UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

EDUARDO CORRÊA RODRIGUES

SISTEMA WEB DE GERENCIAMENTO DE TARIFAÇÃO DO RESTAURANTE UNIVERSITÁRIO DA UFES

VITÓRIA – ES
ABRIL/2017

EDUARDO CORRÊA RODRIGUES

SISTEMA WEB DE GERENCIAMENTO DE TARIFAÇÃO DO RESTAURANTE UNIVERSITÁRIO DA UFES

Parte manuscrita da Proposta de Projeto de Graduação do aluno **Eduardo Corrêa Rodrigues**, apresentada ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Engenharia Elétrica.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Departamento de Engenharia Elétrica

Orientador: Prof. D.Sc Hans Jorg Andreas Schneebelli

VITÓRIA – ES
ABRIL/2017

EDUARDO CORRÊA RODRIGUES

SISTEMA WEB DE GERENCIAMENTO DE TARIFAÇÃO DO RESTAURANTE UNIVERSITÁRIO DA UFES

Parte manuscrita da Proposta de Projeto de Graduação do aluno **Eduardo Corrêa Rodrigues**, apresentada ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Engenharia Elétrica.

Trabalho aprovado. Vitória, ES, 26 de Abril de 2017:

_____

Prof. D.Sc Hans Jorg Andreas Schneebelli
Orientador

_____

Prof. D.Sc Alvaro Cesar Pereira Barbosa
Avaliador

_____

Prof. D.Sc Eliete Maria de Oliveira Caldeira
Avaliador

_____

M.Sc Leandro Muniz Lima
Avaliador

VITÓRIA – ES
ABRIL/2017

AGRADECIMENTOS

Ao Professor Hans, pela paciência, pela oportunidade e pela orientação.

Ao Igor, Leandro e Aline, que me ajudaram muito neste trabalho.

À minha esposa, Jessica, que teve paciência e que me ajudou bastante a concluir este trabalho.

À minha mãe e ao meu irmão, que sempre me apoiaram e, com este trabalho, não foi diferente.

Aos meus amigos, pela força e torcida.

RESUMO

Este trabalho consiste em analisar e desenvolver um Sistema de Tarifação, com interface *Web*, para o Restaurante da Universidade Federal do Espírito Santo (RU). O desenvolvimento dessa pesquisa baseia-se na necessidade do Núcleo de Tecnologia da Informação em obter um sistema que gerencie a tarifação do RU. Para tanto, foram estudados os requisitos e as ferramentas necessárias para a implementação do Projeto. O processo de construção do sistema passou pelas etapas de levantamento de requisitos, especificação de requisitos, definição da arquitetura do sistema, implementação e testes.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

LAN	*Local area network*
LDAP	*Lightweight Directory Access Protocol*
NTI	Núcleo de Tecnologia de Informação
RU	Restaurante universitário
SGBD	Sistema de Gerenciamento de Banco de Dados
TAE	Técnico-Administrativos em Educação
TI	Tecnologia de Informação
UFES	Universidade Federal do Espírito Santo
VPN	*Virtual Private Network*
MVC	modelo-visão-controle
GPL	Licença Pública Geral
PHP	*Hypertext Preprocessor*
XHTML	*eXtensible Hypertext Markup Language*
CSS	*Cascading Style Sheets*
HTML	*Hypertext Markup Language*
API	*Application Programming Interface*
SQL	*Structured Query Language*
PDO	*PHP Data Objects*
CPF	Cadastro de Pessoas Físicas
URL	*Uniform Resource Locator*

SUMÁRIO

1 INTRODUÇÃO

Com o avanço da tecnologia, cada vez mais os processos diários estão sendo automatizados, porém nota-se que algumas atividades das universidades federais não estão acompanhando tal evolução, como no caso do sistema de cobrança do restaurante universitário da UFES. O RU é de suma importância para a comunidade acadêmica. Possui 1.056 lugares e fornece 601.000 refeições por ano e 5.500 refeições por dia (FERNANDES, 2011). Aperfeiçoar tais processos por meio de tecnologias modernas permite a diminuição de custos e torna mais prático o cotidiano de todos que estão envolvidos. Para lidar com a demanda, é necessário um sistema de tarifação que simplifique e possa agilizar todo o processo.

Existem dois tipos de público do RU: interno e externo. O externo engloba pessoas que não têm vínculo com a universidade (visitantes). O interno, por sua vez, é composto pela comunidade acadêmica, quais sejam, estudantes, docentes, técnicos administrativos (TAE) e funcionários de empresas terceirizadas. Os estudantes podem ser classificados em três tipos: estudante com 100% de isenção; 50% de isenção e estudante sem isenção. Docentes e técnicos compõe o grupo dos servidores. Já os usuários especiais englobam os terceirizados e outros como consta na Resolução 56/2014 do Conselho Universitário da UFES. O tipo de categoria ao qual o sujeito pertence determina o preço pago por refeição. A Figura 1 apresenta os preços praticados em Dezembro de 2016.

O desenvolvimento dessa pesquisa baseia-se na necessidade do Núcleo de Tecnologia da Informação (NTI) em obter um sistema que gerencie a tarifação do RU. Este pode ser dividido em duas partes: o sistema embarcado nas catracas, que será desenvolvido em outro projeto e o sistema W*eb*, objeto de estudo em questão.

Figura 1 – Valores por categoria de usuário

Categoria de usuário	Valor pago R$
Estudantes 100% *	Isento
Estudante 50% *	0,75
Estudantes*	1,50
Servidor * (Resolução 27/2016)	9,50
Usuários especiais (Resolução 27/2016)	9,50
Visitantes (Resolução 27/2016)	9,50

Fonte: site do RU/UFES.

1.1 Sistema de tarifação

Atualmente, o restaurante universitário possui um sistema que foi adquirido de um fornecedor externo, e não desenvolvido pela própria universidade. Porém, está tecnologicamente obsoleto e sua licença de uso terminará em breve (informação verbal)1. Além disso, não possui integração com as demais aplicações; não conta com uma interface de consulta de saldo online; e não pode ser atualizado pelo NTI.

O sistema de tarifação clássico, no qual o usuário compra um cupom para cada refeição, necessita de alocação de pessoal suficiente e agilidade nos processos para atender a demanda diária, que é significativa. Por outro lado, um sistema informatizado no qual o usuário compre créditos para utilizar *a posteriori* otimiza todo o processo, evitando filas e tempo de espera desnecessários no caixa do restaurante.

[1]Informação fornecida por Hans Jorg Andreas Schneebelli no Núcleo de Tecnologia da Informação da UFES, em Vitoria, em Junho de 2016.

Tal sistema necessita de um cartão de acesso, objeto de identificação que vincula o usuário. Uma vez carregado, passará por um leitor localizado na catraca, autorizando seu detentor a adentrar no restaurante. Os dados sobre o saldo podem estar armazenados no próprio cartão ou no banco de dados do NTI, sendo esta última a opção mais segura, na medida em que há uma maior dificuldade de adulterar as informações.

O sistema mais adequado, portanto, é o informatizado com as informações de saldo salvos em uma base de dados, o que pressupõe a existência de uma infraestrutura que leve em consideração a segurança, uma vez que gerencia o aspecto monetário do processo. Na Tabela 1 estão apresentadas as características de cada possível sistema.

Tabela 1 – Características de cada possível sistema.

	Acesso Rápido ao RU	Seguro	Atualizável	Baixo custo	Integração com outras aplicações
Sistema de tarifação clássico	Não	Sim	Não	Sim	Não
Sistema atual	Sim	Não	Não	Não	Não
Sistema informatizado com saldo salvo no cartão	Sim	Não	Sim	Sim	Sim
Sistema informatizado com autenticação via cartão e saldo em uma base de dados	Sim	Sim	Sim	Sim	Sim

Fonte: Produção do próprio autor.

1.2 Objetivos

1.2.1 Objetivo geral

Desenvolver um sistema que controle a tarifação do restaurante universitário, isto é, gerencie uma base de dados onde estará contido o cadastro de todos os usuários, assim como seus respectivos saldos. As informações serão obtidas através do serviço de diretório de cadastro (LDAP) da UFES a fim de manter a base sempre atualizada.

1.2.2 Objetivos específicos

a) Identificar e documentar os requisitos do Sistema;
b) Realizar a modelagem do Sistema e documentá-la;
c) Definir a arquitetura e documentá-la;
d) Implementar o Sistema.

1.3 Metodologia

A princípio, será realizado o levantamento de requisitos e a especificação dos mesmos. Em seguida, será feita a definição da arquitetura do sistema. Por fim, será elaborado o Documento de Projeto, contendo a arquitetura do software e esboço detalhado de cada um de seus componentes.

O próximo passo será o estudo de tecnologias e ferramentas necessárias para o desenvolvimento do sistema, tais como: Linguagem de Programação, Banco de Dados, *frameworks*. Uma vez concluída a fase de definições do sistema, e com as ferramentas de desenvolvimento já predeterminadas, ocorrerá sua implementação. Ao final dessas, serão realizados testes para validar o sistema.

1.4 Estrutura do Trabalho

O próximo capítulo deste trabalho, o Segundo, mostra o sistema proposto e seus requisitos; já o Terceiro apresenta como foi realizada a implementação e as ferramentas utilizadas para tal. O Capítulo Quatro relaciona os testes realizados bem como os resultados obtidos e, por fim, o Cinco, resume as conclusões do trabalho.

2 SISTEMA DE CONTROLE DE TARIFAÇÃO

Este capítulo aborda os resultados da Engenharia de Requisitos para a construção do sistema de tarifação do RU. Na Seção 2.1 é apresentado o Escopo do Projeto; na Seção 2.2, as Funcionalidades e na Seção 2.3, é apresentado o Banco de dados. Por fim, na Seção 2.4, 2.5 e 2.6, é apresentado o *framework*, a comunicação com as catracas e a arquitetura, respectivamente.

2.1 Descrição do Escopo

Diariamente, muitas pessoas fazem uso do restaurante universitário. A fim de agilizar o processo de compra e de acesso ao refeitório, existe a necessidade de um sistema de tarifação. Tal sistema terá uma interface *Web* para adicionar, editar e excluir os dados e armazenará as seguintes informações: local e capacidade de pessoas de cada restaurante; local e status de cada catraca; horário e nome de cada tipo de refeição; valor e horário de cada tarifa; data, valor e origem de cada movimentação financeira; nome e prioridade de cada categoria; tempo de vigência da categoria para cada usuário; saldo, identificação e identificação do cartão para cada usuário.

Para aumentar a segurança e garantir a confiabilidade dos dados, as informações sobre o restaurante, a catraca, a tarifa, a categoria e o tipo de refeição só poderão ser excluídas ou editadas quando não tiverem vínculo com outros dados.

O valor cobrado para um usuário depende da categoria que este possui. As categorias têm uma data de início e uma de fim. Caso o indivíduo perca o vínculo antes do fim da vigência, haverá a possibilidade de expirar a categoria. Assim, a data final passa a ser a data do dia corrente.

Por fim, será necessária que a aplicação gere relatórios, quais sejam: fluxo de pessoas e fluxo de caixa.
O usuário do restaurante, chamado nesse projeto de usuário comum, conseguirá fazer consulta de seu saldo através da interface W*eb*. O caixa do restaurante poderá inserir crédito, consultar extrato e definir uma categoria para um usuário. O gestor terá, além das mesmas funções do

caixa, os seguintes acessos: permissão de cadastrar as categorias, os restaurantes, as catracas, as tarifas, os tipos de refeições, gerar relatórios e fazer ajustes nas recargas que obtiverem erro.

O sistema também deve possibilitar uma conexão com as catracas de forma a realizar o decremento do saldo a medida que o indivíduo utilize o cartão.

2.2 Funcionalidades

O usuário quando apresentar o cartão, na entrada do refeitório, para leitura no sensor das catracas, deve ter seu acesso liberada caso tenha saldo suficiente. No caso de acesso liberado, o saldo deve ser ajustado de acordo com a tarifa. Nas situações que não tenha saldo suficiente, o usuário deve se dirigir ao caixa para compra de créditos.

Os usuários deverão poder consultar seus saldos através de um site. Nele terão que digitar seus logins únicos da universidade através de uma consulta do diretório da UFES acessível por meio do protocolo LDAP. O protocolo LDAP "[…] ou Protocolo Leve de Acesso a Diretórios é um conjunto de regras que controla a comunicação entre serviços de diretórios e seus clientes" (TRIGO, 2007). O serviço de LDAP da UFES, que será chamado nesse projeto de LDAP-UFES, utiliza o 389 Directory Server, que é um serviço de diretório baseado no protocolo LDAP, que provê informações em forma de árvore com o objetivo de facilitar o acesso a cada item armazenado. É um sistema de diretório centralizado, visando agregar dados de todas as pessoas que possuem relação com a universidade. Seu principal objetivo é o de prover esses dados às aplicações que necessitem dos mesmos.

O sistema deverá ter, ainda, uma interface de uso dos caixas e gestores. Neste ambiente, todo o controle de caixa é realizado, e inclui atividades como realização de recarga, cadastro de uma nova tarifa, etc. Os caixas e gestores do RU também serão autenticados através do LDAP-UFES, porém, neste caso, terão acesso a área administrativa.

Por fim, o sistema deverá ser capaz de realizar a comunicação com as catracas. Isto é, autenticar as catracas e o usuário, liberar a passagem e, em seguida, realizar o decremento do saldo do usuário.

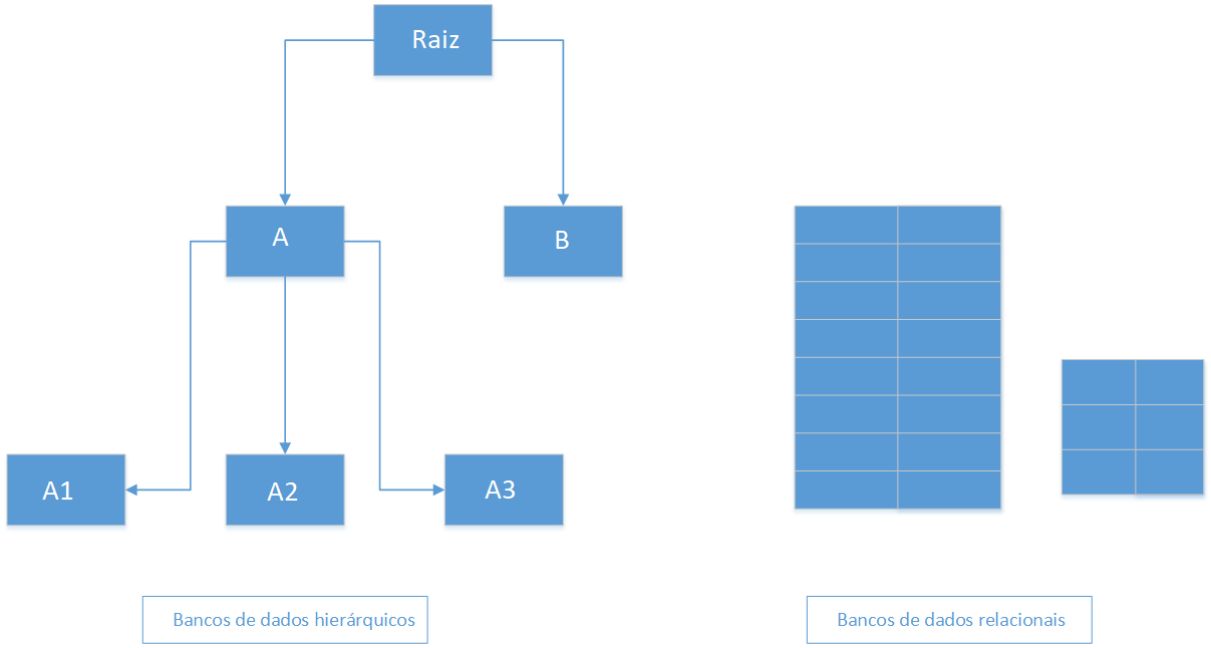
2.3 Armazenamento da Informação de Saldo

Os dados sobre o saldo podem estar armazenados no próprio cartão, no banco de dados ou em ambos. No caso em que o saldo é salvo no cartão tem-se a segurança debilitada, a medida em que há uma maior facilidade de adulterar as informações. No caso em que o dado tanto fica salvo no cartão quanto no banco de dados tem-se um aumento desnecessário na complexidade porque será preciso sincroniza-los. Desta forma, o melhor cenário é com o saldo armazenado em um banco de dados seguro.

Um banco de dados é "[…] uma coleção de dados relacionados" (ELMASRI e NAVATHE, 2010). Os SGBD (Sistema Gerenciador de Banco de Dados) são softwares que controlam de maneira eficaz diversos bancos de dados. (ROB e CORONEL, 2011). Há alguns modelos de SGBD, quais sejam, modelo hierárquico, rede, relacional, dedutivo, entre outros.

Os Bancos de Dados Relacionais possibilitam que os usuários utilizem uma grande variedade de abordagens no tratamento das informações, além de poderem fazer perguntas relacionadas aos negócios por meio de vários pontos. Já nos bancos de dados hierárquicos, os usuários precisam definir as regras de negócios de maneira específica, iniciando a consulta sempre pela raiz. A Figura 2 apresenta a diferenciação dos modelos relacional e hierárquico de acordo com a representação dos dados.

Figura 2 – Bancos de dados relacional e hierárquico.



Fonte: Produção do próprio autor.

Assim, devido a uma maior “abstração de dados e independência entre dados e programas” (ELMASRI e NAVATHE, 2010) nos modelos relacionais, este é o mais adequado para conter as informações dos usuários e suas movimentações financeiras.

Serão utilizadas duas bases de dados. A primeira, chamada de Banco Principal, terá estrutura relacional, será acessada pelas aplicações e conterá os dados de saldo de cada usuário. A segunda, cuja estrutura é hierárquica, já está implementada na universidade, utiliza o protocolo LDAP e contém os dados cadastrais atualizados dos usuários.

2.4 Framework

A UFES adotou alguns padrões para desenvolvimento de software. Entre outros, tem-se o Drupal e o Yii (informação verbal)2.

[2]Informação fornecida por Hans Jorg Andreas Schneebelli no Núcleo de Tecnologia da Informação da UFES, em Vitoria, em Agosto de 2016.

Considerado um *framework* de alta performance, capaz de agregar segurança e velocidade às aplicações, o Yii é, atualmente, um dos mais utilizados do mercado. Com ele, é possível desenvolver desde sites mais simples até as mais complexas aplicações *Web*, baseando-se no padrão arquitetural modelo-visão-controle (MVC). O MVC visa separar a lógica de negócio da interface com o usuário. Assim, os programadores podem mudar facilmente cada parte, sem afetar as outras. No padrão MVC, o modelo representa as informações (os dados) e as regras de negócio, a visão contém elemento de interface com o usuário, como textos ou formulários, e o controle gerencia a comunicação entre o modelo e a visão.

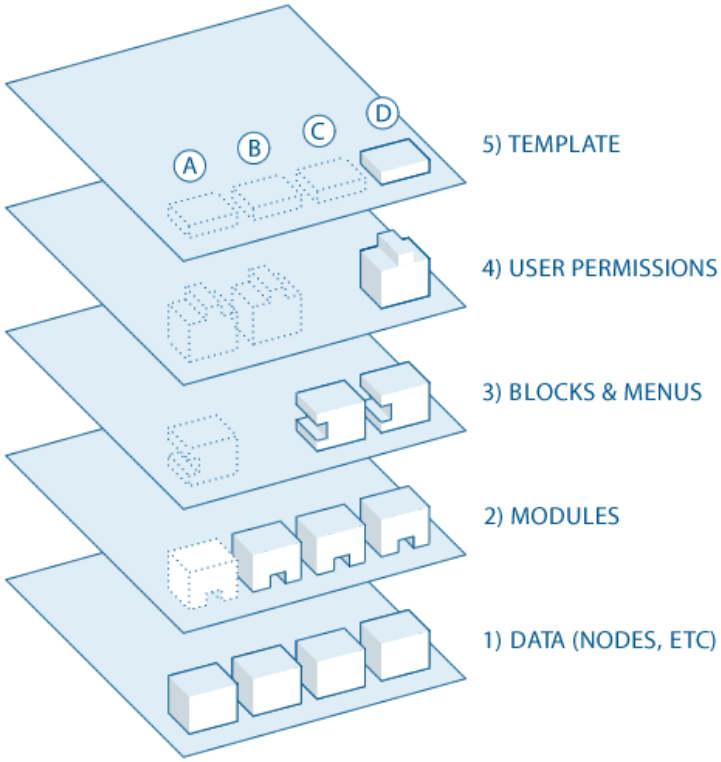
O Drupal é um *framework* modular focado na gerência de conteúdo. Permite desenvolver todo o tipo de *websites* e aplicações. Sua plataforma é de código aberto e é liberado sob a licença GPL (Licença Pública Geral). Possui uma comunidade de programadores, entusiastas e empresas que, em conjunto, desenvolvem e o fazem evoluir para dar respostas às necessidades de utilizadores e clientes. Possui um excelente grau de segurança. É constantemente testado por sua comunidade, de forma a prevenir vulnerabilidades e garantir uma resposta rápida e eficaz a qualquer problema que possa surgir. Outros benefícios dessa plataforma é o multilinguismo, possuindo mais de 90 línguas disponíveis, inclusive o Português (BR), e a possibilidade de usar mais do que uma simultaneamente.

A vantagem da utilização do Drupal neste sistema é a possibilidade de abstração da aparência e do conteúdo durante o desenvolvimento, uma vez que a UFES já possui um padrão em Drupal para *websites* e aplicações *Web*. Isso possibilita que o foco fique nas funcionalidades do sistema. Outra vantagem é que devido a UFES já utilizar o Drupal, os usuários já estão ambientados com a plataforma.

Seu núcleo foi bem projetado com um sistema de “ganchos” (*hooks* ou *callbacks)*, que permite que módulos insiram funcionalidades. O Drupal tem por objetivo prover um núcleo que suporte ser estendido por meio de módulos personalizados. Especificamente, o Drupal é codificado na linguagem PHP e tem como formato primário de fonte de dados os bancos de dados MySQL (ou MariaDB) e PostgreSQL.

Há cinco camadas principais a serem consideradas na arquitetura do Drupal, como mostra a Figura 3. Na base do sistema está a coleção de nós. Um nó é qualquer parte de um conteúdo, como uma página W*eb*, artigo, tópico de um fórum, etc. Na camada seguinte estão os módulos. Estes são arquivos comuns de códigos para rotinas em PHP agregados a arquivos de suporte e que utilizam a arquitetura do Drupal para integração de novos componentes funcionais (Butcher 2008). Sendo assim, um módulo em Drupal é mais uma noção que permite utilizar bons princípios de design e desenvolvimento. Estes princípios pregam uma atenção especial para formatação de entrada e saída de dados, bem como uma preocupação quanto a localização e documentação dos arquivos. Tais módulos podem ser fornecidos com o núcleo do Drupal ou contribuídos por membros da comunidade. Na próxima camada, encontram-se os blocos e menus. Os blocos geralmente fornecem a saída de um módulo. Em seguida, estão as permissões do usuário. Esta é a camada onde são determinados os diferentes tipos de acesso para cada usuário. Na camada superior está o tema do site. Trata-se predominantemente de XHTML (*eXtensible Hypertext Markup Language*) e CSS (*Cascading Style Sheets*).

Figura 3 – Arquitetura do Drupal.



Fonte: www.drupal.org/docs/7/understanding-drupal/overview

O PHP (*Hypertext Preprocessor*), desenvolvido especialmente para a *Web*, é uma linguagem de código aberto. A sua ideia é que seja possível embutir trechos PHP em código HTML. Como o PHP é escrito diretamente junto com o HTML, o servidor fica encarregado de processar e transformar o código misto (PHP e HTML) em um HTML puro antes de retornar a página para o usuário.

Atualmente, mais de 10 milhões de sites no mundo utilizam a linguagem PHP. O diferencial está na sua capacidade de interação com o mundo *Web* (Niederauer, 2004).

Dentre as funcionalidades do PHP, pode-se destacar a criação de sites *Web* dinâmicos, o que possibilita uma interação com o usuário por meio de formulários, parâmetros da URL e links.

O Drupal foi desenvolvido em PHP e os Módulos, objetos desse trabalho, também.

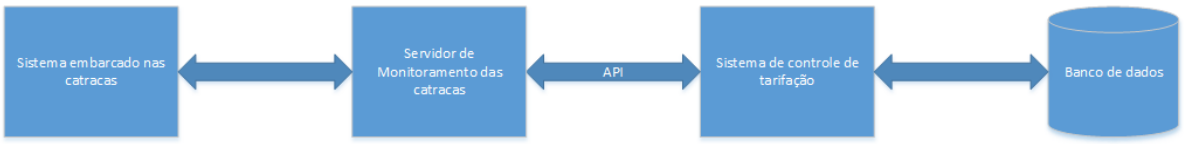
A versão do PHP, utilizada nesse projeto, é 5.4.16.

2.5 Comunicação com as catracas

Neste projeto, será necessário implementar a comunicação entre as catracas e o banco de dados. Uma possível solução é um acesso direto ao banco de dados realizado pelo software embarcado nas catracas, porém isto caracteriza uma falha de segurança, uma vez que o banco de dados estaria vulnerável na rede. Outra solução e o uso de um servidor intermediário (Sistema de controle de tarifação) com uma API implementada. Tal servidor receberia as requisições e acessaria o banco para realizar o ajuste do saldo, fazendo o intermédio. Assim, a solução mas propicia é a segunda. A API (*Application Programming Interface*) é definida como um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por outros sistemas. No contexto da *Web*, uma API é um conjunto definido de mensagens de requisição e resposta HTTP.

Além disso, um segundo servidor é será preciso devido a comunicação com as catracas necessitar de um protocolo de comunicação seguro. Tal protocolo será utilizado na comunicação entre o sistema embarcado nas catracas e o servidor de monitoramento, conforme a Figura 4. O sistema embarcado e o servidor de monitoramento serão desenvolvidos em outro projeto.

Figura 4 - Comunicação entre as catracas e o banco de dados.

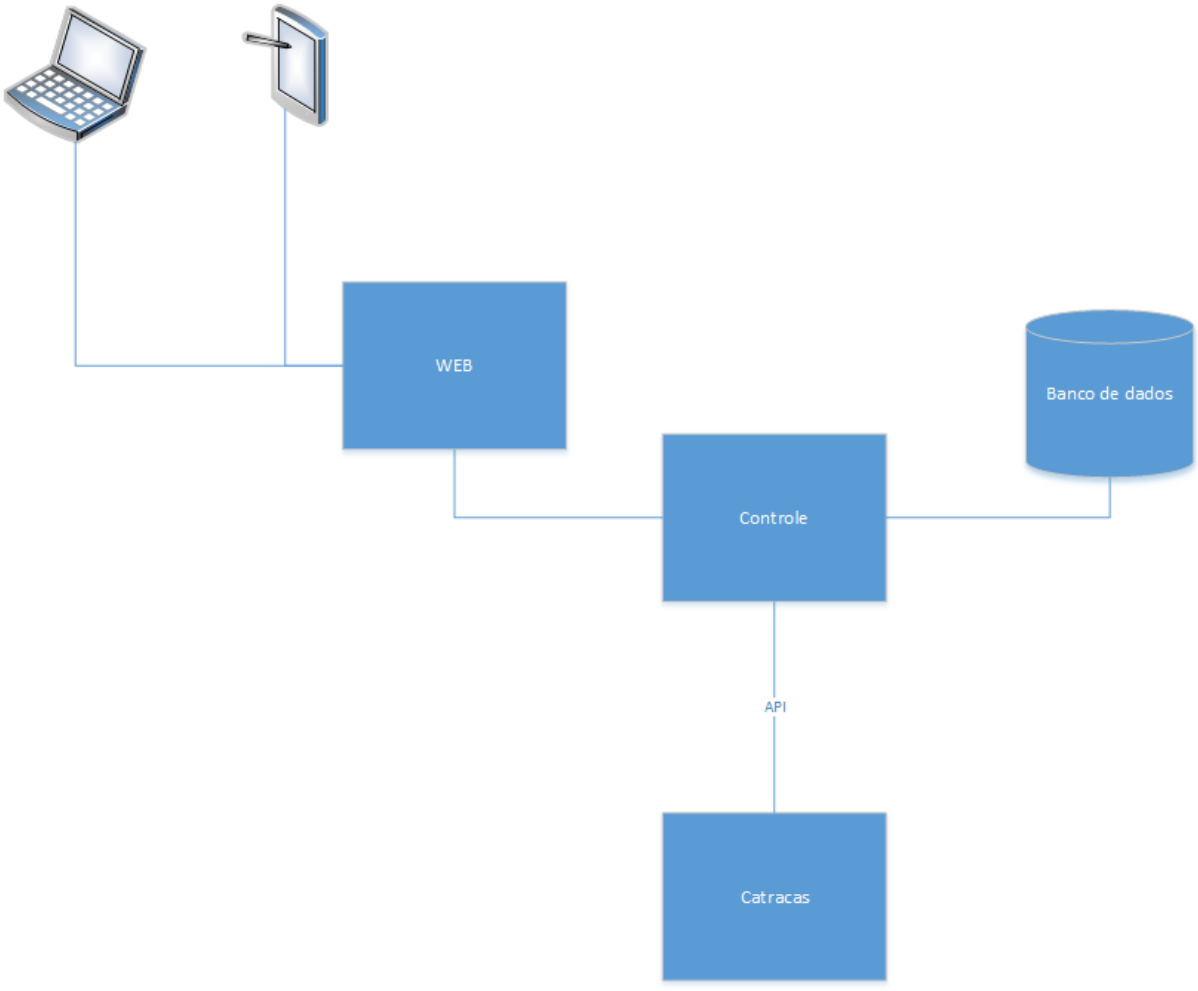


Fonte: Produção do próprio autor.

2.6 Arquitetura

No diagrama a seguir, é possível observar a interface web, chamada de WEB, que possibilitará a consulta de saldo dos usuários. O bloco controle é responsável por gerenciar as informações administrativas e possibilitar a comunicação das catracas, por meio de uma API com o banco de dados.

Figura 5 - Arquitetura do sistema.



Fonte: Produção do próprio autor.

3 IMPLEMENTAÇÃO

Após a identificação e modelagem dos requisitos, ocorre a fase de projeto, no qual o software é modelado de forma que os aspectos tecnológicos que serão utilizados para a implementação do sistema sejam levados em consideração. Esses aspectos envolvem: linguagem de programação, bibliotecas e APIs utilizadas, características de interface com o usuário, arquitetura de software e forma de persistência de dados. Por fim, o produto de software é construído e testado.

Na Seção 3.1, o Modelo de Caso de Uso é descrito. Na 3.2, a arquitetura do sistema é apresentada. Já na 3.3 é mostrado o banco de dados. Na Seção 3.4 é mostrada a camada de abstração de banco de dados. Por fim, na Seção 3.5 é feito o detalhamento dos módulos em Drupal.

3.1 Modelo de Caso de Uso

Os diagramas de casos de uso definem quais funcionalidades um sistema deve oferecer. Tais diagramas possuem dois principais elementos: atores e casos de uso (FALBO, 2014). A Tabela 2 mostra os atores identificados no contexto do sistema.

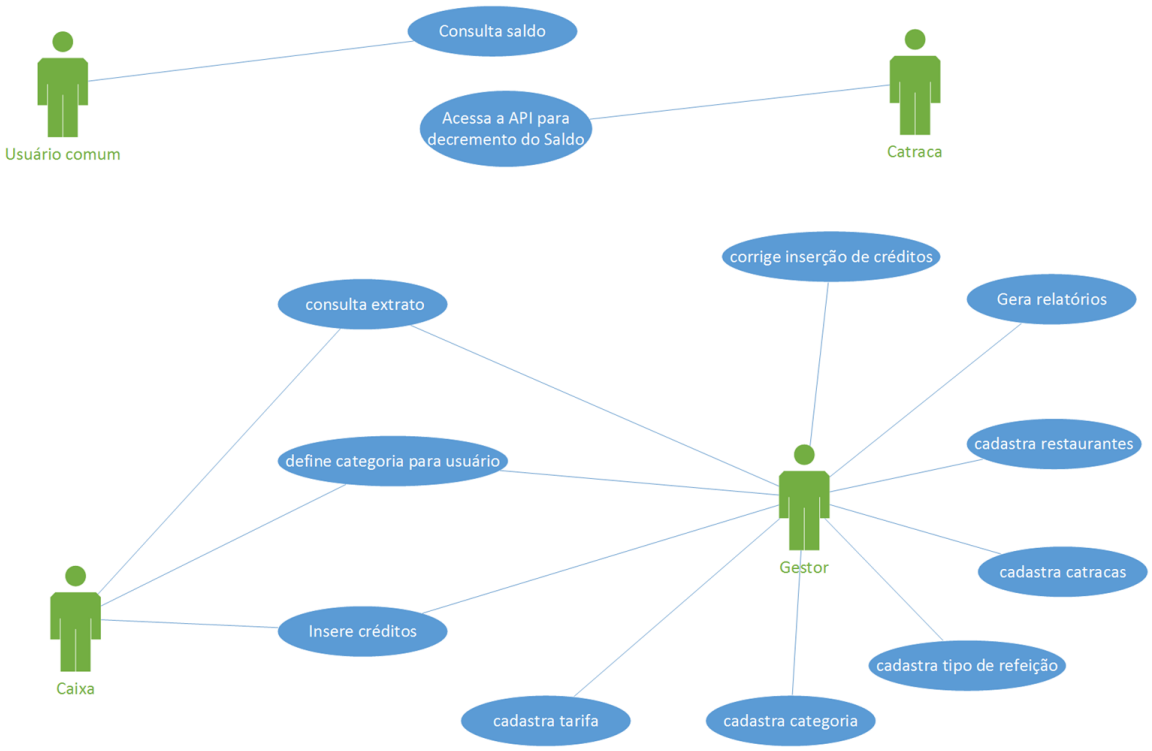
Tabela 2 – Atores.

Ator	**Ação**
Usuário comum	Consulta saldo.
Caixa	Insere créditos, define categoria para usuário, consulta extrato do usuário.
Gestor	Gera relatórios, corrige inserção de créditos, cadastra restaurantes, cadastra catracas, cadastra tipo de refeição, cadastra tarifa, cadastra categoria, define categoria para usuário, consulta extrato do usuário, insere créditos.
Sistema embarcado nas catracas	Acessa a API para decremento do Saldo

Fonte: Produção do próprio autor.

A Figura 6 apresenta os casos de uso do sistema, e em seguida temos suas descrições.

Figura 6 – Diagrama de Casos de Uso.



Fonte: Produção do próprio autor.

Os casos de uso: **cadastra restaurante, cadastra catraca, cadastra tipo de refeição, cadastra tarifa e cadastra categoria** são do tipo cadastrais e incluem alteração, inclusão, consulta e exclusão. A alteração e a exclusão vão depender se o dado salvo já foi utilizado. Por exemplo, se uma tarifa cadastrada já tiver sido utilizada por algum usuário, então ela não poderá ser excluída ou alterada.

O caso de uso **define categoria para usuário** servirá para o caixa ou o gestor vincular uma categoria a um usuário. O **Insere créditos** permitirá que o caixa ou o gestor coloquem créditos para o usuário. O gestor também poderá **gerar relatórios** para determinado período de tempo e **corrigir uma inserção** errada de crédito por meio do corrige inserção de créditos. O caso de uso **consulta saldo** será para os usuários comuns obterem seus saldos.

3.2 Arquitetura do Sistema

O sistema, composto por três módulos feitos em Drupal, controlará a tarifação do restaurante universitário, isto é, gerenciará uma base de dados onde estará contido o cadastro de todos os usuários, assim como seus respectivos saldos. A Figura 7 resume o sistema.
A autenticação dos usuários, para acesso aos módulos, será de responsabilidade do núcleo do Drupal. Este se conecta ao serviço de diretório de cadastro (LDAP) da UFES e realiza a autenticação.

Módulo de Gerência Financeira: Será acessado pelos caixas do restaurante, durante o atendimento, para recarga dos cartões. Este módulo também será responsável por fazer o intermédio entre as catracas e a base de dados. Haverá um sistema embarcado nas catracas da entrada do restaurante, que acessará a API do módulo e fará o decremento do saldo à medida que cada usuário adentre no mesmo. Sistema embarcado este que será desenvolvido em outro projeto.

Três níveis de permissão são necessários neste módulo:
a) administrador, usuário com acesso irrestrito ao sistema;
b) caixa, usuário responsável pela inserção dos créditos;
c) gestor, usuário responsável por gerar relatórios, correção da inserção de créditos;

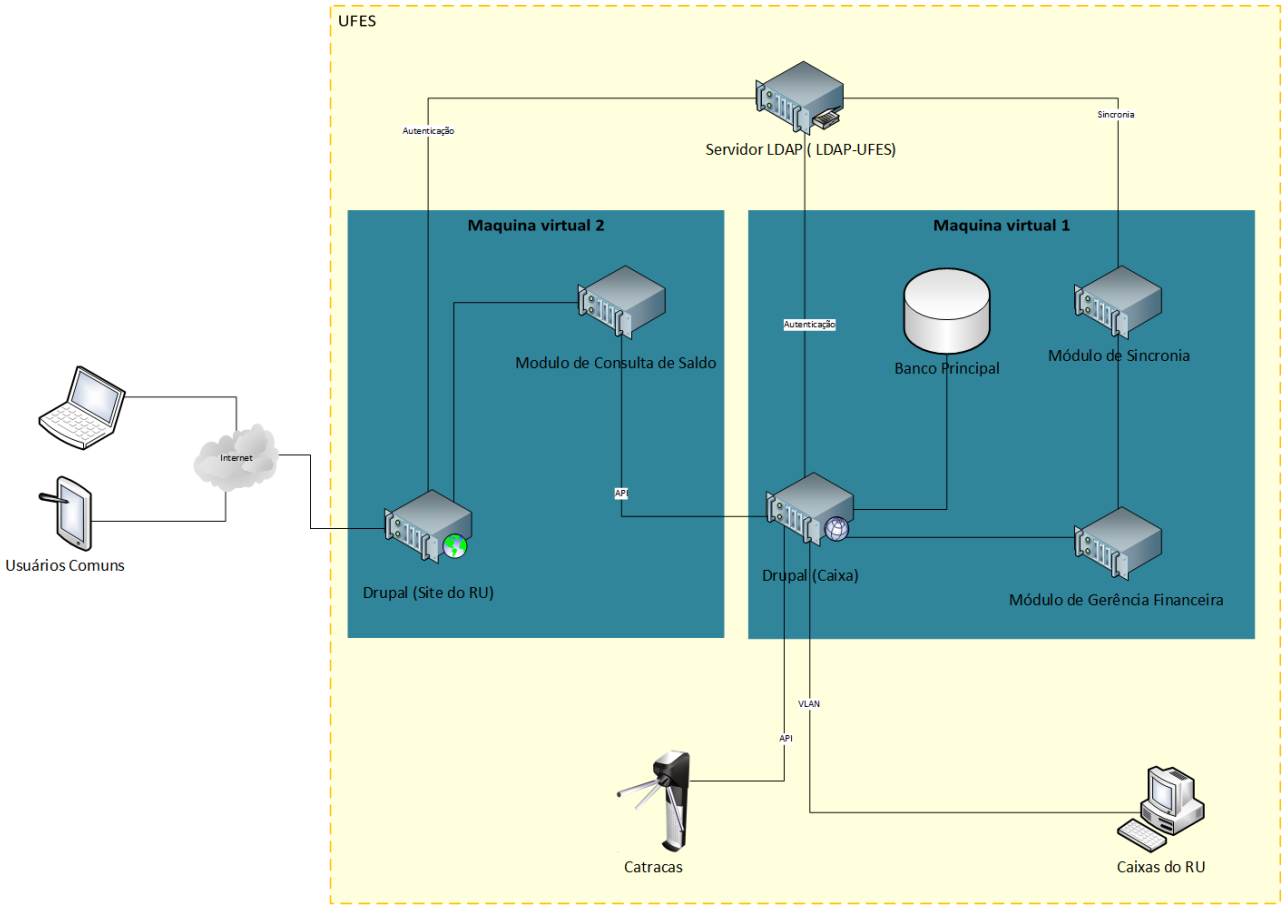
Módulo de Consulta de Saldo: Estará integrado ao site do RU, que divulga diariamente o cardápio, e servirá para consulta de saldo dos usuários.
O usuário deverá se dirigir ao caixa ou fazer um acesso à aplicação online a fim de carregar/recarregar seu cartão e assim ter acesso ao refeitório.

Neste módulo será necessário apenas um nível de permissão:
Usuário comum, que terá acesso apenas a seu saldo e a área de compra de créditos (para imprimir a GRU).

Módulo de Sincronização: é responsável por sincronizar os dados entre o LDAP-UFES e o Banco Principal. Desta forma, obtém-se o vínculo mais atual com a universidade, e assim será cobrado o valor da tarifa correta.

As catracas e o Módulo de Consulta de Saldo realizam o acesso ao Banco Principal através de duas APIs codificadas no Módulo de Gerência Financeira.

Figura 7 – Arquitetura do sistema.

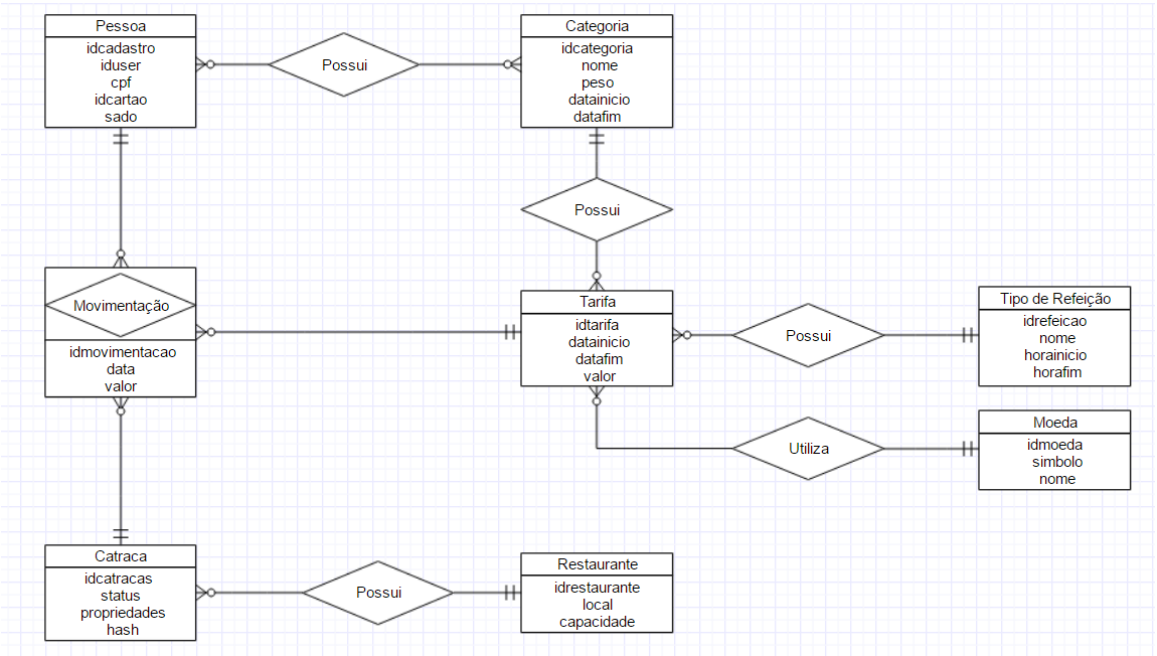


Fonte: Produção do próprio autor.

3.3 Banco de dados

O modelo Entidade-Relacionamento é um modelo de dados conceitual de alto nível, cujos conceitos foram projetados para estarem o mais próximo possível da visão que o usuário tem dos dados, não se preocupando como estes estarão armazenados. O modelo ER é utilizado principalmente durante a fase de projeto do banco de dados. A Figura 8 apresenta o um modelo ER do projeto.

Figura 8 – Modelo Entidade Relacionamento.



Fonte: Produção do próprio autor.

O usuário, representado pela entidade **Pessoa**, poderá ter nenhuma ou até várias categorias. Será cobrado uma tarifa do usuário que utilizar o RU dependendo de sua **Categoria** e Tipo de refeição. A **Categoria** é o vínculo que o usuário tem com a universidade. O **Tipo de refeição** pode ser exemplificado como almoço, jantar, etc.

No momento em que a tarifa é cobrada, gera-se uma **Movimentação** vinculada a uma **Catraca** que, por sua vez, está vinculada a um **Restaurante**.

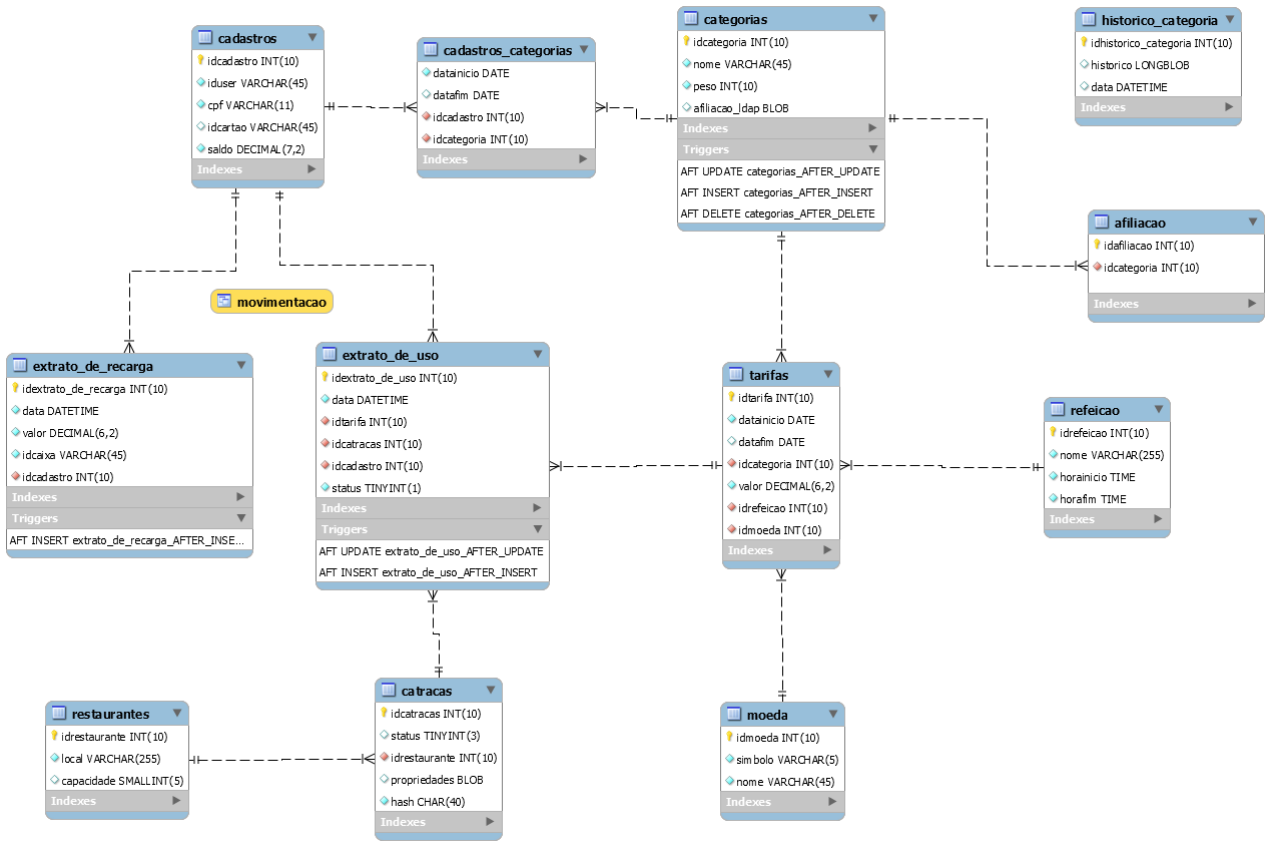
A entidade **Moeda** foi inserida para contemplar algum evento em que seja aceito mais de um tipo de moeda como forma de pagamento ou para uma possível mudança de moeda oficial.

O SGBD selecionado para o projeto foi o MariaDB (versão 5.5.50), um sistema de banco de dados relacional muito semelhante ao MySQL. É um dos servidores de banco de dados mais populares do mundo. Foi feito pelos desenvolvedores originais do MySQL. Sua Plataforma é de código aberto.

“MariaDB é um servidor de banco de dados que oferece a funcionalidade e substituição para o MySQL. [...] Além das funcionalidades básicas do MySQL, MariaDB fornece um rico conjunto de aprimoramentos de recursos, incluindo mecanismos de armazenamento alternativo, otimizações de servidores e patches” (Disponível em www.mariadb.org).

São utilizadas duas bases de dados. A primeira, representada na Figura 9, chamada de Banco Principal, que usa o MariaDB, tem estrutura relacional e é acessada diretamente apenas pelo Módulo de Gerência Financeira. A segunda, chamada de LDAP-UFES, cuja estrutura é hierárquica, já estava implementada na universidade, e contém os dados cadastrais atualizados dos usuários.

Figura 9 – O Banco Principal.



Fonte: Produção do próprio autor.

O Módulo de Gerência Financeira acessa o Banco Principal para inserir e consultar dados nas tabelas. As tabelas **extrato_de_recarga** e **extrato_de_uso** são responsáveis por armazenar os dados de recarga e os dados de acesso ao refeitório, respectivamente. O campo peso na tabela **categorias** é utilizado para definir uma prioridade para os casos em que um usuário possui mais de uma categoria, por exemplo, ser servidor e estudante ao mesmo tempo. A tabela **cadastros_categorias** contém as datas de início e fim de uma categoria para um usuário. A tabela **afiliação** armazena a relação entre uma categoria e uma afiliação (utilizada no LDAP-UFES).

O saldo, na tabela **cadastro**, é atualizado através de um gatilho do banco, exibido na Figura 11, que é iniciado sempre que uma movimentação financeira é inserida na tabela **extrato_de_recarga** ou na tabela **extrato_de_uso**. Outro gatilho, mostrado na Figura 12, salvará os dados na tabela **historico_categoria** sempre que a tabela **categorias** for alterada.

Uma *view*, chamada de Movimentação e mostrada na Figura 10, foi implementada no Banco Principal com o objetivo de centralizar, em uma única consulta, todos os dados financeiros, tanto de entrada quanto de saída.

Figura 10 – *View* Movimentação.

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `movimentacao` AS
    SELECT
        `cadastros`.`iduser` AS `iduser`,
        `extrato_de_recarga`.`data` AS `data`,
        `extrato_de_recarga`.`valor` AS `valor`,
        `extrato_de_recarga`.`idcaixa` AS `operador`,
        1 AS `status`
    FROM
        (`extrato_de_recarga`
        JOIN `cadastros` ON ((`extrato_de_recarga`.`idcadastro` = `cadastros`.`idcadastro`)))
    UNION SELECT
        `cadastros`.`iduser` AS `iduser`,
        `uso`.`data` AS `data`,
        CONCAT('-', `tarifas`.`valor`) AS `valor`,
        CONCAT('Catraca ',
                `uso`.`idcatracas`,
                ' em ',
                `restaurantes`.`local`) AS `operador`,
        `uso`.`status` AS `status`
    FROM
        ((((`extrato_de_uso` `uso`
        JOIN `tarifas` ON ((`uso`.`idtarifa` = `tarifas`.`idtarifa`)))
        JOIN `catracas` ON ((`catracas`.`idcatracas` = `uso`.`idcatracas`)))
        JOIN `restaurantes` ON ((`restaurantes`.`idrestaurante` = `catracas`.`idrestaurante`)))
        JOIN `cadastros` ON ((`uso`.`idcadastro` = `cadastros`.`idcadastro`)))
```

Fonte: Produção do próprio autor.

Figura 11 – Gatilho para atualizar o saldo.

```
CREATE DEFINER=`root`@`localhost` TRIGGER `RU`.`extrato_de_uso_AFTER_INSERT` AFTER INSERT ON `extrato_de_uso` FOR EACH ROW
BEGIN
if (new.status in (0,1)) then
UPDATE cadastros c
SET
    c.saldo = c.saldo - (select valor from tarifas t where t.idtarifa = new.idtarifa)
WHERE
    c.idcadastro = new.idcadastro;
end if;
END
```

Fonte: Produção do próprio autor.

Figura 12 – Gatilho para salvar o histórico das categorias

```
CREATE DEFINER=`root`@`localhost` TRIGGER `RU`.`categorias_AFTER_INSERT` AFTER INSERT ON `categorias` FOR EACH ROW
BEGIN
INSERT INTO `RU`.`historico_categoria`
(`historico`,`data`)
VALUES
((select GROUP_CONCAT(idcategoria,'-',peso) from categorias), current_timestamp());
END
```

Fonte: Produção do próprio autor.

3.4 Camada de abstração do Drupal para banco de dados

Por padrão, o Drupal inclui definição de abstração para MySQL, MariaDB e PostgreSQL. O acesso ao Banco Principal é feito através de uma API do Drupal. Esta fornece uma camada de abstração de banco de dados. A intenção desta camada é preservar a sintaxe SQL, mas também permite aos desenvolvedores uma forma de alavancar funcionalidades mais complexas de maneira unificada. Além disso, ela fornece uma interface estruturada para construir dinamicamente consultas e faz verificações de segurança. O sistema é construído sobre a API de banco de dados PDO (PHP *Data Objects*) do PHP e herda grande parte de sua sintaxe e semântica. A Figura 13, exemplo de uso dessa camada, apresenta uma função do Módulo de Gerência Financeira que consulta, na tabela **cadastros** do Banco Principal, o **iduser** e o **CPF**. Nela, pode-se ver a função `module_load_include` que inclui um arquivo de algum módulo. O arquivo incluído, neste caso, contém as configurações de acesso ao banco. A função `adm_ru_ufes_db_open` direciona a consulta para o banco indicado. As funções `db_select`, `fields`, `condition` são utilizadas para fazer a consulta ao banco e substituem a escrita de uma *query* SQL. A `drupal_json_output` é utilizada para gerar a listagem de CPFs a medida que o usuário comece a digitar. Esta função atualiza os dados sem que haja a necessidade de atualizar a página.

Figura 13 – Exemplo de uso da camada de abstração.

```
14  function adm_ru_ufes_cartao_autocomplete($string = '') {
15    $matches = array();
16    if ($string) {
17      module_load_include('inc', 'adm_ru_ufes', 'adm_ru_ufes.db.api');
18      $current_db = adm_ru_ufes_db_open();
19      $results = db_select('cadastros', 'base');
20      $results->fields('base', array('iduser', 'cpf'));
21      $or = db_or();
22      $or->condition('base.cpf', db_like($string) . '%', 'LIKE');
23      $or->condition('base.iduser', db_like($string) . '%', 'LIKE');
24      $or->condition('base.idcartao', db_like($string) . '%', 'LIKE');
25      $results->condition($or);
26      $results->range(0, 10);
27      adm_ru_ufes_db_close($current_db);
28      $result = $results->execute();
29      foreach ($result as $cartao) {
30        $matches[$cartao->iduser] = check_plain($cartao->cpf);
31      }
32    }
33
34    drupal_json_output($matches);
35  }
36
```

Fonte: Produção do próprio autor.

3.5 Os módulos em Drupal

Um módulo é uma coleção de funções que se agregam ao Drupal com a finalidade de prover novas funcionalidades. Tais funcionalidades são adicionadas por meio da utilização de ganchos existentes no núcleo do sistema, e permitem a execução de códigos em pontos específicos do processamento da página.

Os módulos ficam armazenados no diretório `/modules`. Nesta pasta, não correm o risco de serem sobrescritos ou apagados durante o processo de atualização do Drupal. Assim, foi criado um diretório "`/sites/all/modules/adm_ru_ufes`" que conterá os arquivos do Módulo Gerência Financeira. O mesmo foi feito para os outros módulos.

O primeiro arquivo a ser criado é o `adm_ru_ufes.info`, cujas informações estão na Figura 14. Para cada módulo, é necessário um arquivo `*.info`, que deve ter o mesmo nome do módulo.

Figura 14 – Arquivo adm_ru_ufes.info.

```
name = Restaurante Universitário
description = Modulo de Administração do RU.
core = 7.x
package = UFES

version = "7.x-1.0"

```

Fonte: Produção do próprio autor.

O elemento `name` é usado para exibir o nome do módulo na página de administração dos módulos. O elemento `description` é a descrição do módulo, também exibida na página de administração dos módulos. O elemento `package` define um grupo (ou categoria) para que o módulo seja associado. Na página de administração dos módulos, cada módulo é exibido agrupado pelo `package`. O elemento *core* define a versão do Drupal para qual o módulo é compatível. O elemento `version` define a versão do módulo. (TOMLINSON, Foundation 2012) Em seguida, é necessário criar o arquivo `adm_ru_ufes.module` no mesmo lugar onde se encontra o `adm_ru_ufes.info`, ou seja, no diretório "`/sites/all/modules/adm_ru_ufes`". Tal arquivo conterá a função `adm_ru_ufes_menu()`. Esta função retorna uma matriz associativa, cujas chaves definem caminhos e os valores são uma matriz associativa de propriedades para cada caminho.

3.5.1 Módulo de Gerência Financeira

Este módulo é para uso dos caixas e gestores, no qual os caixas farão as recargas nos cartões dos usuários para acesso ao refeitório, ao passo que os gestores poderão:

- Ajustar recarga (mesmo formulário de fazer recarga, porém apenas o gestor pode inserir valores negativos);
- Listar a movimentação financeira geral ou por usuário;
- Vincular uma categoria a uma afiliação (para sincronia com LDAP-UFES);
- Listar vínculos entre categorias e afiliações;
- Cadastro de uma categoria;
- Listar categorias;
- Edição de uma categoria;
- Listar categorias por usuário;
- Cadastro de uma categoria para um usuário;

- Cadastro de uma tarifa;
- Listar tarifas;
- Edição de uma tarifa;
- Cadastro de uma refeição;
- Listar refeições;
- Edição de uma refeição;
- Cadastro de uma catraca;
- Listar catracas;
- Edição de uma catraca;
- Cadastro de um restaurante;
- Listar restaurantes;
- Edição de um restaurante;
- Exibir relatório com fluxo de pessoas;
- Exibir relatório com o fluxo de caixa.

3.5.1.1 Recarga

Na Figura 15, pode-se observar o código que gera o formulário para recarga dos saldos. A função da Figura 13, exibida anteriormente, é chamada para auto completar o CPF e assim evitar erros de digitação. A Figura 16 expõe a função de validação, que verifica se o identificador do usuário está correto. Verifica, também, se o usuário tem permissão para inserir valores negativos, caso este em que o gestor ajusta uma possível recarga errada. Por fim, verifica se o valor absoluto da recarga é menor que 9999,99 reais.

Ainda sobre a recarga, na Figura 17, podemos observar a função de *submit*, que insere os dados na tabela **extrato_de_recarga**. Na Figura 18, pode-se observar a tela do formulário descrito.

Figura 15 – Código do formulário da recarga.

```php
function adm_ru_ufes_recarga_form($form, &$form_state) {
  $form['uid'] = array(
    '#type' => 'textfield',
    '#title' => t('Usuario'),
    '#autocomplete_path' => 'ru_ufes/cartao/autocomplete',
    '#size' => 60,
    '#maxlength' => 128,
    '#required' => TRUE,
  );

  $form['valor'] = array(
    '#type' => 'textfield',
    '#title' => t('Valor'),
    '#size' => 60,
    '#maxlength' => 7,
    '#required' => TRUE,
  );

  $form['submit'] = array(
    '#type' => 'submit',
    '#value' => t('enviar'),
  );
  $form['submit']['#validate'] = array('adm_ru_ufes_recarga_form_validate');
  $form['submit']['#submit'][] = 'adm_ru_ufes_recarga_form_submit';

  return $form;
}

```

Fonte: Produção do próprio autor.

Figura 16 – Código de validação da recarga.

```
78  function adm_ru_ufes_recarga_form_validate($form, &$form_state) {
79    $iduser = $form_state['values']['uid'];
80    module_load_include('inc', 'adm_ru_ufes', 'adm_ru_ufes.db.api');
81    $current_db = adm_ru_ufes_db_open();
82    $results = db_select('cadastros', 'base');
83    $results->fields('base', array('iduser'));
84    $results->condition('base.iduser', $iduser);
85    adm_ru_ufes_db_close($current_db);
86    $count = $results->countQuery()->execute()->fetchField();
87    if ($count > 1) {
88      form_set_error('uid', t('Desculpa, varios usuários com mesmo registro.'));
89    }
90    elseif ($count == 0) {
91      form_set_error('uid', t('Desculpa, o %uid não foi encontrado.', array('%uid'
        => $iduser)));
92    }
93    $form_state['values']['valor'] = str_replace(',', '.', $form_state['values']['
        valor']);
94    if ((!is_numeric($form_state['values']['valor']) || $form_state['values']['
        valor'] < 0) && !user_access('ajuste de saldo')) {
95      form_set_error('valor', t('Digite um valor maior ou igual a zero.'));
96    }
97    elseif (($form_state['values']['valor'] > 9999.99) || ($form_state['values']['
        valor'] < -9999.99)) {
98      form_set_error('valor', t('Digite um valor menor ou igual a 9999,99.'));
99    }
100 }
```

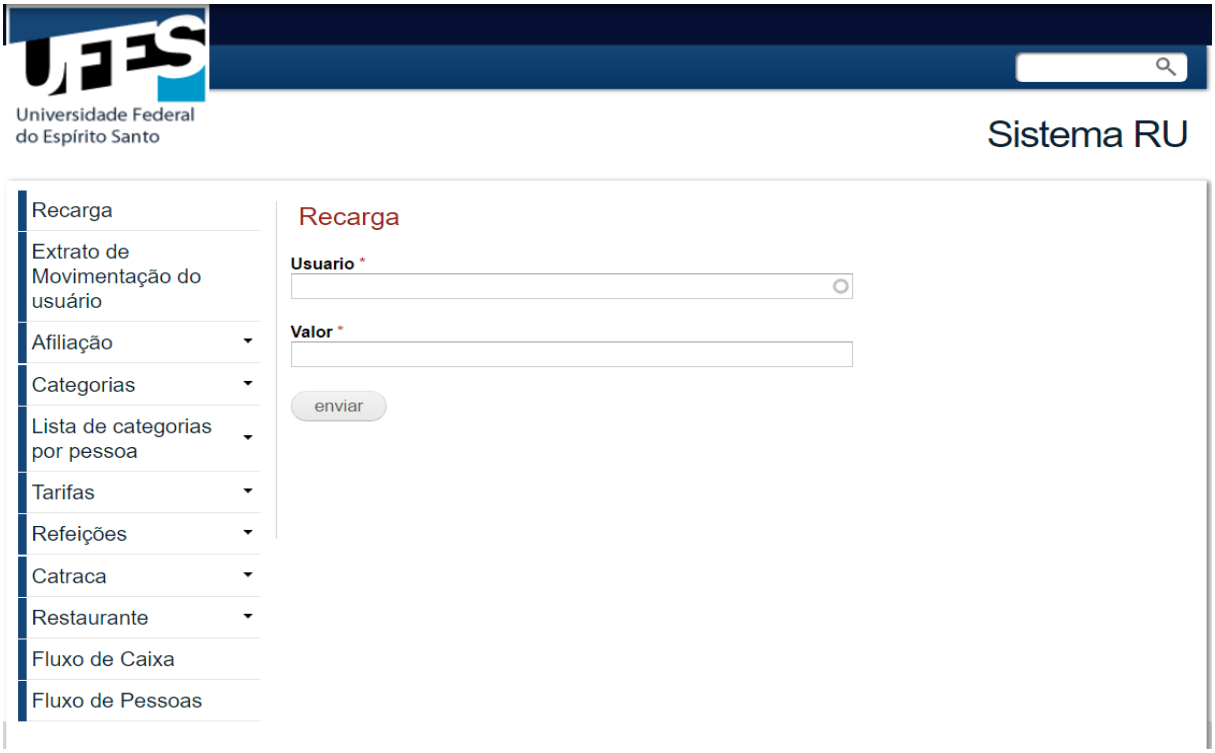
Fonte: Produção do próprio autor.

Figura 17 – Código do *submit* da recarga.

```
107 function adm_ru_ufes_recarga_form_submit($form, &$form_state) {
108   global $user;
109   $iduser = $form_state['values']['uid'];
110   module_load_include('inc', 'adm_ru_ufes', 'adm_ru_ufes.db.api');
111   $current_db = adm_ru_ufes_db_open();
112   $results = db_select('cadastros', 'base');
113   $results->fields('base', array('idcadastro'));
114   $results->condition('base.iduser', $iduser);
115   adm_ru_ufes_db_close($current_db);
116   $result = $results->execute()->fetchAssoc();
117   $dados = array(
118     'idcadastro' => $result['idcadastro'],
119     'idcaixa' => $user->name,
120     'valor' => $form_state['values']['valor'],
121     'data' => date('Y-m-d H:i:s'),
122   );
123
124   module_load_include('inc', 'adm_ru_ufes', 'adm_ru_ufes.db.api');
125   $current_db = adm_ru_ufes_db_open();
126   $results = db_insert('extrato_de_recarga');
127   $results->fields($dados);
128   $results->execute();
129   adm_ru_ufes_db_close($current_db);
130   drupal_set_message(t('Dados salvos com sucesso!'), 'status');
131 }
```

Fonte: Produção do próprio autor.

Figura 18 – Formulário da recarga.



Fonte: Produção do próprio autor.

3.5.1.2 Cadastro, Edição e Listagem

A seguir, na Figura 19, tem-se a criação das páginas referentes a tarifa. O primeiro item cria a página de cadastro de tarifa, que conterá o formulário mostrado na Figura 20. O segundo, cria a página de listagem. Já o terceiro e o quarto criam as páginas para editar e deletar as tarifas, respectivamente. Cada página possuiu a informação `type`. O `MENU_SUGGESTED_ITEM` é um item de menu que será sugerido ao administrador do Drupal, podendo ele adicionar ao site ou não. Já o `MENU_LOCAL_TASK` é um item de menu que descreve uma ação a ser executada em seu item pai. Um exemplo é o caminho "`ru_ufes/tarifa/%/editar`", que executa a tarefa "editar" em "`ru_ufes/tarifa/%`". A informação `Page callback` determina a função que criará a página da *Web* quando o usuário visita o caminho. Já a informação `page arguments` é o conjunto de argumentos a serem passados à função `Page callback`.

Figura 19 – Criação das páginas referentes a tarifa.

```php
function adm_ru_ufes_menu() {
  $items = array();
  $items['ru_ufes/tarifa/novo'] = array(
    'title' => 'Cadastro de Tarifa',
    'description' => 'Cadastro de Tarifa',
    'type' => MENU_SUGGESTED_ITEM,
    'page callback' => 'drupal_get_form',
    'page arguments' => array('adm_ru_ufes_tarifa_form'),
    'access arguments' => array('cadastro de dados'),
    'file' => 'adm_ru_ufes.form.inc',
  );
  $items['ru_ufes/tarifa/lista'] = array(
    'title' => 'Tarifas',
    'description' => 'Tarifas',
    'type' => MENU_SUGGESTED_ITEM,
    'page callback' => 'adm_ru_ufes_tarifalist_form',
    'access arguments' => array('cadastro de dados'),
    'file' => 'adm_ru_ufes.form.inc',
  );
  $items['ru_ufes/tarifa/%/editar'] = array(
    'title' => 'Edição de tarifa',
    'description' => 'Edição de tarifa',
    'type' => MENU_LOCAL_TASK,
    'page callback' => 'drupal_get_form',
    'page arguments' => array('adm_ru_ufes_tarifa_form', 2),
    'access arguments' => array('cadastro de dados'),
    'file' => 'adm_ru_ufes.form.inc',
  );
  $items['ru_ufes/tarifa/%/deletar'] = array(
    'title' => 'Deletar Tarifa',
    'description' => 'Deletar Tarifa',
    'type' => MENU_LOCAL_TASK,
    'page callback' => 'drupal_get_form',
    'page arguments' => array('adm_ru_ufes_tarifa_deletar_form', 2),
    'access arguments' => array('cadastro de dados'),
    'file' => 'adm_ru_ufes.form.inc',
  );
```

Fonte: Produção do próprio autor.

A função principal usada com formulários é `drupal_get_form`, que é usada para formulários apresentados interativamente a um usuário. A `drupal_get_form` lida com a recuperação, processamento e exibição de um formulário HTML renderizado para os módulos automaticamente.

Figura 20 – Código do formulário da tarifa.

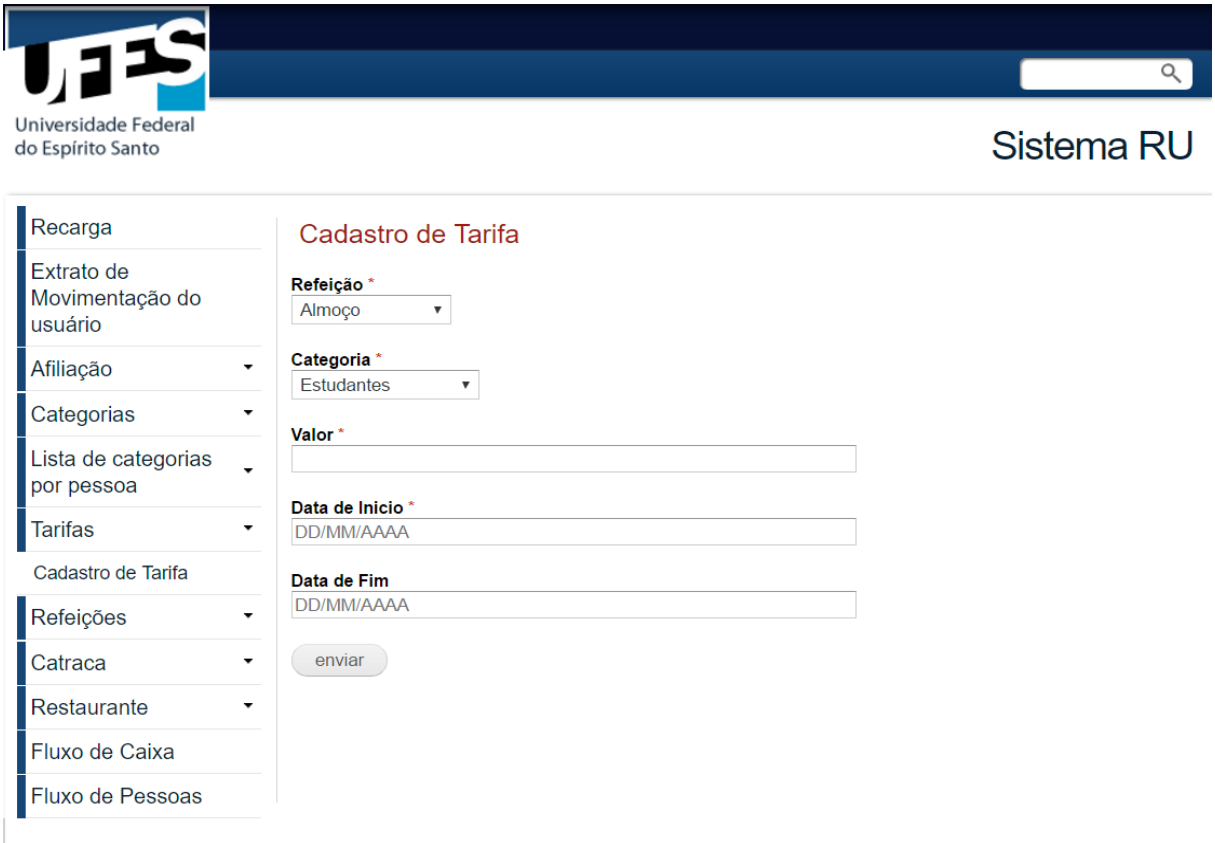
```
819  $form['categoria'] = array(
820    '#type' => 'select',
821    '#title' => t('Categoria'),
822    '#options' => $matches,
823    '#default_value' => $tarifas['idcategoria'],
824    '#required' => TRUE,
825    '#disabled' => adm_ru_ufes_tarifa_used($idtarifa),
826  );
827
828  $form['valor'] = array(
829    '#type' => 'textfield',
830    '#title' => t('Valor'),
831    '#size' => 60,
832    '#maxlength' => 7,
833    '#required' => TRUE,
834    '#default_value' => $tarifas['valor'],
835    '#disabled' => adm_ru_ufes_tarifa_used($idtarifa),
836  );
837  $form['datainicio'] = array(
838    '#type' => 'textfield',
839    '#title' => t('Data de Inicio'),
840    '#size' => 60,
841    '#maxlength' => 128,
842    '#required' => TRUE,
843    '#attributes' => array('placeholder' => t('DD/MM/AAAA')),
844    '#default_value' => empty($tarifas['datainicio']) ? NULL : format_date(
       strtotime($tarifas['datainicio']), 'custom', 'd/m/Y'),
845    '#disabled' => adm_ru_ufes_tarifa_used($idtarifa),
846  );
847  $form['datafim'] = array(
848    '#type' => 'textfield',
849    '#title' => t('Data de Fim'),
850    '#size' => 60,
851    '#maxlength' => 128,
852    '#attributes' => array('placeholder' => t('DD/MM/AAAA')),
853    '#default_value' => empty($tarifas['datafim']) ? NULL : format_date(strtotime
       ($tarifas['datafim']), 'custom', 'd/m/Y'),
854  );
855  $form['submit'] = array(
856    '#type' => 'submit',
857    '#value' => t('enviar'),
858  );
859
860  $form['submit']['#validate'] = array('adm_ru_ufes_tarifa_form_validate');
861  $form['submit']['#submit'][] = 'adm_ru_ufes_tarifa_form_submit';
862
863  return $form;
864 }
```

Fonte: Produção do próprio autor.

O Drupal usa essas funções para obter consistência em seu processamento e apresentação de formulários, simplificando o código e reduzindo a quantidade de HTML que deve ser gerada pelos módulos.

Na Figura 21 é mostrada a tela com o formulário para a criação de uma nova tarifa. A página de edição de uma tarifa é carregada utilizando a mesma função que a página de cadastrar uma nova tarifa, dependendo do parâmetro passado. Alguns itens, na página de edição, poderão estar bloqueados. Isto ocorre devido aos dados bloqueados estarem vinculados a outras informações. Por exemplo, se uma tarifa cadastrada já tiver sido utilizada por algum usuário, então ela não poderá ser alterada. A exclusão ocorre da mesma forma.

Figura 21 – Formulário da tarifa.



Fonte: Produção do próprio autor.

Na Figura 22 é gerada uma tabela com a listagem de todas as tarifas salvas no banco e na Figura 23 podemos ver o resultado.

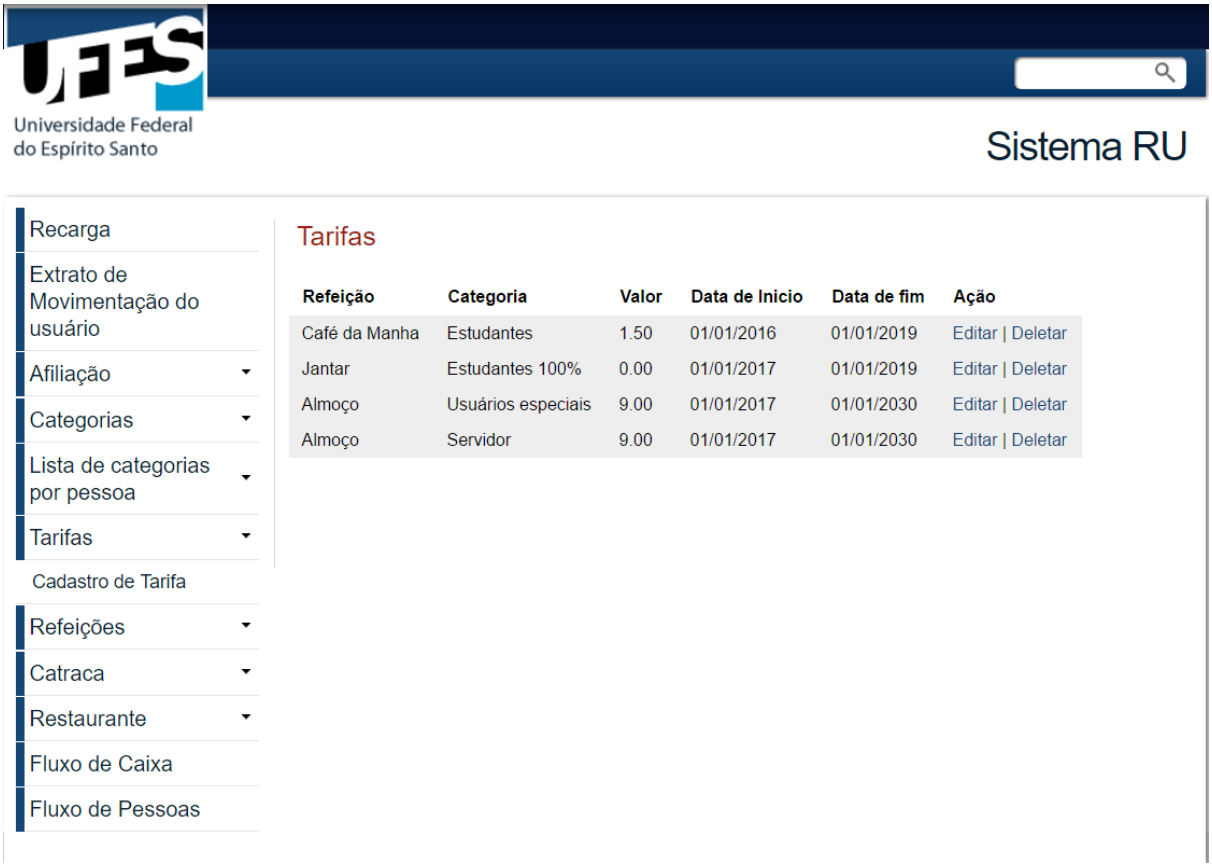
Figura 22 – Código para listagem das tarifas.

```
716 function adm_ru_ufes_tarifalist_form() {
717   $header = array(
718     'Refeição',
719     'Categoria',
720     'Valor',
721     'Data de Inicio',
722     'Data de fim',
723     'Ação',
724   );
725   $rows = array();
726
727   module_load_include('inc', 'adm_ru_ufes', 'adm_ru_ufes.db.api');
728   $current_db = adm_ru_ufes_db_open();
729   $results = db_select('tarifas', 'base');
730   $results->fields('base');
731   adm_ru_ufes_db_close($current_db);
732   $result = $results->execute();
733   foreach ($result as $tarifa) {
734     $current_db = adm_ru_ufes_db_open();
735     $result_cat = db_select('categorias', 'base');
736     $result_cat->fields('base', array('nome'));
737     $result_cat->condition('base.idcategoria', $tarifa->idcategoria);
738     $result_ref = db_select('refeicao', 'base');
739     $result_ref->fields('base', array('nome'));
740     $result_ref->condition('base.idrefeicao', $tarifa->idrefeicao);
741     adm_ru_ufes_db_close($current_db);
742
743     $rows[] = array(
744       check_plain($result_ref->execute()->fetchField()),
745       check_plain($result_cat->execute()->fetchField()),
746       check_plain($tarifa->valor),
747       format_date(strtotime(check_plain($tarifa->datainicio)), 'custom', 'd/m/Y')
            ,
748       format_date(strtotime(check_plain($tarifa->datafim)), 'custom', 'd/m/Y'),
749       l(t('Editar'), 'ru_ufes/tarifa/' . $tarifa->idtarifa . '/editar') . ' | ' .
            l(t('Deletar'), 'ru_ufes/tarifa/' . $tarifa->idtarifa . '/deletar'),
750     );
751   }
752   return theme('table', array('header' => $header, 'rows' => $rows));
753 }
```

Fonte: Produção do próprio autor.

Na figura anterior, pode-se observar a função `theme`. Todos os pedidos de saída temática devem passar por esta função. Ele examina o pedido e encaminha-o para a função ou modelo apropriado do tema. Tal função gera uma saída temática dependendo do parâmetro passado, que nesse caso foi `table`. Os outros parâmetros, para criar a listagem com as tarifas, são o cabeçalho e as linhas.

Figura 23 – Tela com a listagem das tarifas.



Fonte: Produção do próprio autor.

3.5.1.3 APIs

Foram implementadas, ainda, duas APIs: uma para possibilitar as catracas realizarem o decremento do saldo após o uso, e a outra para que o Módulo de Consulta de Saldo possa ter acesso ao Banco Principal, como mostra a Figura 24.

A função `adm_ru_ufes_saldo_api` recebe uma *hash* e um identificador do usuário. A *hash* é utilizada para identificar que a requisição partiu do Módulo de Consulta de Saldo. Por conseguinte, a função faz a consulta no banco e retorna o saldo do usuário. Caso a *hash* enviada esteja errada ou vazia, será devolvida uma mensagem de falha. Se a contagem de resultados da consulta ao banco de dados retornar qualquer valor diferente de um, a seguinte mensagem será enviada: “Usuário não encontrado”.

Figura 24 – API para que o Módulo de Consulta de Saldo acesse o Banco Principal.

```
450 function adm_ru_ufes_saldo_api($hash = NULL, $uid = NULL) {
451   $msgs = array(
452     0 => t('OK!'),
453     1 => t('Falha!'),
454     2 => t('Usuario não encontrado.'),
455   );
456   $result = array();
457   if (empty($hash) || empty($uid) || !in_array($hash, array('
        1477436f8bfd9c9550d8eebf2ea308896c53b099'))) {
458     $result['code'] = 1;
459   }
460   else {
461     module_load_include('inc', 'adm_ru_ufes', 'adm_ru_ufes.db.api');
462     $current_db = adm_ru_ufes_db_open();
463     $results = db_select('cadastros', 'base');
464     $results->fields('base', array('idcadastro', 'saldo'));
465     $or = db_or();
466     $or->condition('base.iduser', $uid);
467     $or->condition('base.idcartao', $uid);
468     $results->condition($or);
469     adm_ru_ufes_db_close($current_db);
470     $count = $results->countQuery()->execute()->fetchField();
471     if ($count != 1) {
472       $result['code'] = 2;
473     }
474     else {
475       $idcadastro = $results->execute()->fetchField(0);
476       $current_db = adm_ru_ufes_db_open();
477       $results = db_select('cadastros', 'base');
478       $results->fields('base', array('saldo'));
479       $results->condition('idcadastro', $idcadastro);
480       adm_ru_ufes_db_close($current_db);
481       $saldo = $results->execute()->fetchField();
482       $result['code'] = 0;
483       $result['saldo'] = $saldo;
484     }
485   }
486   $result['msg'] = $msgs[$result['code']];
487   drupal_json_output($result);
488 }
```

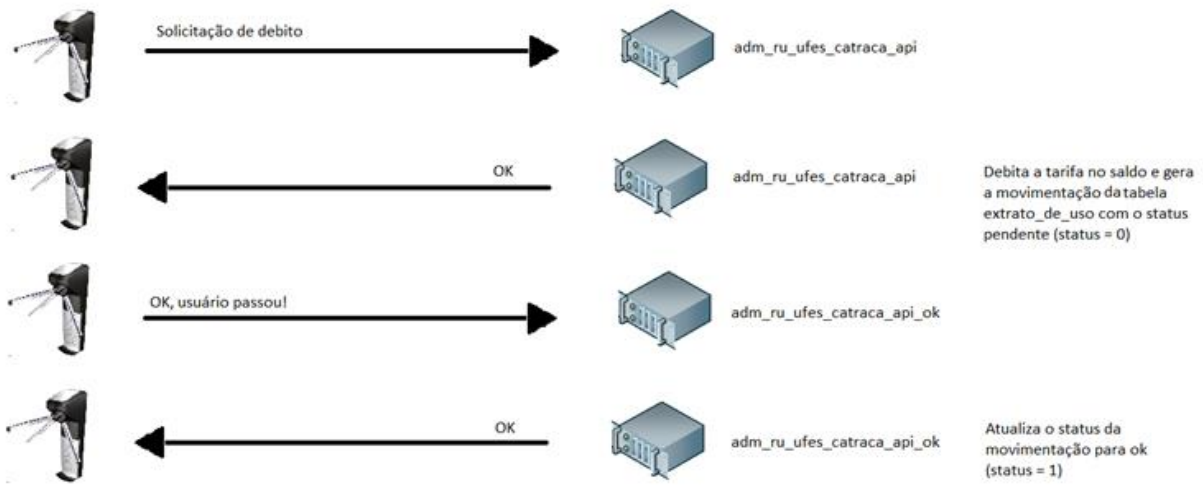
Fonte: Produção do próprio autor.

Na Figura 26 e 27, tem-se a API para possibilitar as catracas realizarem o decremento do saldo. Esta é bastante semelhante a anterior. Porém, a informação da *hash*, neste caso, é obtida através da consulta a tabela catracas. Tal *hash* é informada no momento em que se cadastra uma nova catraca. Se o usuário não tiver saldo suficiente, ou não tiver uma tarifa válida, ou, ainda, a data enviada pela catraca for anterior à do sistema, uma mensagem de erro será transmitida conforme a variável `$msgs`. No caso em que a data enviada seja nula, será salva a data atual do servidor

na tabela **extrato_de_uso**. O parâmetro *forced* desconsidera a verificação de saldo e debita o valor da tarifa, mesmo que a pessoa não possua saldo suficiente. Neste caso, o usuário pode ficar com saldo negativo.

A função `adm_ru_ufes_catraca_api`, ao receber a solicitação de débito, autoriza a entrada do usuário no refeitório e faz o débito no banco. Porém, o status, na tabela **extrato_de_uso**, fica pendente `(status=0)`. A catraca, após a passagem do usuário, envia uma mensagem de confirmação. Assim, a função `adm_ru_ufes_catraca_api_ok`, na Figura 28, altera o status de pendente para ok `(status=1)`. Desta forma, o usuário que tiver seu cartão lido, mas que não adentre no refeitório, não será cobrado. Após a alteração do status, o saldo, na tabela **cadastros**, é atualizado através de uma *trigger* no banco de dados. A Figura 25 ilustra a comunicação.

Figura 25 – Diagrama de sequência.



Fonte: Produção do próprio autor.

Figura 26 – API que possibilita as catracas realizarem o decremento do saldo (Parte 1).

```php
function adm_ru_ufes_catraca_api($hash = NULL, $uid = NULL, $timestamp = NULL, $forced = FALSE) {
  $msgs = array(
    0 => t('OK!'),
    1 => t('Falha!'),
    2 => t('Saldo insuficiente.'),
    3 => t('Usuario não encontrado.'),
    4 => t('Sem tarifa valida.'),
    5 => t('Data inválida.'),
  );
  $result = array();
  if (empty($hash) || empty($uid)) {
    $result['code'] = 1;
  }
  elseif (!empty($timestamp) && $timestamp > time()) {
    $result['code'] = 5;
  }
  else {
    if (empty($timestamp)) {
      $timestamp = time();
    }
    module_load_include('inc', 'adm_ru_ufes', 'adm_ru_ufes.db.api');
    $current_db = adm_ru_ufes_db_open();
    $results = db_select('catracas', 'base');
    $results->fields('base', array('idcatracas'));
    $results->condition('base.hash', $hash);
    $results->condition('base.status', 1);
    adm_ru_ufes_db_close($current_db);
    $count = $results->countQuery()->execute()->fetchField();
    if ($count != 1) {
      $result['code'] = 1;
    }
    else {
      $idcatracas = $results->execute()->fetchField();
      module_load_include('inc', 'adm_ru_ufes', 'adm_ru_ufes.db.api');
      $current_db = adm_ru_ufes_db_open();
      $results = db_select('extrato_de_uso', 'base');
      $results->fields('base', array('idextrato_de_uso'));
      $results->condition('base.idcatracas', $idcatracas);
      $results->condition('base.status', 0);
      adm_ru_ufes_db_close($current_db);
      $resultado = $results->execute();
      foreach ($resultado as $movimentacao) {
        module_load_include('inc', 'adm_ru_ufes', 'adm_ru_ufes.db.api');
        $current_db = adm_ru_ufes_db_open();
        $results = db_update('extrato_de_uso');
        $results->fields(array('status' => 2));
        $results->condition('idextrato_de_uso', $movimentacao->idextrato_de_uso);
        adm_ru_ufes_db_close($current_db);
        $results->execute();
      }
      module_load_include('inc', 'adm_ru_ufes', 'adm_ru_ufes.db.api');
      $current_db = adm_ru_ufes_db_open();
      $results = db_select('cadastros', 'base');
      $results->fields('base', array('idcadastro', 'saldo'));
      $or = db_or();
      $or->condition('base.iduser', $uid);
      $or->condition('base.idcartao', $uid);
      $results->condition($or);
      adm_ru_ufes_db_close($current_db);
      $count = $results->countQuery()->execute()->fetchField();
      if ($count != 1) {
        $result['code'] = 3;
      }
```

Fonte: Produção do próprio autor.

Figura 27 – API que possibilita as catracas realizarem o decremento do saldo (Parte 2).

```
346       else {
347         $idcadastro = $results->execute()->fetchField(0);
348         $saldo = $results->execute()->fetchField(1);
349         module_load_include('inc', 'adm_ru_ufes', 'adm_ru_ufes.db.api');
350         $current_db = adm_ru_ufes_db_open();
351         $results = db_select('tarifas', 't');
352         $results->join('categorias', 'c', 't.idcategoria = c.idcategoria');
353         $results->join('cadastros_categorias', 'cc', 'cc.idcategoria =
              t.idcategoria');
354         $results->join('refeicao', 'r', 'r.idrefeicao = t.idrefeicao');
355         $results->condition('cc.idcadastro', $idcadastro);
356         $results->condition('cc.datainicio', date('Y-m-d', $timestamp), '<=');
357         $or_1 = db_or();
358         $or_1->condition('cc.datafim', date('Y-m-d', $timestamp), '>=');
359         $or_1->isNull('cc.datafim');
360         $results->condition($or_1);
361         $results->condition('t.datainicio', date('Y-m-d', $timestamp), '<=');
362         $or_2 = db_or();
363         $or_2->condition('t.datafim', date('Y-m-d', $timestamp), '>=');
364         $or_2->isNull('t.datafim');
365         $results->condition($or_2);
366         $results->condition('r.horainicio', date('H:i:s', $timestamp), '<=');
367         $results->condition('r.horafim', date('H:i:s', $timestamp), '>=');
368         $results->orderBy('c.peso', 'ASC');
369         $results->orderBy('t.valor', 'ASC');
370         $results->range(0, 1);
371         $results->fields('t', array('idtarifa', 'valor'));
372         adm_ru_ufes_db_close($current_db);
373         $count = $results->countQuery()->execute()->fetchField();
374         if ($count != 1) {
375           $result['code'] = 4;
376         }
377         else {
378           $idtarifa = $results->execute()->fetchField(0);
379           $valor = $results->execute()->fetchField(1);
380           if ($saldo >= $valor || $forced) {
381             $dados = array(
382               'data' => date('Y-m-d H:i:s', $timestamp),
383               'idtarifa' => $idtarifa,
384               'idcatracas' => $idcatracas,
385               'idcadastro' => $idcadastro,
386               'status' => 0,
387             );
388             $current_db = adm_ru_ufes_db_open();
389             $results = db_insert('extrato_de_uso');
390             $results->fields($dados);
391             $results->execute();
392             adm_ru_ufes_db_close($current_db);
393             $result['code'] = 0;
394           }
395           else {
396             $result['code'] = 2;
397           }
398         }
399       }
400     }
401   }
402
403   $result['msg'] = $msgs[$result['code']];
404   drupal_json_output($result);
405 }
```

Fonte: Produção do próprio autor.

Figura 28 – Função de resposta que altera o status da transação.

```
function adm_ru_ufes_catraca_api_ok($hash = NULL) {
  $msgs = array(
    0 => t('OK!'),
    1 => t('Falha!'),
  );
  $result = array();
  if (empty($hash)) {
    $result['code'] = 1;
  }
  else {
    module_load_include('inc', 'adm_ru_ufes', 'adm_ru_ufes.db.api');
    $current_db = adm_ru_ufes_db_open();
    $results = db_select('catracas', 'base');
    $results->fields('base', array('idcatracas'));
    $results->condition('base.hash', $hash);
    adm_ru_ufes_db_close($current_db);
    $count = $results->countQuery()->execute()->fetchField();
    if ($count != 1) {
      $result['code'] = 1;
    }
    else {
      $idcatracas = $results->execute()->fetchField();
      module_load_include('inc', 'adm_ru_ufes', 'adm_ru_ufes.db.api');
      $current_db = adm_ru_ufes_db_open();
      $results = db_update('extrato_de_uso');
      $results->fields(array('status' => 1));
      $results->condition('idcatracas', $idcatracas);
      $results->condition('status', 0);
      adm_ru_ufes_db_close($current_db);
      $results->execute();
      $result['code'] = 0;
    }
  }
  $result['msg'] = $msgs[$result['code']];
  drupal_json_output($result);
}
```

Fonte: Produção do próprio autor.

3.5.1.4 Permissões

A seguir, na Figura 29, são apresentadas as permissões do módulo. Os perfis de usuários, por exemplo, caixa e gestor, são montados pelos administradores do Drupal, a partir destas permissões.

Figura 29 – Permissões do Módulo de Gerência Financeira.

```
493 function adm_ru_ufes_permission() {
494
495   return array(
496     'administrar ru ufes' => array(
497       'title' => t('Configurar modulo do RU'),
498       'description' => t('Perform administration tasks for my module.'),
499     ),
500     'fazer recarga ru ufes' => array(
501       'title' => t('Fazer recarga'),
502       'description' => t('.'),
503     ),
504     'cadastro de dados' => array(
505       'title' => t('Cadastro de dados'),
506       'description' => t('.'),
507     ),
508     'fluxodecaixa' => array(
509       'title' => t('Gerar relatório de fluxo de caixa'),
510       'description' => t('.'),
511     ),
512     'fluxodepessoas' => array(
513       'title' => t('Gerar relatório de fluxo de pessoas'),
514       'description' => t('.'),
515     ),
516     'associacao de categoria' => array(
517       'title' => t('Associação de categoria'),
518       'description' => t('.'),
519     ),
520     'ajuste de saldo' => array(
521       'title' => t('Ajuste de saldo'),
522       'description' => t('.'),
523     ),
524     'cadastro de afiliacao' => array(
525       'title' => t('Cadastro de Afiliação'),
526       'description' => t('.'),
527     ),
528   );
529 }
```

Fonte: Produção do próprio autor.

3.5.1.5 Relatórios

A Figura 30 e 31 contém o código da página Extrato de Movimentação do Usuário. Tal código faz uso da *view* mostrada na Figura 10 e tem como objetivo gerar um relatório (Figura 32) com toda a movimentação financeira, podendo ser filtrado pelo campo Usuário.

Este Módulo ainda é capaz de gerar outros dois relatórios: fluxo de caixa e fluxo de pessoas, mostrado na Figura 33, que funcionam da mesma forma que o Extrato de Movimentação do Usuário.

Figura 30 – Código da página Extrato de Movimentação do Usuário (Parte 1).

```php
function adm_ru_ufes_movimentacao_form($form, &$form_state) {
  $form['filter'] = array(
    '#type' => 'fieldset',
    '#collapsible' => TRUE,
    '#collapsed' => TRUE,
    '#title' => t('Filter option'),
  );
  $form['filter']['uid'] = array(
    '#type' => 'textfield',
    '#title' => t('Usuario'),
    '#autocomplete_path' => 'ru_ufes/cartao/autocomplete',
    '#size' => 60,
    '#maxlength' => 128,
    '#required' => TRUE,
  );

  $form['filter']['submit'] = array(
    '#type' => 'submit',
    '#value' => t('Filter'),
  );

  $header = array(
    array('data' => t('Nome'), 'field' => 'iduser'),
    array('data' => t('Data'), 'field' => 'data', 'sort' => 'desc'),
    array('data' => t('Valor'), 'field' => 'valor'),
    array('data' => t('Operador'), 'field' => 'operador'),
    array('data' => t('Situação'), 'field' => 'status'),
  );

  $status = array(
    0 => t('pendente'),
    1 => t('confirmada'),
    2 => t('estornada'),
  );

  module_load_include('inc', 'adm_ru_ufes', 'adm_ru_ufes.db.api');
  $current_db = adm_ru_ufes_db_open();
  $results = db_select('movimentacao', 'base');
  $results->fields('base');
  if (isset($form_state['storage']['filters']['uid'])) {
    $results->condition('iduser', $form_state['storage']['filters']['uid']);
  }
  $results = $results->extend('TableSort')->orderByHeader($header);
  $results = $results->extend('PagerDefault')->limit(20);
  adm_ru_ufes_db_close($current_db);
  $result = $results->execute();
  $rows = array();
  foreach ($result as $movimentacao) {
    $rows[] = array(
      check_plain($movimentacao->iduser),
      format_date(strtotime(check_plain($movimentacao->data)), 'custom', 'd/m/Y
        H:i:s'),
      check_plain($movimentacao->valor),
      check_plain($movimentacao->operador),
      $status[check_plain($movimentacao->status)],
    );
  }
```

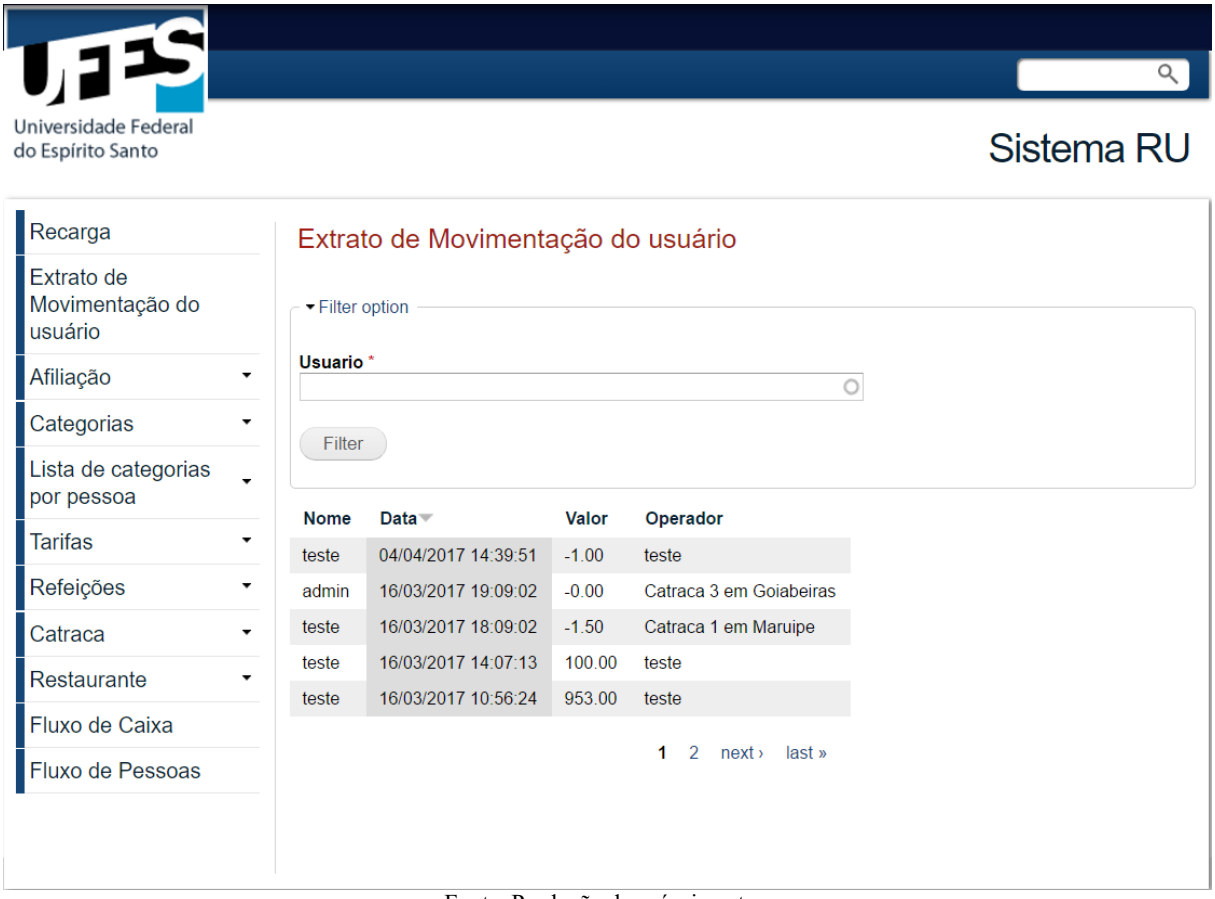
Fonte: Produção do próprio autor.

Figura 31 – Código da página Extrato de Movimentação do Usuário (Parte 2).

```
198   $form['table'] = array(
199     '#theme' => 'table',
200     '#header' => $header,
201     '#rows' => $rows,
202     '#empty' => t('Sem dados para exibir.'),
203   );
204   $form['pager'] = array(
205     '#markup' => theme('pager'),
206   );
207
208   if (isset($form_state['storage']['filters']['uid'])) {
209     module_load_include('inc', 'adm_ru_ufes', 'adm_ru_ufes.db.api');
210     $current_db = adm_ru_ufes_db_open();
211     $results = db_select('cadastros', 'base');
212     $results->fields('base', array('saldo'));
213     $results->condition('iduser', $form_state['storage']['filters']['uid']);
214     adm_ru_ufes_db_close($current_db);
215     $saldo = $results->execute()->fetchField();
216     $form['saldo'] = array(
217       '#markup' => t('Saldo para usuário %uid: %saldo', array('%saldo' => $saldo
          , '%uid' => $form_state['storage']['filters']['uid'])),
218     );
219   }
220
221   $form['submit']['#validate'] = array('adm_ru_ufes_movimentacao_form_validate');
222   $form['submit']['#submit'][] = 'adm_ru_ufes_movimentacao_form_submit';
223
224   return $form;
225 }
226
```

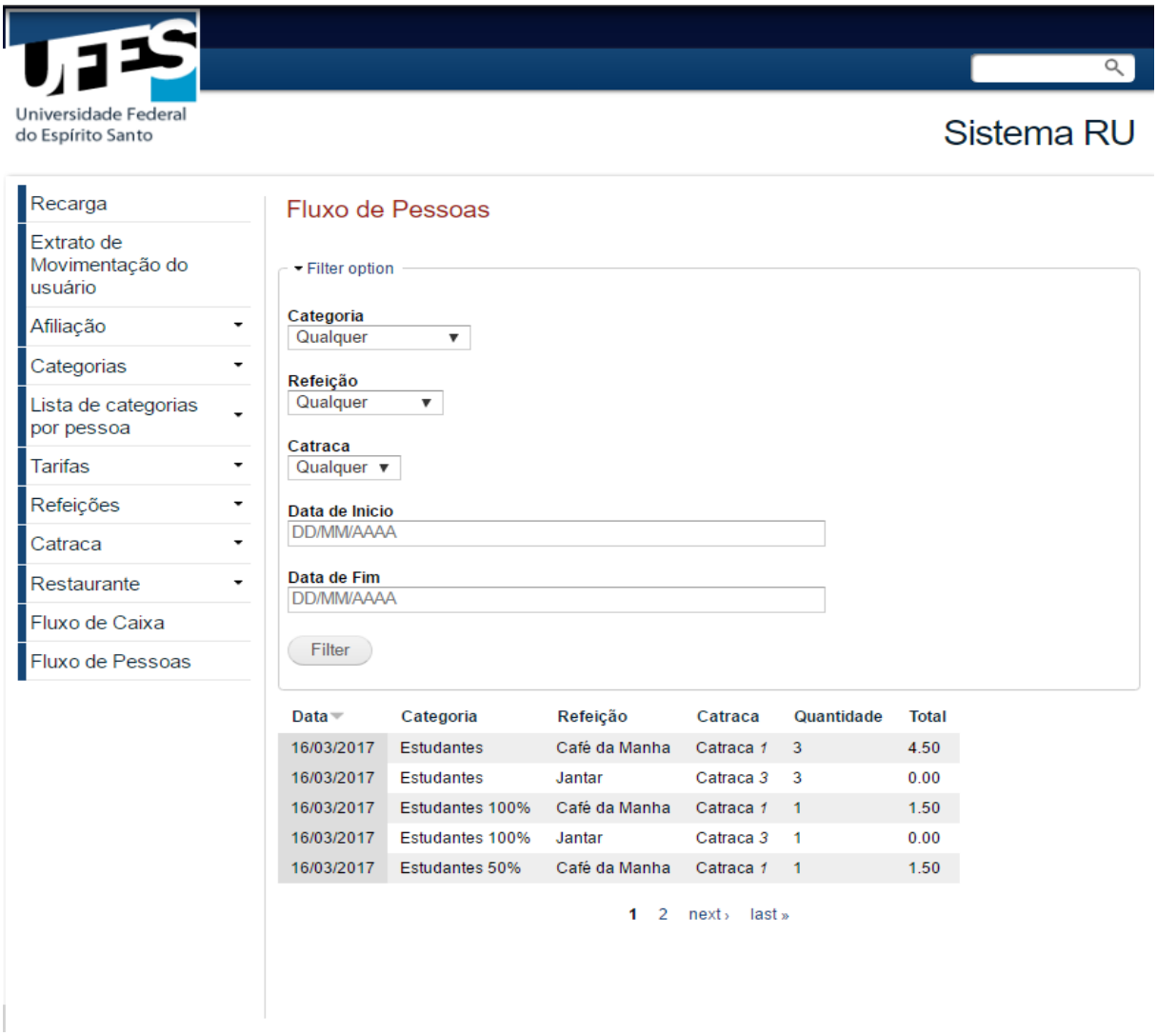
Fonte: Produção do próprio autor.

Figura 32 – Tela da página Extrato de Movimentação do Usuário.



Fonte: Produção do próprio autor.

Figura 33– Tela do relatório Fluxo de Pessoas



Fonte: Produção do próprio autor.

3.5.2 Módulo Consulta de Saldo

Este módulo funciona em conjunto com a API da Figura 24, e faz a requisição do saldo do usuário através da função `ru_ufes_block_view`, na Figura 34. Ficará integrado ao site oficial do RU, que foi construído em Drupal, como mostram as Figuras 36 e 37.

Figura 34 – Função para a requisição do saldo do usuário através da API.

```
11  function ru_ufes_block_info() {
12    // Declaração do bloco de saldo.
13    $blocks['saldo'] = array(
14      'info' => t('Saldo do cartão'),
15      'cache' => DRUPAL_NO_CACHE,
16      // DRUPAL_CACHE_PER_USER opção de cache por usuario.
17    );
18    return $blocks;
19  }
20
21  /**
22   * Implements hook_block_view().
23   */
24  function ru_ufes_block_view($delta = '') {
25    // Verifica permissão no bloco e e seguida exibe o conteudo do Bloco de saldo.
26    if (!user_access('acessar saldo ru ufes')) {
27      return array();
28    }
29
30    global $user;
31    $request = drupal_http_request('http://devsistemaru.ufes.br/adm/ru_ufes/api/
        saldo/1477436f8bfd9c9550d8eebf2ea308896c53b099/'.$user->name);
32    $json_response = drupal_json_decode($request->data);
33    $block = array();
34
35    switch ($delta) {
36      case 'saldo':
37        $block['subject'] = t('Saldo do cartão');
38        if ($json_response['code'] == 0) {
39          $block['content'] = t('Seu saldo é: %saldo', array('%saldo' => $
              json_response['saldo']));
40        }
41        elseif ($json_response['code'] == 2) {
42          $block['content'] = t('Cadastro não encontrado.');
43        }
44        else {
45          $block['content'] = t('Falha na consulta.');
46        }
47        break;
48    }
49    return $block;
50  }
```

Fonte: Produção do próprio autor.

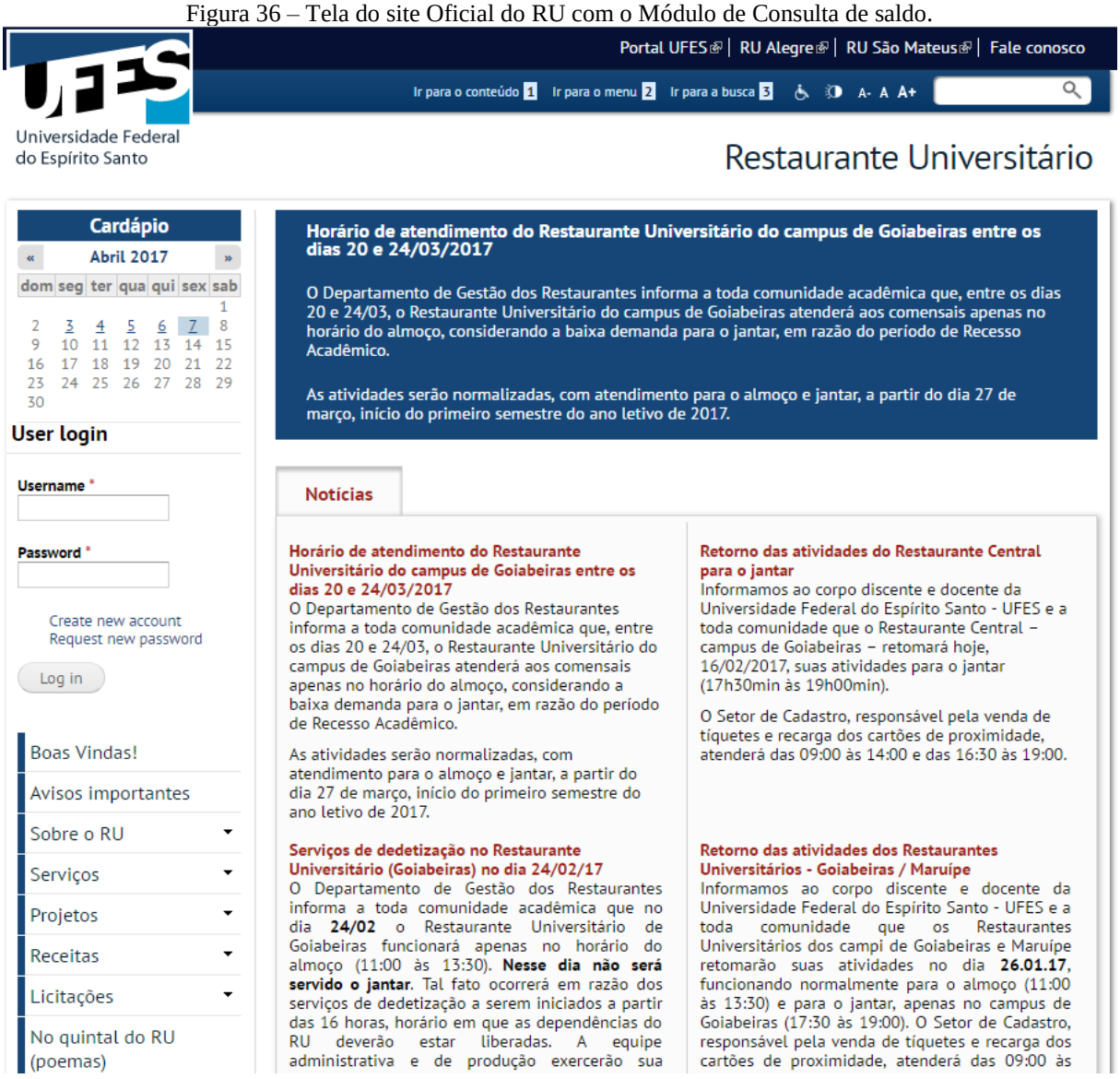
Semelhante ao módulo anterior, porém mais simples, tem-se as permissões deste módulo, mostradas na Figura 35.

Figura 35 – Permissões do Módulo de Consulta de Saldo.

```
85  function ru_ufes_permission() {
86
87    return array(
88      'administrar ru ufes' => array(
89        'title' => t('Configurar modulo do RU'),
90        'description' => t('Perform administration tasks for my module.'),
91      ),
92      'acessar saldo ru ufes' => array(
93        'title' => t('acessar saldo do RU'),
94        'description' => t('.'),
95      ),
96    );
97  }
```

Fonte: Produção do próprio autor.

Figura 36 – Tela do site Oficial do RU com o Módulo de Consulta de saldo.



Fonte: Produção do próprio autor.

Figura 37 – Site oficial do RU exibindo o saldo.

Fonte: Produção do próprio autor.

3.5.3 Módulo de Sincronia com LDAP-UFES

Este módulo é responsável por sincronizar os dados entre o LDAP-UFES e o Banco Principal. Ao fazê-lo, obtém-se o vínculo mais atual com a universidade, isto é, a categoria que o indivíduo pertence, e assim é cobrado o valor da tarifa correta.

Os seguintes campos serão obtidos do LDAP-UFES:

`uid`**:** Define o identificador único do usuário no LDAP. Este campo é definido como sendo uma composição entre o nome e sobrenome. Dado um nome: Renan Carvalho Almeida Dias Silva Tavares Santos, tenta-se gerar os seguintes uids, em ordem: renan.santos renan.c.santos, renan.a.santos, renan.d.santos, renan.s.santos, renan.t.santos, renan.ca.santos, renan.cad.santos,

renan.santos.03 e renan.santos.05759666703. As duas últimas tentativas usam o número do CPF da pessoa.

`brPersonCPF`: Exibe o CPF da pessoa, apenas números.

`EduPersonAffiliation`: Define a afiliação da pessoa. Tal afiliação caracteriza o papel da pessoa dentro da Ufes. Neste campo, aceita-se apenas um número por entrada, sendo que o número define qual é a afiliação. A relação entre números e afiliações segue abaixo:
1) Docente;
2) Técnico-Administrativo;
3) Instrutor;
4) Aluno de Graduação;
5) Aluno de Pós-Graduação;
6) Aluno de Intercâmbio;
7) Aluno Mobilidade;
8) Visitante;
9) Terceirizado;
10) Residente de Medicina;
11) Aluno de Graduação Especial;
12) Professor Colab/Vis. da Pós-Graduação;
13) Secretário da Pós-Graduação;
15) Residente Multi-Funcional;
16) Tutor EAD;
17) Aluno de Graduação EAD.

Os dados constantes no campo `uid` do LDAP são sincronizados com `iduser` da tabela cadastros. Assim como o `cpf` é sincronizado com `brPersonCPF`, e a afiliação com `EduPersonAffiliation`.

É necessário que a consulta ao LDAP retorne apenas usuários ativos. Desta forma, o filtro da consulta deve conter uma condição com os seguintes campos:

`nsAccountKeep`: Consiste em um valor booleano que define se a conta do usuário deve permanecer inalterada (forçado ativo), mesmo que este usuário não possua mais vínculo com a universidade.
`nsAccountLock`: Trata-se de um campo que define se um usuário deve ser bloqueado ou não. Em usuários ativos, este campo é sempre *false*.

A sincronia é feita da seguinte forma:

- É feita uma consulta, requisitando todos os usuários ativos, no LDAP.
- Em seguida, é feita uma consulta no Banco Principal, requisitando todos os usuários.
- Depois, é feita a diferença, a fim de descobrir quais usuários estão no banco principal e não estão no LDAP.
- Com a consulta anterior, é necessário verificar se o CPF já está cadastrado no LDAP e atualizar o `uid` do usuário. Este caso é de uma possível mudança de nome.
- Com as duas últimas operações, é possível expirar as categorias dos usuários que sobram. Estes não estão mais ativos.
- O próximo passo é descobrir todos os usuários que estão no LDAP e não estão no Banco Principal.
- Por fim, cria-se usuários da última operação e sincroniza a categoria de todos os usuários.

O *script* descrito anteriormente é executado de forma periódica através da função de rotina do Drupal.

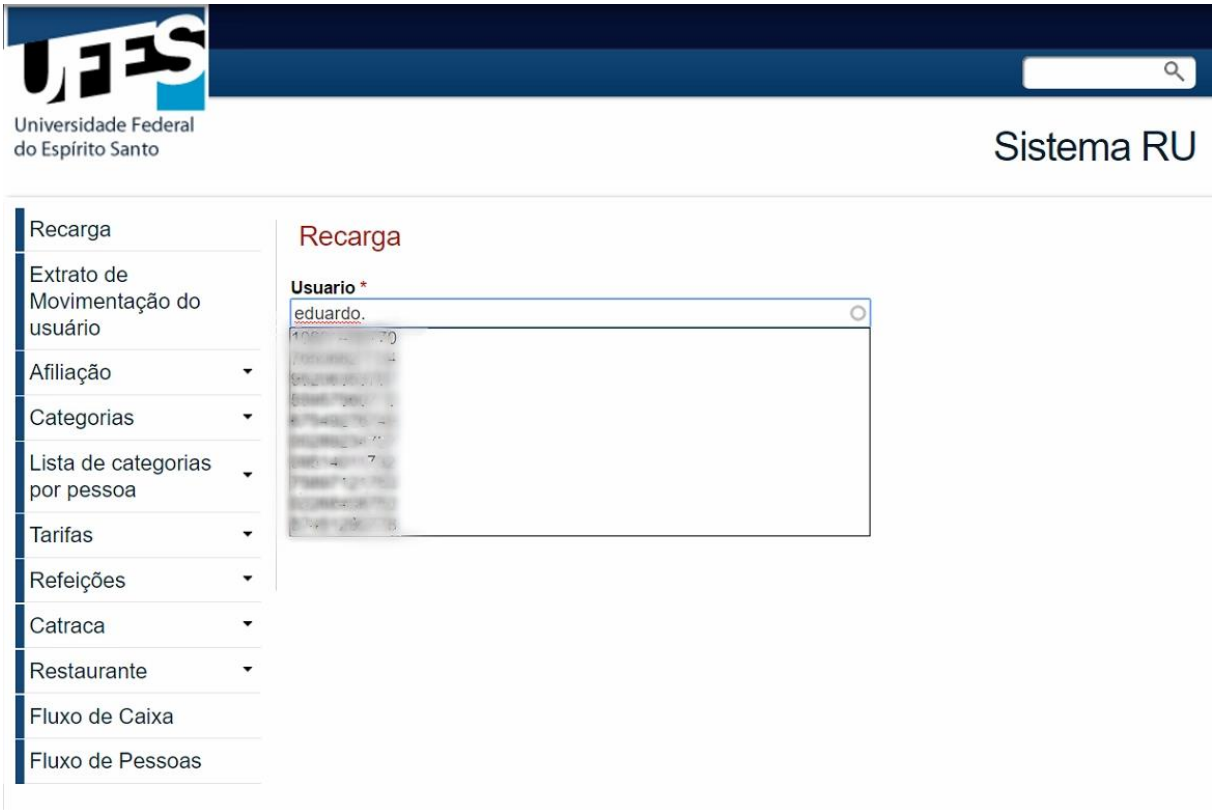
4 VALIDAÇÃO

Para validar o sistema foram feitos alguns testes com o objetivo de verificar se os requisitos para um determinado uso foram cumpridos. Está é uma atividade essencial para se garantir a qualidade do software.

4.1 Simulação de uso Módulo de Sincronia com LDAP-UFES

Foi possível verificar que a sincronia do Banco Principal com o LDAP-UFES aconteceu de forma esperada, pois os dados na tabela cadastros foram atualizados. Isto pode ser observado tanto em uma consulta direta ao banco quanto na tentativa de simular uma recarga, conforme Figura 38.

Figura 38 – Tela de recarga após sincronia.



Fonte: Produção do próprio autor.

,

4.2 Simulação de uso do Módulo de Gerência de Dados

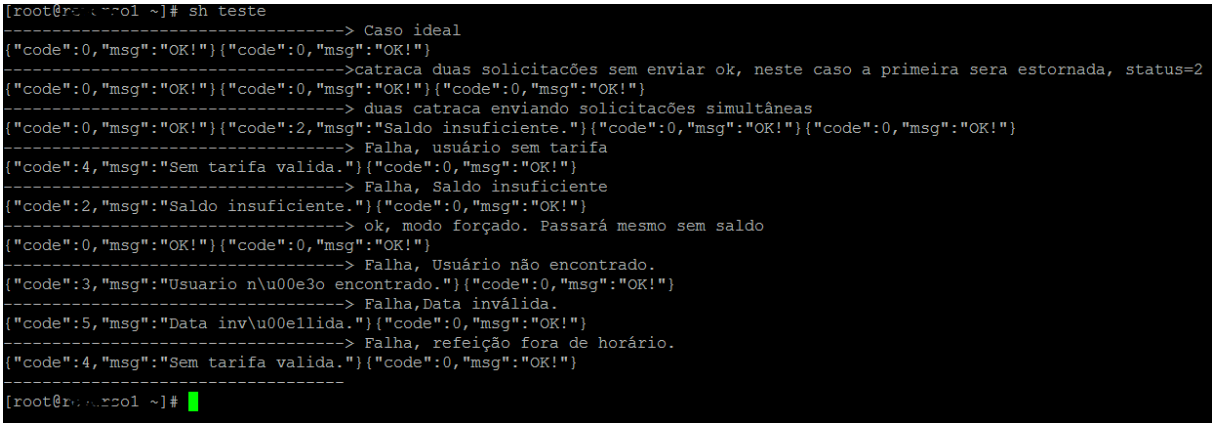
O teste de uso da API das catracas foi realizado através da criação de um *script* (Figura 39). Este simula a comunicação entre várias catracas e o servidor. O teste foi realizado com o *script* rodando em uma máquina virtual diferente, a fim de se aproximar da realidade. A Figura 40 mostra o resultado da rotina.

Figura 39 – Script de teste.

```
echo '----------------------------------> Caso ideal'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca/a7b9ad6028c8527db434330838d1d9478d70bf09/eduardo.c.rodrigues'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca_ok/a7b9ad6028c8527db434330838d1d9478d70bf09/'
echo
echo '---------------------------------->catraca duas solicitacões sem enviar ok, neste caso a primeira sera estornada, status=2'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca/a7b9ad6028c8527db434330838d1d9478d70bf09/eduardo.c.rodrigues'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca/a7b9ad6028c8527db434330838d1d9478d70bf09/admin'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca_ok/a7b9ad6028c8527db434330838d1d9478d70bf09/'
echo
echo '----------------------------------> duas catraca enviando solicitacões simultâneas'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca/bf49c4251456e84e77fa1a4f74453ce5a2dc6eee/eduardo.c.rodrigues'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca/a7b9ad6028c8527db434330838d1d9478d70bf09/admin'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca_ok/bf49c4251456e84e77fa1a4f74453ce5a2dc6eee'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca_ok/a7b9ad6028c8527db434330838d1d9478d70bf09/'
echo
echo '----------------------------------> Falha, usuário sem tarifa'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca/a7b9ad6028c8527db434330838d1d9478d70bf09/teste'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca_ok/a7b9ad6028c8527db434330838d1d9478d70bf09/'
echo
echo '----------------------------------> Falha, Saldo insuficiente'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca/a7b9ad6028c8527db434330838d1d9478d70bf09/eduardo.c.rodrigues'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca_ok/a7b9ad6028c8527db434330838d1d9478d70bf09/'
echo
echo '----------------------------------> ok, modo forçado. Passará mesmo sem saldo '
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca/a7b9ad6028c8527db434330838d1d9478d70bf09/eduardo.c.rodrigues//true'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca_ok/a7b9ad6028c8527db434330838d1d9478d70bf09/'
echo
echo '----------------------------------> Falha, Usuário não encontrado.'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca/a7b9ad6028c8527db434330838d1d9478d70bf09/aaaaaaaa'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca_ok/a7b9ad6028c8527db434330838d1d9478d70bf09/'
echo
echo '----------------------------------> Falha,Data inválida.'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca/a7b9ad6028c8527db434330838d1d9478d70bf09/eduardo.c.rodrigues/1592035409 '
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca_ok/a7b9ad6028c8527db434330838d1d9478d70bf09/'
echo
echo '----------------------------------> Falha, refeição fora de horário.'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca/a7b9ad6028c8527db434330838d1d9478d70bf09/eduardo.c.rodrigues/1492009200'
wget -qO - 'http://devsistemaru.ufes.br/adm/ru_ufes/api/catraca_ok/a7b9ad6028c8527db434330838d1d9478d70bf09/'
echo
echo '----------------------------------'
~
~
~
~
~
~
```

Fonte: Produção do próprio autor.

Figura 40 – Resultado da rotina.



Fonte: Produção do próprio autor.

Ao se fazer todos os testes de uso da catraca e de sincronia, foi possível concluir que o restante do sistema está funcionando, uma vez que os dados inseridos e consultados foram congruentes.

5 CONSIDERAÇÕES FINAIS

Nesse capítulo, são apresentadas as conclusões obtidas durante a análise, projeto e desenvolvimento desse trabalho, bem como as perspectivas para possíveis trabalhos futuros.

5.1 Conclusões

Neste trabalho, foram analisados vários tipos de sistemas de tarifação a fim de obter o que melhor se adequa ao restaurante universitário. Foi realizada a definição do escopo e o levantamento dos requisitos. Os testes foram feitos e verificou-se que os objetivos foram alcançados.

Por fim, a grande novidade foi a criação dos módulos para Drupal, que se mostraram bastante versáteis. Pode-se perceber que o ambiente de desenvolvimento para Drupal está maduro e oferece recursos suficientes para os mais variados usos.

5.2 Trabalhos Futuros

Normalmente, novas necessidades são identificadas no final do desenvolvimento de um software. A manutenção e a evolução do software devem ser um trabalho constante, de forma que o ciclo de vida não finalize na homologação, mas permaneça ao longo do tempo.

Sendo assim, alguns trabalhos surgirão a partir deste. Como evolução do Sistema de Tarifação do RU, pode-se especular a possibilidade de aumentar a segurança, ou seja, criar um quarto módulo voltado para a segurança, com *hash* dinâmicas para as comunicações.

Outra possibilidade de trabalho futuro é a criação de mais relatórios que a gerência do RU possa necessitar.

REFERÊNCIAS BIBLIOGRÁFICAS

389 Directory Server. **Documentation.** Disponível em: <directory.fedoraproject.org>. Acesso em: 17 agosto 2016.

Api Reference | Drupal API. Disponível em: <http://api.drupal.org>. Acesso em: 20 janeiro. 2017.

BUTCHER, Matt. **Learning Drupal 6 Module Development**. Birmingham: Packt Publishing, 2008.

ELMASRI, R.; NAVATHE, S.B. **Sistemas de bancos de dados.** 6**.** ed**.** São Paulo: Pearson, 2010.

FALBO, R. A. **Engenharia de Software**. 2014. Disponível em: <http://www.inf.ufes.br/~falbo/files/Notas_Aula_Engenharia_Software.pdf>.

FERNANDES, Cynthia Perovano. **Histórias Do Trabalho Em Um Restaurante Universitário: Entre Conversas, Panelas E Temperos.** Vitória: UFES, 2011. Tese (Mestrado em Psicologia Institucional) - Programa de Pós-Graduação em Psicologia Institucional, Centro de Ciências Humanas e Naturais, Universidade Federal do Espirito Santo, Vitória, 2011.

MariaDB Foundation. **MariaDB. 2015**. Disponível em: <https://mariadb.org/pt-br/>. Acesso em: 15 fevereiro 2017.

NIEDERAUER, Juliano. **Desenvolvendo websites com PHP.** São Paulo: Novatec (2004).

Restaurante Universitário da Universidade federal do Espirito Santo. **Resolução n° 56/2014.** Disponível em: <www.ru.ufes.br/content/resolução-n°-562014>. Acesso em: 23 dezembro 2016.

Restaurante Universitário da Universidade federal do Espirito Santo. **Valores e Identificação para Acesso.** Disponível em: <www.ru.ufes.br/content/valores-0>. Acesso em: 23 dezembro. 2016.

ROB, P.; CORONEL, C. **Sistemas de Banco de Dados – Projeto, Implementação e Administração.** São Paulo: Cengage Learning, 2011.

TOMLINSON, Todd. **Desenvolvimento em Pro Drupal 7**. Rio de Janeiro: Editora Ciência moderna (2012)

TRIGO, C. H. **OpenLDAP: uma abordagem integrada.** São Paulo: Novatec Editora, 2007.