FileMaker Pro Advanced, o qual possibilita fazer *debug* dos passos desenvolvidos em cada *script* do protótipo. Para programações em php foi utilizado a ferramenta *Text Edit* que é nativa no próprio sistema Mac Os 10.8. Para a parte de testes do XML foi usado o XMLMind. A seguir são apresentados os detalhes destas tecnologias.

3.3.1.1 PHP

A linguagem PHP foi utilizada para as funcionalidades de requisição de XML. Para rodar a linguagem, é necessário um interpretador PHP e um servidor web. No protótipo é utilizado um servidor APACHE versão 2.2.22 (Unix) rodando o interpretador PHP versão 5.4.4. A linguagem também pode ser utilizada em linha de comando como, por exemplo, em pequenos testes durante o desenvolvimento, pode até mesmo ser utilizada como linguagem *desktop* com o PHP-GTK (PHP, 2012).

No Quadro 4 é apresentado o código principal de consulta, através do qual o protótipo realiza a busca das informações remotamente armazenadas. Já no Quadro 5 é apresentada uma classe que traz os dados de uma forma legível para o FileMaker.

```
1. <?php
2. // require_once garante que o arquivo será incluido
3. // uma única vez. Se o arquivo nao foi encontrado da erro
4. require_once('FileMaker.php');
5. require_once('xml_tcc.php');
6. // Verifica se o usuario solicitou algum tipo de filtro
7. $filtros = $_GET;
8. // Estabelece conexao com o FileMaker
9. $fm = new FileMaker('restaurantHi.fmp12', '127.0.0.1',
10.'usuariotcc', 'senhatcc');
11.if (! empty($filtros))
12.{
13.// Aplica as consultas
14.$fmFindCmd = $fm->newFindCommand('layout_exportacao');
15.foreach ($filtros as $campo => $valor)
16.{
17.    if ($campo == 'data_hora')
18.    {
19.            $data = date('d-m-Y', $valor);
20.            $fmFindCmd->addFindCriterion($campo, $valor);
21.    }
22.    else
23.    {
24.         $fmFindCmd->addFindCriterion($campo, $valor);
25.    }
26.}
27.$fmResult = $fmFindCmd->execute();
28.}
29.else
30.{
31.// Busca todos os registros no layout
32.$fmFindAllCmd = $fm>newFindAllCommand(
33.'layout_exportacao');
34.// Retorna todos os registros do layout
35.$fmResult = $fmFindAllCmd->execute();
36.}
37.// Pega o array com todos os registros
38.$records = $fmResult->getRecords();
39.// Cria objeto do layout para pegar
40.//as informaÁıes dos campos
41.$fmLayout = $fmResult->getLayout();
42.$fields = $fmLayout->getFields();
43.// Criar objeto da classe XMLFileMaker
44.$xmlFm = new XMLFileMaker('restaurantHi.fmp12');
45.// Escreve os campos (METADATA) no XML
46.$xmlFm->writeMetadata($fields);
47.// Escreve os registros
48.$xmlFm->writeRecords($records,
49.$fmResult->getFoundSetCount());
50.$xmlFm->showXML();
51.?>
```

Quadro 4 – Construção do arquivo que o protótipo irá chamar para importar

O código PHP responsável pela consulta dos dados no servidor inicia tentando estabelecer conexão com o banco de dados FileMaker, denominado `restaurantHi.fmp12`. A conexão e estabelecida por ocasião da instanciação da classe `FileMaker` na linha 4. O construtor dessa classe recebe como parâmetros o nome do banco de dados, o endereço IP de localização do servidor, o nome do usuário com acesso ao servidor e a senha desse usuário.

Uma vez estabelecida a conexão, o código avalia os filtros de consulta requisitados pelo usuário através da URL. Onde as informações persistidas na URL são recuperadas pelo

PHP através da variável `$_GET` como pode ser visto na linha 7 do quadro 4. Para aplicação dos filtros solicitados, utiliza-se a classe `FileMaker_Command_Find`, conforme linha 14, instanciada por intermédio do método `newFindCommand` na mesma linha, da classe `FileMaker`. Ao método `newFindCommand` é enviado como parâmetro o nome do *layout* no banco de dados FileMaker ao qual deseja-se iniciar a consulta. Cada um dos filtros é aplicado à consulta usando o método `addFindCriterion`, que pode ser vista na linha 20, da classe `FileMaker_Command_Find`. Este método recebe dois parâmetros, sendo o primeiro o nome do campo no *layout* e o segundo o valor que deseja filtrar.

Caso nenhum filtro tenha sido requisitado através da URL, o código realiza a busca de todos os registros no layout. Tal busca é realizada utilizando o método `newFindAllCommand`, linha 32, implementado na classe `FileMaker` e que recebe como parâmetro o nome do *layout* que deseja-se consultar. Este método retorna uma instância da classe `FileMaker_Command_FindAll`.

Ambas as classes de consulta, `FileMaker_Command_Find` e `FileMaker_Command_FindAll`, são descendentes da classe `FileMaker_Command` e, portanto, implementam o método `execute`. Este método realiza a consulta propriamente dita e retorna um objeto do tipo `FileMaker_Result`. Este objeto, por sua vez, contém as informações recuperadas na consulta.

Um objeto do tipo `FileMaker_Result` contém uma coleção de todos os registros resgatados ao layout por ocasião da consulta ao banco de dados, além de outras informações pertinentes ao layout consultado, como a quantidade de registros encontrados, os campos disponíveis.

O código da consulta, utilizado no protótipo produto deste trabalho, executa uma chamada ao método `getRecords` da classe `FileMaker_Result`. Este método retorna uma coleção de objetos do tipo `FileMaker_Record`, armazenados em uma estrutura PHP do tipo `array`. As informações recuperadas são escritas em um documento XML, cuja escrita é responsabilidade da classe `XMLFileMaker`, cuja codificação pode ser visualizada no Quadro 5.

```
1. <<?php
2. require_once('FileMaker.php');
3. /**
4. Classe para exportar os registros para um formato legível
5. ao FileMaker
6. */
7. class XMLFileMaker
8. {
9. private $xmlWriter;
10./**
11.Construtor da classe
12.*/
13.public function __construct($database, $errorCode = 0)
14.{
15.$this->xmlWriter = new XmlWriter();
16.// Alocando memoria para saÌda do XML
17.$this->xmlWriter->openMemory();
18.// Escreve o cabecalho do XML
19.// <?xml version="1.0" encoding="UTF-8">
20.$this->xmlWriter->startDocument('1.0', 'UTF-8');
21.// Escreve a tag principal (raiz) do XML
22.$this->xmlWriter->startElement('FMPXMLRESULT');
23.// Escreve o atributo xmlns para a tag FMPXMLRESULT
24.<FMPXMLRESULT xmlns=
25."http://www.filemaker.com/fmpxmlresult">
26.$this->xmlWriter->writeAttribute('xmlns',
27.'http://www.filemaker.com/fmpxmlresult');
28.$this->xmlWriter->writeElement('ERRORCODE', $errorCode);
29.// Cria a tag PRODUCT e seus atributos
30.$this->xmlWriter->startElement('PRODUCT');
31.$this->xmlWriter->writeAttribute('VERSION',
32.'ProAdvanced 12.0v1');
33.$this->xmlWriter->writeAttribute('NAME', 'FileMaker');
34.$this->xmlWriter->writeAttribute('BUILD', '03-15-2012');
35.$this->xmlWriter->endElement();
36.// Tags criadas com writeElement
37.//sao fechadas automaticamente
38.$this->xmlWriter->startElement('DATABASE');
39.$this->xmlWriter->writeAttribute('NAME', $database);
40.$this->xmlWriter->writeAttribute
41.('TIMEFORMAT', 'k:mm:ss');
42.// $this->xmlWriter->writeAttribute('RECORDS', 0);
43.$this->xmlWriter->
44.writeAttribute('DATEFORMAT', 'D/m/yyyy');
45.$this->xmlWriter->endElement();
46.}
47.public function writeMetadata($fields)
48.{
49.// Criar a tag METADATA para escrever as informações
50.// dos campos do layout
51.$this->xmlWriter->startElement('METADATA');
52.foreach ($fields as $f)
53.{
54.// Escreve a tag FIELD e seus atributos
55.   $this->xmlWriter->startElement('FIELD');
56.$this->xmlWriter->writeAttribute('NAME', $f->getName());
57.$this->xmlWriter->writeAttribute('TYPE', $f->getType());
58.$this->xmlWriter->writeAttribute('MAXREPEAT',
59.$f->getRepetitionCount());
60.$this->xmlWriter->writeAttribute('EMPTYOK', 'YES');
61.$this->xmlWriter->endElement();
62.}
63.// Fecha o METADATA
64.$this->xmlWriter->endElement();
65.}
66.public function writeRecord
67.     ($data, $recordID, $modID = null)
68.{
69.// Escreve a linha do registro
```

```
70.$this->xmlWriter->startElement('ROW');
71.// Define o atributo RECORDID
72.$this->xmlWriter->startAttribute('RECORDID');
73.$this->xmlWriter->text($recordID);
74.$this->xmlWriter->endAttribute();
75.// Define o atributo MODID
76.$this->xmlWriter->startAttribute('MODID');
77.$this->xmlWriter->text($modID);
78.$this->xmlWriter->endAttribute();
79.// Percorre o array com os dados do registro
80.foreach ($data as $campo => $valor)
81.{
82.$this->xmlWriter->writeElement($campo, $valor);
83.}
84.// Fecha a tag do registro (ROW)
85.$this->xmlWriter->endElement();
86.}
87.public function writeRecords($records, $found)
88.{
89.// Cria a tag principal RESULTSET
90.$this->xmlWriter->startElement('RESULSET');
91.// Numero de registros
92.$this->xmlWriter->writeAttribute('FOUND', $found);
93.// Cada linha do array sera um registro
94.foreach ($records as $r)
95.{
96.// Pega os nomes dos campos do registros
97.$campos = $r->getFields();
98.$this->xmlWriter->startElement('ROW');
99.$this->xmlWriter->writeAttribute(
100.     'RECORDID', $r->getRecordId());
101.     $this->xmlWriter->writeAttribute(
102.     'MODID', $r->getModificationId());
103.     // Percorre os campos para pegar os valores
104.     foreach ($campos as $c)
105.     {
106.     $this->xmlWriter->startElement('COL');
107.     $this->xmlWriter->writeElement(
108.     'DATA', $r->getField($c));
109.     $this->xmlWriter->endElement();
110.     }
111.     // Fecha o registro (ROW)
112.     $this->xmlWriter->endElement();
113.     }}
114.     public function showXML()
115.     {
116.     // Nao usar saidas para o navegador
117.     //antes de trocar o cabeçalho
118.     // Exemplos: echo, print, print_r, var_dump
119.     header('Content-type:application/xml; charset=UTF-8');
120.     // Finaliza a tag RAIZ
121.     $this->xmlWriter->endElement();
122.     // Finaliza o documento
123.     $this->xmlWriter->endDocument();
124.     echo $this->xmlWriter->outputMemory(true);
125.     }}?>
```

Quadro 5 – Classes em PHP para retornarem o que FileMaker aceita como XML

A classe `XMLFileMaker`, além do seu construtor, é composta por outros quatro métodos. Cada um desses métodos possui responsabilidades distintas que trabalham em sinergia para construção do documento XML.

O método construtor da classe `XMLFileMaker`, linha 7, cria um objeto do tipo `XmlWriter`, linha 9. Este objeto está disponível junto à biblioteca padrão da linguagem PHP e

fornece um conjunto de funcionalidades para criar documentos XML. Depois de instanciar a classe `XmlWriter`, o construtor inicia a escrita do documento XML criando as *tags* com informações globais, sendo elas a identificação do produto FileMaker e o nome do banco de dados. Tais informações são armazenadas nas *tags* `PRODUCT` e `DATABASE`, linhas 30 e 38, respectivamente.

O código PHP responsável pela consulta, apresentado no Quadro 5, após realizar a instanciação da classe `XMLFileMaker`, executa-se uma chamada ao método `writeMetadata`, linha 47, desta classe. Este método escreve todas as informações pertinentes à estrutura do layout na *tag* `METADATA`, linha 51. Cada campo presente no layout tem suas informações escritas em uma *tag* `FIELD` dentro da *tag* `METADATA`, linha 47. A *tag* `FIELD` contém quatro atributos denominados `NAME`, `TYPE`, `MAXREPEAT` e `EMPTYOK` , linhas 56 a 58, que armazenam o nome do campo, o nome do tipo de dados por ele armazenado, quantidade máxima de repetições e se o campo aceita valores nulos.

Uma vez escritas as informações pertinentes à estrutura do layout, o código responsável pela consulta executa uma chamada ao método `writeRecords`, linha 66, da classe `XMLFileMaker`. Este método recebe dois parâmetros: a coleção de registros encontrados na consulta e a quantidade de registros encontrados. A coleção de registros é um vetor de objetos do tipo `FileMaker_Record`, linha 72. Cada objeto do tipo `FileMaker_Record` contém todas as informações pertinentes a um registro recuperado na consulta. Os registros são escritos dentro de uma `tag` denomindada `RESULTSET`, linha 90. Esta `tag` contém um atributo chamado `FOUND` , linha 92, que armazena o número total de registros encontrados na consulta. Os registros são lidos através da instrução PHP `foreach`, linha 94, utilizada para percorrer os elementos de uma estrutura do tipo *array*. Cada registro encontrado implica escrever uma tag `ROW`, linha 98. Cada `tag ROW` contém dois atributos denominados `RECORDID` e `MODID`, linhas 100 e 102, que armazenam o identificador do registro e o identificador de modificação do registro, respectivamente. O identificador do registro é recuperado através do método `getRecordId`, linha 100, e o identificador de modificação do registro é recuperado pela chamada ao método `getModificationId`, linha 102. Ambos os métodos são implementados na classe `FileMaker_Record`. Cada campo e valor do registro é escrito dentro da `tag ROW`.

3.3.1.2 Fazendo a importação através do XML

A importação é feita através da XML conforme documentação que a FileMaker disponibiliza para uma importação e exportação de dados, com o pequeno detalhe que o padrão é criado pelas classes em PHP e não pelo FileMaker, pois assim o XML abre portas para trocar o padrão a qualquer momento. O Quadro 6 demostra o XML gerado pela classe em PHP.

```
<FMPXMLRESULT xmlns="http://www.filemaker.com/fmpxmlresult">
<ERRORCODE>0</ERRORCODE>
<PRODUCT VERSION="12.0v1" NAME="FileMaker" BUILD="03-15-
2012"/>
<DATABASE NAME="restaurantHi.fmp12" TIMEFORMAT="k:mm:ss"
DATEFORMAT="D/m/yyyy"/>
<METADATA>
<FIELD NAME="Pedido_exportacao::Pedido_Mesa" TYPE="normal"
MAXREPEAT="1" EMPTYOK="YES"/>
<FIELD NAME="Pedido_exportacao::Status" TYPE="normal"
MAXREPEAT="1" EMPTYOK="YES"/>
<FIELD NAME="Pedido_exportacao::data_hora" TYPE="normal"
MAXREPEAT="1" EMPTYOK="YES"/>
<FIELD NAME="Produtos_exportacao::Produto_Nome" TYPE="normal"
MAXREPEAT="1" EMPTYOK="YES"/>
<FIELD NAME="Produtos_exportacao::Produto_Valor"
TYPE="normal" MAXREPEAT="1" EMPTYOK="YES"/>
<FIELD NAME="Produtos_exportacao::Produto_Tipo" TYPE="normal"
MAXREPEAT="1" EMPTYOK="YES"/>
<FIELD NAME="Produtos_exportacao::Subtipo" TYPE="normal"
MAXREPEAT="1" EMPTYOK="YES"/>
<FIELD NAME="Produto_Quantidade" TYPE="normal" MAXREPEAT="1"
EMPTYOK="YES"/>
</METADATA>
<RESULSET FOUND="8">
<ROW RECORDID="10046" MODID="4">
<COL>
<DATA>03</DATA>
</COL>
<COL>
<DATA>Aberto</DATA>
</COL>
<COL>
<DATA>09/12/2012 20:48:56</DATA>
</COL>
…
…
…
<COL>
<DATA>1</DATA>
</COL>
</ROW>
</RESULSET>
</FMPXMLRESULT>
```

Quadro 6 – XML gerado pelas classes em PHP, buscando dados do FileMaker

3.3.1.3 Importando e utilizando o protótipo em FileMaker

Após disponibilizado o padrão XML que o FileMaker aceita, utiliza 3 *scripts* para o protótipo funcionar.

A Figura 14 apresenta o código identado dos botões que estão tanto no menu principal quanto em cada seção/tela do protótipo. Assim, cada botão faz a chamada do *Script* "Ir para Layout" enviando um parâmetro como referência para qual layout o botão deve acionar. Como exemplo, no primeiro `IF`, a função `Get (ScriptParameter)` retorna o parâmetro enviado pelo botão e compara com a String `preferencias`. Caso for isto verdadeiro, o passo seguinte é a função `Go to Layout`, que faz com que a tela vá para o layout especificado na função, com dados vindos da tabela `preferencias`.

Entretanto, se o parâmetro enviado não foi de `preferencia`, cairá fora do primeiro IF e entrará no ELSE, que faz um teste semelhante, perguntando se o parâmetro é "pedidos", para entrar na tela de pedidos. Os próximos testes são referentes a menu, gráficos e metas, levando cada um ao seu devido *layout*.

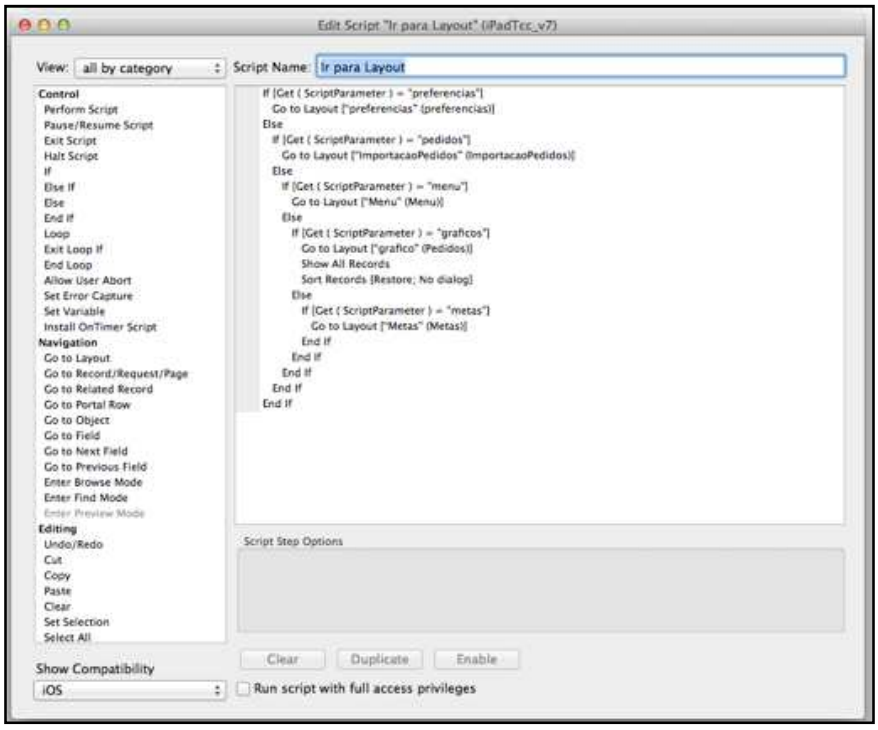


Figura 14 – *Scripts* para ir e vir entre *layouts*

O tratamento aplicado ao *script* do Quadro 7 é de importação do XML que está no servidor, o qual de acordo com as preferências escolhidas no layout de preferências, pode

realizar importação de outro caminho.

Depurando o *script* e explicando-o pode-se perceber que o primeiro comentário é sobre a importação do servidor. Desta maneira executa-se a função que leva o *script* para o layout de `ImportacaoPedidos` que esta associado a tabela com o mesmo nome. Depois é executado a função `Show All Records` para mostrar todos os registros. Em seguida a função `Delete All Records` é acionada, para deletar todos os registros encontrados sem questionar o usuário, por isso a opção de `No dialog` é selecionada. Da forma como o *script* segue, nesta etapa já não haver algum registro antigo na memória, deixando o caminho livre para, em tempo real, obter novas informações do servidor.

O próximo passo é executar a mais importante função deste protótipo, `Import Records,` que recebe o parâmetro `No dialog` e o `preferencias::caminho`, roteiro que importará os dados de acordo com o caminho armazenado na tabela de `preferencias`, do campo `caminho`.

A função `Import` quando executada, só funcionará perfeitamente se o XML estiver de acordo com as regras que a própria FileMaker criou, pois é um padrão adotado que se baseia em arquivos FileMaker de outras versões, sendo há anos o mesmo padrão.

A tabela `ImportacaoPedidos` serve como uma tabela intermediária caso o *script* faça a importação corretamente. Em seguida o *script* faz uma cópia para tabela `pedidos`. Desta maneira, como comentado, são deletados todos os registros da tabela de pedidos para adicionar os dados atualizados que estão armazenados na tabela `ImportacaoPedidos,` já dentro do protótipo.

Executa-se a função que leva o *script* para o *layout* de `Pedidos`, `Go to LayoutPedidos`. É executada a função que mostra todos os registros, `Show All Records` e depois todos os registros desta tabela são deletados com a função `Delete All Records,` também com o parâmetro `No dialog,` para não aparecer pergunta nenhuma na tela para o usuário.

Com a tabela de pedidos vazia, é executado um `Go to layout ImportacaoPedidos` para ir até o layout que contém os dados importados. Depois executa-se a função que leva ao primeiro registro, `Go to Record/Request/Page[First]`. Então executa-se um *loop* que fará a cópia de campo por campo da tabela `ImportacaoPedidos` para `Pedidos`.

Abre-se o *loop* com a função `loop`. Depois é criado neste *loop* uma variável para cada campo da tabela, com o nome do produto em `$Produto_Nome`, com a função `Set Variable` e atribuindo o valor da tabela `ImportacaoPedidos` e do campo respectivo, no caso

`Produto_Nome`, onde `$Produto_Nome` é uma variável do tipo local que está armazenando temporariamente o valor recebido da tabela `ImportacaoPedidos` do campo `Produto_Nome`. Assim continuamente são criadas variáveis para cada campo.

Ao finalizar a criação das variáveis, executa-se a função que direciona o fluxo do *script* para outro layout usando a função `Go to Layout` e passando como parâmetro `Pedidos`, o qual deverá receber os itens da tabela `importacaoPedidos`.

É enviado o comando de `New Record/Request` para criar um novo registro nesta tabela, a qual tem as mesmas características da tabela de `ImpotacaoPedidos`. Desta maneira é possível fazer o processo inverso, definindo a cada campo o valor armazenado na variável criada antes da entrada neste layout.

A função `Set Field` faz que o campo `Produto_Nome` da tabela `Pedidos` receba o valor armazenado na variável local `$Produto_Nome`, lembrando que variáveis locais somente são finalizadas pelo coletor de lixo do FileMaker ao final da execução de um *script*.

É dado continuidade a mais funções para adicionar os valores armazenados em variáveis para dentro dos campos do registro novo da tabela Pedidos. Ao final, um comando de `Go to layout ImportacaoPedidos` é acionado para que o *script* volte a etapa de *loop* e possa ir para o próximo registro. Para fazer isso é executado a função `Go to Record/Resquest/Page` com o parâmetro `Next`, que vai pro próximo registro e `Exit after last`, que permite que se for o último registro que estiver sendo exibido, o *loop* possa ser encerrado.

E ao final do *script*, o comando de `Go to Layout[Original layout]` é executado para fazer a tela do sistema retornar à tela de partida deste *script*, pois foram inserido botões em todas as partes do protótipo que podem executar o *script* de importação, porém, todos devem retornar à mesma tela após executados.

```
1. #Faz a importação do Server
2. Go to Layout [ “ImportacaoPedidos” (ImportacaoPedidos) ]
3. Show All Records
4. Delete All Records [ No dialog ]
5. Import Records [ XML (from calculation): preferencias::caminho ] [
   No dialog ]
6. Show All Records
7. #Deleta tudo da tabela de pedidos atual
8. Go to Layout [ “Pedidos” (Pedidos) ]
9. Show All Records
10.Delete All Records [ No dialog ]
11.#
12.#Faz a cópia completa pra tabela certa
13.#
14.Go to Layout [ “ImportacaoPedidos” (ImportacaoPedidos) ]
15.Go to Record/Request/Page [ First ]
16.Loop
```

```
17.Set Variable [ $Produto_Nome; Value:ImportacaoPedidos::Produto_Nome
   ]
18.Set Variable [ $Produto_Tipo; Value:ImportacaoPedidos::Produto_Tipo
   ]
19.Set Variable [ $status; Value:ImportacaoPedidos::status ]
20.Set Variable [ $Subtipo; Value:ImportacaoPedidos::Subtipo ]
21.Set Variable [ $mesa; Value:ImportacaoPedidos::mesa ]
22.Set Variable [ $Produto_Quantidade;
23.                  Value:ImportacaoPedidos::Produto_Quantidade ]
24.Set Variable [ $Produto_Valor;
   Value:ImportacaoPedidos::Produto_Valor ]
25.Set Variable [ $data_hora; Value:ImportacaoPedidos::data_hora ]
26.Go to Layout [ “Pedidos” (Pedidos) ]
27.New Record/Request
28.Set Field [ Pedidos::Produto_Nome; $Produto_Nome ]
29.Set Field [ Pedidos::Produto_Tipo; $Produto_Tipo ]
30.Set Field [ Pedidos::status; $status ]
31.Set Field [ Pedidos::Subtipo; $Subtipo ]
32.Set Field [ Pedidos::mesa; $mesa ]
33.Set Field [ Pedidos::Produto_Quantidade; $Produto_Quantidade ]
34.Set Field [ Pedidos::Produto_Valor; $Produto_Valor ]
35.Set Field [ Pedidos::data_hora; $data_hora ]
36.Go to Layout [ “ImportacaoPedidos” (ImportacaoPedidos) ]
37.Go to Record/Request/Page [ Next; Exit after last ]
38.End Loop
39.Go to Layout [ original layout ]
```

Quadro 7 – *Script* para importação do XML

Um dos objetivos do protótipo é estimar se as metas foram alcançadas. Assim o usuário poderia adicionar as Metas que deseja alcançar e clicar no botão “Fazer Comparativos“, para executar o *script* que fará a comparação, representado no Quadro 8.

Este *script* começa com a função `Set Error Capture` e o parâmetro `On`, que serve para capturar erros ocorridos durante a execução de todo o *script*, não mostrando-os para o usuário.

A função `Go to Layout Metas`, linha 3, envia o *script* para o layout de metas, apenas para ter certeza que toda continuação do *script* passe dentro desta tabela, caso o usuário execute a partir de outro local.

O comando `Go to Recod/Request/Page[First],` linha 4, faz a tabela do banco de dados trazer o primeiro registro à tona. Logo depois é iniciado um *loop* que fará a execução da pesquisa de cada meta ser apresentada em tela. Para a meta de número 1, é armazenada uma variável do produto a que ele se refere. Outra variável armazenada é a de classificação, para saber do que se trata a meta, se é por ano, por mês ou dia. Uma última variável armazena o valor da classificação.

Na linha 9, é executado o comando para levar o *script* a tabela de Pedidos para que sejam efetuadas as comparações, com o `Go to Layout Pedidos`. Depois é executada a função que deixa a tabela em modo de busca, esperando ter itens vinculado aos campos para efetuar uma busca. O comando `Enter Find Mode[]` faz isso na linha 10.

Como dentro da tabela de pedidos são armazenadas a data e a hora que o pedido foi efetuado no restaurante, a continuação do *script* leva a um teste para saber se a classificação é igual a `Ano`. Se for, a parte verdadeira do `IF` é acionada e o campo `data_hora` recebe apenas o valor da variável `$valor_classificacao`, linha 15. Caso não for, verifica-se se é igual a `Mês`, linha 17. Como o padrão de busca de data e hora necessitam de uma data, internamente na busca é acrescentado um campo de conversão chamado `Date` que necessita de três parâmetros para executar, os quais são mês, dia e ano. Coloca-se o símbolo `*` nos parâmetros que não são usados. Para finalizar, caso também não for uma meta de `Mês`, entra-se no teste pra saber se é `Dia` assim o valor adicionado no campo de busca `data_hora` deve ser `Date ( $valor_classificacao ; "*" ; Year(Get ( CurrentDate )))` , linha 21, onde a função `Year` toma apenas o ano de uma data, que no caso é de outra função chamada `Get (CurrentDate)` a qual retorna a data corrente do sistema operacional, possibilitando armazenar o seu valor.

Antes de finalizar os itens a serem inseridos para a busca, a variável que estava na memória vindo da meta atual chamada `$produto` é adicionado com a função `Set Field[Pedidos::Produto_Nome; $produto]` no campo de nome do produto. Nesta etapa do *script* é chamada a função `Perform Find[],` que executa a busca, conforme linha 27.

Após executada a busca, uma variável é inicializada com a função `Set Variable[$total; Value:Get(FoundCount)]`, linha 28, a qual armazena o valor da quantidade encontrada de registros com as opções procuradas. Neste momento o *script* volta ao *layout* de Metas para adicionar no campo `quantidade_restaurante`, linha 30, do registro atual o total de itens encontrados no período estabelecido pela meta.

Para que o *script* volte a etapa de loop e possa ir para o próximo registro, é acionado a uma função, linha 31, `Go to Record/Resquest/Page [Next; Exit after last]`, que permite encerrar o loop caso seja o último registro que estiver sendo exibido.

```
1. #Fazer pesquisa para cada meta estabelecida
2. Set Error Capture [ On ]
3. Go to Layout [ “Metas” (Metas) ]
4. Go to Record/Request/Page [ First ]
5. Loop
6. Set Variable [ $produto; Value:Metas::produto ]
7. Set Variable [ $classificacao; Value:Metas::Classificacao ]
8. Set Variable [ $valor_classificacao; Value:Metas::valor_calculo ]
9. Go to Layout [ “Pedidos” (Pedidos) ]
10.Enter Find Mode [ ]
11.#
12.#Dependendo da Classificação, seta o campo certo
13.#
14.If [ $classificacao = "Ano" ]
15.Set Field [ Pedidos::data_hora; $valor_classificacao ]
```

```
16.Else
17.If [ $classificacao = "Mês" ]
18.Set Field [ Pedidos::data_hora; Date ( "*" ; "*" ;
   $valor_classificacao ) ]
19.Else
20.If [ $classificacao = "Dia" ]
21.Set Field [ Pedidos::data_hora; Date
22.                ( $valor_classificacao ; "*" ; Year(Get (
   CurrentDate ))) ]
23.End If
24.End If
25.End If
26.Set Field [ Pedidos::Produto_Nome; $produto ]
27.Perform Find [ ]
28.Set Variable [ $total; Value:Get ( FoundCount ) ]
29.Go to Layout [ “Metas” (Metas) ]
30.Set Field [ Metas::quantidade_restaurante; $total ]
31.Go to Record/Request/Page [ Next; Exit after last ]
32.End Loop
```

Quadro 8 – *Script* para fazer comparativo de metas

3.3.2 Operacionalidade da implementação

O protótipo tem como finalidade disponibilizar ao usuário de iPad com FileMaker instalado e o protótipo carregado, a visualização dos pedidos feitos no restaurante que está com o servidor online. Neste capítulo é possível verificar as funcionalidades exemplificadas.

Houve um foco em desenhar uma interface simples e de fácil acesso. Desta maneira foi criado um menu com botões para acessar as áreas desenvolvidas. Ao se fazer necessário mais opções internas de cada área, são disponibilizados botões para navegação e para executar a importação de novos dados.

Nesta sessão são apresentadas as telas em formato tutorial, para entender todo funcionamento do protótipo. Será utilizado como exemplo um restaurante fictício, com dados previamente cadastrados no sistema.

3.3.2.1 Cadastrando novo usuário

Para adicionar um novo usuário, é necessário a utilização do FileMaker Pro, o qual permite adicionar usuários através do gerenciador de usuário, conforme Figura 15.

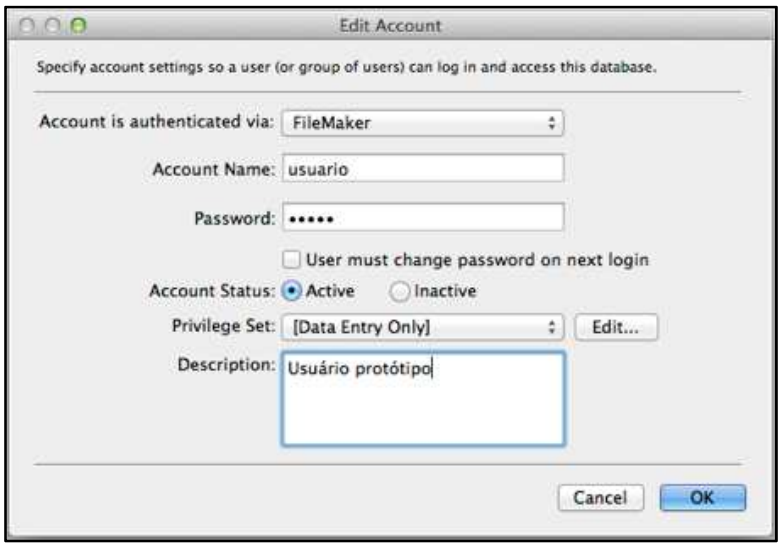


Figura 15 – Cadastro de usuário

3.3.2.2 Entrando no protótipo desenvolvido

Com FileMaker Go baixado do appStore (sistema de distribuição de aplicativos oficial da Apple), utiliza-se o aplicativo iTunes, que roda tanto em Windows quanto em Macs para instalar o protótipo dentro do iPad. Após transferido ao dispositivo, clica-se sobre o protótipo que está na memória do iPad. O nome do protótipo na lista é iPadTcc, Figura 16.

Figura 16 – Tela de *login* do protótipo

3.3.2.3 Acessando o sistema

É solicitado *login* e senha para acesso, conforme Figura 17.

Figura 17 – Tela de *login* do protótipo

3.3.2.4 Tela inicial no protótipo

A primeira tela, conforme a Figura 18 é o menu principal, que contém botões para ir aos sub-menus de preferências, pedidos, metas e gráficos.

Figura 18 – Menu principal do protótipo

3.3.2.5 Sub-menu de preferências

O arquivo que fica dentro do server do restaurante recebendo do php o arquivo XML tem o nome de `exportação`. Assim dentro do protótipo coloca-se o nome do arquivo em um dos campos que constituem a URL de acesso.

Também é necessário selecionar se esta internamente na rede ou externamente, escolhendo entre o IP interno e o IP externo do server, conforme Figura 19.

O arquivo está em um servidor com a possibilidade de haver mais sistemas. Por isso, este foi colocado em uma pasta chamada `iPadOut,` que contém os outros arquivos para fazer o protótipo funcionar.

Ao finalizar as escolhas, é apresentado a URL que fará a importação do XML.

Figura 19 – Sub-menu de preferencia do protótipo

3.3.2.6 Importando XML

Como apresentado na Figura 19, existe um botão de importação, chamado `Importar`

`XML` , o qual está presente em vários locais do sistema, pois ai pode-se ver informações do restaurante em tempo real apenas clicando neste botão para trazer as informações do servidor para dentro do banco de dados do protótipo dentro do iPad.

Se a conexão estiver lenta, permite-se ver uma pop-up de transferência das informações. Caso contrário, a Figura 20 passa despercebida.

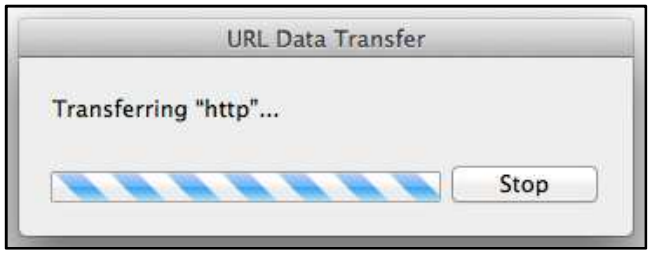


Figura 20 – Transferindo XML para o protótipo

3.3.2.7 Sub-menu de pedidos

Esta tela traz informações de mais de uma tabela do banco de dados, pois um pedido é constituído por produtos e tipos, e após ser importado, há necessidade é de avaliar os pedidos e não as demais tabelas.

Todos os pedidos que aparecem na Figura 21, estão no protótipo neste instante, pois foram importados quando o usuário clicou no botão `importar XML`.

É possível navegar entre os pedidos para saber tudo que foi requisitado pelos clientes do restaurante, basta clicar sobre os botões de <<, que leva para o primeiro pedido, < que leva para o pedido anterior, > que leva para o próximo e >> que leva até o último pedido feito até a importação.

Figura 21 – Tela de pedidos do RestauranteHi no protótipo

3.3.2.8 Metas

O protótipo comporta uma tabela para definição de metas, as quais devem ser construídas com dados como: descrição da meta, quantidade, se a meta é mensal, anual ou vendas até o momento e o produto escolhido. Assim é apresentado um gráfico conforme a Figura 22, que tem uma coluna para mostrar o quanto o restaurante vendeu no período e quanto foi definido como meta.

Figura 22 – Metas do protótipo

3.3.2.9 Cadastrando metas

Para acrescentar uma nova meta, basta selecionar a ultima linha da lista de metas e detalhar a nova meta, alimenta-se a descrição, quantidade, tempo e tipo de produto, conforme Figura 23.

Figura 23 – Cadastrando nova meta

3.3.2.10 Itens de maior consumo

É apresentado na Figura 24 os itens de maior consumo no restaurante com suas respectivas porcentagens. A regra é comparar a quantidade de vezes que o produto foi pedido com a quantidade estimada na meta. Para isso são somados as quantidades em cada pedido e como um todo apresentado na tela do sub-menu de `Gráficos` do protótipo.

Figura 24 – Itens de maior consumo

3.3.2.11 Tipos de pedido

Conforme Figura 25, é apresentado a porcentagem de pedidos feitos por tipo. Onde o restaurante classificou os tipos em pratos, bebidas e sobremesas.

Figura 25 – Tipos de pedidos (prato / bebida / sobremesa)

3.4 RESULTADOS E DISCUSSÃO

A maior parte da pesquisa deste trabalho focou na troca das informações entre o banco de dados que fica dentro do servidor com o protótipo que está longe deste servidor. Assim a troca de informações baseou-se em criar um padrão que o próprio FileMaker reconhecesse, facilitando a comunicação.

Foi necessário disponibilizar uma interface que estimule e facilite o uso do protótipo, tendo em vista que usuários de quaisquer áreas possam utilizá-lo, como administradores de empresas.

Comparando-se com trabalhos correlatos, este protótipo tomou corpo e apresentou um diferencial no que diz respeito à facilidade de acesso e usabilidade, onde os trabalhos correlatos não previram a adequação ao cenário móvel.

Para identificar a aceitação e a adequação de uso do protótipo, aplicou-se um questionário de usabilidade para quatro administradores de restaurante. O questionário foi aplicado oralmente enquanto apresentava-se o protótipo. Este questionário foi aplicado no dia 05/11/2012.

O questionário foi elaborado com base na ISO/IEC 9126 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2003), que é uma norma ISO para qualidade de produto de software, que se enquadra no modelo de qualidade das normas da família 9000. Dentre os resultados mais relevantes na aplicação do questionário, percebeu-se a adequação do sistema ao que inicialmente se propôs, tendo em vista a aderência de variados sistemas operacionais móveis, conforme mostra Figura . Para saber se o protótipo seria bem aceito, foi questionado qual dispositivo o administrador utiliza no momento, em seu trabalho. Duas pessoas utilizavam o iPad da Apple, uma utilizava o Galaxy da Sansung e o outro não tinha *tablet*. Todos os usuários fizeram o teste no iPad do autor deste trabalho.

Conforme a Figura 27, os usuários usariam em seus estabelecimentos este protótipo e conforme a Figura 28 percebe-se que os usuários identificaram facilmente o resultado que atingiriam com cada funcionalidade do sistema. Quanto à usabilidade, Figura 29, os resultados foram positivos, concluindo que todos acharam usual a área de metas. Já sobre a eficiência como um todo, Figura 30, foi elogiado em todos os casos o tempo de resposta, que levou em média 3 segundos para obter 8 registros. Para concluir foi solicitado que o usuário criasse sua própria meta, a qual todos, após rápido treinamento, concordaram no quesito facilidade de manutenção, conforme Figura 31.

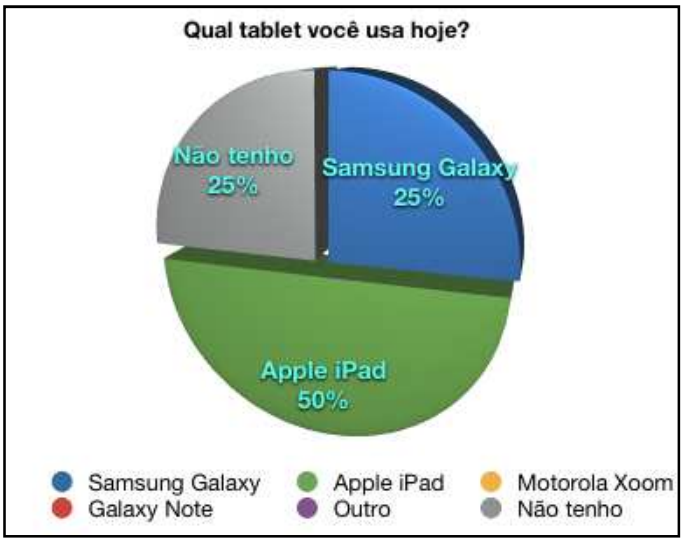


Figura 26 – Respostas em relação ao *tablet* utilizado

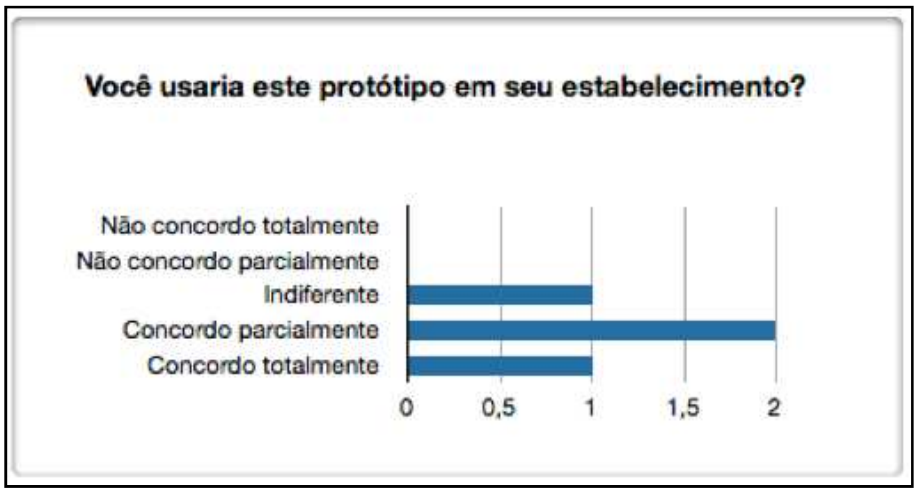


Figura 27 – Respostas em relação a utilização do protótipo

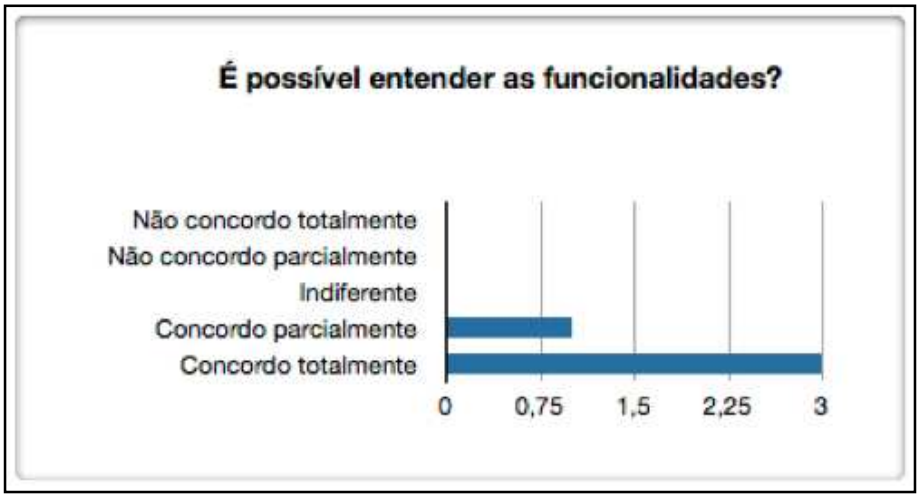


Figura 28 – Respostas em relação ao entendimento das funcionalidades do sistema

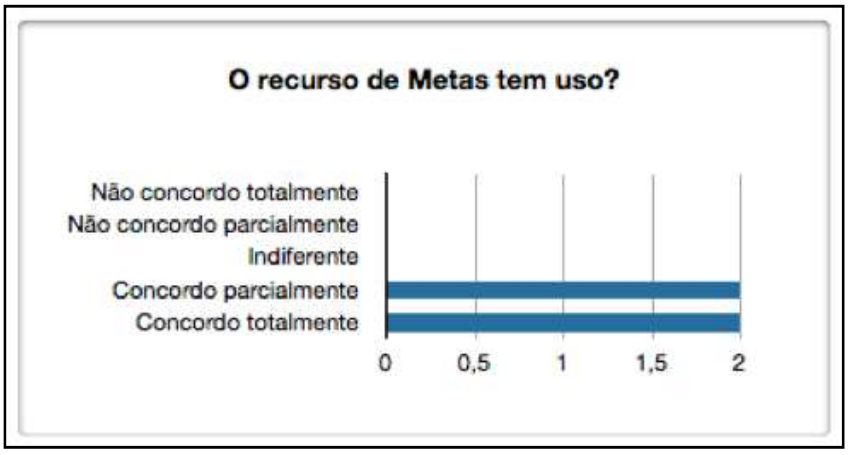


Figura 29 – Respostas em relação a usabilidade das metas

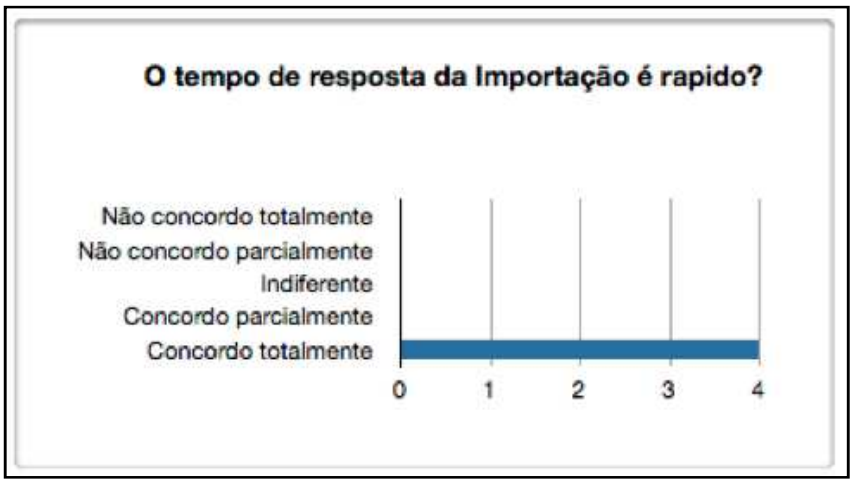


Figura 30 – Respostas em relação a eficiência do sistema

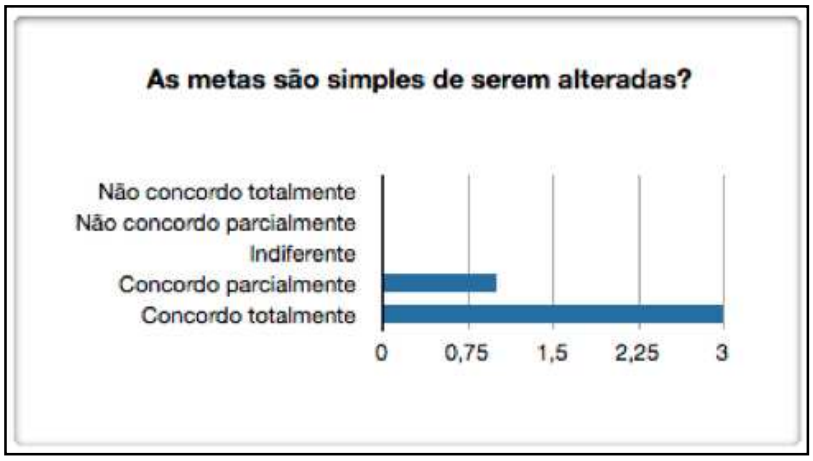


Figura 31 – Respostas em relação a manutenibilidade do sistema

4 CONCLUSÕES

Este trabalho teve como finalidade estudar e desenvolver uma ferramenta baseada no padrão XML a um baixo custo, que em questionários, percebeu-se a mobilidade gerada aos gestores de restaurantes, possibilitando o acesso a informações em tempo real remotamente.

Para agilizar a tarefa de gerenciamento, foi desenvolvida uma interface gráfica focada na facilidade de uso, mesmo sendo usada por pessoas com pouco conhecimento em informática. A interface é composta com botões de fácil manipulação e com apresentação de relatórios com números e gráficos.

Foram disponibilizadas informações de vendas até uma hora atual, metas de vendas e itens de maior consumo do restaurante através de um servidor remoto. Os dados de cada pedido armazenados no sistema do restaurante foram exportados para arquivos no formato XML. Informações importadas a partir do banco de dados FileMaker do servidor foram armazenadas no iPad. Para auxiliar no gerenciamento das informações obtidas, foram construídas telas e relatórios gráficos apresentando informações de cunho analítico.

O trabalho desenvolvido permite a visualização e gerenciamento das informações da empresa que utiliza o sistema RestauranteHi remotamente de forma simplificada.

A integração do FileMaker com outra tecnologia, no caso a linguagem PHP e o formato XML, mostrou-se de alta complexidade. O maior desafio foi compreender a tecnologia disponibilizada pela FileMaker para realizar tarefas utilizando PHP. Muito por conta da escassez de documentação e outros trabalhos similares. As mensagens de erro disparadas pela tecnologia não facilitavam a detecção da falha no protótipo. Foram inúmeros testes de conexão, até conseguir realizar as consultas necessárias.

Os trabalhos correlatos analisados ajudaram a determinar como o padrão XML foi utilizado, a fim de simplificar a visualização dos dados do sistema. Souza contribuiu com o estudo que viabiliza a integração com outros sistemas, enquanto Piscke ajudou na separação de dados lógicos de negócios da interface do usuário. Os trabalhos correlatos apresentaram aplicativos com foco nos sistemas de informação, mais especificamente utilizando-se o tipo Sistema de Processo Transacional (SPT).

Sendo o iPad uma das tecnologias mais recentes que modificaram a forma do usuário acessar conteúdos rapidamente, fazer uso dele para visualizar relatórios e traçar análises empresariais contribui para o avanço da tecnologia móvel que vem se estabelecendo nos últimos anos.

O esforço despendido neste trabalho contribuiu para o surgimento de oportunidades profissionais. Muito do conhecimento obtido durante a realização deste, já está sendo utilizado no desenvolvimento de produtos comerciais.

O conhecimento obtido e documentado foi, durante o processo, compartilhado com a comunidade FileMaker, despertando o interesse de outros desenvolvedores que aguardam a finalização deste trabalho para aplicar as técnicas em seus projetos.

Os conceitos de banco de dados que durante a vida acadêmica foram aprendidos puderam ser vistos na prática, assim como a necessidade de analisar o projeto antes, durante e depois com levantamento de requisitos, modelagem dos dados e pesquisa de mercado.

4.1 EXTENSÕES

Durante a especificação e implementação do protótipo surgiram algumas sugestões de implementações extensivas ao desenvolvido neste trabalho.

Inicialmente, sugere-se a implementação de versões para rodarem nativamente em sistemas operacionais móveis os quais o FileMaker não é instalável, pois como o padrão criado é baseado em XML, o qual pode ser portável entre diferentes plataformas, basta que sejam alteradas no protótipo algumas classes para que ele funcione em dispositivos como Samsung Galaxy, Motorola Xoom e Galaxy Note, conseguindo obter os mesmos resultados.

Sugere-se também a criação de uma tecnologia escrita sobre a linguagem PHP para fazer o processo inverso ao trabalho. Esta tecnologia deve permitir o gerenciamento das informações no banco de dados FileMaker a partir de um website. O gerenciamento compõem-se por inclusão, exclusão, alteração, visualização dos dados entre outros.

E como última sugestão de extensão, sugere-se usar do mesmo padrão para gerenciamento remoto de websites, pois desta maneira não tem-se à necessidade de ter um webservice, e sim uma troca de informações baseada em XML.

REFERÊNCIAS BIBLIOGRÁFICAS

ALUR, Deepak; CRUPI, John; MALKS, Dan. **Core J2EE patterns**: as melhores práticas e estratégias de design. Tradução Altair Dias Caldas de Moraes. Rio de Janeiro: Campus, 2004.

APPLE. **Apple Incorporation**: iPad human interface guidelines. Cupertino: Apple Incorporation, 2010a.

APPLE. **Apple Incorporation**: iPad user guide. Cupertino: Apple Incorporation, 2010b.

BATISTA, Emerson O. **Sistema de informação**: o uso consciente da tecnologia para o gerenciamento. São Paulo: Saraiva, 2004.

BILBAO, Henrique M. B.; FRANZ, Rubens R. **O empreendedor de visão**. São Paulo: Atlas, 2009.

DALFOVO, Oscar; TAMBORLIN, Norberto. **Business intelligence**. São Paulo: Clube de Autores, 2010.

DAVENPORT, Thomas H.; PRUSAK, Laurence. **Conhecimento empresarial**. Tradução: Lenke Peres. Rio de Janeiro: Campus, 1999.

ELLIOT, James; LOY, Marc. **Getting productive with XMLMind.** California, USA, 2007. Disponível em: <http://www.xml.com/lpt/a/1706>. Acesso em: 20 fev. 2012.

BOWERS, Bob, **How to use FileMaker API and PHP.** California, USA, 2011. Disponível em: <http://help.filemaker.com/app/answers/detail/a_id/6555/~/how-to-use-filemaker-api-and-php>. Acesso em: 16 dez. 2012.

GAMMA, Erich et al. **Padrões de projeto**: soluções reutilizáveis de software orientado a objetos. Tradução Luiz A. Meirelles Salgado. Porto Alegre: Bookman, 2000.

HEITLINGER, Paulo. **O guia prático da XML**: conceitos, exemplos, prática e aplicações da linguagem universal. São Paulo: Centro Atlântico PT, 2001.

HENRY, Patrick D. **FileMaker Pro 11 – getting started**. Califórnia: FileMaker Inc, 2010**.**

JARRAGH. **RestaurantApp.** Espoo, Finlândia, 2011. Disponível em: <https://projects.forum.nokia.com/QMLRestaurantApp/browser/RestaurantApp/qml/common/RestaurantApp/content/restaurant.xml?rev=10a60fdab0023393b3ebe7c5102fedb8ada732e7>. Acesso em: 20 maio 2012.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS**. NBR ISO/IEC9126-1**: engenharia de software – qualidade de produto – parte 1: modelo de qualidade. Rio de Janeiro, 2003.

MCLAUGHLIN, Brett. **Java & XML**. 2nd ed. Califórnia: O'Reilly, 2001.

NUNES, Radilene; BARBOSA, Rodrigo; LUZIRENE, Maria. **Gestão de serviços e fornecedores na cadeia de alimentação**. São Paulo, 2011. Disponível em: <http://www.fam2011.com.br/site/revista/pdf/ed7/art03.pdf>. Acesso em: 17 maio 2012.

PADOVEZE, Clóvis L. **Contabilidade gerencial**: um enfoque e sistemas de informação contábil. São Paulo: Atlas, 1997.

PHP**. O que o PHP pode fazer?** EUA, 2012. Disponível em: <http://br2.php.net/manual/pt_BR/intro-whatcando.php>. Acesso em: 10 fev. 2012.

PHP.NET**. O que é PHP?**. Disponível em: <http://php.net/manual/pt_BR/intro-whatis.php>. Acesso em: 20 fev. 2012.

PISCKE, Leandro S. **Framework em Java para geração de telas no modelo CRUD baseado em XML e objetos remotos utilizando a arquitetura MVC e padrões**. 2007. 67 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

PROSSER, Susan; GRIPMAN, Stuart. **FileMaker Pro 11:** the missing manual. Califórnia: O'Reilly, 2010.

RUAS, Roberto L.; PINHEIRO, Ivan A. **Sua empresa é competitiva?:** diagnóstico de competitividade para as micro e pequenas empresas comerciais. Brasília: Sebrae, 1996.

LIMA, Ronaldo F.; **Como o iPad pode ajuda-lo nos negócios.** 2012. Disponível em: <http://www.saibre.com.br/2012/07/como-o-ipad-pode-ajuda-lo-nos-negocios>. Acesso em: 19 dez. 2012

SOUZA, Ricardo F. de. **Software para planejamento de unidade didáticas estruturado através de XML SCHEMA**. 2005. 86 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

STAIR, Ralph M. **Princípios de sistemas de informação**. Rio de Janeiro: LTC, 1998.

WORLD WIDE WEB CONSORTIUM. **Requirements for XML Schema 1.1.** [S.l.], 2003. Disponível em: <http://www.w3.org/TR/2003/WD-xmlschema-11-req-20030121/>. Acesso em: 16 dez. 2012.

XMLMIND**. O que é o editor de XML XMLMind?** EUA, 2012. Disponível em: < http://www.xmlmind.com/xmleditor/>. Acesso em: 05 mar. 2012.

YODER, Joseph; JOHNSON, Ralph; WILSON, Quince. Connecting business objects to relational database. In: CONFERENCE ON PATTERNS LANGUAGES OF PROGRAMS, 5th, 1998, Monticello. **Proceedings...** Illinois: Dept. of Computer Science, 1998. p. 9-11. Disponível em: <http://test.joeyoder.com/Research/objectmappings/Persista.pdf>. Acesso em: 20 maio 2012.