**FACITEC**

**DIEGO FERREIRA DA SILVA**

**ARQUITETURA ORIENTADA A SERVIÇOS PARA DESENVOLVIMENTO DE SISTEMAS**

BRASÍLIA - DF

2012

**DIEGO FERREIRA DA SILVA**

**ARQUITETURA ORIENTADA A SERVIÇOS PARA DESENVOLVIMENTO DE SISTEMAS**

Trabalho de Conclusão de Curso apresentado à Facitec como parte dos requisitos para conclusão e pós-graduação no curso de Engenharia de Software da Facitec.

Orientador: Robson Oliveira Albuquerque, Doutor.

BRASÍLIA – DF

2012

**DIEGO FERREIRA DA SILVA**

**ARQUITETURA ORIENTADA A SERVIÇOS PARA DESENVOLVIMENTO DE SISTEMAS**

Trabalho de Conclusão de Curso apresentado à Facitec como parte dos requisitos para conclusão e pós-graduação no curso de Engenharia de Software da Facitec.

Área de Concentração: Arquitetura de Sistemas

Aprovada em:

Banca Examinadora:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Dedicatória**

A meus pais, Marlucia Cândida da Silva e Osvaldo Ferreira Filho e minha namorada Mariana Araujo Alves por todo apoio e compreensão nestes e outros momentos da minha vida.

**Agradecimentos**

Quero agradecer a todos os professores em especial ao professor Robson pelo incentivo e paciência, a todos os amigos que fiz durante o curso, todos que de certa forma contribuíram para que chegasse até aqui.

Obrigado.

**Resumo**

Os profissionais de desenvolvimento de software estão sempre em busca de novos recursos que proporcionem softwares de maior qualidade, com baixo custo e fácil manutenção.

Desde os primórdios da informática veem surgindo constantemente novas tecnologias que visam aumentar a produtividade e qualidade na construção de sistemas de informação.

Existe um conceito surgido recentemente, a Arquitetura Orientada a Serviços *(Service Oriented Architecture),* que trouxe grandes benefícios para as organizações que necessitam de sistemas de informação para sobreviver, ou seja, quase todas. Este conceito trouxe grandes benefícios para quem o utiliza, propiciando softwares menos desacoplados e mais robustos.

Da mesma forma que ocorreu com outros conceitos surgidos na computação, como a Orientação a Objetos, existem muitas pessoas apostando neste conceito como sendo a solução definitiva para os problemas encontrados na integração de sistemas.

Este estudo procura avaliar este novo conceito na engenharia de software e os impactos causados por ele no mundo de desenvolvimento de software.

Foram apresentados as principais metodologias utilizadas na integração de sistemas presentes no mercado, apontado seus pontos fortes e pontos fracos. Foi feito um estudo sobre a Arquitetura Orientada a Serviços, demonstrando como ela funciona e quais os benefícios que ela traz para a engenharia de software. Foi demonstrado como esta arquitetura trabalha, seus protocolos o meio de comunicação, quais a opções que os desenvolvedores tem em termos de tecnologia ao se utilizar esta arquitetura.

Foi feito um estudo de caso na construção de um sistema que utiliza princípios apresentados na Arquitetura Orientada a Serviços.

O sistema desenvolvido possui a funcionalidade de controle de finanças pessoais, o sistema possui dois módulos, um módulo que atua no ambiente da internet e outro módulo feito para dispositivos móveis.

Palavras chave: SOA, Web Services, Engenharia de Software, Integração de Sistemas.

**Abstract**

The software development professionals are always looking for new software features that provide higher quality, low cost and easy maintenance.   
Since the early days of computing see constantly emerging new technologies to increase productivity and quality in building information systems.   
There is a concept called emerged recently, Service Oriented Architecture, which brought great benefits for organizations that require information systems to survive, or almost all. This concept has brought great benefits to those using, providing software less uncoupled and more robust.   
Just as occurred with other concepts that arise in computing, such as object orientation, there are many people betting on this concept as the ultimate solution to the problems encountered in systems integration.   
This study seeks to evaluate this new concept in software engineering and the impacts caused by it in the world of software development.   
There presented the main methodologies used in systems integration in the market, pointing their strengths and weaknesses. A study was made on the Service Oriented Architecture, demonstrating how it works and what benefits it brings to software engineering. It was demonstrated how this architecture works, their means of communication protocols, which the options that developers have in terms of technology when using this architecture.   
Has made a case study in building a system that uses principles presented in the Service Oriented Architecture.

Keywords: SOA, Web Services, Software Engineering, Systems Integration.

**LISTA DE FIGURAS**

[Figura 1 - Linguagens utilizadas no mercado. 17](#_Toc330240215)

[Figura 2 - Exemplo de serviços de integração entre sistemas 18](#_Toc330240216)

[Figura 3 - Integração Via Socket TCP/UDP 19](#_Toc330240217)

[Figura 4 - Comportamento de requisição-resposta do HTTP 21](#_Toc330240218)

[Figura 5 - Requisição HTTP 21](#_Toc330240219)

[Figura 6 – Resposta HTTP 21](#_Toc330240220)

[Figura 7 – Integração Via Web Service 22](#_Toc330240221)

[Figura 8 – Exemplo de uma arquitetura baseada em serviços 25](#_Toc330240222)

[Figura 9 – Mensagem Soap 28](#_Toc330240223)

[Figura 10 – Troca de informações SOAP 30](#_Toc330240224)

[Figura 11 - Resposta REST Serviço Consulta Aluno 34](#_Toc330240225)

[Figura 12 – Diagrama de casos de uso da aplicação Web 38](#_Toc330240226)

[Figura 13 – Diagrama de casos de uso da aplicação Mobile 39](#_Toc330240227)

[Figura 14 – Diagrama de classes do pacote *br.com.financialmobile.bb* 40](#_Toc330240228)

[Figura 15 – Diagrama de classes do pacote *br.com.financialmobile.converter* 41](#_Toc330240229)

[Figura 16 – Diagrama de classes do pacote *br.com.financialmobile.dao* 42](#_Toc330240230)

[Figura 17 – Diagrama de classes do pacote *br.com.financialmobile.exceptions* 43](#_Toc330240231)

[Figura 18 – Diagrama de classes do pacote *br.com.financialmobile.model* 43](#_Toc330240232)

[Figura 19 – Diagrama de classes do pacote *br.com.financialmobile.model.to* 44](#_Toc330240233)

[Figura 20 – Diagrama de classes do pacote *br.com.financialmobile.model.util* 44](#_Toc330240234)

[Figura 21 – Diagrama de classes do pacote *br.com.financialmobile.security* 45](#_Toc330240235)

[Figura 22 – Diagrama de classes do pacote *br.com.financialmobile.security.util* 45](#_Toc330240236)

[Figura 23 – Diagrama de classes do pacote *br.com.financialmobile.service* 46](#_Toc330240237)

[Figura 24 – Diagrama de classes do pacote *br.com.financialmobile.servlets* 47](#_Toc330240238)

[Figura 25 – Diagrama de classes do pacote *br.com.financialmobile.servlets.xml.bean* 47](#_Toc330240239)

[Figura 26 – Diagrama de classes do pacote *br.com.financialmobile.util* 48](#_Toc330240240)

[Figura 27 – Diagrama de classes do pacote *br.com.financialmobile.webservices* 49](#_Toc330240241)

[Figura 28 – Diagrama de classes do pacote *br.com.financialmobile.exception* 49](#_Toc330240242)

[Figura 29 – Diagrama de classes do pacote *br.com.financialmobile.filters* 50](#_Toc330240243)

[Figura 30 – Diagrama de classes do pacote br.com.financialmobile.model 51](#_Toc330240244)

[Figura 31 – Diagrama de classes do pacote br.com.financialmobile.model.to 52](#_Toc330240245)

[Figura 32 – Diagrama de classes do pacote br.com.financialmobile.model.to 52](#_Toc330240246)

[Figura 33 – Diagrama de classes do pacote br.com.financialmobile.service 53](#_Toc330240247)

[Figura 34 – Diagrama de classes do pacote br.com.financialmobile.ui 54](#_Toc330240248)

[Figura 35 – Diagrama de classes do pacote br.com.financialmobile.utils 55](#_Toc330240249)

[Figura 36 - Tela de autenticação do sistema Web 56](#_Toc330240250)

[Figura 37 – Tela de cadastro de usuários no sistema Web 56](#_Toc330240251)

[Figura 38 – Tela de recuperação de senha no sistema Web 56](#_Toc330240252)

[Figura 39 – Tela inicial do sistema na Web 57](#_Toc330240253)

[Figura 40 – Tela de consulta de gastos 57](#_Toc330240254)

[Figura 41 – Tela de inclusão/alteração de gastos 58](#_Toc330240255)

[Figura 42 – Tela de consulta de usuários 58](#_Toc330240256)

[Figura 43 – Tela de cadastro/alteração de dados do usuário 58](#_Toc330240257)

[Figura 44 – Tela de consulta de categorias 59](#_Toc330240258)

[Figura 45 – Tela de cadastro/alteração de categorias 59](#_Toc330240259)

[Figura 46 – Tela de Relatório de Gastos 60](#_Toc330240260)

[Figura 47 – Tela de apresentação do sistema mobile 60](#_Toc330240261)

[Figura 48 – Tela de autenticação do sistema mobile 61](#_Toc330240262)

[Figura 49 – Tela inicial do sistema 61](#_Toc330240263)

[Figura 50 – Tela de inserção de gastos 62](#_Toc330240264)

[Figura 51 – Tela de consulta de gastos 62](#_Toc330240265)

[Figura 52 – Tela de exibição de resultados da consulta 63](#_Toc330240266)

[Figura 53 – Tela de opções pós consulta 63](#_Toc330240267)

[Figura 54 – Tela de edição de gastos 64](#_Toc330240268)

[Figura 55 – Tela de consulta do relatório 64](#_Toc330240269)

[Figura 56 – Relatório de gastos agrupado por categoria 65](#_Toc330240270)

[Figura 57 – Custo do relatório desmembrado 65](#_Toc330240271)

[Figura 58 – Tela inicial no momento do sincronismo 66](#_Toc330240272)

**LISTA DE ABREVIATURAS**

**SOA -** *Service Oriented Architecture*

**TCP -** *Transmission Control Protocol*

**HTTP -** *HyperText Transfer Protocol*

**WWW -** *World Wide Web*

**XML -** *eXtensible Markup Language*

**REST -** *Representational State Transfer*

**UML -**  *Unified Modeling Language*

**SOAP -** *Simple Object Access Protocol*

**TI -** *Tecnologia da Informação*

**SW -** *Software*

**IBM -** *International Business Machines*

**WSDL -** [*Web Service Definition Language*](http://www.w3.org/TR/wsdl)

**JSON -** *JavaScript Object Notation*

**SUMÁRIO**

[1 Introdução 13](#_Toc330236232)

[1.2 Motivação 13](#_Toc330236233)

[1.4 Objetivos Específicos 14](#_Toc330236234)

[1.5 Estrutura do Trabalho 14](#_Toc330236235)

[2 Integração de Sistemas 16](#_Toc330236236)

[2.1 Integração Via Sockets 19](#_Toc330236237)

[2.2 Integração Via Troca de Mensagens HTTP 20](#_Toc330236238)

[2.3 Integração Via Web Service 22](#_Toc330236239)

[3 Arquitetura Orientada a Serviços 23](#_Toc330236240)

[3.1 Baixo Acoplamento 24](#_Toc330236241)

[3.2 Independência de Plataforma ou Linguagem 25](#_Toc330236242)

[3.3 Reuso de Código 26](#_Toc330236243)

[3.4 Redução de Redundâncias 26](#_Toc330236244)

[3.5 Redução de Custo de Manutenção 27](#_Toc330236245)

[3.6 Web Services 27](#_Toc330236246)

[3.6.1 SOAP 27](#_Toc330236247)

[3.6.2 REST 31](#_Toc330236248)

[4 Modelagem do Sistema 34](#_Toc330236249)

[4.1 Introdução 34](#_Toc330236250)

[4.2 Visão Geral 34](#_Toc330236251)

[4.3 Ferramentas Utilizadas 34](#_Toc330236252)

[4.3.1 Padrões Utilizados 36](#_Toc330236253)

[4.4 Diagramas UML 36](#_Toc330236254)

[4.4.1 Casos de Uso 36](#_Toc330236255)

[4.4.2 Diagrama de Classes 38](#_Toc330236256)

[4.4.2.1 Web 38](#_Toc330236257)

[4.4.2.2 Mobile 48](#_Toc330236258)

[4.5 Aplicativo Financial Mobile 54](#_Toc330236259)

[4.5.1 Telas do Aplicativo e Utilização Módulo Web 54](#_Toc330236260)

[4.5.2 Telas do Aplicativo e Utilização Módulo Mobile (Emulador) 59](#_Toc330236261)

[4.6 Plano de Teste 65](#_Toc330236262)

[4.7 Considerações 66](#_Toc330236263)

[5 CONCLUSÃO 66](#_Toc330236264)

[5.1 Trabalhos Futuros 67](#_Toc330236265)

[Referências 68](#_Toc330236266)

[APÊNDICE A – DESCRIÇÃO DOS CASOS DE USO 70](#_Toc330236267)

[SUMÁRIO 70](#_Toc330236268)

[Descrição dos Casos de Uso 71](#_Toc330236269)

[SUMÁRIO 79](#_Toc330236270)

[Introdução 80](#_Toc330236271)

[Requisitos de Teste 81](#_Toc330236272)

[Estratégia de Testes 83](#_Toc330236273)

[Roteiro de Simulação de Testes 85](#_Toc330236274)

[Testes de Integridade das Informações 85](#_Toc330236275)

[Teste Funcional e de Interface 86](#_Toc330236276)

[Teste de Configuração e Segurança 91](#_Toc330236277)

[Teste de Instalação 92](#_Toc330236278)

# 1 Introdução

Atualmente empresas necessitam dos sistemas de informação para desenvolvimento e controle na gestão de todos os setores e recursos. O investimento nesta área geralmente acaba sendo maior do que o esperado, e nem sempre o retorno é maior ou igual ao valor investido. Diante deste cenário, ás empresas estão sempre em busca de soluções que visam baixar custos e trazer melhores resultados. Um ponto estratégico que traz inúmeros resultados positivos para as empresas é a reutilização de sistemas. Normalmente esta prática pode ocorrer através da integração de serviços existentes em um sistema específico a ser reutilizado. Porém, isso nem sempre acontece e, muitas vezes existe um sistema que faz uma determinada atividade e esta atividade tem que ser reescrita em outro sistema que precisa efetuar a mesma ação, simplesmente por estarem em versões diferentes ou por serem escritos em linguagens distintas, fatos que geram incompatibilidade entre ambos.

Conforme Silva (2006) abordou em sua monografia a Orientação a Serviços surgiu para suprir tal necessidade, unindo diversos conceitos e práticas sob um único paradigma, através da exposição de regras de negócio através de serviços.

Esse paradigma é conhecido como Arquitetura Orientada a Serviços ou **SOA** e, em essência, trata-se de um paradigma criado com o objetivo de desenvolver sistemas modularizados, o que traz diversos benefícios ao produto final.

Ao invés de desenvolver grandes aplicações como um único bloco, pode-se encapsular algumas funções importantes e disponibilizá-las na forma de serviços em uma rede.

A arquitetura propõe padrões e práticas de desenvolvimento para possibilitar que os serviços (web services) possam interagir adequadamente em um ambiente tão heterogêneo quanto a Internet, ou em outra rede qualquer.

Como qualquer arquitetura baseada em componentes, SOA mantém a independência de cada um deles e define, apenas como será feita a comunicação entre eles.

## 1.2 Motivação

A ausência de padronização nas áreas de TI é um fato observado diante de inúmeras experiências. Quem trabalha nesta área pode constatar isso no dia-dia, observando que a grande preocupação das empresas com soluções no presente, muitas vezes as levam a esquecer do futuro, ou seja, na urgência de atender à necessidade de um cliente, acabam não dando tanta importância para o que viria acontecer após a construção do sistema, as manutenções, atualizações e possíveis integrações com outros sistemas. Esse “descaso” pode causar muitos danos financeiros à organização, pois futuramente dependendo da necessidade do cliente em uma regra de negócio que o sistema criado não se adéque, pode ser necessário a reescrita de código ou até mesmo a reimplementação de um módulo.

Para tanto este trabalho apresenta um sistema desenvolvido utilizando alguns princípios da Arquitetura Orientada a Serviços, destacando-se as vantagens que este paradigma oferece como baixo acoplamento, independência de plataforma ou linguagem, reusabilidade de código, redução de redundâncias e custo de manutenção.

1.3 Objetivo Geral

Aplicar os conceitos da Arquitetura Orientada a Serviços (SOA), no desenvolvimento de um aplicativo simples, prático e eficaz, para controle finaceiro pessoal, integrando as plataformas Web e Mobile.

## 1.4 Objetivos Específicos

* Implementar módulo Web, permitindo segurança através da autenticação de usuários.
* Possibilitar entradas com informações e alterações de custos e gastos.
* Implementar um módulo para dispositivos móveis que ofereça conforto, comodidade e mobilidade ao usuário para entrar com dados referente a seus gastos nos sistema de onde estiver.
* Criar relatórios nos sistemas Web e Mobile, permitindo que o usuário consiga ver detalhadamente o controle de seus gastos.

## 1.5 Estrutura do Trabalho

O presente trabalho, que destaca as etapas para desenvolvimento de um sistema, utilizando a arquitetura SOA, encontra-se dividido e organizado nos seguintes capítulos:

O Capítulo 2 (dois) apresenta o conceito de integração de sistemas e algumas maneiras possíveis de se fazer isso, sua importância nas organizações que utilizam sistemas de informação.

O Capítulo 3 (três) apresenta o paradigma de Arquitetura Orientada a Serviços, o que é SOA, abordando suas principais características como baixo acoplamento, independência de plataforma ou linguagem, reuso de código, redução de redundâncias e redução de custo de manutenção, faz uma abordagem sobre Web Services que são fortemente utilizados na Arquitetura Orientada a Serviços, neste capítulo serão apresentados os principais protocolos utilizados, SOAP e REST.

O Capítulo 4(quatro) apresenta a documentação do sistema proposto.

O Capítulo 5(cinco) conclui o trabalho e apresenta propostas para futuras implementações.

# 2 Integração de Sistemas

Às vezes podemos nos perguntar, existe realmente a necessidade de integrarmos os sistemas, poderíamos re-escrever tudo na linguagem que utilizamos e padronizar tudo em uma só linguagem. Porém, se pensarmos por esse lado isso é impossível, pois nenhuma empresa consegue isso, pois as linguagens são alteradas, surgem outras melhores, com benefícios que leva a substituição da linguagem que está sendo utilizada no momento por outra.

IBM (2006) caracteriza a integração de sistemas da maneira tradicional como um processo árduo e falho, segundo ela o processo se divide basicamente nos seguintes passos:

* Necessidade de negócio.
* Procurar documentação existente.
* Verificar dados para importação.
* Analisar mudanças necessárias, encontrar impactos no código, planejar mudanças.
* Desenvolver rotinas para transformar dados e validação.
* Manualmente validar resultados, verificar integrações, ajustar inconsistências e publicar relatórios.

O resultado disso tudo nem sempre é o esperado, o fluxo pode ser tornar desordenado na arquitetura dos sistemas das empresas.

Existem barreiras que impedem que o negócio seja flexível e que haja o reuso:

* Falta de processos e padrões.
* Políticas de arquitetura limitadas.
* Foco em aplicações pontuais que suportam necessidades de negócio redundantes.
* Infraestrutura construída sem um plano.

É preciso quebrar as aplicações em blocos reutilizáveis de serviços, possibilitando combinar blocos para atender necessidades de negócios e reutilizar blocos para atender novas necessidades. IBM (2006)

Para isso é necessário a criação de um processo de desenvolvimento de software comum e orientado ao negócio, modernizando a arquitetura das aplicações para uma arquitetura aberta, orientada a serviços, com isso haverá uma revitalização gradual das interfaces das aplicações para o consumo e entrega de serviços. IBM (2006)

Um exemplo real:

Ford Motors:

* 50.000 programas batch são executados toda noite.
* Cada batch tem entre 5 e 20 passos.
* Cada passo é um módulo que possui entre 1.000 e 50.000 linhas de código COBOL

Fonte: Mendes, 2010.

Diante de fatos como mencionado acima podemos chegar a conclusão de que a re-escrita é inviável, temporalmente, economicamente e tecnicamente.

A nossa realidade:

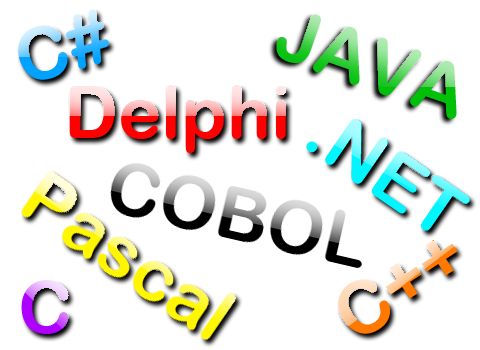


Figura - Linguagens utilizadas no mercado.

Fonte: Mendes, 2010.

Integração é a peça chave para que empresas possam reaproveitar e integrar regras de negócio implementadas em sistemas distintos.

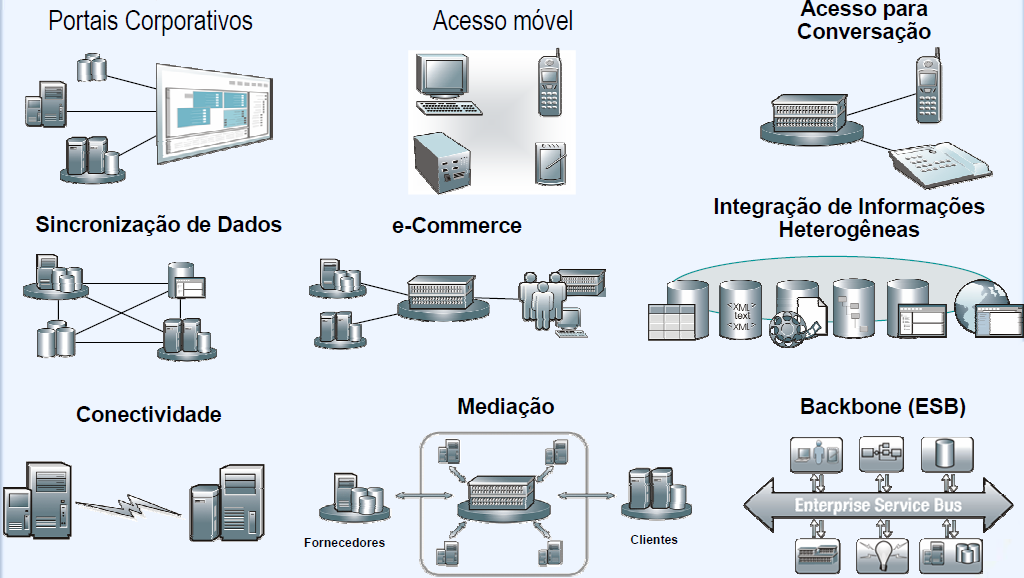


Figura - Exemplo de serviços de integração entre sistemas

Fonte: Mendes, 2010.

A figura acima exibe uma gama de sistemas e plataformas distintas que podem ser integradas.

Segundo Junior (2007) em Integração de Sistemas com Java, no passado, grandes empresas trabalhavam com aplicações centralizadas em servidores e com acesso via terminais burros, onde os aplicativos e dados eram integrados.

Este processo foi alterado ao longo do tempo, devido ao alto custo para manutenção e expansão destes equipamentos.

Equipamentos mais potentes foram disponibilizados para o usuário final, servidores mais baratos com outros sistemas operacionais foram incluídos nas redes das companhias e a internet com aplicações via browser vieram para completar e deixar mais complexo o ambiente corporativo.

Hoje podemos encontrar em uma empresa:

* Diversos bancos de dados(DB2, Oracle, MySQL, etc.).
* Aplicações cliente servidor, desenvolvidas em Delphi, VB e Java.
* Aplicações web, desenvolvidas em PHP, .Net e Java.
* Aplicações legadas utilizando-se dos velhos terminais.

No estágio atual, um usuário precisa acessar diversos sistemas para incluir um mesmo dado ou realizar diversas consultas e cálculos para terminar uma determinada tarefa.

Esta situação dá abertura a dados inconsistentes e muito retrabalho, uma situação, por exemplo, em que o usuário tenha que entrar com os mesmos dados em sistemas diferentes para efetivar uma operação.

Integração de sistemas vem para auxiliar a integridade de dados e a produtividade de usuários, executando em BackOffice tarefas que o usuário realizaria manualmente.

## 2.1 Integração Via Sockets

Sockets é uma comunicação TCP/UDP, ou seja, com sockets você pode criar uma comunicação entre aplicações, utilizando um protocolo próprio, ou pode comunicar-se com outras aplicações conhecendo seus protocolos. Por exemplo, uma determinada aplicação escrita em Java pode conversar com aplicações escritas em outras linguagens como C++, Delphi, etc. E pode também interagir com equipamentos físicos (catracas, controles de ponto, etc.) caso os mesmos utilizem o protocolo TCP ou UDP.

A utilização de Sockets para integração de sistemas não é usualmente recomendada, devido a grande quantidade de código que é necessário ser escrito para que se tenha um sistema confiável.

Este recurso é recomendado somente em casos que realmente seja necessário sua utilização, como nos casos de integração direta com softwares embarcados como, por exemplo, softwares de catracas, controle de ponto, devido ao grande trabalho de codificação utilizando esta metodologia.

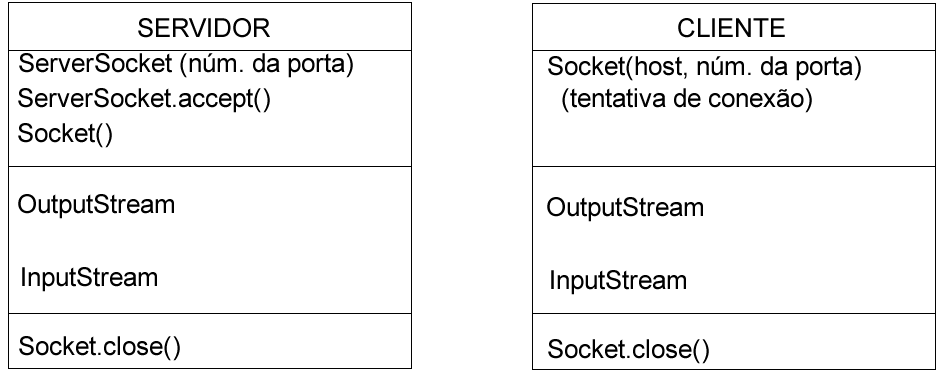


Figura - Integração Via Socket TCP/UDP

Fonte: Glauco, Fagundes, 2010.

Na figura acima é possível entender como ocorre o processo de comunicação entre sistemas utilizando Socket TCP/UDP.

* Primeiramente ocorre uma tentativa de estabelecimento de conexão (onde é informado endereço e porta do servidor).
* Após, basta enviar dados para o fluxo de conexão (não é preciso informar endereço e porta novamente).
* Baseado no modelo de fluxos, um Socket utiliza dois fluxos: um fluxo de entrada e outro de saída.
* Um processo envia dados a outro simplesmente escrevendo no fluxo de saída associado com o socket.
* Um processo recebe dados escritos por outro processo lendo de um fluxo de entrada associado a um socket, Silva (2006).

### 2.2 Integração Via Troca de Mensagens HTTP

Até a década de 1990, a Internet era usada primordialmente por pesquisadores, acadêmicos e estudantes universitários para se interligar com hospedeiros remotos, transferir arquivos de hospedeiros locais para hospedeiros remotos e vice-versa, enviar e receber notícias e enviar e receber correio eletrônico. No início da década de 1990 entrou em cena uma nova aplicação, importantíssima, a World Wide Web. A Web é a aplicação da Internet que chamou a atenção do público em geral. Ela transformou drasticamente a maneira como pessoas interagem dentro e fora de seus ambientes de trabalho.

A Web utiliza o protocolo HTTP em seu núcleo, podemos dizer que ele está no coração da Web. O HTTP é implementado em dois programas: um programa cliente e outro servidor. Os dois programas, executados em sistemas finais diferentes, conversam um com o outro por meio da troca de mensagens HTTP. O HTTP define a estrutura dessas mensagens e o modo como o cliente e o servidor as trocam.

O HTTP usa o TCP como seu protocolo de transporte subjacente. O cliente HTTP primeiramente inicia uma conexão TCP com o servidor. Uma vez estabelecida a conexão, os processos do browser e do servidor acessam o TCP por meio de suas interfaces sockets. No lado cliente a interface socket é a porta entre o processo cliente e a conexão TCP; no lado servidor, ela é a porta entre o processo servidor e a conexão TCP. O cliente envia mensagem de requisição HTTP para sua interface socket e recebe mensagens de resposta HTTP de sua interface socket. De maneira semelhante, o servidor HTTP recebe mensagens de requisição de sua interface socket e envia mensagens de resposta para sua interface socket, Kurose, Ross (2005).

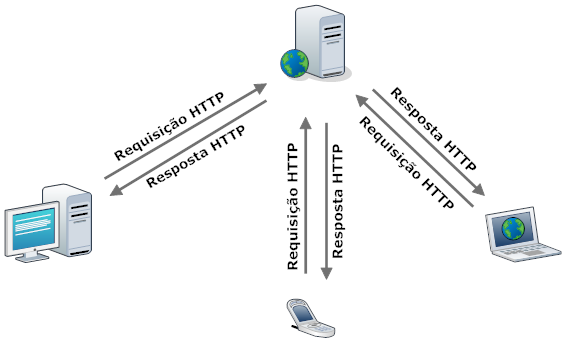


Figura - Comportamento de requisição-resposta do HTTP

Na figura acima, podemos ver como é o comportamento do protocolo HTTP diante de uma requisição, este protocolo trabalha da seguinte forma, o cliente envia uma requisição para o servidor, esta requisição possui um formato como o exemplo abaixo:

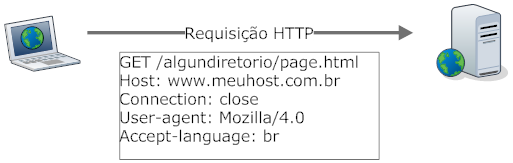


Figura - Requisição HTTP

Após o servidor receber a requisição o mesmo efetua o processamento solicitado e devolve uma resposta para o cliente:

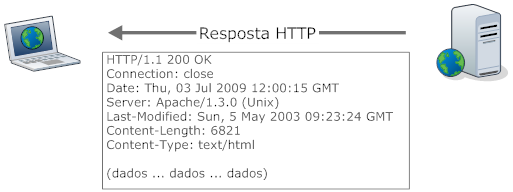


Figura – Resposta HTTP

Os dados retornados pelo servidor são em formato texto, portanto pode ser utilizado por qualquer sistema que consiga fazer uma requisição HTTP. O protocolo HTTP é comumente utilizado para integrar sistemas de plataformas diferentes, pois não exige um grande esforço para esta integração.

### 2.3 Integração Via Web Service

Web Service é a mais recente maneira para efetuar a integração entre sistemas, através deles é possível integrar sem distinção de plataforma ou linguagem. Seus protocolos utilizam XML (Extensible Markup Language) para empacotar e desempacotar as informações que são trafegadas através dos protocolos disponíveis para sua utilização, sendo assim aplicações feitas em linguagens diferentes trocam informações entre si e no momento desta troca a informação é convertida para a linguagem universal XML.

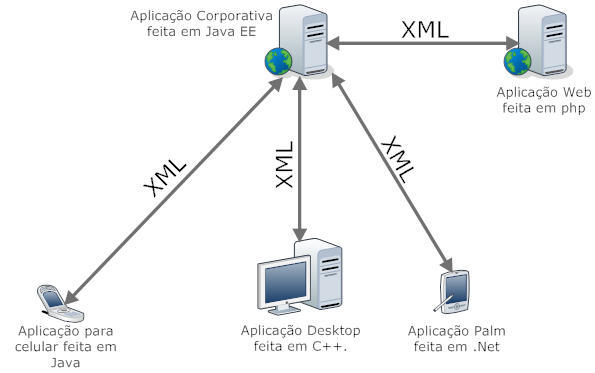


Figura – Integração Via Web Service

A figura anterior mostra aplicações feitas em linguagens distintas comunicando-se entre si, por meio de troca de mensagens XML, como é feito na utilização de Web Service.

O XML é a base dos Web Services, ele fornece a descrição, o armazenamento, o formato da transmissão para troca de dados. A sintaxe XML utilizada especifica como os dados são representados genericamente.

Web Service é a maneira mais indicada para integração de sistemas no momento atual, devido a sua praticidade e confiabilidade para integração.

# 3 Arquitetura Orientada a Serviços

A Arquitetura Orientada a Serviços (SOA) é um modelo de desenvolvimento de software que visa permitir que os serviços de um processo de negócio sejam integrados mais facilmente.

Um componente de um processo de negócio é uma atividade comum, algo que é realizado frequentemente naquele processo de negócio específico. Definimos esta atividade como função de negócio.

O objetivo é implementar um sistema que represente o negócio do cliente, dividindo este negócio em processos e estes em atividades. Silva (2006).

SOA pode ter diferentes interpretações:

* Diretor de Negócios: Uma tecnologia que cria um ambiente de negócio ágil e provê vantagem competitiva ou maior valor.
* Gerente TI: É um conjunto de processos, estruturas e diretrizes de governança que permite alinhar TI às necessidades do negócio.
* Arquiteto SW: É uma arquitetura de software baseada em padrões abertos que permite integrar aplicações novas e existentes.
* Desenvolvedor: Um framework baseado em Web Services que permite invocar objetos remotamente utilizando em um ambiente homogêneo baseado em troca de mensagens XML.

IBM (2006)

Algumas características do SOA:

* Atividades de negócio são realizadas através de uma série de serviços que possuem maneiras bem definidas de “pedir” e “responder” informações.
* Não interessa como o serviço foi implementado, contanto que ele responda aos comandos da forma correta com a qualidade necessária.
* O serviço precisa ser adequadamente seguro e confiável, além de rápido o suficiente para atender as demandas.
* Isto faz de SOA uma tecnologia ideal para ser utilizada em um ambiente de TI que possua hardware e software de múltiplos fabricantes.

IBM (2006)

Segundo (ROSEN, M., LUBLINSKY, B., SMITH,T.,K., BALCER.J.MARC., 2008) uma forma de avaliar o que SOA promete é olhar para as motivações e expectativas das pessoas que estão envolvidas em atividades SOA. Em uma pesquisa em 2006 realizado pelo Consórcio Cutter, as motivações para se utilizar SOA incluíam uma série de razões técnicas e comerciais. As motivações mais comuns foram: agilidade, flexibilidade, reutilização, racionalização de dados, integração e redução de custos.

Algumas das repostas mais relevantes a pesquisa foram:

* Reutilização de ativos estratégicos em todos os aplicativos de diversos departamentos.
* Necessidade de prestar um apoio mais ágil aos processos de negócio, lidar eficientemente com mudanças no processo de negócio.
* Gerenciamento das informações de uma forma global
* Rapidez e facilidade de implantação do projeto, a preocupação com a duplicação de métodos de negócio entre os projetos.
* Eficiência em termos de confiabilidade e custo de desenvolvimento.
* Reunir diversas linhas de negócio em muitas áreas geográficas com velocidade mais rápida para o mercado.
* Integrar sistemas legados.

Segundo (JUSUTTIS, M., NICOLAI, 2007), SOA é um conceito para grandes sistemas distribuídos. Para entender SOA, você tem que entender as propriedades de grandes sistemas distribuídos.

Primeiro, os sistemas têm de lidar com grandes legados. Você não pode introduzir SOA através da concepção de tudo a partir do zero. Você tem que lidar com o fato de que a maioria dos sistemas que estão em uso permanecerá em uso. Isto também significa que o estabelecimento de SOA não é um projeto de como projetar um novo sistema. Trata-se de alterar a estrutura de um sistema existente, o que significa que tem que lidar com plataformas antigas e problemas de compatibilidade. Na verdade, o SOA é uma abordagem para a manutenção de sistemas de grande porte.

## 3.1 Baixo Acoplamento

Cada atividade do negócio é tratada como um serviço, ou seja, um componente independente que poderá ser utilizado quantas vezes forem necessárias em partes diversas do sistema ou por outros sistemas.

Vemos que SOA é, essencialmente, uma arquitetura baseada em componentes, onde cada componente preserva a sua independência e interage, apenas, através de interfaces bem definidas. Silva (2006).

## 3.2 Independência de Plataforma ou Linguagem

Outro ponto importante a se ressaltar é que não há limitações prévias em relação ao modo de implementar um serviço. Não há limitações em relação às tecnologias, linguagens ou plataformas a serem utilizadas, por exemplo. O desenvolvedor deve apenas especificar a declaração do serviço os tipos de dados de entrada e saída. Os detalhes da implementação do serviço serão abstraídos de quem os utiliza.

Com a utilização da Arquitetura Orientada a Serviços é possível alcançar a interoperabilidade, que é a possibilidade de sistemas coexistirem e comunicarem-se independente de fabricantes ou tecnologias. Com esse fator alcançado é possível produzir soluções muito mais flexíveis e de melhor qualidade já que não encontramos dificuldades de comunicação entre sistemas diversos. Hansen (2007).

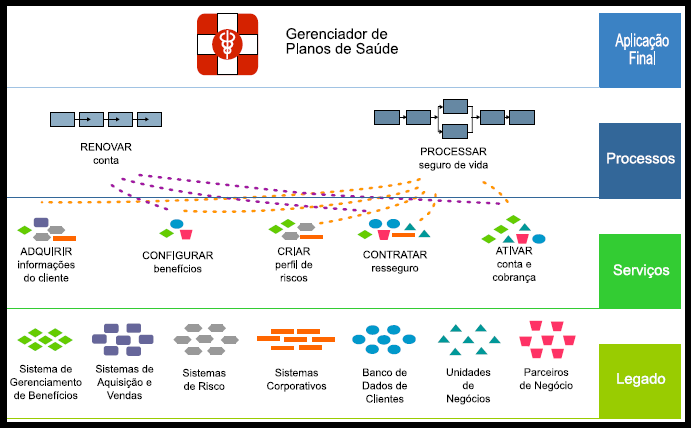


Figura – Exemplo de uma arquitetura baseada em serviços

Fonte: Silva, 2006

Na figura 3, temos um exemplo simplificado do modo como um sistema poderia ser implementado utilizando os conceitos relacionados à orientação a serviços.

Na camada inferior temos todos os sistemas que o cliente utiliza atualmente, sistemas que talvez já não atendam a necessidade do negócio do cliente, pois o mundo dos negócios é muito dinâmico e muda freqüentemente. Os mesmos não conseguem acompanhar esta mudança sem gerar um custo alto para o cliente devido a rigidez na sua construção.

Um sistema é como uma construção civil, com a diferença de poder haver uma depreciação no sentido de atender as expectativas com muito mais rapidez que um prédio, por exemplo.

No entanto mesmo esses sistemas depreciados são importantes para o cliente, e não é possível desfazer-se deles.

Esse é um exemplo de um contexto em que a orientação a serviços pode ser aplicada. Podemos transformar cada função do negócio do cliente em um serviço independente, criamos estes serviços acessando estes sistemas do cliente e retirando as informações necessárias, disponibilizando-as para quem as necessita.

Vale ressaltar que se este sistema já estivesse sido feito com base na orientação a serviços desde o inicio da sua construção, talvez não teríamos a camada inferior, pois essa camada seria abstraída pela camada de serviços, que é a que precisamos conhecer para utilizá-los. Silva (2006)

## 3.3 Reuso de Código

A reutilização de código é um fator altamente recomendado pela Engenharia de Software, porém não sendo sempre possível atingi-la em sua totalidade, principalmente em sistemas distintos que por ventura tenham que executar a mesma ação.

Através da orientação a serviços vimos que isto é plenamente possível, pois os serviços são totalmente independentes.

## 3.4 Redução de Redundâncias

Comumente vemos situações em que uma importante regra de negócio é implementada em mais de um sistema, isso é comum porque os sistemas normalmente são construídos de forma independente, sem pensar uns nos outros. Com a arquitetura orientada a serviços isso é eliminado, pois a regra é disponibilizada para quem precisar utilizar em forma de serviço. A lógica não pertence apenas a um único sistema, mas sim a todos os sistemas que necessitem utilizá-la.

## 3.5 Redução de Custo de Manutenção

Este fator está bastante ligado ao item anterior, o que acontece é que como a redução de redundância acontece, se por ventura houver alguma alteração em alguma regra de negócio que necessite a alteração de código de algum serviço, este código somente terá que ser alterado em um único lugar, se não utilizarmos orientação a serviços, esse código terá que ser alterado em todos os sistemas que implementa tal regra.

## 3.6 Web Services

No início de 2001 quando Ariba, IBM e Microsoft publicaram o WSDL 1.1 (Web Services Description Language), Web Services foram pensados como uma maneira de construir aplicação distribuídas mais facilmente. Esta tecnologia permite que aplicações se integrem diretamente com outras aplicações. E a idéia básica é que essa conexão se dê sem que seja necessário efetuar grandes customizações nas próprias aplicações. Além disso, uma das premissas fundamentais é que o padrão usado pelas conexões seja aberto e independente de plataforma tecnológica ou linguagens de programação.

Um WebSservice, portanto, é um componente de software, ou uma unidade lógica de aplicação, que se comunica através de tecnologias padrões de Internet. Esse componente provê dados e serviços para outras aplicações. Essa tecnologia combina os melhores aspectos do desenvolvimento baseado em componentes e a Web. Como componentes, representam uma funcionalidade implementada em uma 'caixa-preta', que pode ser reutilizada sem a preocupação de como o serviço foi implementado. As aplicações acessam os Web Services através de protocolos e formatos de dados padrões, como HTTP, XML, SOAP e REST.

## 3.6.1 SOAP

SOAP é um protocolo leve destinado a troca de informações estruturadas em um ambiente distribuído, ele utiliza o XML para definir a estrutura da mensagem que pode ser trocada também em outros protocolos subjacentes, ele foi criado para ser independente de qualquer linguagem em especial.

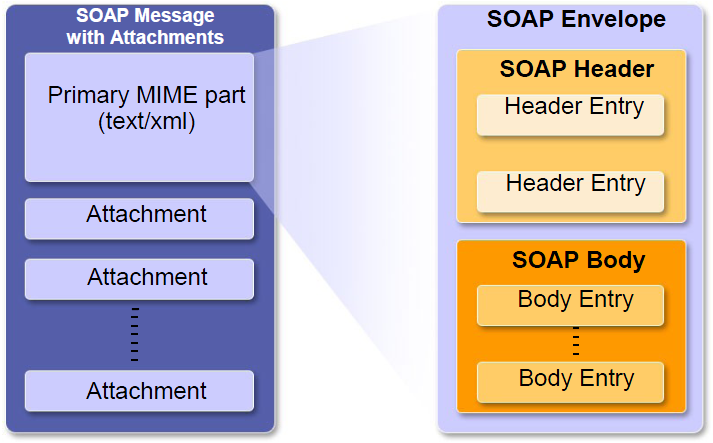


Figura – Mensagem Soap

Fonte: Sun, 2006

Na imagem acima podemos ver em detalhes do que consiste um envelope SOAP, primeiramente vemos na mensagem que ela possui uma parte inicial onde contém a mensagem no formato XML, esta mensagem é dividida em dois blocos o Header e Body, esses blocos podem ter n filhos para representar suas informações, o header não é obrigatório, nele normalmente são adicionadas informações sobre a mensagem, configurações e etc. No SOAP body são colocadas informações referentes ao conteúdo da mensagem em sí e dentro dele também é inserido uma tag especial para indicar se houve erro na mensagem o <soap:fault>, que é utilizado nas mensagens SOAP de resposta. A mensagem possui esse formato devido a uma padronização estipulada pelo órgão W3C, para que todas as implementações desse protocolo tivesse este formato.

O protocolo SOAP possui duas versões a 1.1 e 1.2 e segue abaixo uma listagem com as diferenças entre o protocolo 1.1 e 1.2.

* SOAP 1.2 não permite que qualquer elemento se ja inserido após o corpo, enquanto SOAP 1.1 permite que isso seja feito.
* SOAP 1.2 não permite que o atributo env:encondingStyle[[1]](#footnote-2) apareça dentro da tag env:Envelope[[2]](#footnote-3), enquanto SOAP 1.1 permite que ele seja exibido em qualquer elemento. SOAP 1.2 possui elementos específicos em que este atributo possa ser usado.
* SOAP 1.2 defina uma nova tag env:NotUnderstod[[3]](#footnote-4) no elemento header para transmitir informação sobre um bloco de cabeçalho obrigatório que não poderiam ser processados.
* SOAP 1.2 fornece um novo código de falha DataEncodingUnknown
* Os namespacs definidos pelos dois protocolos são diferentes
* SOAP 1.2 substitui o atributo env:actor[[4]](#footnote-5) pelo env:role, mas essenciamente com a mesma semântica.
* SOAP 1.2 define um novo atributo, env:relay, para os blocos de cabeçalho para indicar se os blocos de cabeçalho não transformados devem ser enviados.
* SOAP 1.2 define dois novos papéis, “none” e “ultimateReceiver”, juntamente com um modelo mais detalhado de processamento de como estes se comportam.
* SOAP 1.2 retirou a notação de “.” Para os códigos de falha, que agora são simplesmente um nome em XML Qualificado, onde o prefixo do namespace é o namespace do envelope.
* SOAP 1.2 substituiu os códigos de erro “client” e “server” pelos códigos “sender” e “receiver”.

(W3, 2010)

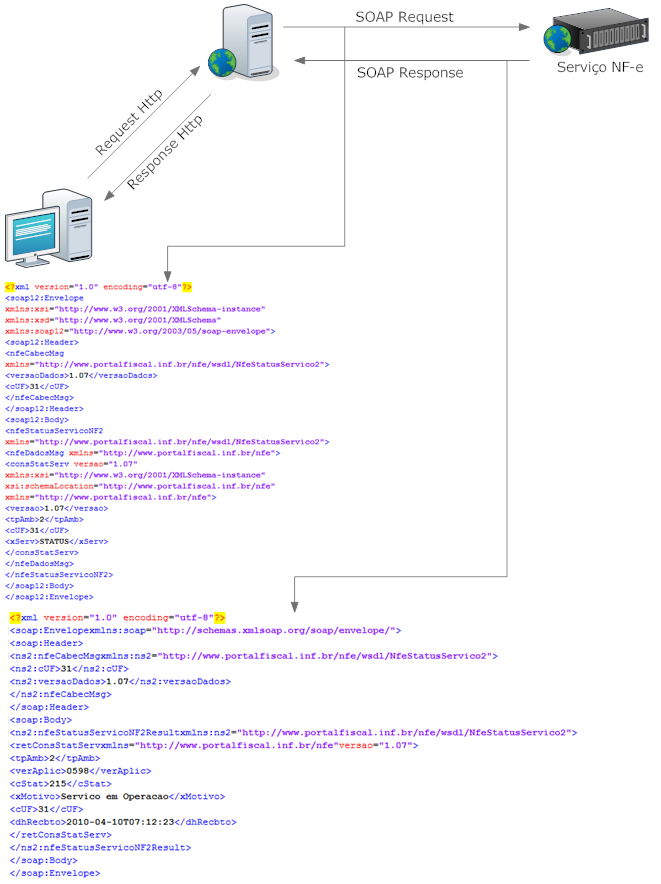


Figura – Troca de informações SOAP

Na imagem acima podemos ver um exemplo real de uma troca de mensagens utilizando SOAP 1.2, neste caso o cliente é um browser acessando a aplicação no servidor web que por sua vez consome os serviços de outro servidor utilizando o protocolo SOAP.

Com este protocolo os serviços ficam totalmente desacoplados de plataforma ou linguagem de programação.

## 3.6.1.1 WS-Security

Em abril de 2002, a Microsoft Corporation, a IBM Corporation e a VeriSign Inc. uniram-se e publicaram um conjunto de novas especificações de segurança denominadas WS-Security para serviços Web, com o objetivo de que as empresas pudessem criar e construir aplicações de serviços Web com ampla interoperabilidade.

Em março de 2004, uma nova especificação do padrão WS-Security foi publicada, como “Web Services Security: SOAP Message Security 1.0”, pelo grupo OASIS(Organization for the Advancement of Structured Information Standards) que descreve avanços na mensagem SOAP, ao especificar um perfil no uso da Assinatura XML(XML Signature) e da Criptografia XML(XML Encryption) para garantir integridade e confidencialidade para mensagens SOAP (OASIS, 2012).

WS-Security suporta, integra e unifica vários modelos, mecanismos e tecnologias de segurança em uso no mercado, permitindo que vários sistemas possam interoperar em plataformas e linguagens neutras.

As novas especificações de segurança definem um conjunto de padrões para extensões SOAP ou para cabeçalhos de mensagens, utilizados para oferecer maior integridade, confidencialidade e autenticação das mensagens com o sistema de transmissão de mensagens SOAP.

A autenticação está relacionada à identificação do chamador. O WS-Security usa tokens de segurança para manter essas informações com um cabeçalho de segurança da mensagem SOAP. A integridade da mensagem é obtida com assinaturas digitais XML. Isso garante que partes da mensagem não tenham sido adulteradas após a assinatura do originador. A confidencialidade da mensagem é baseada na especificação de criptografia XML e garante que partes correspondentes da mensagem só possam ser compreendidas pelo(s) destinatário(s) desejado(s).

A segurança é oferecida no próprio protocolo de troca de mensagens – SOAP. Com este recurso é possível uma mensagem passar por vários serviços Web ao longo do seu processamento. Suponha que um cliente A faça uma chamada a um serviço Web B, que faz outra chamada a um serviço Web C. Com informações de segurança embutido no protocolo de comunicação é possível atender os requisitos de segurança para este exemplo e muitas aplicações exigem (ROSENBERG, 2004).

## 3.6.2 REST

O termo REST veio de uma tese de doutorado sobre a web escrita por Roy Fielding, publicado em 2000, é uma técnica de engenharia de software para troca de mensagens em sistemas distribuídos.

REST não é considerado uma arquitetura para desenvolvimento de software, mas sim um conjunto de normas que quando aplicada no desenvolvimento de sistemas cria uma arquitetura confiável e desacoplada para troca de mensagens entre mesmos ou sistemas distintos.

Existem algumas restrições que definem um sistema REST:

* O sistema deve ser cliente – servidor.
* Não deve manter o estado da conexão, ou seja, as requisições devem ser independente umas das outras.
* Deve implementar um sistema de cache e sua estrutura de rede deve suportar diversos níveis de cache.
* Tem que ser uniformemente acessível, cada recurso deve possuir um único endereço de acesso válido utilizando um conjunto de métodos pré-definidos: GET, POST, PUT e DELETE.
* Tem que ser construído em camadas e deve suportar escalabilidade.
* Deve fornecer código sobre demanda, embora esta é uma restrição opcional, aplicações podem ser construídas para carregar o código em tempo de execução, sob demanda, por exemplo os applets java.

Estas restrições não ditam qual tecnologia utilizar, elas só definem como os dados são transferidos entre os componentes e quais os benefícios de seguir estas restrições. Portanto um sistema REST pode ser implementado em qualquer arquitetura de rede disponível. REST é basicamente uma arquitetura sustentável, renovável e distribuída.

Um recurso REST é algo que é endereçável pela internet. Por endereçável, entendemos que os recursos podem ser acessados e transferidos entre clientes e servidores. Posteriormente, um recurso é um mapeamento lógico temporal de um conceito para o problema domínio para o qual estamos a implementar uma solução.

Os serviços REST são implementados normalmente sobre o protocolo HTTP, devido a essa implementação pode ser transmitido basicamente qualquer tipo de informação. Os dados são trafegados utilizando TCP/IP e o navegador sabe interpretar os fluxos binários por causa do valor do Content-Type inserido no cabeçalho da mensagem de resposta do protocolo HTTP. Consequentemente em um sistema REST a representação de um recurso depende do tipo informado na chamada ao serviço (MIME type), que é especificado no protocolo da mensagem de requisição.

Existe uma semântica estabelecida ao nível da infraestrutura. No caso da Web, intermediários como as caches e proxies, podem agir de acordo com a operação executada e contribuir para a escalabilidade de toda a infraestrutura, sem afetar a definição da respectiva operação abstrata. Por exemplo, o método GET, idempotente por definição, pode ser servido por caches intermédias sem intervenção do servidor responsável pelo recurso. Por outro lado, os métodos POST, PUT e DELETE só podem ser processados pelo servidor uma vez que alteram

o estado do recurso.

Segundo a abordagem tradicional ao desenvolvimento de Serviços Web, os recursos estão escondidos atrás de métodos que podem ser invocados. Num Serviço Web compatível com a arquitetura REST os recursos a disponibilizar devem ser expostos e ter uma identificação global.

Um exemplo de URL’s de serviços REST pode ser visto abaixo:

*http://endereco.exemplo.com.br/informacoesaluno/123456*

*http://endereco.exemplo.com.br/informacoesaluno/4564654*

*http://endereco.exemplo.com.br/informacoesaluno/6545454*

Depois de expostos os recursos, a interação é realizada utilizando as operações genéricas do protocolo HTTP. Assim, para obter a representação da informação relativa a um estudante, utiliza-se o método GET sobre o URL concreta. Na Figura 11 ilustra-se essa situação.

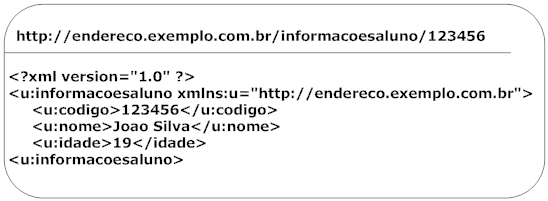


Figura - Resposta REST Serviço Consulta Aluno

Os métodos restantes do protocolo HTTP permitem: pedir a criação de uma representação para um novo recurso (POST); atualizar a representação de um recurso existente (PUT); apagar um recurso (DELETE). Enquanto que os métodos DELETE e PUT operam diretamente sobre um recurso identificado por uma URL, o método POST é aplicado sobre uma URL disponibilizada pelo servidor para a criação de novos recursos (ex: *http://endereco.exemplo.com.br/informacoesaluno/*).

De uma forma resumida, REST representa um modelo de como a Web deve funcionar de forma a garantir os requisitos inicialmente estabelecidos, em particular as propriedades de escalabilidade e heterogeneidade.

Na construção de serviços Web, esses serviços necessitam ser acessados em algum

lugar na Web por uma aplicação cliente. Uma forma de acessar um serviço Web é fazer com que a aplicação cliente conheça a URI (Uniform Resource Identifier) do serviço, desta maneira caracteriza o modo estático de localizar e acessar um serviço. Entretanto, quando a aplicação cliente não detém a localização de um serviço Web, este pode ser descoberto antes de ser acessado, caracterizando o modo dinâmico de descobrir a localização de um serviço. O UDDI provê um método padronizado para a publicação e descoberta de informações sobre serviços Web. É uma iniciativa de criar um framework padrão para descrição de serviços, descoberta de negócios e integração de serviços comerciais focando na pesquisa e na arquitetura orientada a serviços (CHAPPELL, 2002).

# 4 Modelagem do Sistema

## 4.1 Introdução

O levantamento dos requisitos para modelagem do aplicativo deu-se a partir de entrevistas realizadas com pessoas que já utilizavam ferramentas de controle de finanças pessoais e de observação dos aplicativos existentes. O direcionamento tomado foi atender as necessidades sem deixar que a utilização da ferramenta ficasse complexa para o usuário. A interação é mínima mas suficiente, ficando a cargo do *software* a maioria das rotinas.

A documentação desta etapa foi realizada utilizando o padrão UML – *Unified Modeling Language* – dando origem aos Digramas de Caso de Uso e de Classes, descritos nesta seção.

## 4.2 Visão Geral

O sistema desenvolvido tem como finalidade controlar as finanças pessoais do usuário. O usuário pode interagir com o sistema através de duas interfaces, com acesso a internet utilizando um browser ou através do aplicativo instalado no celular. As duas interfaces possuem basicamente as mesmas funcionalidades, permitindo ao usuário que informe ou altere informações e gere relatórios. O sistema permite ao usuário a maior comodidade e conforto.

## 4.3 Ferramentas Utilizadas

Para o desenvolvimento da aplicação foi adotado a linguagem de programação Java (Sun, 2010), o ambiente de desenvolvimento Eclipse (Eclipse, 2010) e o container web Apache Tomcat(Tomcat, 2010) com Jboss Embedded(Jboss Embedded, 2010) instalado para desenvolver a aplicação Web. Para a aplicação mobile foi utilizado o ambiente de desenvolvimento NetBeans(Netbeans, 2010) com JavaMe SDK 3.0 instalado. Além da familiaridade com a linguagem e seu potencial de portabilidade entre sistemas operacionais, o Java tem uma rica base de bibliotecas e *frameworks[[5]](#footnote-6)* que facilitam o processo de desenvolvimento, como as que foram utilizadas no projeto:

* JSF 1.2 - Api para desenvolvimento de aplicações Web em Java que propicia um modelo de desenvolvimento baseado em componentes, ou seja, o estado da página é mantido no servidor possibilitando que seja vinculado componentes da página diretamente a objetos Java.
* JBoss Rich Faces – Framework para desenvolvimento de aplicações ricas para internet, este frameworkd oferece vários componentes para desenvolvimento Web com JSF que facilitam e aumentam a produtividade do desenlvolvedor.
* JBoss Seam – Framework para desenvolvimento web em JSF, este framework adiciona ao JSF novos recursos, ele completa o JSF possibilitando ao desenvolvedor a utilização de recursos que o JSF puro não provê, deixando o código mais enxuto e elegante.
* Spring Framework – Framework para desenvolvimento de aplicações Java que propicia ao desenvolvedor várias funcionalidades, como por exemplo: Gerenciamento de transações, Injeção de dependências e outros recursos.
* Spring Security – Framework para autenticação e controle de acesso de aplicações Java, este framework oferece vários recursos de autenticação e controle de acesso baseado e perfis.
* Hibernate – Framework para mapeamento objeto relacional é um facilitador para se trabalhar com banco de dados em Java, oferece recursos para mapeamento de tabelas do banco de dados em objetos Java e diversos facilitadores para criação de consultas dinâmicas e permite que o sistema seja facilmente portável de um banco de dados para outro.
* Extjs – Biblioteca Java script para criação de interfaces web.
* JQuery – Biglioteca Java script crossbrowser que propicia uma sintaxe mais simples para utilização de Javascript.
* JAX-WS – Implementação Sun para se trabalhar com web services utilizando o protocolo SOAP em Java.
* LWUIT – Api JME para desenvolvimento de interfaces Java para celulares.
* KSOAP – Api para criação de clientes de web services SOAP em celulares utilizando a linguagem Java.
* FLOGGY – Api para persistência de dados em dispositivos móveis utilizando Java.
* MAVEN – Ferramenta para integração contínua de projetos, com esta ferramenta um projeto desenvolvido pode ser aberto em qualquer IDE que consiga ler projetos neste padrão.
* AJAX – Técnica para troca de mensagens assíncronas entre o cliente e o servidor, com esta técnica é possível que aja troca de mensagens entre o browser e o servidor de forma assíncrona, sem que o usuário tenha que esperar uma resposta.

## 4.3.1 Padrões Utilizados

Para desenvolvimento do sistema foram utilizados alguns padrões de projetos como:

* DAO – Este é um padrão que consiste em manter toda estrutura de acesso a banco de dados em uma camada independente das demais camadas do projeto.
* SERVICE – Este padrão utiliza uma forma de exposição de métodos de regra de negócios através de serviços para a camada de visualização.

## 4.4 Diagramas UML

## 4.4.1 Casos de Uso

O diagrama de Casos de Uso é utilizado para auxiliar a comunicação entre analistas e o cliente do *software* que está sendo especificado. Ele demonstra as principais funcionalidades que o aplicativo deve ter sem entrar em detalhes técnicos ou na solução do problema, sempre do ponto de vista do usuário (Guedes, 2008).

O diagrama do aplicativo é caracterizado pela presença de dois atores. De um lado o usuário comum, que é o usuário com permissão somente para a utilização do sistema. Do outro o *usuário administrador* que é a o usuário com acessos administrativos com permissões para alteração de dados de todos os usuários e para cadastro de categorias.

O sistema desenvolvido permite que o usuário faça o controle de seus gastos do dia-dia, através de duas interfaces acessando pela internet com um browser ou utilizando um dispositivo móvel com o aplicativo instalado.

Para desenvolvimento deste sistema foi utilizando princípios definidos na Arquitetura Orientada Serviços, expondo métodos de regra de negócios para o acesso via dispositivo móvel.

Nas Figuras 11 e 12 podemos ver como os atores interagem com os casos de uso e suas interdependências. Estes casos de uso são detalhados no Apêndice A.



Figura – Diagrama de casos de uso da aplicação Web



Figura – Diagrama de casos de uso da aplicação Mobile

### 4.4.2 Diagrama de Classes

O diagrama de classes é uma representação da estrutura e relações das classes que servem de modelo para os objetos ou entidades (Guedes, 2008). Aqui ele foi dividido em pacotes de forma a organizar as classes.

O projeto foi divido em camadas, sendo a primeira uma camada responsável por acesso ao banco de dados, a segunda que é a camada de serviços que interage diretamente com a camada de acesso a dados é responsável por expor métodos de regra de negócios para a camada de visualização e por último a camada de visualização que é onde foi desenvolvido toda interface com o usuário.

## 4.4.2.1 Web



Figura – Diagrama de classes do pacote *br.com.financialmobile.bb*

As classes representadas no pacote acima são responsáveis por manter o estado das páginas web, nesse contexto cada página web possui uma classe para representar seu estado e fazer a interface da página com as camadas subseqüentes.



Figura – Diagrama de classes do pacote *br.com.financialmobile.converter*

As classes do pacote acima são classes conversoras, responsáveis por converter campos de texto das páginas em objetos Java.



Figura – Diagrama de classes do pacote *br.com.financialmobile.dao*

As classes do pacote acima são responsáveis pelas operações do sistema com o banco de dados, a classe principal deste pacote é a AbstractDao que possui todos os métodos básicos como inserção, consulta, atualização e delete de um objeto específico, esta classe é genérica de forma que as outras classes que herdam dela não precisem reimplementar estes métodos.



Figura – Diagrama de classes do pacote *br.com.financialmobile.exceptions*

As classes do pacote acima são utilizadas para tratamento de exceções no sistema.



Figura – Diagrama de classes do pacote *br.com.financialmobile.model*

As classes do diagrama acima são utilizadas para fazer o mapeamento objeto relacional entre as tabelas do sistema no banco de dados e os objetos Java no sistema.



Figura – Diagrama de classes do pacote *br.com.financialmobile.model.to*

A classe do pacote da Figura 19 é para ser utilizada como objeto de transição de dados, neste caso possui apenas uma classe a CustoTO devido ao fato de ser uma classe utilizada para representar um relatório no sistema.



Figura – Diagrama de classes do pacote *br.com.financialmobile.model.util*

O pacote da Figura 20 possui classes que são utilitárias para se trabalhar com os objetos do pacote model, normalmente os métodos destas classes são todos estáticos e são utilizados para operações que não fazem parte da regra de negócios do sistema.



Figura – Diagrama de classes do pacote *br.com.financialmobile.security*

O pacote acima possui classes utilizadas para autenticação e controle de acesso no sistema. Este pacote possui apenas duas classes, CustomUserDetaisService que extende uma classe de autenticação do framework spring security e reimplementa dois métodos. A classe UsuarioSecurity e a classe utilizada para representação do usuário logado no sistema.



Figura – Diagrama de classes do pacote *br.com.financialmobile.security.util*

O pacote possui uma classe com um método estático que á utilizada para recuperar o usuário logado no sistema.



Figura – Diagrama de classes do pacote *br.com.financialmobile.service*

As classes do pacote acima são utilizadas como serviços no sistema, elas são uma interface de acesso da camada de visualização e é através delas que o sistema disponibiliza os serviços para os consumidores.



Figura – Diagrama de classes do pacote *br.com.financialmobile.servlets*

As classes do pacote acima são servlets utilizados pela página de login do sistema para autenticação e cadastro de usuários.



Figura – Diagrama de classes do pacote *br.com.financialmobile.servlets.xml.bean*

A classe do pacote acima é utilizada pelos servlets do sistema para parsear informações de Java para o formato JSON.



Figura – Diagrama de classes do pacote *br.com.financialmobile.util*

As classes do pacote acima são classes utilitárias utilizadas pelo sistema para operações em comum.



Figura – Diagrama de classes do pacote *br.com.financialmobile.webservices*

As classes do pacote acima representam os werbservices disponibilizados pelo sistema para integração com o sistema mobile.

## 4.4.2.2 Mobile



Figura – Diagrama de classes do pacote *br.com.financialmobile.exception*

As classes do pacote da Figura 28 são exceções que podem ser disparadas caso ocorra algum erro no processamento de alguma operação mobile.



Figura – Diagrama de classes do pacote *br.com.financialmobile.filters*

As classes do pacote acima são filtros utilizados para pesquisa no sistema mobile, como no celular não possuímos o recurso de banco de dados relacional é necessário criar filtros para consulta de informações.



Figura – Diagrama de classes do pacote br.com.financialmobile.model

O pacote acima possui as classes que representam nossas entidades dos objetos salvos na memória do celular.



Figura – Diagrama de classes do pacote br.com.financialmobile.model.to

O pacote acima contem classes que serão utilizadas apenas para transição de valores, objetos que representam um estado de uma operação, neste caso um relatório.



Figura – Diagrama de classes do pacote br.com.financialmobile.model.to

O pacote possui classes utilitárias referente as entidades do sistema, utilizadas para conversão de objetos.



Figura – Diagrama de classes do pacote br.com.financialmobile.service

O pacote acima possui as classes responsáveis por acessar os serviços esternos e internos da aplicação.



Figura – Diagrama de classes do pacote br.com.financialmobile.ui

O pacote acima possui as classes responsáveis pela interface do sistema no celular.



Figura – Diagrama de classes do pacote br.com.financialmobile.utils

As classes do pacote acima são classes utilitárias do sistema no celular, elas possuem métodos estáticos que podem ser invocados por qualquer classe do sistema.

## 4.5 Aplicativo Financial Mobile

Nesta sessão será apresentado o aplicativo em sua totalidade, suas telas, suas funcionalidades e como o mesmo deve ser utilizado. O aplicativo exige total iteração do usuário, toda ação do usuário gera uma reação imediata no sistema. Todas as telas referente as funcionalidades do sistema nos dois módulos serão exibidas a seguir.

### 4.5.1 Telas do Aplicativo e Utilização Módulo Web

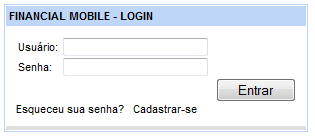


Figura - Tela de autenticação do sistema Web

A figura 36 mostra a tela de autenticação do sistema na internet. É para esta tela que o sistema redireciona o usuário quando o sistema é acessado pela primeira vez.

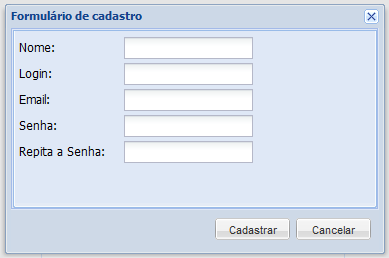


Figura – Tela de cadastro de usuários no sistema Web

Ao clicar no link Cadastrar-se exibido na Figura 35, o sistema exibe a tela da imagem acima para que o usuário preencha suas informações e efetue o cadastro no sistema.

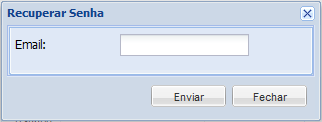


Figura – Tela de recuperação de senha no sistema Web

Caso o usuário não se lembre da senha basta que o mesmo clique no link Esqueceu sua senha?,exibido na Figura 35 para que o sistema exiba a tela da imagem anterior, possibilitando que o usuário recupere sua senha através de seu email.



Figura – Tela inicial do sistema na Web

Após o usuário efetuar a autenticação no sistema ele é redirecionado para a tela referente a imagem acima. Esta tela traz um resumos dos custos pagos nos últimos 30 dias e dos custos a pagar nos próximos 30 dias.



Figura – Tela de consulta de gastos

A imagem anterior representa a tela onde o usuário pode efetuar a consulta de gastos, filtrando pelos parâmetros exibidos no canto superior da tela.

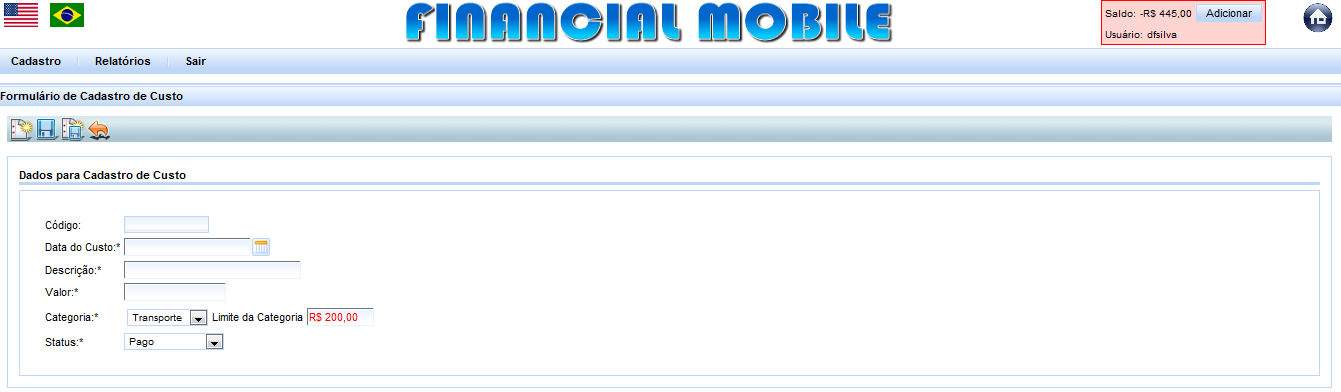


Figura – Tela de inclusão/alteração de gastos

A imagem acima exibe a tela de cadastro/alteração de gastos.

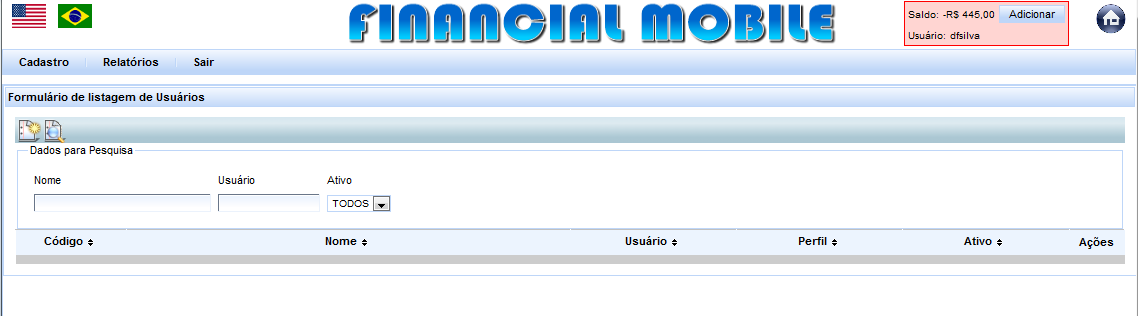


Figura – Tela de consulta de usuários

A imagem acima exibe a tela de consulta de usuários no sistema, caso o usuário que esteja acessando esta tela seja o administrador do sistema, ele consegue consultar todos os usuários, caso seja um usuário comum, ele consegue ver somente seu usuário.

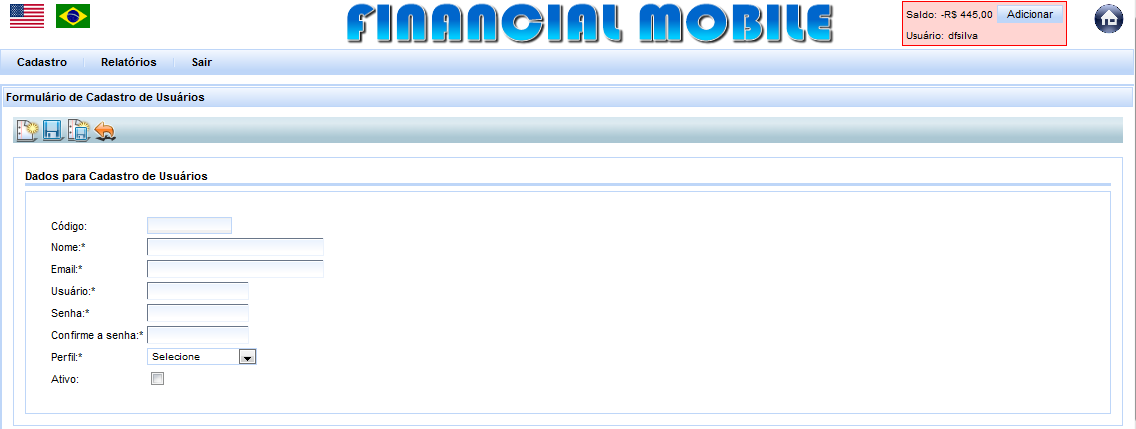


Figura – Tela de cadastro/alteração de dados do usuário

A imagem acima exibe a tela de cadastro/alteração de dados do usuário dentro do sistema.

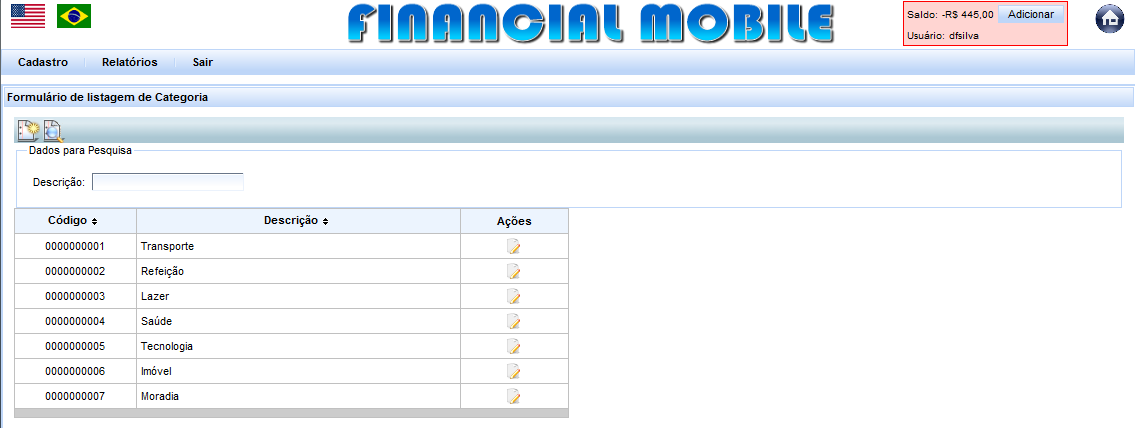


Figura – Tela de consulta de categorias

A imagem acima exibe a tela de consulta de categorias, somente usuários com perfil administrativo conseguem acessar esta funcionalidade.

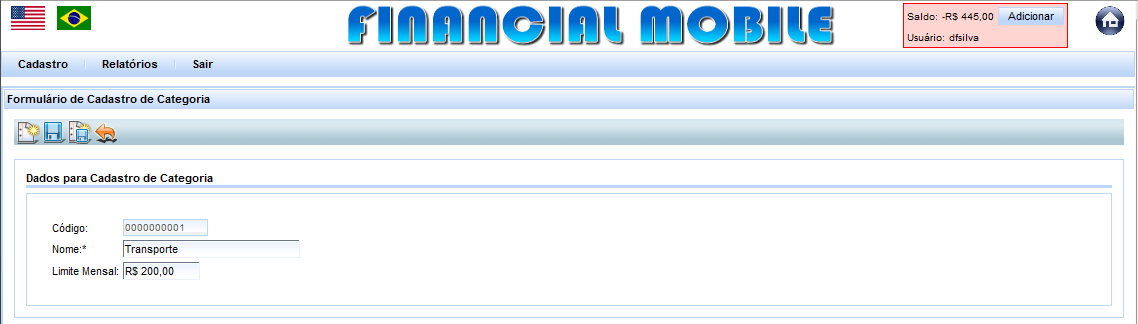


Figura – Tela de cadastro/alteração de categorias

A imagem acima exibe a tela de cadastro/alteração de categorias, somente usuários com perfil administrativo conseguem acessar esta funcionalidade.

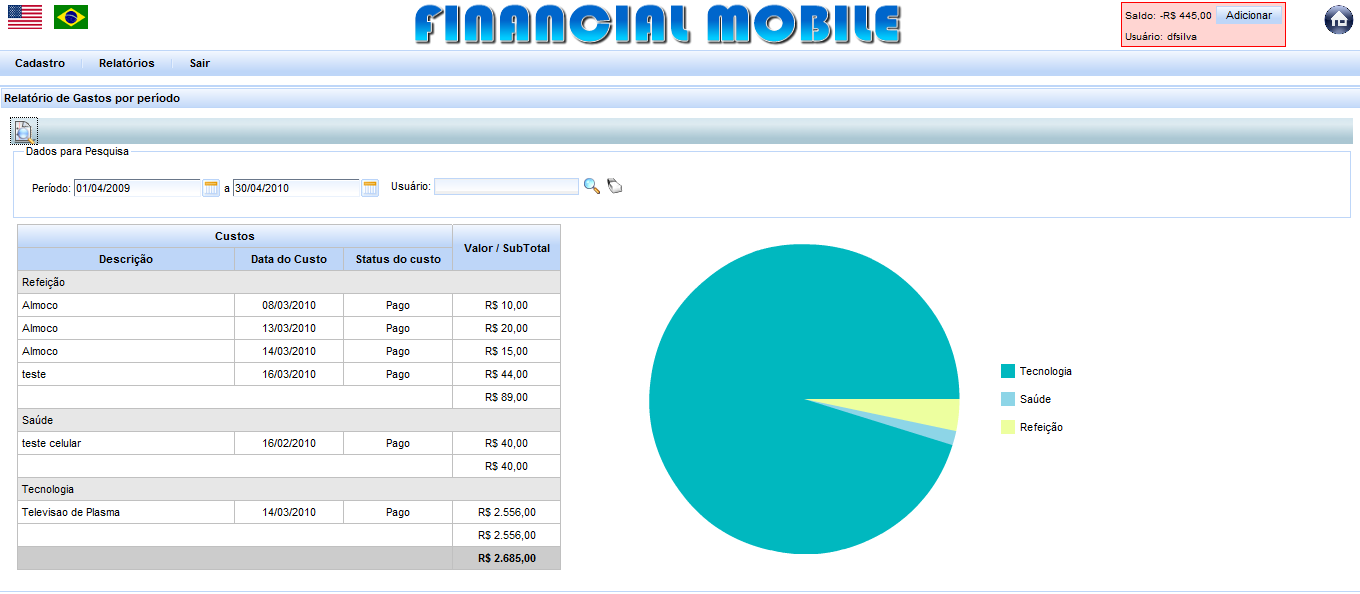


Figura – Tela de Relatório de Gastos

A imagem acima exibe a tela de relatório de gastos, através desta funcionalidade o usuário consegue ver de forma clara onde estão seus maiores gastos.

## 4.5.2 Telas do Aplicativo e Utilização Módulo Mobile (Emulador)



Figura – Tela de apresentação do sistema mobile

A imagem da Figura 47 exibe a tela de apresentação do sistema, esta tela é exibida para o usuário no momento em que ele acessa o aplicativo no celular.

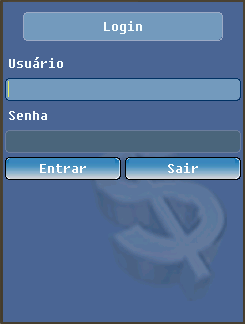


Figura – Tela de autenticação do sistema mobile

A imagem da Figura 48 exibe a tela de autenticação de usuários no sistema móbile, esta tela só é exibida para o usuário uma única vez após a instalação do aplicativo no celular.



Figura – Tela inicial do sistema

A imagem da Figura 49 exibe a tela inicial do sistema, esta tela é exibida para o usuário logo após sua autenticação no aplicativo.



Figura – Tela de inserção de gastos

A imagem da Figura 50 exibe a tela de inserção de gastos, através desta tela o usuário poderá informar gastos onde quer que esteja, independente de ter conexão com a Internet ou não.

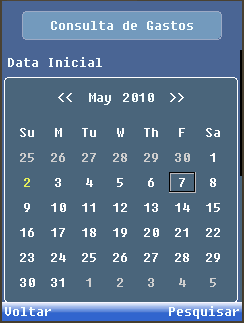


Figura – Tela de consulta de gastos

A imagem da Figura 51 exibe a tela de consulta de gastos.



Figura – Tela de exibição de resultados da consulta

A imagem da Figura 52 exibe a tela de listagem dos resultados da tela de consulta.

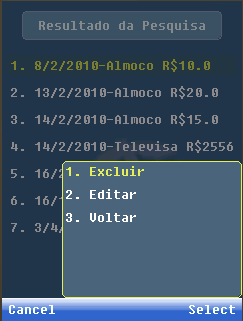


Figura – Tela de opções pós consulta

A imagem da Figura 53 exibe a tela de opções dos itens exibidos na consulta.

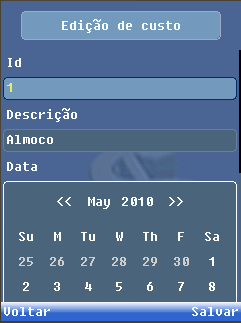


Figura – Tela de edição de gastos

A imagem da Figura 54 exibe a tela de edição de gastos.



Figura – Tela de consulta do relatório

A imagem da Figura 55 exibe a tela de consulta para geração do relatório de gastos.

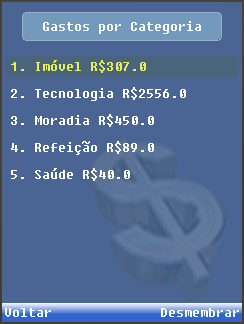


Figura – Relatório de gastos agrupado por categoria

A imagem da Figura 56 mostra a tela de exibição do relatório de gastos, os gastos são agrupados por categoria.

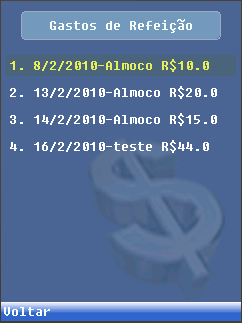


Figura – Custo do relatório desmembrado

A imagem da Figura 57 exibe a tela do relatório de gastos com os valores desmenbrados.

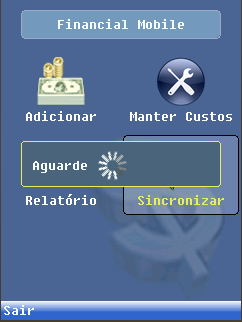


Figura – Tela inicial no momento do sincronismo

A imagem da Figura 58 exibe a tela inicial do sistema no momento em que o usuário pressiona a opção Sincronizar, neste momento o aplicativo no dispositivo móvel efetua um sincronismo das informações entre ele e o servidor de aplicações na internet.

## 4.6 Plano de Teste

O plano permite constatar a funcionalidade levantada nos requisitos e compreende a maior porcentagem do esforço técnico do processo de *software*. Nesse processo é testado tanto os casos de uso quanto especificações suplementares.

Esse plano define estratégias de teste, testando as funcionalidades propostas, a usabilidade, confiabilidade, configuração, segurança e instalação. Segue-se aqui, de forma adequada ao porte do projeto, o que a metodologia *Rational Unified Process®[[6]](#footnote-7)* descreve como plano de testes. O documento detalhado e o relatório de execução dos testes se encontram no Apêndice B.

Foi uma etapa muito importante no projeto. Ajudou na constatação de algumas falhas como um erro que ocorreu na funcionalidade de cadastro de usuários quando o ambiente foi alterado para outro servidor, demonstradas no relatório, e contribuiu para a melhoria das funcionalidades no geral.

## 4.7 Considerações

As ferramentas utilizadas para desenvolvimento do aplicativo são todas open source[[7]](#footnote-8), resultando neste sistema que também é open source. Apesar da desta primeira versão do sistema possui apenas as funcionalidades básicas, sua estrutura dividida em camadas, construído com princípios de orientação a serviços permite que sua expansão se torne fácil e prática.

Sua interface usual e bastante intuitiva não oferece dificuldades para o usuário, o sistema não requer instalação, requer apenas uma cópia de um arquivo para o dispositivo móvel, tornando-o bastante prático, funcional e seguro.

# 5 CONCLUSÃO

Diante do cenário atual em que vivemos diante de muitas opções sobre linguagens de programação e recursos para criação de sistemas, levando em consideração que este leque de opções tende a ficar maior com o tempo, a Arquitetura Orientada a Serviços se encaixa perfeitamente com este cenário, propiciando uma integração homogênea entre ambientes distintos, trazendo grandes benefícios e facilidades na construção de aplicações corporativas em ambientes distribuídos. A Arquitetura Orientada a Serviços apesar de estar consolidada no mercado ainda sofre algumas evoluções por parte das tecnologias que normalmente são utilizadas para a implementação desta arquitetura, um exemplo disso é o protocolo utilizado nos Web Services, que inicialmente tínhamos apenas o SOAP, hoje já temos outras opções e uma delas é o REST que foi comentada em um capítulo deste trabalho.

No que tange ao desenvolvimento do software proposto, os objetivos foram alcançados. A comunicação entre ao sistema web e o sistema mobile se mostrou eficaz e confiável com a utilização de Web Services, para o usuário isso trouxe grande comodidade levando até ele a possibilidade de utilizar o sistema em dois ambientes. A interface com o usuário nos dois ambientes se mostrou simples e intuitiva, a instalação do aplicativo no aparelho celular se mostrou muito prática bastando que o usuário copie um arquivo após fazer o download na internet.

A arquitetura utilizada na construção do sistema com frameworks estáveis e robustos proporciona ao sistema um ambiente confiável na sua execução do sistema.

O sistema também atinge o objetivo proposto de utilização de padrões presentes na Arquitetura Orientada a Serviços, no qual foi utilizado na forma de expor os serviços do sistema web para consumo destes através da aplicação mobile.

Em súmula, a aplicação apresenta de forma prática e confiável uma maneira simples para que o usuário possa controlar seus gastos do dia-dia, e quanto mais utilizada mais o usuário percebe o quanto é útil o sistema.

## 5.1 Trabalhos Futuros

Esta primeira versão do sistema apresenta apenas as funcionalidades básicas proposta pela idéia do software.

Como melhorias futuras pretende-se desenvolver um módulo específico para compras de supermercado, permitindo que o usuário armazene os produtos comprados e o valor total da compra, isso irá auxiliar para conferência do valor da compra com o valor cobrado no caixa e servirá como comparativo para compras futuras, podendo comparar preços de produtos e o preço total da compra.

Neste módulo quando o usuário finaliza a compra no supermercado, automaticamente aquele valor entra como um custo para o usuário.

# Referências

ECLIPSE, **IDE para Desenvolvimento Java**. Disponível em <http://www.eclipse.org>. Acesso em 21 de Abril de 2012.

GUEDES G., **UML: Uma abordagem prática.** 3.ed. São Paulo: Novatec, 2008. 336p.

GLAUCO A. L., Fagundes L. L., **Sockets em Java.** Disponível em: <http://www.exatec.unisinos.br/~glaucol/arquivos/Jogos-RedesDeComputadoresAula-Sockets.pdf>, acessado em 16 de Março de 2012.

HANSEN, Mark D., **SOA Using Java Web Services**, Pearson Education, Crawfordsville, p. 4-30, 2007

IBM, **Arquitetura Orientada a Serviços**. Disponível em <http://www.pr.senai.br/posgraduacao/uploadAddress/Introducao%20ao%20SOA[31574][4843].pdf>. Acesso em 10 de Maio de 2012.

JUNIOR, Alfredo L. Santos, **Integração de Sistemas com Java**, Brasport, Rio de Janeiro, V.1, p. 80-108, 2007.

JBOSS EMBEDDED, **Api para containers web disporem de serviços de um servidor de aplicações**. Disponível em <http://community.jboss.org/wiki/EmbeddedJBoss> Acesso em 21 de Abril de 2012.

JUSUTTIS, M., NICOLAI **SOA in Practice** 1.ed. Sebastopol: O’Reilly, 2007. 5p.

KUROSE, F., J., Ross, K., W., **Computer networking : a top-down approach featuring the internet,** Addison-Wesley, Boston, 3 Ed., p.68-92, 2005.

LEONARDO R. NUNES, **Sockets em Java**. Disponível em: <http://www.sumersoft.com/publicacoes/SocketsEmJAVA.pdf>, acessado em 16 de Março de 2012.

MENDES, Marco A. S., **Estratégias de Integração e Arquitetura SOA com a Plataforma Java.** Disponível em: <http://www.marcomendes.com/ArquivosBlog/EstrategiaIntegracao\_SOA\_PlataformaJava\_templateMGJUG.pdf>, acessado em 16 de Março de 2012.

NETBEANS, **IDE para Desenvolvimento Java.** Disponível em <http://netbeans.org/> Acesso em 21 de Abril de 2012.

ROSEN, M., LUBLINSKY, B., SMITH,T.,K., BALCER.J.MARC., **Service-Oriented Architecture and Disign Strategies** 1.ed. Indianaplis: Wiley, 2008. 46p.

SANDOVAL, Jose, **RESTful Java Web Services**, Packt Publishing Ltd, Olton, p.5 – 20, 2009.

SILVA, Filipe, M., **SOA – Arquitetura Orientada a Serviços**, Universidade de São Paulo, Instituto de Matemática e Estatística, Prof. Dr. Fábio Kon, 2006.

SUN, Sun Microsystems, **Java Web Services Tutorial**, Sun Microsystems,, Santa Clara,, p. 120, 2006

SUN, **Sun Microsystems**. Disponível em <http://www.sun.com>. Acesso em 21 de Abril 2012.

TOMCAT, **Container Web para aplicações Java**. Disponível em <http://tomcat.apache.org/>. Acesso em 21 de Abril de 2012.

W3, **SOAP ESPECIFICATION.** Disponível em: <http://www.w3.org/TR/2007/REC-soap12-part0-20070427/#L4697>, acessado em 10 de Abril de 2012.

WIKIPEDIA, **Wikipedia, the free encyclopedia**. Disponível em: < http://en.wikipedia.org/>. Acesso em 21 de Abril de 2012.

CHAPPELL, David; JEWEL, Tyler. Java Web Services. [S.I.]: O’Reilly, 2002.

OASIS. Web Services Security: Soap Message Security 1.0. Disponível em: <http://www.oasis-open.org/committees/download.php/5531/oasis-200401-wss-soapmessage-security-1.0.pdf>. Acesso em 01 de Junho de 2012.

ROSENBERG, Jothy.; REMY, David. Securing Web Services with WS-Security. USA. Sams Publishing, 2004.

# APÊNDICE A – DESCRIÇÃO DOS CASOS DE USO

# SUMÁRIO

[1 Descrição dos Casos de Uso 74](#_Toc262242545)

[1.1 RF01-Cadastro de Usuario 74](#_Toc262242546)

[1.2 RF02-Recupera Senha 74](#_Toc262242547)

[1.3 RF03-Efetua Login 75](#_Toc262242548)

[1.4 RF04-Manter Gastos 75](#_Toc262242549)

[1.5 RF05-Relatorio de Gastos Simples 76](#_Toc262242550)

[1.6 RF06-Manter Categorias 77](#_Toc262242551)

[1.7 RF07-Manter Usuarios 78](#_Toc262242552)

[1.8 RF08-Efetuar Login 79](#_Toc262242553)

[1.9 RF09-Adicionar Gasto 79](#_Toc262242554)

[1.10 RF10-Manter Gastos 80](#_Toc262242555)

[1.11 RF11-Relatorio de Gastos 80](#_Toc262242556)

[1.12 RF12-Sincronizar Dados 80](#_Toc262242557)

[1.13 RF13-Limitar Gastos por Categoria 81](#_Toc262242558)

[1.14 RF14-Relatório de Média de Gastos por Categoria 81](#_Toc262242559)

# Descrição dos Casos de Uso

* 1. UC01-Cadastro de Usuario

Este caso de uso descreve o processo de cadastramento de usuário.

|  |
| --- |
|  |
| **Cenário Alternativo**  **Campos Obrigatórios**   1. O usuário acessa o sistema 2. O usuário clica no link cadastrar-se 3. O sistema exibe uma janela solicitando os dados para cadastramento do usuario 4. O usuário não entra com algum dos seguintes dados: Nome, Login, Senha, Confirmação de senha e e-mail. 5. O usuário pressiona o botao salvar. 6. O sistema exibe uma mensagem solicitando ao usuário que informe os campos obrigatórios. |
| **Cenário Principal**  **Cenário Principal**   1. O usuário acessa o sistema 2. O usuário clica no link cadastrar-se 3. O sistema exibe uma janela solicitando os dados para cadastramento do usuario 4. O usuário entra com os seguintes dados: Nome, Login, Senha, Confirmação de senha e e-mail. 5. O usuário pressiona o botao salvar. 6. O sistema exibe uma mensagem informando que os dados foram salvos com sucesso. |

* 1. UC02-Recupera Senha

Este caso de uso descreve o processo de recuperação de senha de um usuário.

|  |
| --- |
|  |
| **Cenário Principal**  **Cenário Principal**   1. O usuário acessa o sistema 2. O usuário pressiona o botão recuperar senha. 3. O sistema exibe um formulário para que o usuário informe o email cadastrado. 4. O usuário informa o email e pressiona o botao enviar. 5. O sistema exibe uma mensagem informando que a senha foi enviada para o email informado. |
| **Cenário Alternativo**  **Email nao cadastrado**   1. O usuário acessa o sistema 2. O usuário pressiona o botão recuperar senha. 3. O sistema exibe um formulário para que o usuário informe o email cadastrado. 4. O usuário informa o email e pressiona o botao enviar. 5. O sistema exibe uma mensagem informando que o email informando não possui cadastro no banco de dados. |

* 1. UC03-Efetua Login

Este caso de uso descreve o processo de efetuar login no sistema.

|  |
| --- |
|  |
| **Cenário Principal**  **Cenário Principal**   1. O usuário acessa o sistema 2. O usuário entra com os seguintes dados: Login e Senha. 3. O usuário pressiona o botão entrar. 4. O sistema redireciona o usuário para a página inicial do sistema. |
| **Cenário Alternativo**  **Usuário ou senha inválidos**   1. O usuário acessa o sistema 2. O usuário entra com os seguintes dados: Login e Senha. 3. O usuário pressiona o botão entrar. 4. O sistema exibe uma mensagem informando ao usuário que houve um erro na autenticação do usuário. |

* 1. UC04-Manter Gastos

Este caso de uso define o processo de cadastro e manutenção de custos.

|  |
| --- |
|  |
| **Cenário Alternativo**  **Campos Obrigatórios**   1. O usuário acessa o sistema 2. O usuário efetua o login no sistema 3. O usuário clica no link manter custos 4. O usuário clica no botão adicionar custo 5. O sistema exibe uma tela para que o usuáro informe os dados do gasto 6. O usuário não entra com algum dos seguintes dados: Valor do Gasto, Descricao do Gasto, Data do Gasto, Categoria do Gasto, Status do gasto. 7. O usuário pressiona o botao salvar 8. O sistema exibe uma mensagem informando ao usuário os campos obrigatórios que não foram informados. |
| **Cenário Principal**  **Cenário Principal**   1. O usuário acessa o sistema 2. O usuário efetua o login no sistema 3. O usuário clica no link manter custos 4. O usuário clica no botão adicionar custo 5. O sistema exibe uma tela para que o usuáro informe os dados do gasto 6. O usuário entra com o os seguintes dados: Valor do Gasto, Descricao do Gasto, Data do Gasto, Categoria do Gasto, Status do gasto. 7. O usuário pressiona o botao salvar 8. O sistema exibe uma mensagem informando ao usuário que os dados foram salvos com sucesso. |
| **Cenário Alternativo**  **Editar Custo**   1. O usuário acessa o sistema 2. O usuário efetua o login no sistema 3. O usuário clica no link manter custos 4. O usuário entra com um dos seguintes campos para pesquisa: data inicial e final, descrição do custo, categoria do custo, status do custo. 5. O usuário pressiona o botão pesquisar 6. O sistema lista os custo em uma tabela de acordo com a pesquisa. 7. O usuário clica no botão editar referente ao custo exibido na tabela. 8. O sistema redireciona o usuário para a tela de edição. 9. O usuário altera os dados e pressiona no botão salvar. 10. O sistema exibe uma mensagem informando que os dados foram atualizados com sucesso. |
| **Cenário Alternativo**  **Excluir custo**   1. O usuário acessa o sistema 2. O usuário efetua o login no sistema 3. O usuário clica no link manter custos 4. O usuário entra com um dos seguintes campos para pesquisa: data inicial e final, descrição do custo, categoria do custo, status do custo. 5. O usuário pressiona o botão pesquisar 6. O sistema lista os custo em uma tabela de acordo com a pesquisa. 7. O usuário clica no botão excluir referente ao custo exibido na tabela. 8. O sistema exibe uma mensagem solicitando a confirmação da exclusão do item. 9. O usuário confirma a exclusão. 10. O sistema exclui o item e informa o usuário removendo-o da tabela de pesquisa. |

* 1. UC05-Relatorio de Gastos Simples

Este caso de uso define o processo de geracao de relatório de gastos no sistema

|  |
| --- |
|  |
| **Cenário Principal**  **Cenário Principal**   1. O usuário efetua login no sistema [RF03] 2. O usuário clica no link relatório de gastos 3. O sistema exibe uma tela com os filtros necessários para geração do relatório. 4. O usuário preenche os campos data inicial, data final  e descrição do gasto. 5. O usuário pressiona o botão buscar 6. O sistema apresenta em uma tabela os valores gastos dentro daquele periodo e com aquela descrição, sumarizando os valores e agrupando por categoria. |

* 1. UC06-Manter Categorias

Este caso de uso descreve o processo de adição e manutenção de categorias em um sistema.

|  |
| --- |
|  |
| **Cenário Alternativo**  **Campos obrigatórios**   1. O usuário acessa o sistema 2. O usuário efetua o login no sistema[RF03] 3. O usuário clica no link manter categorias 4. O usuário clica no botão adicionar categoria 5. O sistema exibe uma tela para que o usuáro informe os dados de cadastro. 6. O usuário não entra com algum dos seguintes dados: descrição da categoria. 7. O usuário pressiona o botão salvar 8. O sistema exibe uma mensagem informando ao usuário os campos obrigatórios que não foram informados. |
| **Cenário Principal**  **Cenário Principal**   1. O usuário acessa o sistema 2. O usuário efetua o login no sistema 3. O usuário clica no link manter categorias 4. O usuário clica no botão adicionar categoria 5. O sistema exibe uma tela para que o usuáro informe os dados da categoria 6. O usuário entra com o os seguintes dados: descrição do gasto. 7. O usuário pressiona o botao salvar 8. O sistema exibe uma mensagem informando ao usuário que os dados foram salvos com sucesso. |
| **Cenário Alternativo**  **Editar categoria**   1. O usuário acessa o sistema 2. O usuário efetua o login no sistema 3. O usuário clica no link manter categoria 4. O usuário entra com um dos seguintes campos para pesquisa: descricao da categoria 5. O usuário pressiona o botão pesquisar 6. O sistema lista os itens em uma tabela de acordo com a pesquisa. 7. O usuário clica no botão editar referente ao item exibido na tabela. 8. O sistema redireciona o usuário para a tela de edição. 9. O usuário altera os dados e pressiona no botão salvar. 10. O sistema exibe uma mensagem informando que os dados foram atualizados com sucesso. |
| **Cenário Alternativo**  **Excluir Categoria**   1. O usuário acessa o sistema 2. O usuário efetua o login no sistema 3. O usuário clica no link manter categoria 4. O usuário entra com um dos seguintes campos para pesquisa: descricao da categoria 5. O usuário pressiona o botão pesquisar 6. O sistema lista os itens em uma tabela de acordo com a pesquisa. 7. O usuário clica no botão excluir referente ao item exibido na tabela. 8. O sistema solicita a confirmação da exclusão 9. O usuário confirma a exclusão 10. O sistema remove o item excluido. |

* 1. UC07-Manter Usuários

Este caso de uso permite a inclusão e manutenção de usuários.

|  |
| --- |
|  |
| **Cenário Alternativo**  **Campos Obrigatórios**   1. O usuário acessa o sistema 2. O usuário efetua o login no sistema[RF03] 3. O usuário clica no link manter usuários 4. O usuário clica no botão adicionar usuário 5. O sistema exibe uma tela para que o usuáro informe os dados do usuário. 6. O usuário não entra com algum dos seguintes dados: Nome, Login, Senha, Confirmação de senha e e-mail. 7. O usuário pressiona o botao salvar. 8. O sistema exibe uma mensagem solicita que o usuário informe os campos obrigatórios. |
| **Cenário Principal**  **Cenário Principal**   1. O usuário acessa o sistema 2. O usuário efetua o login no sistema[RF03] 3. O usuário clica no link manter usuários 4. O usuário clica no botão adicionar usuário 5. O sistema exibe uma tela para que o usuáro informe os dados do usuário. 6. O usuário entra com o os seguintes dados: nome do usuário, email do usuário, login, senha e ativo. 7. O usuário pressiona o botao salvar 8. O sistema exibe uma mensagem informando ao usuário que os dados foram salvos com sucesso. |
| **Cenário Alternativo**  **Editar Usuário**   1. O usuário acessa o sistema 2. O usuário efetua o login no sistema 3. O usuário clica no link manter usuários 4. O usuário entra com um dos seguintes campos para pesquisa: Nome do usuário, login, ativo. 5. O usuário pressiona o botão pesquisar 6. O sistema lista os itens em uma tabela de acordo com a pesquisa. |
| **Cenário Alternativo**  **Excluir usuário**   1. O usuário acessa o sistema 2. O usuário efetua o login no sistema 3. O usuário clica no link manter usuários 4. O usuário entra com um dos seguintes campos para pesquisa: Nome do usuário, login, ativo. 5. O usuário pressiona o botão pesquisar 6. O sistema lista os itens em uma tabela de acordo com a pesquisa. 7. O usuário clica no botão excluir referente ao item exibido na tabela. 8. O sistema solicita a confirmação da exclusão 9. O usuário confirma a exclusão 10. O sistema remove o item excluido. |

* 1. UC08-Efetuar Login

Este caso de uso permite o usuário efetuar login no aparelho celular

|  |
| --- |
|  |
| **Cenário Alternativo**  **Campos Obrigatórios**   1. O usuário acessa o sistem 2. O usuário não informa um dos seguintes campos: Usuário ou Senha. 3. O usuário pressiona o botão entrar 4. O sistema exibe uma mensagem informando que o campo usuário e senha são de preenchimento obrigatório. |
| **Cenário Principal**  **Cenário Principal**   1. O usuário acessa a aplicação no celular 2. O usuário informa o campo usuário e senha 3. O usuário clica no botão entrar |

* 1. UC09-Adicionar Gasto

Este caso de uso permite o usuário lançar um custo no celular

|  |
| --- |
|  |
| **Cenário Principal**  **Cenário Principal**   1. O usuário pressiona o botão lançar gastos 2. O sistema exibe uma tela para que o usuáro informe os dados do gasto 3. O usuário entra com o os seguintes dados: Valor do Gasto, Descricao do Gasto, Data do Gasto, Categoria do Gasto, Status do gasto. 4. O usuário pressiona o botao salvar 5. O sistema exibe uma mensagem informando ao usuário que os dados foram salvos com sucesso. |

* 1. UC10-Manter Gastos

Este caso de uso permite ao usuário remover, atualizar e editar um gasto.

|  |
| --- |
|  |
| **Cenário Principal**  **Cenário Principal**   1. O usuário acessa o sistema 2. O usuário pressiona o botão manter custos 3. O sistema exibe um formulário com os campos data inicial e data final. 4. O usuário escolha a data inicial e data final e pressiona o command buscar. 5. O sistema exibe uma lista de custos naquele periodo 6. O usuário seleciona o item desejado e clica no botao editar 7. O sistema redireciona o usuário para uma tela com os dados do custo prontos para edição 8. O usuário altera os dados necessário e clica em salvar 9. O sistema emite uma mensagem informando que os dados foram salvos com sucesso. |
| **Cenário Alternativo**  **Exclusao de gastos**   1. O usuário acessa o sistema 2. O usuário pressiona o botão manter custos 3. O sistema exibe um formulário com os campos data inicial e data final. 4. O usuário escolha a data inicial e data final e pressiona o command buscar. 5. O sistema exibe uma lista de custos naquele periodo 6. O usuário seleciona o item desejado e clica no botao exluir. 7. O sistema exibe uma mensagem solicitando a confirmação da exclusão do custo 8. O usuário confirma a exclusão 9. O sistema então passa este custo para o status excluido e na próxima sincronização o sistema elimina esta custo do sistema. |

* 1. UC11-Relatório de Gastos

Este caso de uso permite que o usuário gere um relatório de gastos do sistema

|  |
| --- |
|  |
| **Cenário Principal**  **Cenário Principal**   1. O usuário acessa o sistema e clica no botão relatórios 2. O sistema exibe uma tela com os filtros data inicial e data final. 3. O usuario informa a data inicial e final  e pressiona o command pesquisar. 4. O sistema exibe um relatório em forma de gráficos na tela do celular com os custos gastos naquele período informado, agrupando por categorias. |

* 1. UC12-Sincronizar Dados

Este caso de uso permite que o usuário sincronize os dados do celular com o servidor de aplicações.

|  |
| --- |
|  |
| **Cenário Alternativo**  **Cenário Principal**   1. O usuário acessa o sistema 2. O usuário pressiona o botão sincronizar gastos 3. O sistema solicita o usuário que confirme a conexão com a internet 4. O usuário efetua a confirmação 5. O sistema então efetua a atualização dos custos, categorias, status e informações do usuário com o servidor web. 6. O sistema exibe uma mensagem informando que a atualização foi bem sucessida. |
| **Cenário Alternativo**  **Erro na atualização**   1. O usuário acessa o sistema 2. O usuário pressiona o botão sincronizar gastos 3. O sistema solicita o usuário que confirme a conexão com a internet 4. O usuário efetua a confirmação 5. O sistema então efetua a atualização dos custos, categorias, status e informações do usuário com o servidor web. 6. O sistema exibe uma mensagem informando que houve um erro ao tentar efetuar as atualizações, informando também o código de erro para que fique mais fácil identificar o tipo de erro que ocorreu. |

* 1. UC13-Limitar Gastos por Categoria

Este caso de uso permite que o usuário limite seus gastos por categoria.

|  |
| --- |
|  |
| **Cenário Principal**  **Cenário Principal**   1. O usuário acessa suas categorias de gastos. 2. O usuário insere um valor x mensal que poderá ser gasto naquela categoria 3. Toda vez que o usuário lançar um novo custo o sistema irá verificar se este valor cadastrado já foi ultrapassado, caso sim ele exibirá um alerta para o usuário informando que o valor foi ultrapassado neste mes e solicitando a confirmação para o cadastro. |

* 1. UC14-Relatório de Média de Gastos por Categoria

Este caso de uso permite que um usuário gere um relatório de gastos com a média de gastos nos ultimos meses dividido por categorias.

|  |
| --- |
|  |
| **Cenário Principal**  **Cenário Principal**   1. O usuário efetua login no sistema [RF03] 2. O usuário clica no link relatório de média de Gastos 3. O sistema exibe uma tela com os filtros necessários para geração do relatório. 4. O usuário preenche os campos data inicial, data final  e descrição do gasto. 5. O usuário pressiona o botão buscar 6. O sistema apresenta em uma tabela e um grafico com a média mensal dos gastos no período selecionado. |

APÊNDICE B – PLANO DE TESTES

# SUMÁRIO

[Introdução 79](#_Toc261384133)

[Objetivo 79](#_Toc261384134)

[Informações Detalhadas 79](#_Toc261384135)

[Escopo 79](#_Toc261384136)

[Identificação do Projeto 80](#_Toc261384137)

[Requisitos de Teste 80](#_Toc261384138)

[Testes de Integridade das Informações 80](#_Toc261384139)

[Teste Funcional e Interface 81](#_Toc261384140)

[Teste de Configuração e Segurança 81](#_Toc261384141)

[Teste de Instalação 81](#_Toc261384142)

[Estratégia de Testes 82](#_Toc261384143)

[Tipos de Teste 82](#_Toc261384144)

[Teste de Integridade das Informações 82](#_Toc261384145)

[Teste Funcional e Interface 82](#_Toc261384146)

[Teste de Configuração e Segurança 83](#_Toc261384147)

[Teste de Instalação 83](#_Toc261384148)

Introdução

Objetivo

O teste de software corresponde a mais alta porcentagem de esforço técnico do processo de software. O objetivo do teste de software é descobrir erros. Para alcançar esse objetivo, uma série de passos de testes – testes de unidade, integração, validação e de sistema – é planejada e executada. Os testes de unidade e de integração concentram-se na verificação funcional de um componente e na incorporação de componentes em uma estrutura de programa. Os testes de validação demonstram a rastreabilidade aos requisitos do software e os testes de sistema validam o software depois de ter sido incorporado a um sistema maior.

Cada passo de teste é realizado por uma série sistemática de técnicas de teste que ajudam no projeto de casos de teste. Com cada passo de teste, o nível de abstração no qual o software é considerado é ampliado.

Informações Detalhadas

O Financial Mobile é um sistema para controle de finanças pessoais que propicia ao usuário a comodidade de utilizar o sistema em duas interfaces, através de um browser o usuário pode acessar o endereço eletrônico do sistema e utilizar suas funcionalidades, assim como efetuar o download do aplicativo para celular, ambas as interfaces possuem as mesmas funcionalidades, visando dar ao usuário total liberdade para utilizar o sistema onde quer que ele esteja.

Escopo

Todos os casos de uso do sistema requerem interação direta com o usuário e o projeto é orientado a objetos e necessita de um servidor web. Tendo em vista estas premissas a estratégia é a realização de testes no sistema acompanhando os requisitos.

Utilizando esta abordagem espera-se verificar se a operabilidade do aplicativo atende aos requisitos, gerando um sistema estável e seguro para a utilização.

Identificação do Projeto

A tabela abaixo identifica a documentação, bem como sua disponibilidade. Ela é utilizada para desenvolver o plano de teste:

|  |  |  |
| --- | --- | --- |
| Documento  (e versão / data) | Criado ou Disponível | Observações |
| Descrição dos Casos de Uso | SIM |  |
| Diagrama de Classes | SIM |  |
| Roteiro de Instalação | SIM |  |
| Manual do Usuário | SIM |  |

Requisitos de Teste

A seguir são identificados os itens (casos de uso e requisitos funcionais) que foram identificados como objetivos de teste.

**Casos de Uso**

|  |
| --- |
| Nome do Caso de Uso |
| UC01-Cadastro de Usuario |
| UC02-Recupera Senha |
| UC03-Efetua Login |
| UC04-Manter Gastos |
| UC05-Relatorio de Gastos Simples |
| UC06-Manter Categorias |
| UC07-Manter Usuarios |
| UC08-Efetuar Login - Celular |
| UC09-Adicionar Gasto - Celular |
| UC10-Manter Gastos - Celular |
| UC11-Relatorio de Gastos - Celular |
| UC12-Sincronizar Dados - Celular |
| UC13-Limitar Gastos por Categoria |
| UC14-Relatório de Média de Gastos por Categoria |

Testes de Integridade das Informações

Como a base de dados do sistema web é o MYSQL e no celular os dados são armazenados utilizando arquivos binários, é necessário verificar se as informações estão ficando coerentes após o sincronismo entre as duas plataformas:

* Verificar se as informações gravadas nas duas plataformas foram sincronizadas corretamente.

Teste Funcional de Interface

* Verificar se as sugestões são exibidas na tela no tempo determinado pelo usuário.
* Verificar se as configurações gravadas na tela de configurações refletem no sistema.
* Verificar se as telas estão sendo exibidas corretamente
* Verificar se as informações das buscas estão coerentes com o filtro
* Verificar se a interface apresenta problemas de usabilidade.

Teste de Configuração e Segurança

* Verificar se o sistema funciona de forma esperada em celulares com JME disponível.
* Verificar se há discrepâncias de visualização em diferentes tipos de monitor, resolução de tela e configuração da área de trabalho.
* Verificar a compatibilidade com os principais browsers(Internet Explorer, Firefox, Chrome)
* Verificar se o controle de acesso e autenticação estão corretos
* Verificar se dados são expostos de forma segura e se algum dos métodos expõe algum tipo de falha em segurança.

Teste de Instalação

* Verificar se o sistema funciona após seguir o roteiro de instalação fornecido nas condições e requisitos mínimos de hardware e software especificados.

# 

Estratégia de Testes

Tipos de Teste

Teste de Integridade das Informações

Objetivo do Teste:

• Garantir que as informações sincronizadas entre o dispositivo móvel e o sistema Web ocorra sem distorção das mesmas.

Técnica:

• Efetuar inserção de dados através do sistema web e do sistema móbile, após isto efetuar um sincronismo das informações através do sistema móbile e observar se os dados estão coerentes.

Critérios de Conclusão:

• Todos os métodos e procedimentos de sincronismo das informações funcionam conforme projetado e sem corromper os dados.

Considerações Especiais:

• Os processos deverão ser disparados manualmente.

• Deve ser feito teste com um número baixo de inserções, 2 registros em cada ambiente, de forma que possa ser mais fácil verificar se houve alguma incoerência nas informações.

Teste Funcional de Interface

Objetivo do Teste:

• Garantir a funcionalidade adequada de objetivo do teste, incluindo navegação, entrada de dados, processamento e recuperação.

Técnica:

• Executar cada caso de uso, fluxo de caso de uso ou método usando dados válidos e inválidos, a fim de verificar o seguinte:

• Os resultados esperados ocorrerão quando forem usados dados válidos.

• As mensagens de erro/aviso apropriadas serão exibidas quando forem usados dados inválidos.

• Cada regra será adequadamente aplicada.

Critérios de Conclusão:

• Todos os testes planejados foram executados.

• Todos os defeitos identificados foram abordados.

Considerações Especiais:

• Nenhuma

Teste de Configuração e Segurança

Objetivo do Teste:

• Verificar se o controle de acesso e segurança do sistema estão de acordo com os requisitos e a compatibilidade do sistema com os ambientes propostos.

Técnica:

• Realizar testes com usuário de perfis diferentes e verificar se seus acessos estão de acordo com a documentação.

• Realizar testes com acessando o sistema via web em diversos browsers e resoluções diferentes.

• Realizar testes no sistema móbile em celulares distintos com resoluções diferentes.

Critérios de Conclusão:

• O resultado dos testes deve estar em conformidade com os casos de uso.

Considerações Especiais:

• Nenhuma.

Teste de Instalação

Objetivo do Teste:

• Verificar se as instruções incluídas no manual de instalação são suficientes para a correta instalação do software.

Técnica:

• Testar a instalação do módulo móbile em celulares diferentes e verificar se o sistema ficou acessível.

Critérios de Conclusão:

• O sistema executa suas transações corretamente.

Considerações Especiais:

• Nenhuma

Roteiro de Simulação de Testes

Testes de Integridade das Informações

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Roteiro de Simulação de Testes** | | | | | | |
| **Descrição do Ambiente** | |  | | | | |
| **Configuração da Máquina** | | Sistema Operacional Windows 7 Core 2 Duo 2.2 GHz 4GB 64 bits | | | | |
| **Período de Execução** | | 26/04/2010 à 28/04/2010 | | | | |
| **Categoria** | | Teste de Integridade das Informações | | | | |
| Ordem | Nome da Simulação | | Simulação | Resultado Esperado | Situação | Observação |
| 1 | **Browser / Celular** | | {O usuário acessa o browser de sua preferência}   1. O usuário acessa o endereço eletrônico do sistema. 2. O usuário efetua login no sistema 3. O usuário efetua o lançamento de um custo 4. O usuário acessa o sistema no seu celular 5. O usuário acessa o aplicativo FinancialMobile 6. O usuário efetua o lançamento de um gasto 7. O usuário efetua o sincronizmo das infomarções no celular. 8. O usuário volta no browser e efetua uma consulta. 9. O usuário vai até o sistema no celular e executa a mesma consulta na web. | É esperedo que o sistema exiba o mesmo resulta idêntico nas duas plataformas | Passou |  |

Teste Funcional e de Interface

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Roteiro de Simulação de Testes** | | | | | | | |
| **Descrição do Ambiente** | |  | | | | | |
| **Configuração da Máquina** | | Sistema Operacional Windows 7 Core 2 Duo 2.2 GHz 4GB 64 bits | | | | | |
| **Período de Execução** | | 27/04/2010 à 28/04/2010 | | | | | |
| **Categoria** | | Teste de Integridade das Informações | | | | | |
| Ordem | Nome Caso de Uso | | Nome da Simulação | Simulação | Resultado Esperado | Situação | Observação |
| 1 | RF01-Cadastro de Usuario | | Cadastro de Usuário | 1. O usuário acessa o sistema utilizando um browser 2. O sistema redireciona o usuário para a tela de autenticação 3. O usuário clica no botão cadastrar 4. O sistema abre uma janela para cadastro do usuário 5. O usuário preenche os dados e clica no botão cadastrar. | O sistema exibe uma mensagem informando que o usuário foi cadastrado com sucesso. | Falhou | Endereço do servlet estava incorreto gerando um erro no cadastramento. Corrigido. |
| 2 | RF02-Recupera Senha | | Recuperar Senha | 1. O usuário acessa o sistema utilizando um browser 2. O sistema redireciona o usuário para a tela de autenticação 3. O usuário clicar no botão recuperar senha 4. O sistema abre uma janela para que o usuário informe seu email. 5. O usuário informa seu email e pressiona o botão enviar. 6. O sistema envia um email para o usuário com sua senha. | O usuário recebe sua senha através de seu email. | Passou |  |
| 3 | RF03-Efetua Login | | Efetuar Login | 1. O usuário acessa o sistema utilizando um browser 2. O sistema redireciona o usuário para a tela de autenticação 3. O usuário informa seu usuário e senha e clica no botão ok. | O sistema redireciona o usuário para a tela inicial do sistema. | Passou |  |
| 4 | RF04-Manter Gastos | | Manter Gastos | 1. O usuário executa os passos do item 3 2. O usuário clica no menu manter gastos 3. O sistema exibe o usuário para a tela de consulta/adição de gastos. 4. O usuário clica no botão novo 5. O sistema redireciona o usuário para a tela de cadastro de custo. 6. O usuário preenche os campos e clica no botão salvar. | O sistema salva os dados do custo informado. | Passou |  |
| 5 | RF05-Relatorio de Gastos Simples | | Relatório de Gastos na Web | 1. O usuário executa os passos do item 3 2. O usuário acessa o menu de relatório de gastos 3. O usuário informa os dados de filtro e pressiona no botão gerar relatório 4. O sistema gera o relatório de gastos | O sistema exibe para o usuário o relatório de gastos | Passou |  |
| 6 | RF06-Manter Categorias | | Manter Categorias | 1. O usuário executa os passos do item 3 2. O usuário acessa o menu de manter categoria 3. O sistema exibe para o usuário a tela de consulta/adição de categorias. 4. O usuário clica no botão nova categoria 5. O sistema exibe para o usuário a tela de adição de categorias. 6. O usuário preenche os dados e clica no botão salvar. | O sistema efetua o cadastro da categoria informada | Passou |  |
| 7 | RF07-Manter Usuarios | | Manter Usuário | 1. O usuário executa os passos do item 3 2. O usuário acessa o menu manter usuário 3. O sistema exibe a tela de consulta/edição de usuários. 4. O usuário efetua a busca 5. O sistema exibe o usuário logado caso o perfil do usuário não tenha permissão para editar outros usuários. 6. O usuário clica no usuário para edição 7. O sistema exibe uma tela com os dados do usuário. 8. O usuário altera os dados e clica no botão salvar. | O sistema efetua a alteração dos dados do usuário. | Passou |  |
| 8 | RF08-Efetuar Login – Celular | | Login no Celular | 1. O usuário acessa o sistema pelo celular 2. O sistema exibe a tela de login 3. O usuário informa o login e senha e pressiona o botão logar. 4. O sistema exibe a tela inicial do sistema | O usuário acessa o aplicativo no celular. | Passou |  |
| 9 | RF09-Adicionar Gasto – Celular | | Adicionar Gasto no Celular | 1. O usuário acessa o aplicativo no celular 2. O usuário clica no botão adicionar gasto 3. O sistema exibe a tela de adição de gastos 4. O usuário preenche os dados do gasto e clica no botão salvar. | O sistema registra o gasto do usuário no celular. | Passou |  |
| 10 | RF10-Manter Gastos – Celular | | Manter Gasto no Celular | 1. O usuário acessa o sistema no celular 2. O usuário clica no botão consultar gastos 3. O usuário preenche os filtros e faz a consulta 4. O sistema exibe uma lista com o resultado 5. O usuário clica no item que deseja editar 6. O sistema exibe o formulário com os valores para edição 7. O usuário altera os dados e clica no botão salvar. | O sistema atualiza os dados do gasto no celular. | Passou |  |
| 11 | RF11-Relatorio de Gastos – Celular | | Relatório de Gastos no Celular | 1. O usuário acessa o sistema no celular 2. O usuário clica no botão relatório de gastos 3. O sistema exibe uma tela para que o usuário insira os filtros 4. O usuário informa os valores e pressiona o botão consultar 5. O sistema exibe o relatório no formato de tabela no celular. | O sistema exibe o relatório para o usuário no celular. | Passou |  |
| 12 | RF12-Sincronizar Dados – Celular | | Sincronizar dados | 1. O usuário acessa o sistema no celular 2. O usuário clica no botão sincronizar dados 3. O sistema se comunica com o sistema web sincronizando os dados com o usuário. | O sistema sincroniza os dados da web com o celular. | Passou |  |
| 13 | RF13-Limitar Gastos por Categoria | | Limite de Gastos por categoria | 1. O usuário executa o passo número 3 2. O usuário acessa a tela de manutenção de categorias 3. O usuário informa o limite da categoria escolhida e clica no botão salvar | O sistema registra o limite para aquela categoria. | Passou |  |
| 14 | RF14-Relatório de Média de Gastos por Categoria | | Relatório de Média de Gastos por Categoria | 1. O usuário executa o passo numero 3 2. O usuário clica no menu gastos por categoria 3. O sistema exibe a tela para filtro 4. O usuário preenche o filtro e clica no botão consultar 5. O sistema exibe o relatório de Média de Gastos por Categoria. | O sistema exibe o relatório para o usuário. | Passou |  |

Teste de Configuração e Segurança

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Roteiro de Simulação de Testes** | | | | | | |
| **Descrição do Ambiente** | |  | | | | |
| **Configuração da Máquina** | | Sistema Operacional Windows 7 Core 2 Duo 2.2 GHz 4GB 64 bits | | | | |
| **Período de Execução** | | 28/04/2010 à 28/04/2010 | | | | |
| **Categoria** | | Teste de Integridade das Informações | | | | |
| Ordem | Nome da Simulação | | Simulação | Resultado Esperado | Situação | Observação |
| 1 | Autenticação/Controle de acesso Web | | 1. O usuário acessa o sistema via browser. 2. O usuário é automaticamente redirecionado para a tela de Login. 3. O usuário digita seu login e senha corretos 4. O sistema efetua o login do usuário redirecionando para a página inicial liberando os menus de acordo com seu perfil | O Login é efetuando liberando os acessos de acordo com o perfil do usuário | Passou |  |
| 2 | Autenticação/Controle de acesso Web | | 1. O usuário acessa o sistema via browser. 2. O usuário é automaticamente redirecionado para a tela de Login. 3. O usuário digita seu login e senha incorretos 4. O sistema exibe uma mensagem informando que o usuário e senha estão inválidos | O Login não é permito exibindo uma mensagem de erro para o usuário | Passou |  |

Teste de Instalação

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Roteiro de Simulação de Testes** | | | | | | |
| **Descrição do Ambiente** | |  | | | | |
| **Configuração da Máquina** | | Sistema Operacional Windows 7 Core 2 Duo 2.2 GHz 4GB 64 bits | | | | |
| **Período de Execução** | | 28/04/2010 à 28/04/2010 | | | | |
| **Categoria** | | Teste de Integridade das Informações | | | | |
| Ordem | Nome da Simulação | | Simulação | Resultado Esperado | Situação | Observação |
| 1 | Instalação Celular | | 1. O usuário faz o download do aplicativo Java para celular(arquivo .jar) 2. O usuário copia o arquivo para o celular 3. O usuário acessa pelo celular | O sistema é aberto diretamente no celular | Passou |  |
|  |  | |  |  |  |  |

1. Tipo de codificação do envelope [↑](#footnote-ref-2)
2. Tag que indica um envelope dentro do SOAP [↑](#footnote-ref-3)
3. Tag que indica um erro definida no header do envelope [↑](#footnote-ref-4)
4. Tag que indica um perfil de comunicação para quem está consumindo o webservice [↑](#footnote-ref-5)
5. *Framework* (ou arcabouço) é uma abstração que une códigos comuns entre vários projetos de *software* provendo uma funcionalidade genérica (Wikipedia, 2010). [↑](#footnote-ref-6)
6. É um processo de engenharia de software. Sua meta é garantir a produção de *software* de alta qualidade que atenda às necessidades dos usuários dentro de um cronograma e orçamento previsíveis. [↑](#footnote-ref-7)
7. O termo open source é um tipo de licença cujo sua definição indica que o software possui código aberto, tendo qualquer pessoa acesso a este código. [↑](#footnote-ref-8)