

ANDERSON MOLES DE MATOS  
JUN MIYAZAKI  
RODRIGO ALESSANDRO AKIRA NAKAMA

APLICAÇÃO DA COMPUTAÇÃO UBÍQUA E DE UM MIDDLEWARE  
ORIENTADO A SERVIÇOS PARA O GERENCIAMENTO DA  
ALIMENTAÇÃO DOMÉSTICA

São Paulo  
2009

ANDERSON MOLES DE MATOS  
JUN MIYAZAKI  
RODRIGO ALESSANDRO AKIRA NAKAMA

APLICAÇÃO DA COMPUTAÇÃO UBÍQUA E DE UM MIDDLEWARE  
ORIENTADO A SERVIÇOS PARA O GERENCIAMENTO DA  
ALIMENTAÇÃO DOMÉSTICA

Trabalho de conclusão de curso de  
graduação apresentado à Escola  
Politécnica da Universidade de São Paulo  
para a obtenção do título de Engenheiro

São Paulo  
2009

ANDERSON MOLES DE MATOS  
JUN MIYAZAKI  
RODRIGO ALESSANDRO AKIRA NAKAMA

APLICAÇÃO DA COMPUTAÇÃO UBÍQUA E DE UM MIDDLEWARE  
ORIENTADO A SERVIÇOS PARA O GERENCIAMENTO DA  
ALIMENTAÇÃO DOMÉSTICA

Trabalho de conclusão de curso de  
graduação apresentado à Escola  
Politécnica da Universidade de São Paulo  
para a obtenção do título de Engenheiro

Área de concentração:  
Engenharia de Computação

Orientador: Professor Doutor  
Carlos Eduardo Cugnasca

São Paulo  
2009

## DEDICATÓRIA

Dedicamos este trabalho a nossos familiares

## AGRADECIMENTOS

Gostaríamos de agradecer em primeiro lugar ao Professor Doutor Carlos Eduardo Cugnasca por ter aceitado nos orientar neste projeto de formatura, por estar sempre disponível para resolver problemas conosco e se prontificar a fazer reuniões mesmo por videoconferência.

Gostaríamos de agradecer aos Professores Doutores Paulo Sérgio Cugnasca e João Batista Camargo Júnior por terem confiado em nossas capacidades, permitindo-nos começar o trabalho de formatura na França.

Gostaríamos de agradecer a todos os professores que nos lecionaram durante toda nossa formação acadêmica.

"O futuro se faz agora  
E cada erro é uma vitória  
Pois a derrota não existe  
Não há conquista sem labuta  
A vida é uma infinita luta  
Onde só perde quem desiste"

Douglas Rafael

## RESUMO

O assunto tratado neste relatório é a automação doméstica, mais especificamente a gestão do estoque de alimentos de uma residência por computador. O fruto principal deste trabalho é um software que utiliza tecnologias crescentes no mercado (i.e. identificação por radiofrequência) e se baseia numa área da computação chamada de *computação ubíqua* (ou *pervasiva*). Esse tipo de computação possui a característica de tornar a interação dos sistemas computacionais transparentes para com os usuários, auxiliando assim as tarefas do dia-a-dia do ser humano, e é uma área em franco desenvolvimento. O sistema desenvolvido utiliza uma tecnologia em *código aberto* destinada exclusivamente à integração de sistemas residenciais, o *middleware Amigo*. Na expectativa do crescimento de aplicações destinadas a essa área que utilizem este mesmo *middleware*, o projeto estaria então pronto para ser integrado com outras aplicações residenciais. Os resultados obtidos no planejamento e aqueles do desenvolvimento deste projeto foram bastante satisfatórios, uma vez que a implementação foi realizada de forma quase completa e os desafios técnicos e temporais foram praticamente todos superados pela equipe. Como conclusão, é possível prever um grande desenvolvimento na área de automação residencial nos próximos anos, sendo, portanto, o assunto tratado neste projeto de alto interesse.

Palavras Chave: Engenharia de Computação. Arquitetura Orientada a Serviços. Computação Pervasiva. Computação Ubíqua. Computação Invisível. Middleware Amigo. Automação Residencial. Casa Inteligente.

## ABSTRACT

The main subject of this report is home automation; more specifically computer aided household food stock management. The main deliverable of this project is a stock management software implemented with state of the art technologies (such as RFID) and is linked to a computer domain called *pervasive computing*. This sort of computing is characterized by the enhancement of human-computer interactions, turning computing systems practically "invisible" and giving men support to common daily tasks. The software has been developed using an open source technology which is exclusively designated to residence automation, the "*Amigo*" middleware. As demand for intelligent houses increases, the developed software would be ready to be easily integrated to other home automation systems using the same "*Amigo*" middleware. The project's planning and development was a success; all major timing and technical issues were resolved together by the team and the major project specifications were successfully implemented. As a conclusion, the team is very satisfied with the result obtained and the forecast of a leap on home automation technologies puts this project's subject in a very important position.

Key Words: Computation Engineering. SOA (Service Oriented Architecture). Pervasive Computing. Ubiquitous Computing. Invisible Computing. Amigo Middleware. Home Automation. Intelligent Houses.

## LISTA DE ILUSTRAÇÕES

Figura 1 - Evolução da utilização do computador pelo homem, de 1940 a 2005 .....	20
Figura 2 - Diagrama encontra-liga-executa .....	26
Figura 3 - Arquitetura em camadas do middleware Amigo .....	30
Figura 4 - Exemplo de etiqueta RFID .....	34
Figura 5 - A Smart Kitchen promove uma culinária saudável ao usuário .....	37
Figura 6 - Interface homem-máquina da Smart Kitchen .....	39
Figura 7 - WBS referente ao escopo do projeto .....	41
Figura 8 - Cronograma do projeto .....	43
Figura 9 - Cronograma do projeto por participante .....	44
Figura 10 - Arquitetura de Hardware do sistema .....	45
Figura 11 - Arquitetura de camadas do sistema .....	46
Figura 12 - Work Breakdown Structure dos módulos do sistema .....	48
Figura 13 - Diagrama de casos de uso do sistema .....	49
Figura 14 - Diagrama de base de dados do sistema .....	56
Figura 15 – Telas da interface homem-máquina do sistema: Tela Inicial, Ingrediente Inserido, Lista de compras e Gerenciar tipos de ingredientes .....	58
Figura 16 - Telas da interface homem-máquina do sistema: Gerenciar Receitas e Inserir receita .....	59
Figura 17 - Telas iniciais. Apresentação dos ingredientes atualmente em estoque. ....	68
Figura 18 – Descrição de uma receita. ....	69
Figura 19 - Resultados dos testes funcionais .....	70

## LISTA DE TABELAS

Tabela 1 - Avaliação do Computador Pessoal.....	62
Tabela 2 - Avaliação do sistema operacional .....	63
Tabela 3 - Avaliação da linguagem de programação .....	64
Tabela 4 - Avaliação do banco de dados.....	65
Tabela 5 - Avaliação do ambiente de desenvolvimento.....	66
Tabela 6 – Tabela do orçamento destinado ao projeto.....	86

## LISTA DE ABREVIATURAS

**IEEE** – Instituto de Engenheiros Eletricistas e Eletrônicos

**OSGi** – Open Services Gateway initiative

**PC** - Personal Computer

**PDA** - Personal Digital Assistant

**RFID** – Radio Frequency Identification

**SOA** – Service Oriented Architecture

## SUMÁRIO

<b>1</b>	<b>Introdução .....</b>	<b>16</b>
1.1	Objetivos .....	16
1.2	Motivação .....	17
1.3	Organização .....	19
<b>2</b>	<b>Conceitos e tecnologias .....</b>	<b>20</b>
2.1	Computação Pervasiva .....	20
2.1.1	Paradigma da Computação Pervasiva.....	20
2.1.2	Computação Transparente .....	21
2.2	Arquitetura Orientada a Serviços (SOA).....	23
2.2.1	Processo de utilização de serviços (BOISSIER, 2007).....	26
2.3	O middleware Amigo .....	27
2.3.1	Desenvolvimento voltado para o usuário .....	28
2.3.2	Aplicações Amigo .....	28
2.3.3	Arquitetura Amigo .....	29
2.3.4	O framework OSGi (OSGI, 2009) .....	31
2.4	O sistema RFID .....	32
2.4.1	Conceito.....	32
2.4.2	Aplicação no projeto .....	33
2.5	Java.....	35
2.5.1	Java ME .....	35
2.6	Estado da arte .....	37
<b>3</b>	<b>Planejamento e metodologia.....</b>	<b>40</b>
3.1.1	Metodologia .....	40

3.1.2	Planejamento .....	41
3.1.3	Do escopo do projeto .....	41
3.1.4	Das restrições .....	42
3.1.5	Dos riscos .....	42
3.2	Divisão de papéis .....	42
3.3	Cronograma.....	43
3.4	Cronograma por participante .....	44
<b>4</b>	<b>Especificação .....</b>	<b>45</b>
4.1	Arquitetura de Hardware .....	45
4.2	Arquitetura de Camadas.....	46
4.3	Escopo funcional do sistema .....	48
4.4	Diagrama de casos de uso .....	49
4.5	Casos de Uso .....	49
4.5.1	Módulo Computador Pessoal.....	50
4.5.2	Módulo Assistente Digital Pessoal (PDA) .....	55
4.6	Base de Dados .....	55
4.7	Interfaces homem-máquina .....	58
<b>5</b>	<b>Recursos e Infraestrutura .....</b>	<b>60</b>
5.1	Infraestrutura Hardware.....	61
5.1.1	Middleware de comunicação.....	61
5.1.2	Computador Pessoal .....	62
5.1.3	Leitor RFID.....	62
5.2	Infraestrutura Software .....	63
5.2.1	Sistema operacional .....	63

5.2.2	Linguagem de programação .....	64
5.2.3	Banco de Dados .....	65
5.2.4	Ambiente de desenvolvimento .....	66
5.2.5	Ambiente de testes .....	67
5.2.6	Ferramenta de gestão de projeto .....	67
<b>6</b>	<b>Resultados e Discussões .....</b>	<b>68</b>
6.1	Capturas de tela .....	68
6.2	Resultados.....	69
6.3	Conclusão dos testes .....	70
<b>7</b>	<b>Considerações finais .....</b>	<b>72</b>
7.1	Dificuldades .....	72
7.1.1	Das dificuldades conceituais.....	72
7.1.2	Da tecnologia e ferramentas.....	73
7.2	Gerenciamento de projeto e cronograma .....	74
7.3	Conclusão .....	75
7.4	Extensões do sistema .....	76
<b>8</b>	<b>Referências.....</b>	<b>77</b>
<b>9</b>	<b>Referências complementares .....</b>	<b>80</b>
	<b>GLOSSÁRIO .....</b>	<b>81</b>
	<b>APÊNDICE A – Plano de testes funcionais .....</b>	<b>82</b>
	<b>APÊNDICE B – Resultados dos testes funcionais.....</b>	<b>84</b>
	<b>APÊNDICE C – Detalhes do cálculo do orçamento de projeto .....</b>	<b>86</b>

# 1 Introdução

## 1.1 Objetivos

O objetivo deste projeto de formatura é de aplicar os conhecimentos adquiridos ao longo da formação acadêmica da Escola Politécnica (e.g. Gestão de Projetos, Engenharia de Software, Teoria de Algoritmos, Banco de Dados, Redes de Computadores) na elaboração de um produto que auxilie a gestão da alimentação de uma família.

O projeto propõe uma solução para o gerenciamento de alimentos em uma geladeira. Esse produto está inserido num contexto de uma *smart house*, ou casa inteligente, sistema que se propõe a trazer facilidades aos moradores, sem que para isso os mesmos tenham que ser profundos conhecedores de tecnologia.

Para a realização deste projeto será utilizado um *middleware* orientado a serviços, o *Middleware Amigo*. Esse *Middleware*, dedicado à gestão de uma casa inteligente, permite um desenvolvimento mais rápido e direto, já que é possível utilizar as interfaces de comunicação já criadas.

Ao final do projeto será obtido um produto de software a ser embutido em equipamentos domésticos. É importante assegurar que o produto final seja útil e de fácil uso para o consumidor final. Além disso, ele deve ser escalável, permitindo uma expansão para outros equipamentos domésticos e para novas tecnologias.

## 1.2 Motivação

Ao se estudar a evolução do paradigma da computação, é possível classificá-la em três fases principais. Na primeira, o processamento da informação era realizado de forma centralizada por *mainframes*. Uma só máquina deveria atender às necessidades de um grande número de pessoas. Em seguida, com o advento do computador pessoal (PC), a computação democratizou-se e uma máquina era destinada a uma pessoa. Na terceira fase, com avanços tecnológicos em *hardware*, *software* e telemática, a contribuição dos dispositivos computacionais expandiu-se, desde aparelhos celulares até máquinas de lavar. Portanto, hoje há inúmeros computadores a serviço de cada pessoa (WEISER, 1996).

Assim, a tendência que se percebe é o aumento do número de computadores por usuário e a crescente complexidade em se gerenciar muitas máquinas que funcionam simultaneamente. Com isso, as pesquisas na área de computação pervasiva têm os objetivos de fazer com que todos esses computadores comuniquem-se entre si produzindo serviços mais inteligentes

Além disso, a tecnologia pervasiva faz parte de uma filosofia na qual o uso do computador, e mais especificamente da tecnologia doméstica, seja altamente transparente ao utilizador. Realizar as tarefas em um contexto de computação pervasiva deve ser algo altamente natural e, no melhor dos casos, o utilizador nem mesmo percebe que está interagindo com um computador ou com um sistema inteligente.

O presente projeto de formatura inspira-se nessa tendência e nas últimas pesquisas da área de computação pervasiva, propondo um sistema que interliga um assistente digital pessoal (PDA), um PC e um leitor RFID acoplado a estoques de alimentos (geladeira, despensa etc.), para gerir e melhorar a alimentação de uma família. No mais, este projeto de formatura complementa o trabalho apresentado no capítulo “Conceitos e tecnologias”, no subcapítulo “Estado da arte” (CHI, 2007), no qual é proposto um sistema de cozinha inteligente que promove uma alimentação saudável por meio de sensores que identificam as características nutricionais de cada ingrediente de uma receita. Esse complemento se deve ao fato de que no

trabalho proposto por Chi (2007) não há um sistema de gestão de estoque de alimentos, de gestão de receitas ou de gestão de datas de validade, funcionalidades essas que são propostas pelo presente projeto de formatura.

O problema da alimentação de uma família foi escolhido pois é um problema cotidiano, devido à falta de tempo para o bom planejamento do que se quer comer, o que se deve comprar, ou o descuido em se verificar quais produtos estão prestes a vencer. Assim, busca-se não somente melhorar a qualidade dos pratos que são ingeridos, melhorando a saúde, mas também não se preocupar com produtos vencendo ou listas de compras a redigir, diminuindo o estresse mental.

Num segundo momento, pode-se vislumbrar a integração ao sistema de outros eletrodomésticos da cozinha (como forno elétrico ou forno de micro-ondas), e, num momento posterior, a integração com outros dispositivos por toda a casa (como televisão, aparelho de som ou sistema de iluminação), criando uma rede doméstica de inteligência ambiente (Ambient Intelligence).

### 1.3 Organização

No capítulo 1 (Introdução), apresentou-se os objetivos do presente projeto, a motivação para realizá-lo e a descrição da estrutura deste documento.

No capítulo 2 (Conceitos e tecnologias), faz-se uma revisão geral da literatura relacionada à computação pervasiva e ao conceito de Arquitetura Orientada a Serviços (SOA). Além disso, apresenta-se o middleware Amigo, a tecnologia RFID, o framework OSGi (e sua implementação de código aberto chamada Oscar) e a linguagem de programação Java™ (e sua vertente para dispositivos móveis chamada Java ME™).

O capítulo 3 apresenta a especificação funcional do projeto, descrevendo-se os casos de uso e as interfaces homem-máquina.

O capítulo 4 apresenta a especificação técnica do projeto, com a arquitetura de hardware, a arquitetura em camadas do sistema e a definição da base de dados.

No capítulo 5 são descritas as infraestruturas de hardware e software que viabilizam este trabalho.

No capítulo 6 é apresentada a metodologia adotada para a condução deste projeto.

No capítulo 7 apresenta-se os testes funcionais realizados, com a metodologia de testes adotada e os resultados dos testes.

No capítulo 8 são feitas as considerações finais.

## 2 Conceitos e tecnologias

### 2.1 Computação Pervasiva

#### 2.1.1 Paradigma da Computação Pervasiva

A origem da computação pervasiva (ou ubíqua) é intimamente ligada à evolução do paradigma da computação. Durante a história da computação, percebe-se a mudança no modo de utilização do computador pelo homem, devido em partes pela evolução da tecnologia e por uma demanda de sistemas cada vez mais específicos.

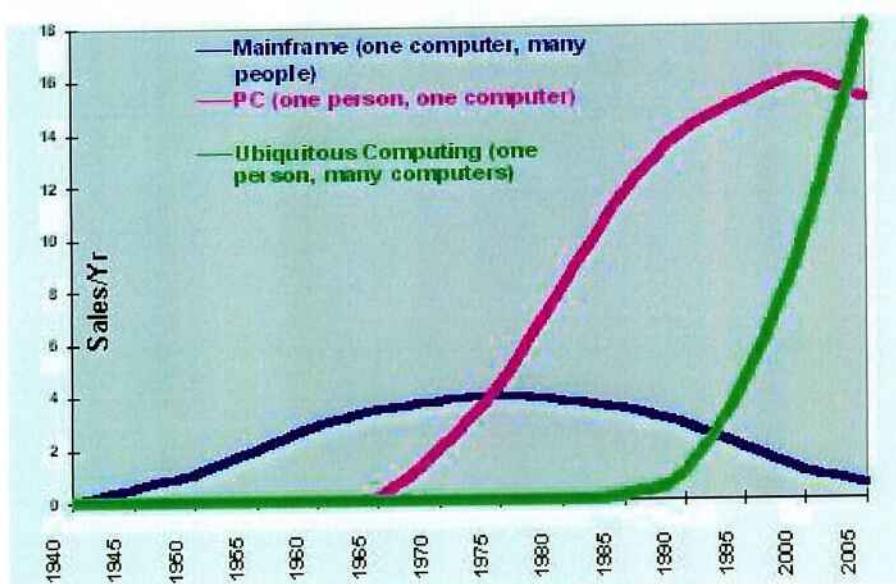


Figura 1 - Evolução da utilização do computador pelo homem, de 1940 a 2005

A figura acima<sup>1</sup> mostra a evolução da utilização do computador pelo homem, de 1940 a 2005, tendo nas ordenadas o volume de vendas de computadores. As três curvas são interpretadas da seguinte maneira:

- Azul: Mainframes. Um computador para várias pessoas. Era como a computação existia no início.
- Rosa: PC pessoais. Um computador para uma pessoa.

<sup>1</sup> Figura extraída de <http://sandbox.xerox.com/ubicomp/> em 20/11/09

- Verde: Computação pervasiva. Vários computadores para uma mesma pessoa.

Em um seminário em 1991, Mark Weiser<sup>2</sup> descrevia a computação pervasiva como a “criação de ambientes saturados de capacidade de computação e comunicação integrados com usuários humanos”. Esta visão era bem a frente de seu tempo, um tempo no qual a Internet era praticamente inexistente.

Desde 1991 a tecnologia deu um salto que tornou possível a criação do ambiente descrito por Weiser. Sistemas de computação integrados a outros sistemas por meio da rede mundial de computadores são amplamente difundidos e estão presentes em quase todos os lugares.

O tema discutido neste projeto de graduação engloba a computação pervasiva na medida em que explora o ambiente doméstico inteligente. Até o ano de 2009, os ambientes inteligentes domésticos não estão completamente desenvolvidos, dando margem a novos sistemas que ofereçam uma maior automação residencial.

### **2.1.2 Computação Transparente**

Outro fato importante é a maneira como a computação é explorada pelo ser humano. Na história da computação verifica-se que cada vez mais a interface de comunicação entre o homem e a máquina vem se desenvolvendo. O limite desta evolução seria um ambiente no qual seria impossível perceber a existência da máquina, ou seja, a utilização da computação se tornaria natural com o ser humano (SOYLU; DE CAUSMAECKER; DESMET, 2009).

Todo este paradigma está intimamente ligado com o conceito de computação pervasiva. À medida que os sistemas de computação são difundidos, e que o homem é forçado a interagir cada vez mais com esses sistemas, é necessário o desenvolvimento de interfaces de comunicação mais claras e intuitivas.

No contexto da automação doméstica, por exemplo, a naturalidade de comunicação estaria no fato de controlar a temperatura interna com um simples

---

<sup>2</sup> WEISER, MARK D. (23/07/1952 – 27/04/1999) – Cientista chefe da Xerox, considerado o pai da computação pervasiva.

comando de voz, ou poder controlar o estoque de alimentos na residência por meio de simples toques na tela de sua geladeira.

Quatro princípios devem ser respeitados quando se pensa em um projeto de computação pervasiva. Estes princípios são baseados nos estudos de Weiser e dizem que:

- i. **A função do computador é ajudar o homem a fazer algo:** o computador não pode ser a origem de problemas ou complicar a execução de uma tarefa pelo homem;
- ii. **O melhor computador é um computador quieto, um serviço invisível:** seguindo a teoria da computação transparente, o homem não deve (ou praticamente não deve) notar a presença de um sistema de computação. Ele deve ser o mais transparente possível;
- iii. **Quanto mais intuitivo, melhor; o computador deve estender o inconsciente do homem:** o computador deve ajudar na tomada de decisões e informar ao usuário apenas aquilo que lhe possa ser útil.
- iv. **Tecnologia deve criar calma:** um dos atributos da computação pervasiva é oferecer tranquilidade ao homem.

## 2.2 Arquitetura Orientada a Serviços (SOA)

Segundo Lublinsky (2007), não há um consenso sobre a melhor definição do conceito de arquitetura orientada a serviços (SOA). Com tantas definições existentes, é difícil capturar a verdadeira essência do acrônimo SOA. Assim, SearchWebServices.com promoveu um concurso para se escolher a melhor definição de SOA. Entretanto, mesmo assim não foi possível eleger a melhor definição de SOA, uma vez que seu significado é diferente para pessoas diferentes, como se pode ver a seguir.

Para um engenheiro de desenvolvimento, SOA é um modelo de programação completo, com padrões, ferramentas e tecnologias.

Para um engenheiro de testes, SOA é uma forma de modularizar e, conseqüentemente, simplificar o processo de teste do sistema.

Para um chefe de projeto, SOA é uma forma de se promover um massivo desenvolvimento paralelo.

Para um arquiteto de aplicações, SOA é um conjunto de princípios e padrões arquiteturais que viabilizam certas características desejáveis em softwares, como modularidade, encapsulamento, baixo acoplamento (*loose coupling*), reuso etc.

Finalmente, para um analista de negócios, SOA é um conjunto de serviços que podem ser usados para construir aplicações e expô-las aos clientes e empresas parceiras.

Para complementar essas definições um tanto subjetivas e poder entender de uma forma mais precisa o significado de SOA, deve-se atentar primeiramente para a letra A desse acrônimo, ou seja, deve-se entender o que significa arquitetura de software. Uma definição bem aceita de arquitetura de software é a fornecida pelo IEEE, no padrão 1471-2000 (IEEE Recommended Practice for Architectural Description of Software-Intensive Systems):

"Arquitetura (de software) é a organização fundamental de um sistema refletida em seus componentes, nas relações entre os componentes e

também com o ambiente externo, e nos princípios que guiam o projeto e a evolução do sistema."

Tomando-se por base essa definição, e considerando-se também as duas outras letras do acrônimo (Orientação a Serviços), chega-se à conclusão de que uma arquitetura orientada a serviços é uma arquitetura na qual o conceito de "serviços" define a forma como são elaborados os componentes do sistema, como eles se comunicam entre si e com o ambiente externo, e como o projeto e a evolução do sistema são organizados.

Segundo OPEN GROUP, 2006, serviço é:

- Uma representação lógica de uma atividade de negócio repetitiva que possui uma saída (um resultado, um produto) especificada.
- Pode ser composto de outros serviços.
- É uma caixa preta do ponto de vista do consumidor do serviço.

Entretanto, para que um serviço atenda às necessidades de uma organização, ele deve obedecer a certos critérios, os quais são especificados a seguir (BALZER, 2004):

- Reuso: o serviço deve ser reutilizável por diferentes pessoas, em diferentes situações ou para diferentes aplicações.
- Granularidade: para que seja possível o reuso do serviço, sua granularidade não deve ser grande demais, o que o tornaria aplicável a apenas alguns problemas específicos. Já se sua granularidade for muito fina, problemas de desempenho do sistema surgiriam, uma vez que muitos serviços teriam que ser invocadas para atender a uma necessidade do consumidor, acarretando um excesso de tráfego de mensagens na rede por meio da qual os serviços se comunicam. Portanto, deve-se buscar um meio termo de granularidade.
- Modularidade: os serviços devem ser modulares, cada um desenvolvendo uma atividade específica, com baixo acoplamento entre os serviços, isto é, um serviço deve depender pouco de outros serviços para atingir seus objetivos.

- *Composability*: os serviços devem poder ser facilmente compostos a partir de outros serviços, e orquestrados de acordo com a necessidade do consumidor do serviço.
- *Componentization*: os componentes que compõe o serviço devem ser encapsulados dentro de uma interface padrão de serviços, para que seja possível seu reuso fora de seu ambiente nativo de execução.
- Conformidade com padrões: a conformidade com padrões de interface e de comunicação, possibilitam ao serviço ser reutilizado independentemente da tecnologia que foi usada na sua construção, ou seja, se um serviço foi feito em Java, ele pode ser utilizado por um consumidor de serviço escrito em .Net.
- Identificação e categorização, provisão e entrega de serviços: os serviços devem estar claramente identificados e especificados num anuário de serviços, acessível aos consumidores de serviços, possibilitando um processo adequado de provisão e entrega de serviços. Esse processo está esquematizado a seguir, e denomina-se encontra-liga-executa (*find-bind-execute*).

### 2.2.1 Processo de utilização de serviços (BOISSIER, 2007)

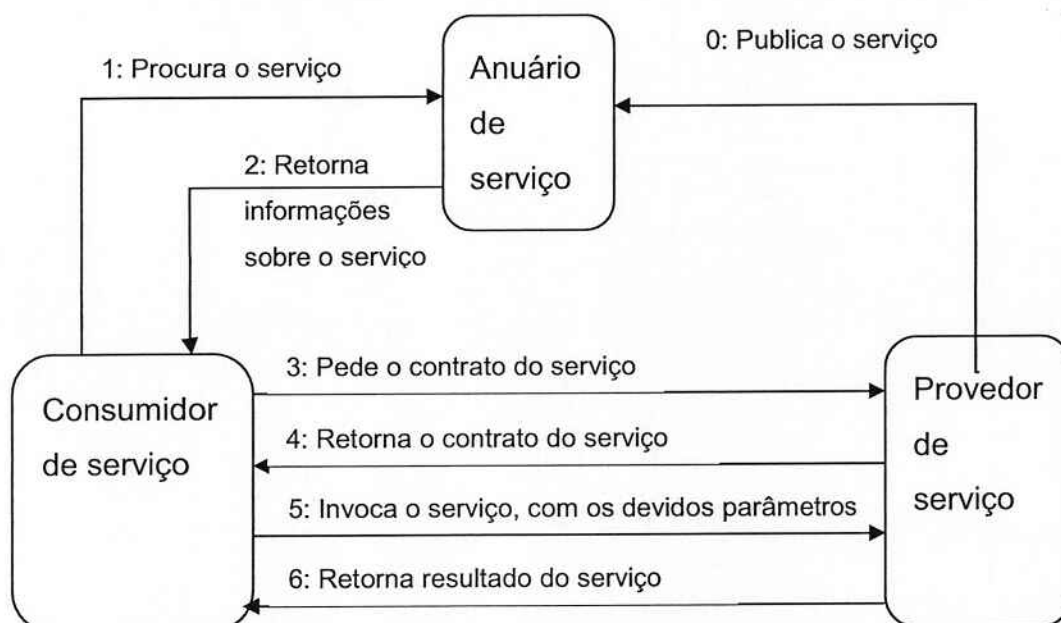


Figura 2 - Diagrama encontra-liga-executa

A seguir, define-se os conceitos chaves relacionados ao diagrama acima:

- Consumidor de serviço: aplicação cliente que se liga a um provedor de serviços e invoca suas funções.
- Provedor de serviço: aplicação que provê e executa serviços.
- Anuário de serviços: anuário que regrupa os serviços disponíveis no sistema, tratando as questões de publicação e descoberta de serviços.
- Contrato de serviço: definição da maneira em que o serviço deve ser chamado, especificando o nome do serviço e os parâmetros necessários.

No item seguinte, apresenta-se o middleware Amigo, cuja arquitetura é orientada a serviços.

## 2.3 O middleware Amigo

O Amigo é um middleware orientado a serviços em código aberto destinado exclusivamente à automação residencial. Por ser de código aberto, ele não proporciona custos com licenças e detém por trás uma comunidade de desenvolvimento ativa, o que garante sua constante evolução. Além disso, ele facilita a missão de integração dos componentes da arquitetura, auxiliando também possíveis extensões do projeto (para fornos inteligentes, sistemas de iluminação, entre outros).

O projeto Amigo reúne quinze das principais empresas e institutos de pesquisa europeus em desenvolvimento de software móvel e de redes domésticas, eletroeletrônicos e eletrodomésticos (como France Telecom, Philips e Telefônica). Eles se uniram neste projeto integrado a fim de se explorar o potencial do mercado de residências inteligentes, assim como para se criar uma infraestrutura tecnológica padrão que viabilize uma casa inteligente de forma flexível e simples.

A rede doméstica já existe, obviamente, em aplicações específicas no PC para comunicação com outros PCs e sistemas de entretenimento doméstico, mas a sua capacidade de realmente mudar a vida das pessoas é ainda marcada por procedimentos de instalação complexos, falta de interoperabilidade entre equipamentos de diferentes fabricantes e ausência de serviços ao usuário.

Ao se concentrar na resolução destas questões essenciais, o projeto Amigo visa superar os obstáculos à aceitação generalizada desta nova tecnologia, facilitando cada vez mais o desenvolvimento de ambientes domésticos inteligentes.

O projeto Amigo desenvolve um middleware que integra dinamicamente sistemas heterogêneos para assegurar a interoperabilidade entre serviços e dispositivos. Por exemplo, eletrodomésticos (sistemas de aquecimento, sistemas de iluminação, máquinas de lavar, frigoríficos), equipamentos multimídia e representantes de jogadores (que se comunicam por meio de UPnP), assim como dispositivos digitais pessoais (laptops, PDAs) são conectados em rede doméstica para trabalhar de maneira interoperável. Esta interoperabilidade entre diferentes domínios de aplicação também pode ser estendida entre casas de locais diferentes.

### **2.3.1 Desenvolvimento voltado para o usuário**

Os usuários podem ter dificuldade em se imaginar vivendo num ambiente tão futurístico como o proposto pela computação pervasiva. Como as exigências dos usuários poderão mudar e evoluir ao longo do tempo, à medida que as pessoas ficarem cada vez mais familiarizadas com ambientes conectados, o projeto Amigo adotou um processo de design centrado no usuário para trabalhar com essa dinâmica da evolução das necessidades.

### **2.3.2 Aplicações Amigo**

O projeto desenvolve aplicações em diferentes domínios, para mostrar o potencial e os benefícios da arquitetura orientada a serviços do middleware para desenvolvedores de aplicativos. Algumas destas aplicações são a segurança, o gerenciamento da informação, o entretenimento no lar e um ambiente doméstico alongado, no qual várias casas estão conectadas. Um dos principais objetivos é fornecer aos usuários finais serviços que lhes permitam compartilhar atividades e experiências de uma forma simples e personalizada.

Eles podem compartilhar a visita de sua residência, por exemplo, com amigos e parentes, ou outras comunidades sociais que estão em outros locais. Assim, por exemplo, é possível para os pais que estão em viagem de negócios compartilharem atividades diárias com os filhos em casa, contando suas histórias para dormir, assistindo televisão juntos ou jogando jogos eletrônicos com eles. Isto é, eles podem compartilhar a sua presença independentemente de sua localização.

### 2.3.3 Arquitetura Amigo

A arquitetura que o “Amigo” segue é orientada a serviços, permitindo o desenvolvimento de softwares como serviços que são prestados e consumidos conforme a demanda de clientes.

A vantagem desta abordagem reside no fácil acoplamento dos componentes de software que compõem uma aplicação. Mecanismos de descoberta de serviços podem ser usados para localizar e selecionar a funcionalidade que um cliente está procurando.

O projeto “Amigo” aceita um grande número de protocolos de comunicação, de forma interoperável, tornando possível para os programadores selecionar um protocolo que seja mais apropriado aos seus objetivos.

O middleware “Amigo” pode ser dividido em três partes principais:

- **Base Middleware (Middleware de Base):** contém as funcionalidades necessárias para facilitar um ambiente de rede. Ele fornece a semântica para se comunicar e encontrar os serviços disponíveis dos dispositivos de rede, incluindo os que são baseados na comunicação existente.
- **Intelligent User Services (Serviços ao Usuário Inteligentes):** contém as funcionalidades necessárias para facilitar um ambiente em rede doméstica. A informação obtida nesta camada é adaptada à situação atual do usuário e às mudanças no contexto.
- **Programming and Deployment Framework (Frameworks para programação e implantação):** contém módulos que facilitam o desenvolvimento de serviços “Amigo” em .NET ou Java, fornecendo suporte para a interoperabilidade, segurança e a descrição dos serviços disponíveis para desenvolvedores.

Essas três partes se inserem no diagrama de arquitetura em camadas do middleware Amigo apresentado abaixo.

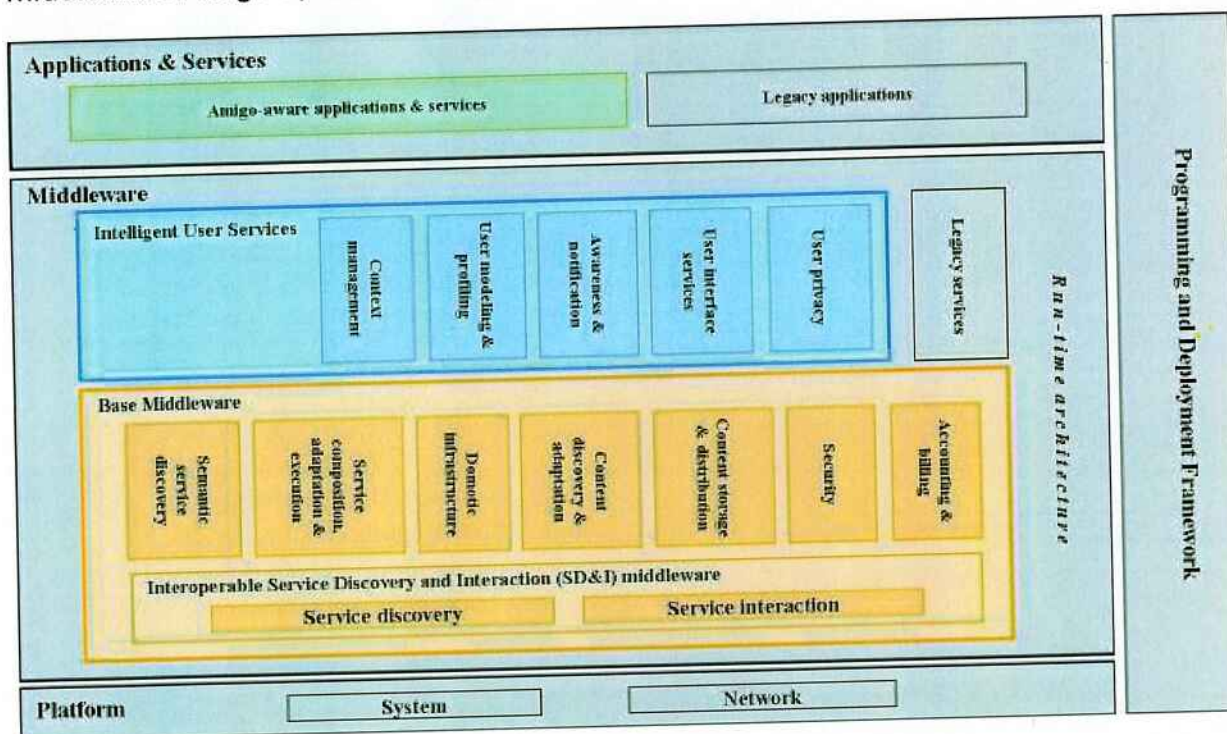


Figura 3 - Arquitetura em camadas do middleware Amigo

#### 2.3.4 O framework OSGi (OSGI, 2009)

No caso de o middleware Amigo ser utilizado por aplicações na linguagem Java™, o que é o caso do presente trabalho, faz-se necessário a utilização do framework OSGi (acrônimo de Open Services Gateway initiative, a qual é a organização que criou esse framework).

Esse framework possibilita a construção ágil de aplicações Java™ complexas a partir de componentes pequenos, reusáveis e colaborativos. Além disso, com esse framework é possível mudar os componentes que compõe uma aplicação dinamicamente, sem a necessidade de reinicialização do sistema. No mais, um componente pode descobrir outro componente dinamicamente para iniciar um processo de colaboração.

Em suma, o framework OSGi possibilita a aplicação da arquitetura orientada a serviços a ambientes Java, unindo os benefícios dessa arquitetura (como a agilidade e flexibilidade de desenvolvimento) aos atrativos da linguagem Java (como a portabilidade e a orientação a objetos que ela oferece).

O framework OSGi desenvolveu uma série de padrões de interface de componentes para funções comuns, como comunicação com servidores HTTP, jornal de log, segurança, administração de usuário, gestão de arquivos XML etc. Existem no mercado várias aplicações que implementam o framework OSGi. No presente trabalho, adotou-se a implementação chamada Oscar, por ser de código aberto e com uma comunidade de desenvolvedores bastante ativa.

## 2.4 O sistema RFID

### 2.4.1 Conceito

RFID é um sistema de rádio-identificação que utiliza rádio etiquetas capazes de armazenar uma considerável quantidade de dados. Estas etiquetas são de tamanho reduzido (podem ser etiquetas autoadesivas) podendo ser incorporadas em objetos ou produtos ou mesmo implantadas em organismos vivos. Estas “rádio-etiquetas” são compostas de uma antena associada a um “chip” eletrônico que lhe permite receber e enviar dados a um emissor/receptor.

As etiquetas são, na maioria dos casos, dispositivos passivos, ou seja, não precisam de nenhum tipo de bateria para funcionar. A energia necessária a seu funcionamento é fornecida pelo leitor de dados no momento da leitura. Em seu formato mais simples, a etiqueta envia ao leitor apenas uma identificação composta de 96 bits que representa seu número de identificação único. Uma etiqueta comum é composta dos seguintes elementos:

- Uma antena;
- Um microchip;
- Um substrato ou uma encapsulação;

O sistema de identificação RFID já está amplamente difundido, sendo encontrado em diversas aplicações dentre elas:

- Acesso aos transportes públicos (BOM da prefeitura de São Paulo);
- Sistema de cobrança de pedágio (Sem-Parar);
- Acesso a prédios comerciais;
- Maratonas e competições esportivas em geral;

Existem diversos setores com alto potencial para a aplicação desta tecnologia entre eles:

- Identificação de cartas nos correios;

- Controle do estado de fabricação de um produto (por exemplo, na cadeia de produção de automóveis)
- Identificação de produtos no caixa de um supermercado;

Muitas destas aplicações já poderiam ser desenvolvidas hoje em dia, porém o preço deste tipo de solução não é viável economicamente, ficando por enquanto apenas no campo teórico.

#### **2.4.2 Aplicação no projeto**

O elemento de base para o funcionamento da arquitetura apresentada neste relatório se baseia na tecnologia RFID. Uma etiqueta RFID estará presente em cada alimento, possibilitando o armazenamento de informações relevantes sobre eles tais como:

- Nome do produto;
- Data de validade;
- Descrição;
- Composição;
- Quantidade/Unidade de medida;

Cada alimento receberá uma etiqueta RFID própria, sendo assim possível um controle total sobre as informações do produto. Diferentemente da tecnologia atual em Código de Barra, as etiquetas serão únicas para cada elemento (por exemplo, dois pacotes de bolacha da mesma marca e do mesmo tipo terão etiquetas distintas se tiverem sido fabricadas em lotes distintos).

Os números de endereços existentes de etiquetas RFID são de  $16^{10}$ , ou seja, pouco mais de 1 trilhão de endereços. Isto porque seu endereçamento é feito por um código composto por 10 dígitos hexadecimais. Além disso, essas etiquetas possuem memórias internas que hoje em dia podem chegar a 1kbyte, ou seja oito mil bits, que podem servir para identificação completa do produto. Com este número de endereços estas etiquetas podem, com folga, endereçar todos os produtos existentes no mundo por muito tempo.

A etiqueta RFID é uma ótima solução para esta arquitetura devido a diversos fatores:

**Fácil uso:** uma etiqueta RFID pode ser uma etiqueta de espessura mínima. Não há impactos no design dos produtos.

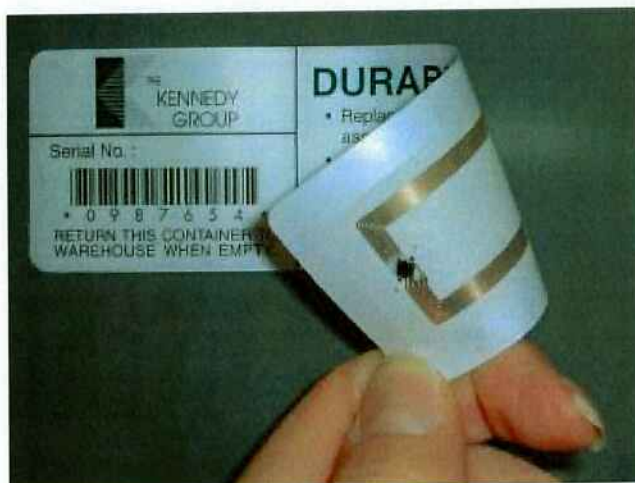


Figura 4<sup>3</sup> - Exemplo de etiqueta RFID

**Unicidade:** não existem duas etiquetas RFID com o mesmo número de série

**Quantidade de informações:** é possível armazenar alguns Kb de informação

**Redução do custo:** as etiquetas estão custando cada vez mais barato. Sua viabilidade neste tipo de aplicação se torna, portanto, cada vez maior.

Para completar a arquitetura RFID é necessário integrar um leitor RFID. Este leitor é o responsável por coletar as informações das etiquetas RFID e transmiti-la a outro componente que será responsável por tratar essas informações.

Existem no mercado diversos tipos de leitores RFID. Devido a seu pequeno tamanho eles podem ser facilmente acoplados em eletrodomésticos de uma residência. Como o protocolo RFID é único para todos os fabricantes, não existe limitação quanto à escolha de marca. A avaliação a ser feita neste caso é o preço, design e o modo de operação do leitor.

Existem no mercado diversos tipos de leitores RFID. A diferença entre eles está em sua potência, isto é, quão próximo uma etiqueta RFID deve estar para que ela

<sup>3</sup> Fonte: <http://www.themajorlearn.info/pic/RFID2.jpg>

possa ser lida. O preço tende a aumentar drasticamente à medida que a potência de leitura aumenta.

Para a arquitetura proposta, o leitor RFID deve ter potência suficiente para ler etiquetas aproximadas a uma distância de alguns centímetros. Existem diversos tipos de fabricantes no mercado que atendem esta exigência. Não realizamos uma pesquisa mais aprofundada sobre este ponto, pois utilizamos o leitor disponibilizado pelo departamento PCS para realizar nosso protótipo.

## **2.5 Java**

Java é uma linguagem de programação de alto nível, imperativa e orientada a objetos. Ela foi criada na década de 90, na empresa Sun Microsystems, por um grupo liderado por James Gosling, o qual é considerado o pai da linguagem Java.

Um dos diferenciais da linguagem Java se dá pelo fato de ela não ser compilada para código nativo da máquina onde o programa é executado, mas para um código intermediário, os bytecodes, que são então interpretados pela Máquina Virtual Java (JVM) instalada na máquina do usuário. Assim, é possível criar programas em Java que podem ser executados em virtualmente qualquer máquina, desde que exista nela uma JVM instalada. Disso decorre sua alta portabilidade.

Outro diferencial da linguagem Java é sua orientação a objetos. Um programa em Java pode ser considerado como uma coleção de objetos se comunicando entre si através de chamadas mútuas de métodos. Cada objeto tem certo tipo, estado e comportamento, os quais são definidos por uma classe. O tipo é o próprio nome da classe. O estado é determinado pelos valores dos atributos de um objeto em dado momento. E comportamento de um objeto é definido por seus métodos (SIERRA; BATES, 2006).

### **2.5.1 Java ME**

Teoricamente, um programa em Java pode ser executado em qualquer plataforma, desde que haja uma JVM nessa plataforma, como explicado anteriormente. No entanto, ao se considerar as limitações em memória e poder de processamento de dispositivos móveis, tais como aparelhos celulares, percebe-se que isso não é totalmente verdade.

Portanto, para atender ao nicho de dispositivos móveis, a Sun Microsystems criou em 1999 a plataforma J2ME, com o objetivo de permitir o desenvolvimento de aplicativos para essa gama de dispositivos.

Nessa plataforma, a JVM é implementada numa versão bastante reduzida. No mais, foi preciso limitar os recursos da linguagem Java. Essas medidas foram tomadas para que a linguagem Java se torne coerente com o hardware de dispositivos como aparelhos celulares.

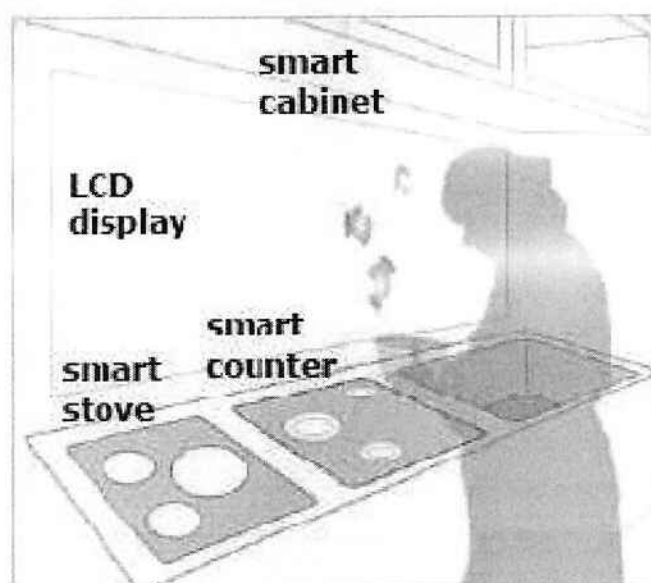
## 2.6 Estado da arte

Diversos projetos de pesquisa foram realizados com o intuito de se desenvolver cozinhas inteligentes.

Fez-se uma revisão de alguns desses projetos e selecionou-se um projeto nessa linha de pesquisa como demonstração do atual estado da arte. Este projeto foi realizado pela Universidade Nacional de Taiwan e apresentado na Conferência sobre Fatores Humanos em Sistemas Computacionais (San Jose, USA) e chama-se Enabling Nutrition-Aware Cooking in a Smart Kitchen (CHI, 2007).

A ideia principal nesse projeto é de elaborar uma cozinha inteligente que melhora o processo de se preparar um prato através do fornecimento de características nutricionais de cada ingrediente que compõe o prato, promovendo uma alimentação mais saudável.

Para tanto, esse sistema é equipado com sensores que detectam as atividades de preparo de prato.

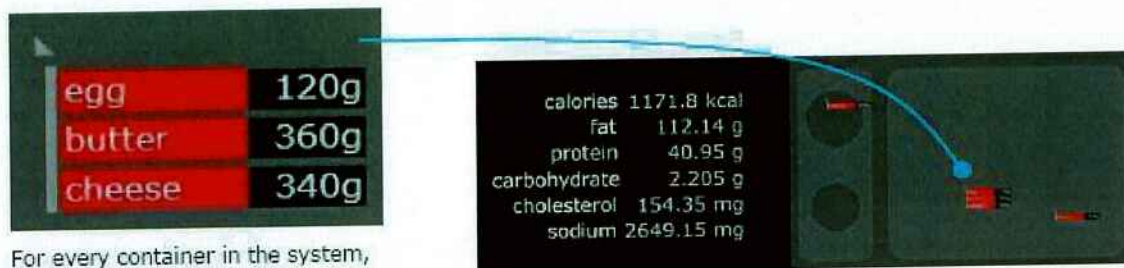


**Figura 5 - A Smart Kitchen promove uma culinária saudável ao usuário**

Nesse projeto, foi realizado um protótipo composto dos seguintes elementos:

- Plano de trabalho inteligente: Com a ajuda de 4 sensores localizados nos 4 cantos do plano de trabalho, é possível detectar a posição dos recipientes colocados sobre o plano de trabalho e calcular o peso dos ingredientes contidos em cada recipiente.
- Forno inteligente: O forno possui duas bocas, uma na parte anterior, outra, na posterior. Em baixo de cada boca se encontra um sensor que detecta por diferença de peso se um novo ingrediente foi adicionado à panela sobre a boca.
- O armário inteligente: Os recipientes (pratos, tigelas, panelas) que são colocados ou retirados do armário são detectados por meio de sensores localizados no armário que calculam a variação de peso a cada inserção ou remoção de recipientes no armário.
- Reconhecimento de ingredientes: No presente estado do protótipo, o reconhecimento de ingredientes se dá de forma manual pelo usuário. Mas estuda-se formas alternativas, como a utilização de etiquetas RFID. O objetivo desse sistema de reconhecimento de ingredientes é de rastrear a movimentação dos diferentes ingredientes, e assim se ter um maior controle por parte do sistema do processo de preparo do prato.

- Interface homem-máquina:



For every container in the system, the interface shows weight information about food ingredients the container has.

**figure 5.** User interface of our smart kitchen system. We display nutrition facts about latest ingredients on the left hand side, and overall ingredient information on the right hand side.

**Figura 6 - Interface homem-máquina da Smart Kitchen**

A interface ao usuário é composta de uma tela LCD que mostra os detalhes nutricionais de um determinado ingrediente, assim como informações gerais de cada recipiente.

### 3 Planejamento e metodologia

#### 3.1.1 Metodologia

A presente metodologia é baseada no texto “Técnicas de Desenvolvimento de Projetos Baseados em Microprocessadores” (CUGNASCA, 2006). Ele faz parte da bibliografia recomendada da disciplina Laboratório de Processadores I, ministrada no currículo de Engenharia de Computação da Escola Politécnica da USP.

Em se tratando de um projeto genérico de desenvolvimento de sistemas, é possível dividi-lo nas partes de projeto de *software* e projeto de *hardware*, partes que serão integradas no planejamento do trabalho. O presente projeto, enquanto protótipo, tem um caráter de *software* mais acentuado que o de *hardware*. O projeto do segundo restringe-se à implementação do leitor RFID, limitando-se fora isso à instalação dos aplicativos nos diversos dispositivos. Tal parte do sistema é feita de maneira avulsa ao projeto de *software*.

Esse, por sua vez, é desenvolvido em uma abordagem *top-down*, na medida em que se busca implementar primeiro as funções mais gerais (ou transcendentais) do sistema, para depois decompô-lo em módulos mais específicos e detalhados, mas com grau de complexidade menor. Dessa forma, a parte de comunicação feita através do *middleware* Amigo é testada e implementada *a priori*, seguida da implementação paralela dos módulos cliente e servidor, de acordo com o nível de prioridade de cada função. Segundo esse nível de prioridade, por exemplo, o desenvolvimento para PDA é feito *a posteriori*, de acordo com o avançamento do projeto.

### 3.1.2 Planejamento

O planejamento deste projeto foi baseado no modelo de plano de projeto preconizado na disciplina de Gerência, Qualidade e Engenharia de Software, ministrada pelo Prof. Dr. Kechi Hirama, no curso de Engenharia da Computação da Escola Politécnica da USP.

### 3.1.3 Do escopo do projeto

Para se definir o escopo do projeto, foram definidos os seguintes artefatos, como ilustrado na figura abaixo:

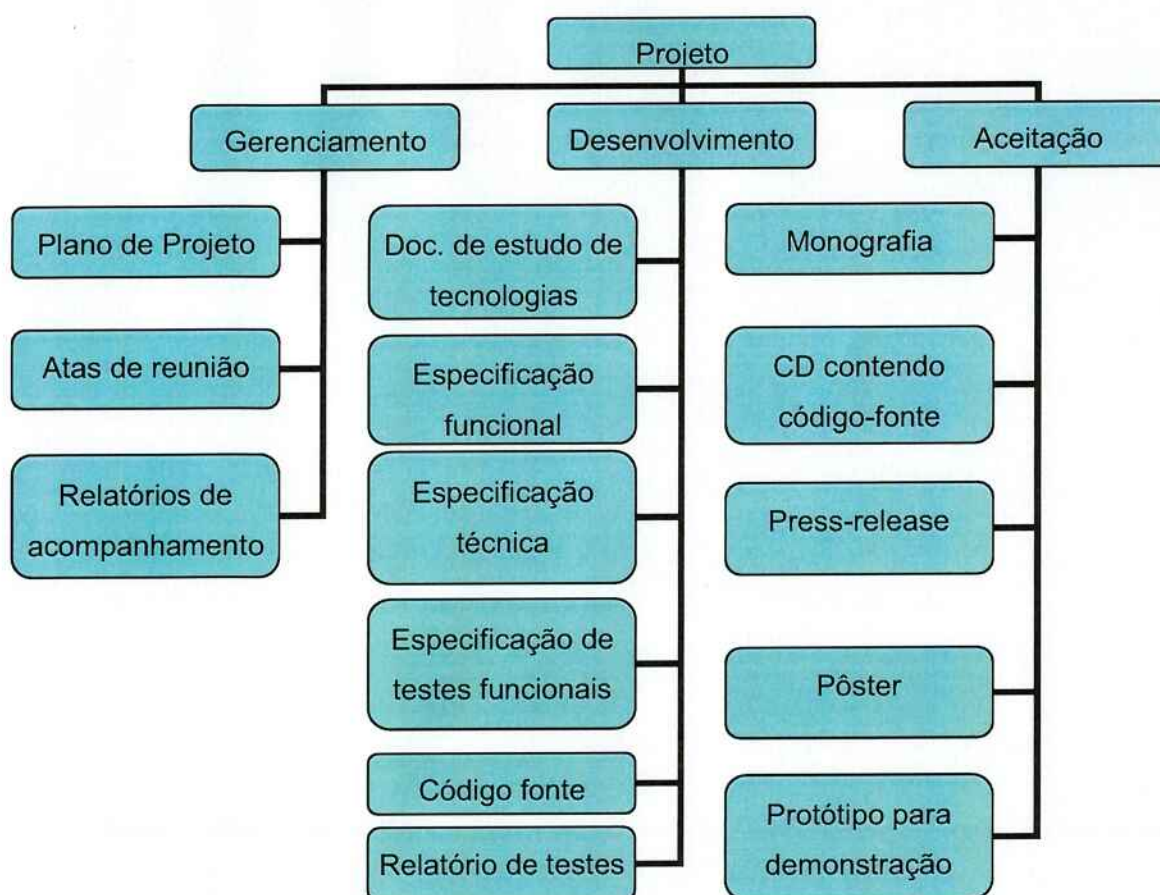


Figura 7 - WBS referente ao escopo do projeto

### 3.1.4 Das restrições

- Uso de soluções gratuitas ou fornecidas pela universidade.
- Aplicação de conceitos aprendidos ao longo do curso de engenharia da computação.
- Prazo de entrega imutável (não pode haver atraso)

### 3.1.5 Dos riscos

- Alteração de requisitos
- Inadequação às especificações
- Complexidade técnica da solução visada
- Indisponibilidade da infraestrutura de testes de integração e de demonstração

## 3.2 Divisão de papéis

A divisão dos papéis dos participantes do projeto foi feita em acordo comum entre todos. Tentamos dividir as tarefas de acordo com as habilidades pessoais e para que o projeto tomasse uma forma mais profissional.

A divisão feita foi a seguinte:

- 1) Anderson Moles de Matos: Chefe de projeto. Responsável por coordenar as atividades, definir um plano de desenvolvimento do projeto, planificar reuniões e assegurar o bom andamento do projeto.
- 2) Jun Miyazaki: Responsável Qualidade. Responsável pela qualidade dos relatórios entregues e pela qualidade final do produto entregue.
- 3) Rodrigo Nakama: Responsável Técnico. Responsável pela parte de programação e configuração de hardware necessária ao projeto. Deve coordenar as atividades necessárias a esse fim.

É importante deixar claro que, mesmo com a divisão de papéis, todos os integrantes do projeto participam de todas as atividades de projeto.

### 3.3 Cronograma

O cronograma abaixo mostra as atividades realizadas a partir do início do projeto. Identificam-se no diagrama duas grandes fases do projeto:

- **Especificação:** Fase inicial de planejamento e determinação do escopo do projeto. Esta fase teve duração de um mês.
- **Desenvolvimento:** Fase de implementação do projeto, dividida em 6 partes.
- **Monografia:** É realizada ao mesmo tempo que o desenvolvimento do projeto. Isso permite a realização de pequenas mudanças de escopo e de acompanhar a evolução do projeto.

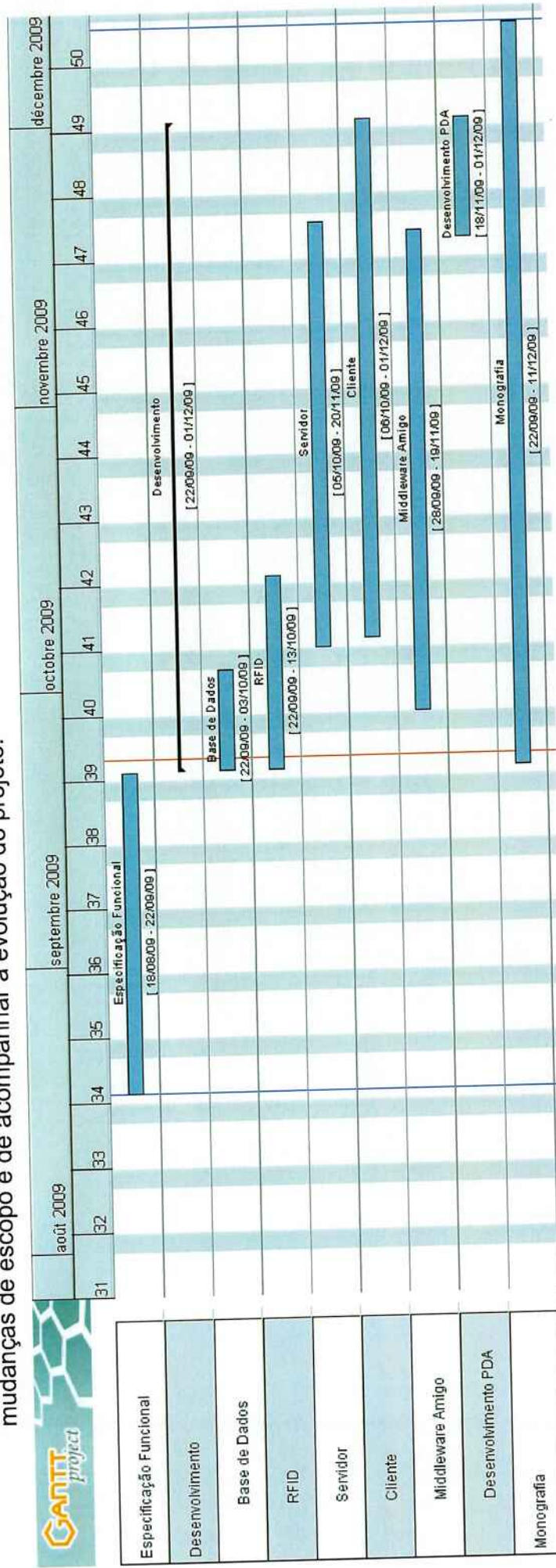


Figura 8 - Cronograma do projeto

3.4 Cronograma por participante

As responsabilidades e tarefas de cada um dos integrantes foram divididas após a definição do cronograma de desenvolvimento a fim de desenvolver o projeto de uma maneira mais dinâmica. Uma pessoa é responsável por uma parte da implementação, mas também fornece suporte a outras tarefas. Isso permite uma integração maior dos módulos e a redução de incoerências.

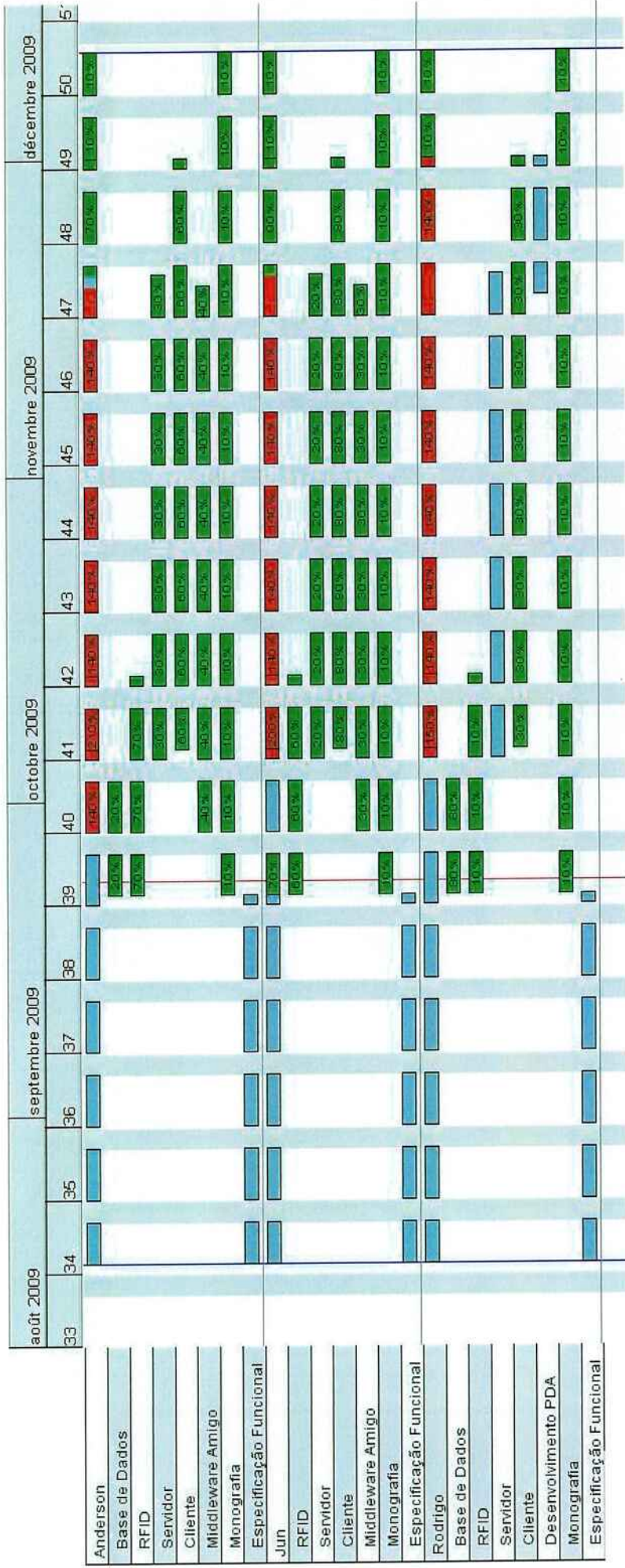
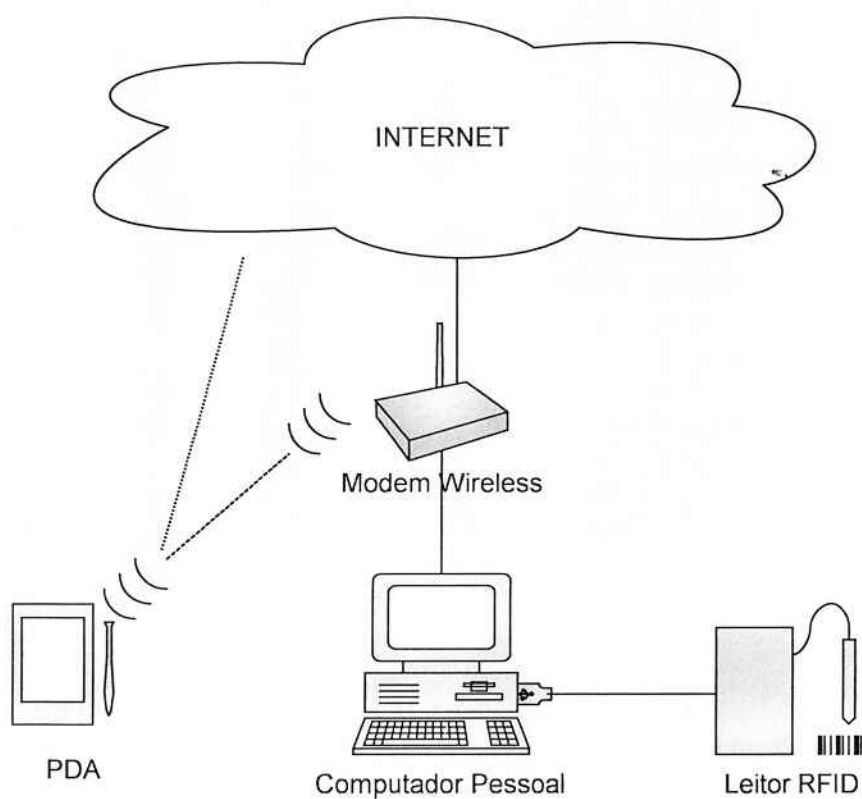


Figura 9 - Cronograma do projeto por participante

## 4 Especificação

### 4.1 Arquitetura de Hardware

A figura abaixo apresenta a arquitetura de Hardware proposta para o projeto. Seus componentes e utilidades são explicadas a seguir.



**Figura 10 - Arquitetura de Hardware do sistema**

- **Computador pessoal:** É o dispositivo utilizado para armazenamento de dados vindos do leitor RFID e por hospedar um servidor e um cliente Amigo para acesso de dados.
- **Leitor RFID:** Equipamento utilizado para realizar as leituras das etiquetas RFID. Sua conexão com o computador central é feita através de um cabo USB.

- **Modem Wireless:** Dispositivo utilizado para criar a rede sem fio doméstica.
- **PDA:** Dispositivo que contém um cliente Amigo, utilizado para se obter a lista de compras atualizada.
- **Internet:** O sistema será conectado à rede mundial de computadores para que o PDA possa ter acesso à funcionalidade de obtenção da lista de compras atualizada, a partir de um lugar remoto.

## 4.2 Arquitetura de Camadas

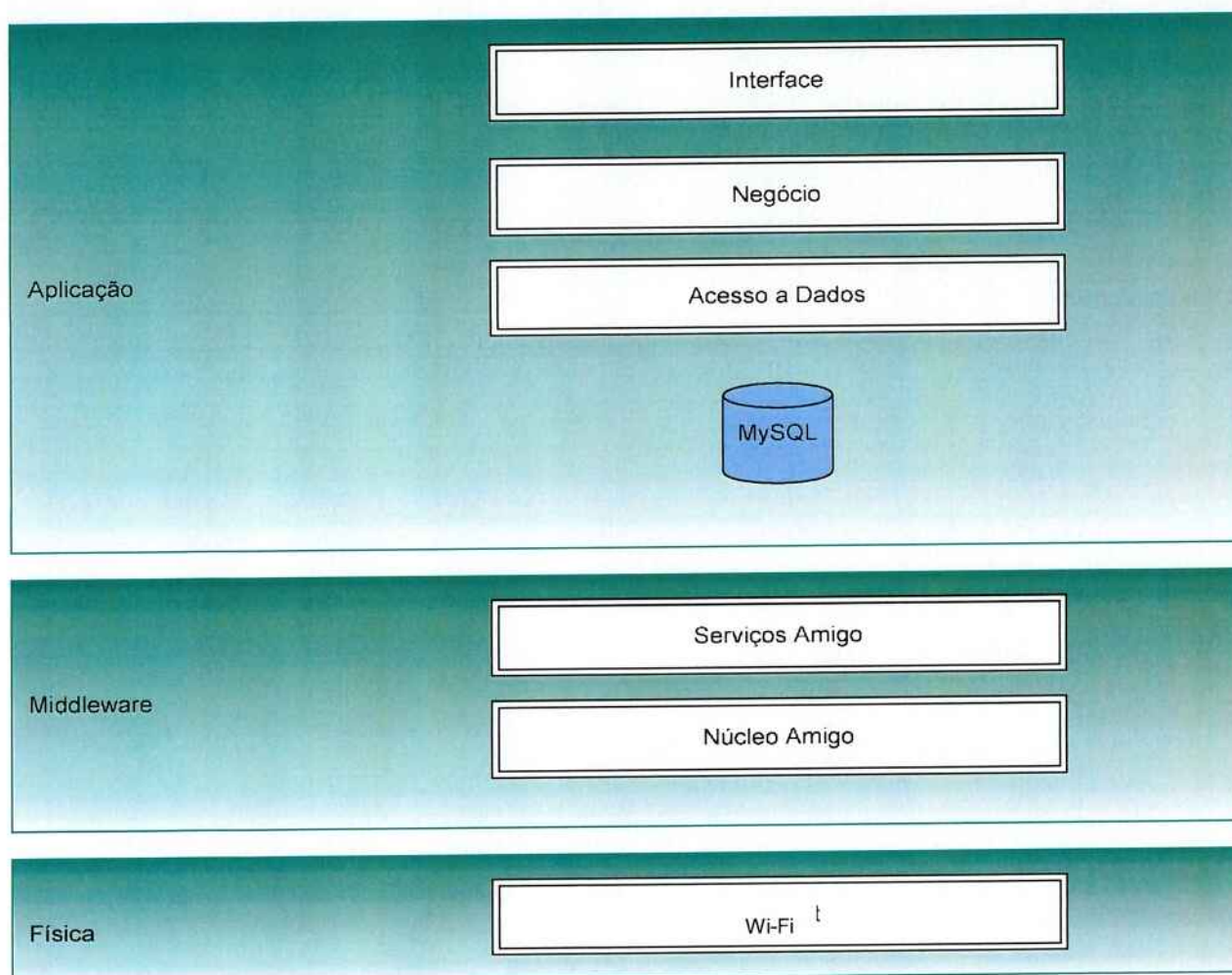


Figura 11 - Arquitetura de camadas do sistema

- Dividimos a arquitetura do software em 3 camadas (física, middleware e aplicação) de acordo com as necessidades técnicas de nossa aplicação. Cada uma dessas camadas são módulos individuais e tem objetivos complementares.
  - **Camada de aplicação:** É composta de componentes (bundles) que podem ou ser fornecedores de serviço (formando a camada de negócio), ou consumidores de serviço (formando a camada de interface). Existe ainda um componente do tipo fornecedor de serviço que provê o acesso à base de dados MySQL (formando a camada de acesso a dados).
  - **Camada Middleware:** É a camada responsável pela comunicação entre os diversos dispositivos ligados à rede. O middleware utilizado é o “Amigo”, específico para automação residencial. Dentro dessa camada existe o núcleo do middleware e os serviços que são disponibilizados para a comunicação entre componentes.
  - **Camada física:** Trata-se da rede sem fio local proporcionada pelo roteador wireless, através da qual os diferentes equipamentos da residência se comunicam.

### 4.3 Escopo funcional do sistema

O escopo funcional do sistema é apresentado na figura abaixo. Este escopo será detalhado na sequência.

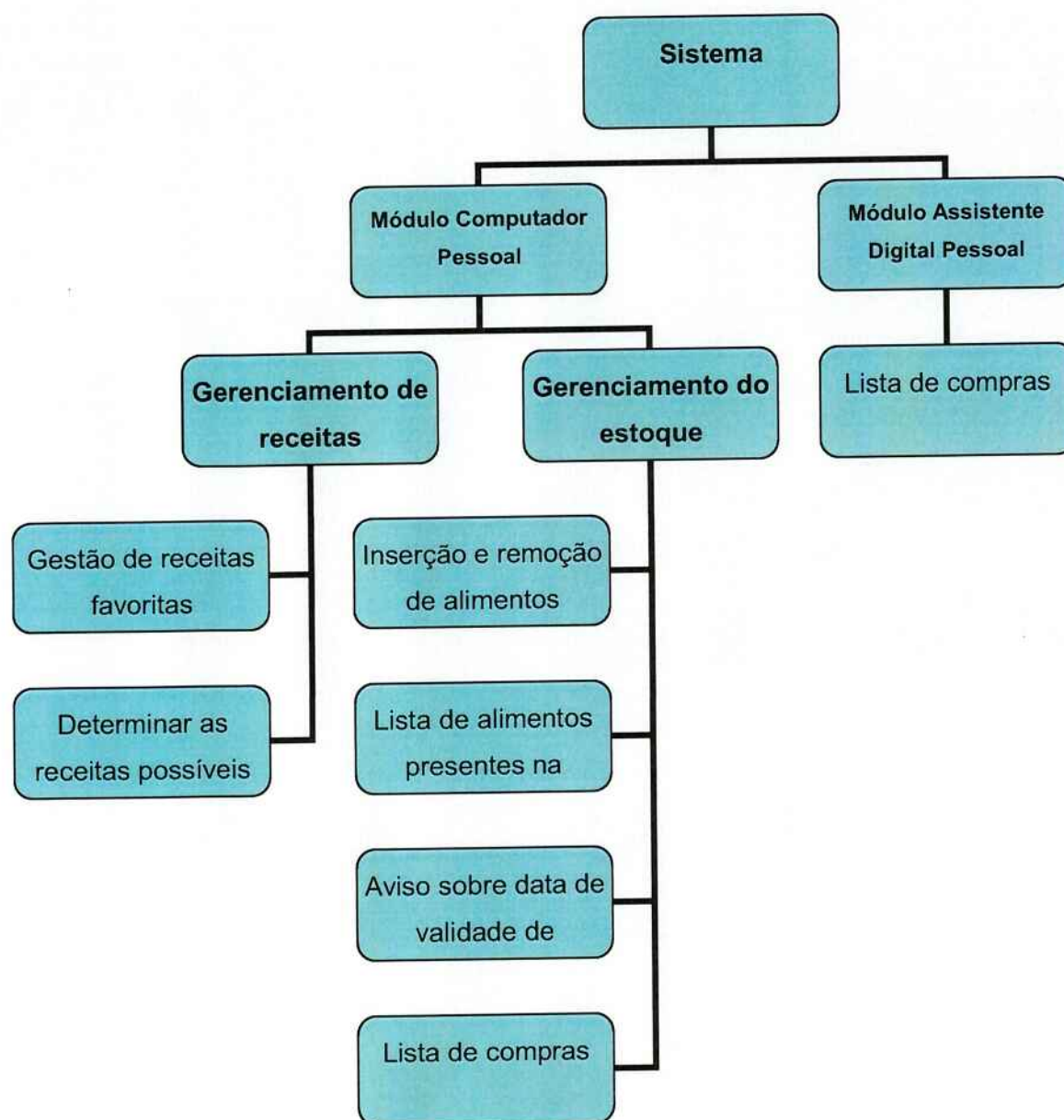


Figura 12 - Work Breakdown Structure dos módulos do sistema

#### 4.4 Diagrama de casos de uso

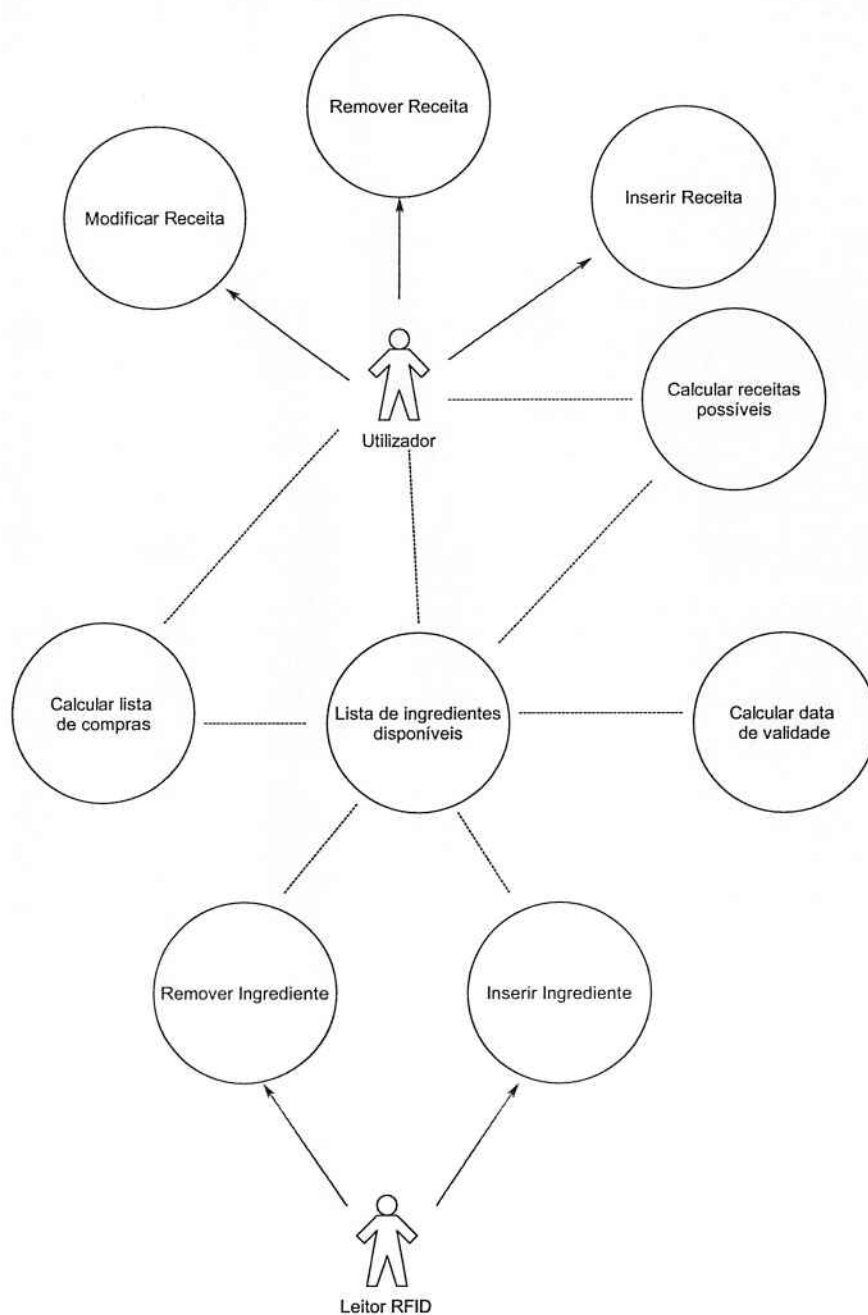


Figura 13 - Diagrama de casos de uso do sistema

#### 4.5 Casos de Uso

Cada módulo possui suas próprias especificações, divididas em pequenos *Casos de Uso*. Para apresentar de modo mais claro estas especificações, utilizamos metodologia semelhante à especificação de casos de uso:

- 1) Resumo da especificação
- 2) Fluxo de eventos
- 3) Limites

#### **4.5.1 Módulo Computador Pessoal**

Este módulo é dividido em duas grandes partes, que são a *o gerenciamento de receitas* e o *gerenciamento do estoque*.

##### **4.5.1.1 Gerenciamento de receitas**

O gerenciamento de receitas é um módulo que possibilita ao usuário de salvar receitas em um banco de dados interno. Com este banco de dados de receitas a geladeira poderá oferecer ao usuário diversos tipos de facilidade como as que serão descritas a seguir.

###### **4.5.1.1.1 Gestão de receitas favoritas**

Permite ao usuário inserir, remover e modificar receitas que ficam salvas em um banco de dados.

###### **a) Inserção de receita**

1) **Resumo:** Insere uma nova receita no banco de dados de receitas

###### **2) Fluxo de Eventos:**

- a. O utilizador escolhe a opção inserir receita
- b. O sistema responde mostrando uma tela para inserção de dados sobre a receita (Nome da receita, ingredientes utilizados, quantidade necessária de cada ingrediente, modo de preparação)
- c. O utilizador preenche os dados e valida a inserção
- d. O sistema mostra a validação da receita inserida

###### **3) Limites**

- a. Nos ingredientes utilizados para a receita, o sistema mostra uma lista com os ingredientes cadastrados no sistema (e que são reconhecidos pelo sistema)

###### **b) Remoção de receita**

1) **Resumo:** Permite a remoção de receitas do banco de dados

###### **2) Fluxo de Eventos:**

- a. O utilizador escolhe a opção remoção de receita
- b. O sistema responde mostrando uma lista com as receitas disponíveis
- c. O Utilizador escolhe uma das receitas mostradas na lista ou escolhe a opção busca de receitas
  - i. Na opção busca de receitas, o utilizador digita o nome de uma receita, clica no botão busca e a lista de receitas é atualizada com as receitas contendo o nome colocado na busca.
- d. Após a escolha da receita, o utilizador clica no botão remover receita
- e. O sistema pede validação do usuário
- f. O usuário valida a receita e ela é removida

### **3) Limites:**

Nenhum

### **c) *Modificação de receita***

**1) Resumo:** Permite a visualização da receita por um utilizador

#### **2) Fluxo de Eventos:**

- a. O utilizador escolhe a opção modificar receita
- b. O sistema responde mostrando uma lista com as receitas disponíveis
- c. O Utilizador escolhe uma das receitas mostradas na lista ou escolhe a opção busca de receitas
  - i. Na opção busca de receitas, o utilizador digita o nome de uma receita, clica no botão busca e a lista de receitas é atualizada com as receitas contendo o nome colocado na busca
- d. Após a escolha da receita, o utilizador clica no botão modificar receita
- e. O sistema abre uma página mostrando toda a descrição da receita. Além disso, o sistema mostra quais ingredientes estão disponíveis na geladeira para realizar a receita

- f. O usuário pode modificar a receita. Para isso ele faz a modificação e clica no botão salvar
- g. O sistema pede validação do usuário para modificação da receita
- h. O usuário valida e a receita é modificada

**3) Limites:**

- a. Para escolher os ingredientes utilizados para a receita, o sistema mostra uma lista com os ingredientes cadastrados no sistema (e que são reconhecidos pelo sistema)

**4.5.1.1.2 Determinar as receitas possíveis com os alimentos disponíveis**

- 1) Resumo:** O sistema retorna ao utilizador uma lista das receitas possíveis com os alimentos que estão atualmente na geladeira

**2) Fluxo de Eventos:**

- a. O utilizador clica no botão "Dica de receitas"
- b. O sistema faz uma busca na base de dados de receitas e nos ingredientes atualmente na geladeira.
- c. Após a busca, o sistema mostra uma lista das receitas disponíveis com os alimentos presentes na geladeira. O sistema propõe a receitas favoritas de acordo com o perfil do usuário
- d. O utilizador escolhe uma receita.
- e. O sistema mostra os detalhes da receita, como para o caso de modificação de receita.

**3) Limites:**

- a. O sistema não fará uma verificação para saber se as quantidades dos ingredientes disponíveis são suficientes para realizar a receita

**4.5.1.2 Gerenciamento de estoque de ingredientes**

**4.5.1.2.1 Inserção e remoção de alimentos**

**a) Inserção de ingrediente**

- 1) Resumo:** Um novo alimento é inserido na estoque do usuário

**2) Fluxo de Eventos:**

- a. O utilizador passa o produto na frente do leitor RFID

- b. O sistema responde e uma tela para inserção de informações sobre o produto é apresentada ao usuário
- c. O utilizador valida e o produto é inserido

**3) Limites:**

Prevemos que as informações sobre o produto, que no momento estão previamente configuradas na base de dados, serão inseridas automaticamente quando etiquetas RFID destinadas a cada produto forem disponibilizadas no mercado.

**b) Remoção de ingrediente**

- 1) Resumo:** Um ingrediente que estava na geladeira é removido do sistema

**2) Fluxo de Eventos:**

- a. O utilizador passa o produto na frente do leitor RFID
- b. Se o produto já tiver sido passado uma vez no leitor RFID, esta segunda passagem fará o ingrediente ser removido do sistema

**3) Limites:**

**4.5.1.2.2 Lista de alimentos presentes na geladeira**

- 1) Resumo:** Esta opção permite ao usuário visualizar uma lista com os alimentos que estão dentro da geladeira

**2) Fluxo de Eventos:**

- a. O utilizador escolhe a opção visualizar lista
- b. O sistema responde mostrando uma lista de alimentos com informações como prazo de validade e informações nutricionais

**3) Limites:**

Nenhum

**4.5.1.2.3 Aviso sobre data de validade de alimentos**

- 1) Resumo:** Um aviso é enviado automaticamente ao utilizador (por email ou por um aviso no desktop do computador pessoal) para avisar que alguns alimentos estão fora da data de validade. O utilizador pode

configurar um intervalo de tempo antes da data de validade do produto para que a notificação de validade lhe seja enviada.

**2) Fluxo de Eventos:**

- a. O utilizador configura de antemão um intervalo de tempo para que o lembrete de validade seja enviado ao utilizador
- b. O sistema verifica uma vez por dia a data de validade de todos os produtos presentes na geladeira
- c. Para produtos com datas de validade fora do intervalo de tempo do lembrete, o sistema identifica estes produtos e notifica o utilizador por dois meios:
  - i. Notificação via email
  - ii. Notificação no desktop do computador pessoal

**3) Limites:**

É possível também que o utilizador configure o modo de notificação a ser utilizado.

O utilizador pode também configurar o intervalo de tempo para que a verificação das datas de validade seja feita pelo sistema.

**4.5.1.2.4 Lista de compras**

- 1) Resumo:** De acordo com os ingredientes favoritos do utilizador, o sistema atualiza, diariamente, uma lista com os ingredientes que estão em falta na geladeira

**2) Fluxo de Eventos:**

- a. O utilizador configura parâmetros para cada um dos ingredientes, informando as quantidades mínimas que devem existir de cada um deles na geladeira
- b. O sistema compara a lista de produtos atualmente na geladeira com as quantidades definidas pelo usuário.
- c. O usuário recebe uma notificação para informar o usuário sobre os ingredientes em falta.

**3) Limites:**

## **4.5.2 Módulo Assistente Digital Pessoal (PDA)**

### **4.5.2.1 Lista de compras**

**1) Resumo:** Carrega no PDA do utilizador uma lista de compras atualizada. Para cada item da lista, exibe uma checkbox para o utilizador indicar se o produto foi tratado (p. ex., colocado no carrinho de compras) à medida que ele realiza suas compras.

**2) Fluxo de Eventos:**

- a. O utilizador pressiona o botão “Obter lista de compras” em seu PDA.
- b. O sistema envia ao PDA a lista de compras.
- c. O PDA apresenta ao utilizador a lista dos itens a serem comprados, com suas respectivas checkboxes.
- d. O utilizador marca as mercadorias que comprou.

**3) Limites:**

Nenhum

## **4.6 Base de Dados**

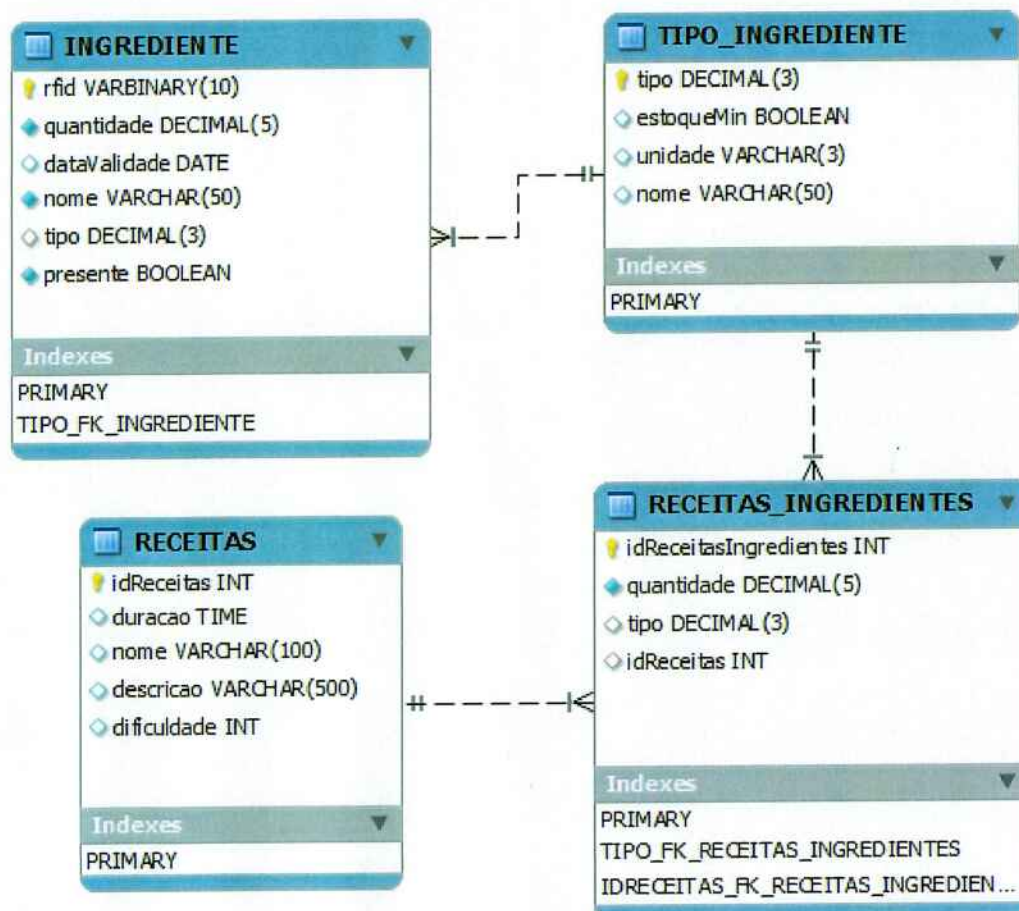


Figura 14 - Diagrama de base de dados do sistema

Existem 4 tabelas na base de dados, a saber:

- **INGREDIENTE**

Nesta tabela serão armazenados os dados referentes a cada item do estoque de alimentos, com os seguintes atributos:

- 1) rfid: chave primária - trata-se de uma sequência de 10 Bytes que estará gravada em cada etiqueta RFID, e que identificará o produto de forma única.
- 2) quantidade: a quantidade do produto (volume ou peso), cuja unidade de medida estará registrada na tabela TIPO\_INGREDIENTE.
- 3) dataValidade: a data de validade do produto.
- 4) nome: o nome do produto.

- 5) tipo: chave estrangeira proveniente da tabela TIPO\_INGREDIENTE.
- 6) presente: variável booleana que indica se o produto está ou não presente no estoque.

- TIPO\_INGREDIENTE

Nesta tabela serão armazenados os dados referentes ao tipo do produto, que consistirão em:

- 1) tipo: chave primaria - código numérico que identifica o tipo do produto (p. ex. 1 para leite, 2 para carne etc.).
- 2) estoqueMin: variável booleana que indica se para o tipo em questão precisa-se haver uma quantidade mínima do produto em estoque permanentemente (destinado a alimentos essenciais como leite ou arroz).
- 3) unidade: a unidade padrão de medida da quantidade para cada tipo de produto (p. ex. kg, ml etc.).
- 4) nome: o nome do tipo de ingrediente.

- RECEITAS

Nesta tabela estarão registrados os dados referentes ao banco de receitas, que terá os seguintes atributos:

- 1) idReceitas: chave primária.
- 2) duração: duração de preparo da receita
- 3) nome: nome da receita.
- 4) descrição: descrição da receita.
- 5) dificuldade: nível de dificuldade da receita.

- RECEITAS\_INGREDIENTES

Criou-se esta tabela pois a relação da tabela RECEITAS com a tabela

TIPO\_INGREDIENTE é n X n. Nesta tabela serão armazenados os seguintes dados:

- 1) idReceitasIngredientes: chave primaria

- 2) quantidade: quantidade de cada ingrediente da receita
- 3) tipo: chave estrangeira proveniente da tabela TIPO\_INGREDIENTE
- 4) idReceitas: chave estrangeira proveniente da tabela RECEITAS

#### 4.7 Interfaces homem-máquina

### Tela Inicial

Lista de Compras

Gerenciar Receitas

Gerenciar Tipos de Ingredientes

Lista de ingredientes com datas de validade. Dar destaque aos alimentos que estão próximos da sua data de validade.

Atualizar

### Ingrediente Inserido

Nome do ingrediente

Tipo

Data de validade

Quantidade

Estoque

Ok

### Lista de compras

Nome do tipo de ingrediente

### Gerenciar tipos de ingredientes

Tipo	Quantidade em estoque	Unidade	Estoque Mínimo?
------	-----------------------	---------	-----------------

Validar

Cancelar

**Figura 15 – Telas da interface homem-máquina do sistema: Tela Inicial, Ingrediente Inserido, Lista de compras e Gerenciar tipos de ingredientes**

Gerenciar Receitas	Inserir receita
<div><div>Inserir</div><div>Remover</div></div>	<div><div>Voltar</div><div>Validar</div></div>
<div><div>Lista de receitas com caixa de seleção à esquerda para selecionar e remover. Destaque dado às receitas que não podem ser realizadas por falta de ingrediente</div><div><div>Ver</div><div>Ver</div><div>Ver</div><div>Ver</div><div>Ver</div><div>Ver</div></div></div>	<div><div>Nome da receita</div><div>Lista de ingredientes à escolher à través de uma caixa de seleção</div><div>Modo de preparação</div></div>

Figura 16 - Telas da interface homem-máquina do sistema: Gerenciar Receitas e Inserir receita

## 5 Recursos e Infraestrutura

Os recursos e infraestrutura utilizados no projeto seguiram uma metodologia de avaliação técnica. Cada um dos itens foi avaliado segundo alguns pontos:

1. **Conhecimento técnico pessoal:** Este ponto é considerado de vital importância para o desenvolvimento do projeto, já que o tempo que se tem para o desenvolvimento do projeto é extremamente curto.

**Ponderação: 10**

2. **Facilidade de utilização:** Trata-se do quão fácil é aplicar a tecnologia.

**Ponderação: 8**

3. **Facilidade de integração com os sistemas domésticos:** Avalia se a solução é facilmente empregável em um sistema embutido de uma geladeira, por exemplo.

**Ponderação: 6**

4. **Grau de reconhecimento no mercado:** Avalia se a tecnologia é reconhecida e bastante utilizada no mercado.

**Ponderação: 7**

5. **Custo de desenvolvimento e de obtenção da tecnologia:** Avalia se a tecnologia está disponível para ser aplicada a um baixo custo (ao menos para a fase de testes).

**Ponderação: 10**

6. **Grau de sofisticação da solução:** Avalia se a solução é sofisticada em termos de inovação.

**Ponderação: 3**

Estes seis pontos de avaliação foram julgados os mais coerentes e condizentes com os objetivos do projeto, tendo em vista as restrições em termos de tempo, recursos e conhecimentos.

O procedimento adotado foi o seguinte:

1. Determinar tipos de infraestrutura e recursos a serem empregados;
2. Para cada tipo de infraestrutura e recursos, determinar quais são as tecnologias disponíveis no mercado;
3. Realizar uma avaliação individual e em seguida comparativa entre as tecnologias com relação aos pontos definidos acima. A avaliação é feita pela atribuição de notas com a ponderação que julgamos conveniente para cada um dos pontos de análise.
4. Escolher uma tecnologia por infraestrutura e recurso, segundo as notas recebidas e a uma avaliação global da tecnologia.

Julga-se que este tipo de estudo é o mais adequado para o projeto, já que ele permite justificar mais claramente as escolhas feitas. O levantamento das tecnologias utilizáveis permite se obter um leque maior de opções e se saber qual opção é a mais adequada.

Resumo dos recursos e infraestrutura necessários ao projeto:

- 1) Ambiente de desenvolvimento
- 2) Linguagem de programação
- 3) Middleware de comunicação
- 4) Simulação da geladeira
- 5) Computador pessoal
- 6) Leitor RFID

## **5.1 Infraestrutura Hardware**

### **5.1.1 Middleware de comunicação**

Para o middleware de comunicação escolhemos desde o princípio o Middleware Amigo, próprio para aplicações de computação pervasiva, logo excelente para o projeto.

### 5.1.2 Computador Pessoal

	Notebook	PC de mesa	PDA
Conhecimento técnico pessoal	9	9	5
Facilidade de utilização	9	9	6
Facilidade de integração com os sistemas domésticos	8	5	10
Grau de reconhecimento no mercado	9	9	9
Custo de desenvolvimento e de obtenção da tecnologia	10	10	5
Grau de sofisticação da solução	6	4	10
<b>NOTA FINAL</b>	<b>8,9</b>	<b>8,3</b>	<b>6,8</b>

**Tabela 1 - Avaliação do Computador Pessoal**

Para simular o computador pessoal de uma residência decidiu-se então pelo notebook, já que é mais fácil em termos de instalação e transporte.

### 5.1.3 Leitor RFID

Utilizar-se-á um leitor de cartões RFID, disponibilizado pela Escola Politécnica, para simular o que seria um identificador RFID de uma geladeira

inteligente. Para realizar a apresentação final do projeto, espera-se utilizar este mesmo leitor RFID dentro de uma geladeira comum.

## 5.2 Infraestrutura Software

### 5.2.1 Sistema operacional

	Windows	Linux
Conhecimento técnico pessoal	9	3
Facilidade de utilização	8	4
Facilidade de integração com os sistemas domésticos	7	8
Grau de reconhecimento no mercado	8	7
Custo de desenvolvimento e de obtenção da tecnologia	10 <sup>4</sup>	10
Grau de sofisticação da solução	3	10
<b>NOTA FINAL</b>	<b>8,2</b>	<b>6,6</b>

Tabela 2 - Avaliação do sistema operacional

<sup>4</sup> O sistema operacional Windows é gratuito para os estudantes da Escola Politécnica. Para uma aplicação comercial, este software teria que ser pago e a nota final seria diferente.

O sistema operacional escolhido foi então o Windows™.

### 5.2.2 Linguagem de programação

	Java	.NET	C++
Conhecimento técnico pessoal	7	3	5
Facilidade de utilização	8	8	4
Facilidade de integração com os sistemas domésticos	8	7	6
Grau de reconhecimento no mercado	8	9	9
Custo de desenvolvimento e de obtenção da tecnologia	10	10	10
Grau de sofisticação da solução	5	5	5
<b>NOTA FINAL</b>	<b>8,0</b>	<b>7,1</b>	<b>6,7</b>

**Tabela 3 - Avaliação da linguagem de programação**

A linguagem de programação escolhida foi Java™, sobretudo pela intimidade que a equipe do presente projeto tem com esta linguagem.

### 5.2.3 Banco de Dados

	My SQL	SQL Server
Conhecimento técnico pessoal	7	5
Facilidade de utilização	8	8
Facilidade de integração com os sistemas domésticos	-	-
Grau de reconhecimento no mercado	8	9
Custo de desenvolvimento e de obtenção da tecnologia	10	5
Grau de sofisticação da solução	8	8
<b>NOTA FINAL</b>	<b>7,1</b>	<b>5,7</b>

**Tabela 4 - Avaliação do banco de dados**

Para a escolha do banco de dados duas grandes opções se apresentam de acordo com os conhecimentos pessoais de cada integrante da equipe deste projeto. Devido ao fato de que o Banco de Dados MySQL é gratuito, ele se mostra muito mais adequado ao projeto se comparado ao SQL Server.

### 5.2.4 Ambiente de desenvolvimento

	Eclipse	Net Beans
Conhecimento técnico pessoal	9	7
Facilidade de utilização	8	8
Facilidade de integração com os sistemas domésticos	-	-
Grau de reconhecimento no mercado	8	8
Custo de desenvolvimento e de obtenção da tecnologia	10	10
Grau de sofisticação da solução	8	8
<b>NOTA FINAL</b>	<b>7,6</b>	<b>7,1</b>

Tabela 5 - Avaliação do ambiente de desenvolvimento

A escolha do ambiente Eclipse se deu devido ao fato que o conhecimento técnico pessoal é superior ao do Net Beans. Julga-se que para os outros fatores os dois bancos de dados são equivalentes.

### 5.2.5 Ambiente de testes

O ambiente de testes é dividido em duas partes: testes unitários e testes funcionais. A escolha do teste funcional foi o aplicativo web **Testlink** e o do teste unitário o **JUnit**. Para realizar a escolha não se usou a metodologia utilizada para os outros casos, pois não se possuía um leque de opções.

### 5.2.6 Ferramenta de gestão de projeto

Gantt Project permite realizar o cronograma do projeto e seguir sua evolução. É uma ferramenta gratuita, sendo esse um dos motivos principais de sua escolha.

## 6 Resultados e Discussões

O conjunto de testes funcionais foi realizado utilizando-se a ferramenta Testlink. Esta ferramenta é dedicada exclusivamente à construção de relatórios de teste e pode ser utilizada de maneira colaborativa já que ela é acessada via browser.

Para as finalidades deste trabalho, duas pessoas trabalharam nos testes (Anderson e Jun), cada um testando algumas funcionalidades do software desenvolvido.

O planejamento de testes (vide “APÊNDICE A”.) foi feito a partir das especificações de casos de uso definidas nesse relatório. A seguir são apresentados os resultados dos testes e um gráfico que avalia as funcionalidades que foram implementadas e que funcionam corretamente.

### 6.1 Capturas de tela

As capturas abaixo mostram a versão final da interface do software desenvolvido.

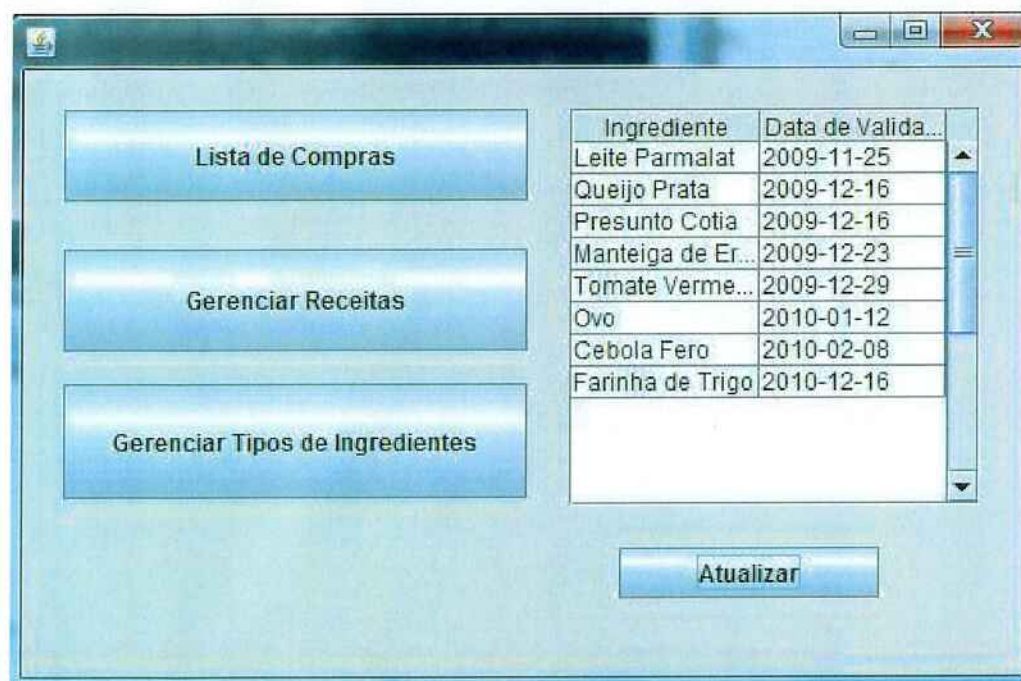
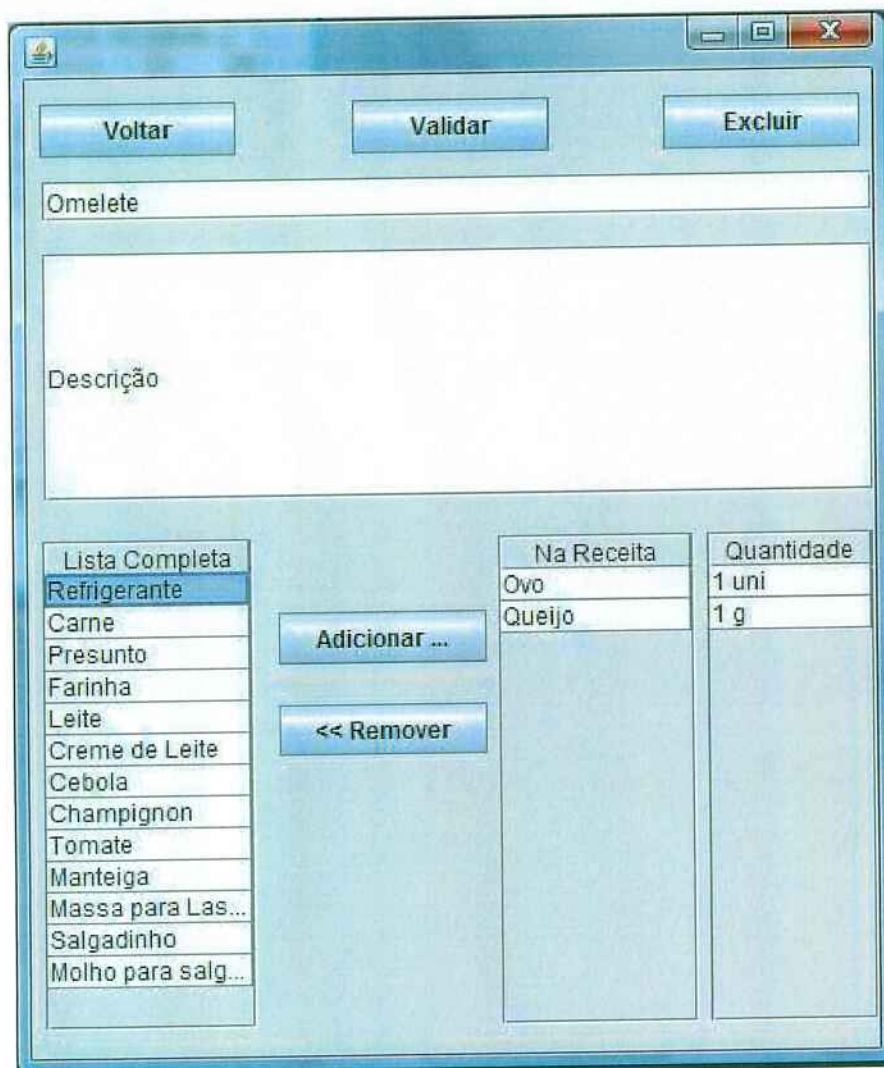


Figura 17 - Telas iniciais. Apresentação dos ingredientes atualmente em estoque.



Voltar Validar Excluir

Omelete

Descrição

Lista Completa	Adicionar ...	<< Remover	Na Receita	Quantidade
Refrigerante			Ovo	1 uni
Carne			Queijo	1 g
Presunto				
Farinha				
Leite				
Creme de Leite				
Cebola				
Champignon				
Tomate				
Manteiga				
Massa para Las...				
Salgadinho				
Molho para salg...				

Figura 18 – Descrição de uma receita.

## 6.2 Resultados

Vide “APÊNDICE B”.

### 6.3 Conclusão dos testes

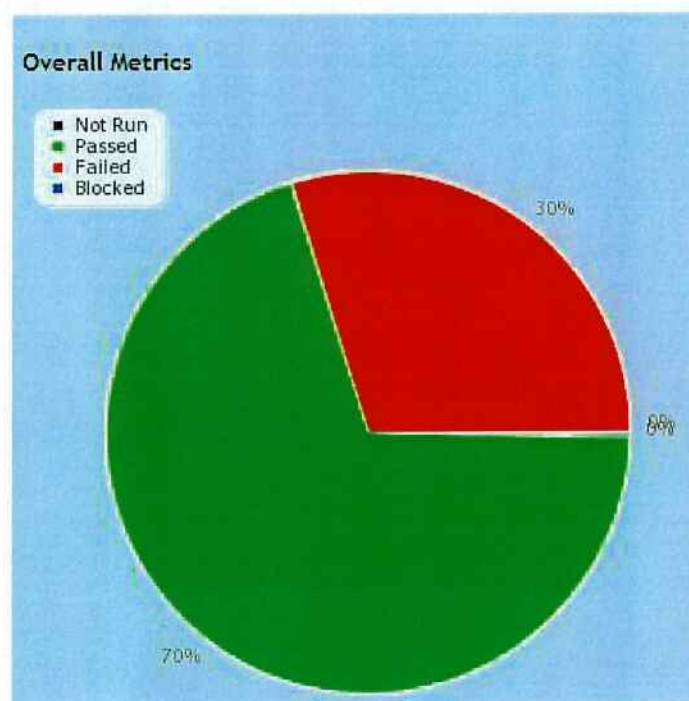


Figura 19 - Resultados dos testes funcionais

O gráfico acima mostra a percentagem de testes válidos. Observa-se que 30% dos testes foram rejeitados e o motivo em todos os casos foi a não implementação da funcionalidade. Estes 30% correspondem a 3 especificações que, em análise rigorosa, são funcionalidades periféricas às funcionalidades principais propostas pelo software. São elas:

- **Modificação de receita:** mesmo que não exista uma funcionalidade direta para modificação de receita, é possível excluir uma receita e depois inseri-la novamente.
- **Notificação de datas de validade:** nenhuma notificação sob forma de mensagem na tela ou e-mail é enviada ao utilizador para notificá-lo das datas de validades dos produtos. É, entretanto, possível de controlar as datas de validades pela tela Inicial já que a lista de produtos presentes na geladeira é ordenada segundo as datas de validades.

- **Lista de compras no assistente digital:** esta funcionalidade não foi implementada devido a limitações de tempo e limitações técnicas.

## **7 Considerações finais**

### **7.1 Dificuldades**

Este trabalho de conclusão de curso representou um desafio aos seus realizadores devido às dificuldades de diversas ordens encontradas durante o desenvolver do projeto. A complexidade do tema, as limitações das fontes documentais, a ignorância dos envolvidos, as restrições no calendário, a disponibilidade dos recursos, os erros – previsíveis e imprevisíveis – e as mudanças técnicas, de planejamento e de escopo foram fatores que dificultaram o bom andamento do projeto. Ao mesmo tempo, agregaram experiências enriquecedoras para os autores do projeto, que foram confrontados a situações para eles inéditas cuja solução significava o cumprimento das metas estabelecidas.

#### **7.1.1 Das dificuldades conceituais**

Se uma inspiração para a proposição do tema de Computação Ubíqua foi uma experiência acadêmica de um dos alunos com o tema, o domínio dos conceitos desse paradigma não era total e limitava-se aos princípios mais básicos e primários. Esse embasamento teórico limitado muitas vezes retardou o andamento do projeto, principalmente em suas especificações iniciais, quando a preocupação de seguir à risca todas as preconizações idealizadas na teoria era constante e um objetivo inicial.

O paradoxo entre a teoria e a prática acentuou-se quando limitações materiais e tecnológicas impediram uma aplicação rígida dos preceitos de Computação Ubíqua. A saber: o fato de existir um sistema de computação onipresente e até certo ponto invisível ao usuário era inicialmente um dos requisitos criados para fins unicamente de correção teórica. No entanto, restrições sobretudo tecnológicas no que diz respeito aos módulos RFID impediam a observação de tal premissa: não havia dispositivos a preço acessível que permitissem a leitura dos dados das etiquetas sem uma atuação ativa e consciente do usuário.

Para que o projeto não se restringisse a um protótipo cuja aplicação estaria condicionada ao desenvolvimento futuro de outras tecnologias, não havendo portanto um interesse comercial e prático imediato, o conceito de transparência do sistema de Computação Ubíqua não foi rigorosamente respeitado, sendo necessária certa interação do usuário quando da inserção de ingredientes e das parametrizações/atualizações do sistema (por exemplo, para inserção de receitas).

Entretanto, buscou-se seguir à medida do possível a teoria, sem descaracterizar completamente um sistema de Computação Ubíqua, que serviu dessa feita como uma inspiração para o sistema proposto e não um fim por si só.

### 7.1.2 Da tecnologia e ferramentas

A rigor, para a funcionalidade do sistema proposto, uma rede convencional bastaria para realizar a comunicação dos diferentes módulos do sistema. Porém, como já foi discutido anteriormente, tal sistemas apresenta desvantagens em relação à utilização do *middleware* Amigo e não permite a implementação de um sistema de Computação Ubíqua. A adoção de tal *framework*, portanto, permitiu o estabelecimento de uma arquitetura condizente com o paradigma supracitado e com escalabilidade suficiente para permitir possíveis evoluções do sistema.

No entanto, tal escolha representou um acréscimo considerável no grau de dificuldade do trabalho. O fato de se tratar de um projeto com alto grau de especificidade e de divulgação relativamente limitada restringiu o leque de possibilidades de fontes de pesquisa ao *síte* do projeto. A documentação pareceu de certo modo insuficiente, tendo sido necessário recorrer a testes para elucidar o funcionamento do *middleware*.

A compreensão do método de comunicação dos dados foi um dos problemas encontrados. O funcionamento da comunicação é explicado por meio de exemplos que dão uma ideia geral de como codificar as classes de comunicação, mas são limitados e não respondem a todas as questões e necessidades levantadas no decorrer deste projeto.

Outro ponto crítico foi a realização do módulo da base de dados, em que a documentação foi igualmente insuficiente e a solução baseou-se em pesquisas realizadas em fontes externas às oficiais.

Além do *middleware*, dificuldades técnicas de ordem geral aconteceram, majoritariamente em virtude da inexperiência dos realizadores do projeto, razão pela qual podem ser consideradas previsíveis. Apenas duas ressalvas relativas à pane de um computador e subsequente perda de trabalho e as dificuldades de compatibilidade entre unidades com diferentes sistemas operacionais.

## 7.2 Gerenciamento de projeto e cronograma

Os percalços encontrados ocasionaram dificuldades no gerenciamento do projeto e perturbações no seu cronograma. Esses atrasos tiveram suas consequências potencializadas quando se considera que o calendário deste projeto era diferente dos demais alunos da disciplina, sendo o conteúdo de duas matérias (Projeto de Formatura I e II) condensados no período de uma.

A utilização do *middleware* Amigo foi o ponto em que a maior parte dos problemas ocorreram, devido às dificuldades nas partes de comunicação de dados e de integração da base de dados. Para contornar tais imprevistos, os esforços tiveram de ser realocados e o cronograma revisto, tendo sido necessário recorrer a reuniões fora dos horários planejados.

À medida em que o cronograma tornou-se mais restrito e o tempo escasso, a divisão de tarefas foi mais rigorosamente estabelecida e seguida. As reuniões (internas e com o orientador), passaram a ser mais planejadas e direcionadas e as discussões e decisões feitas documentadas de forma mais pragmática.

Para que o projeto cumprisse as metas estabelecidas mitigando os atrasos encontrados, foi preciso ter um especial cuidado o respeito das prioridades de desenvolvimento pré-definidas, mesmo que para tanto predileções pessoais e re-transferência de tarefas e responsabilidades tenham sido necessárias. As funcionalidades complementares indicadas nesta monografia não puderam ter sido implementadas, mas seus autores consideram levá-las adiante conforme o sucesso do trabalho.

### 7.3 Conclusão

Existem três aspectos a se considerar como conclusão deste projeto. O primeiro diz respeito à compreensão geral do tema da Computação Ubíqua que pôde ser apreendida. O segundo refere-se à análise dos resultados obtidos neste projeto. Por fim, destacam-se os conhecimentos obtidos pelos alunos que conduziram este trabalho, em relação com a sua formação acadêmica e profissional.

Em se tratando do tema da Computação Ubíqua, pode-se dizer que ainda não é possível disponibilizar um sistema acessível para o gerenciamento da alimentação doméstica, devido a fatores tecnológicos e mercadológicos. Dispositivos de leitor RFID potentes o suficiente para permitir a implementação de tal sistema ainda não existem a preço razoável para clientes não empresariais. Além do alcance reduzido, é preciso ainda customizar os sistemas RFID (etiqueta e leitor) para as diferentes aplicações, o que representa esforços consideráveis.

Quanto à infraestrutura necessária, é mister que os produtos já venham com as informações registradas em sua etiqueta RFID, o que ainda não acontece. Tal cenário, entretanto, não parece ser tão futurista, se considerar-se que esforços já foram feitos nesse sentido, o que explica o fato de que se partiu da premissa de que tais etiquetas já contivessem tais dados. Uma outra possibilidade seria a adição das etiquetas por parte do usuário ou a utilização de recipientes específicos com RFID para o armazenamento dos ingredientes.

Os resultados obtidos estiveram de acordo com o especificado, havendo poucos desvios. Melhorias são sempre possíveis, mas por parte dos realizadores do projeto os resultados foram satisfatórios. Um ponto a se lamentar é a não realização das funcionalidades complementares do sistema, que não acrescentariam muito em termo de sua arquitetura, mas teriam valores na exposição do projeto.

No tocante do aprendizado pessoal, pode-se destacar a introdução a uma nova área do conhecimento da Computação, a aplicação de conceitos de gestão de projetos de desenvolvimento de Sistemas de Informação, a sensibilização às problemáticas do trabalho em equipe, o exercício de conhecimentos teóricos em programação Java e a realização pessoal de ter realizado um projeto de seu interesse e em concordância com a sua formação.

## 7.4 Extensões do sistema

Como perspectivas de evolução deste projeto, sugere-se as seguintes funcionalidades:

- Criar uma estrutura de dados em XML para as receitas, para que seja facilitado o processo de importação e exportação de receitas entre diferentes sistemas, possibilitando o compartilhamento de receitas entre diferentes usuários.
- Integrar o forno ao sistema para que ele se auto-programe de acordo com a receita a preparar.
- Integrar o forno de micro-ondas ao sistema para que ele se auto-programe de acordo com a receita a preparar.
- Para evitar os erros de digitação na hora de inserir os ingredientes, na tela de cadastro de receita, e para padronizar os nomes de ingredientes, deve-se fazer com que o usuário possa escolher os ingredientes a partir de uma lista pré-definida organizada em categorias. Para tanto, deve-se criar (ou procurar) uma ontologia padrão para os ingredientes.
- Na tela de cadastro de receita, após a escolha do ingrediente, o sistema deve propor por meio de uma combobox uma lista de possíveis unidades de medida para o dado ingrediente. Por exemplo, para sal, deve-se propor gramas, colheres de sopa, colheres de chá ou pitadas. Ao gravar na base de dados, o sistema deve converter as quantidades para uma unidade padrão. Para tanto, deve-se criar (ou procurar) uma ontologia para unidades de medida e associar essa ontologia à ontologia dos ingredientes.

## 8 Referências

AMIGO CHALLENGE – TUTORIALS. Tutoriais sobre como desenvolver aplicações pervasivas usando o projeto Amigo. Disponível em: <<http://www.extra.research.philips.com/amigochallenge/tutorials.htm>>. Acesso em: 02 mai. 2009.

BALZER, Y. Improve your SOA project plans. IBM, 2004. Disponível em: <<http://www.ibm.com/developerworks/webservices/library/ws-improvesoa/>>. Acesso em: 01 dez. 2009.

BELL, G. ; DOURISH, P. Yesterday's tomorrows: notes on ubiquitous computing's dominant vision. Disponível em <<http://www.ics.uci.edu/~jpd/ubicomp/BellDourish-YesterdaysTomorrows.pdf>> Acesso em 05/12/09

BOISSIER, O. Web Services – Introduction. Master Web Intelligence. Ecole Nationale Supérieure des Mines de Saint-Etienne. Saint-Etienne, 2007.

CAPTAIN'S UNIVERSE – Site com tutorial para utilização de RFID com Java Disponível em: <<http://www.captain.at/howto-java-serial-port-javax-comm-rxtx.php>> Acesso em 15/11/09

CHI, P.; CHEN, J.; CHU, H.; CHEN, B. Enabling Nutrition-Aware Cooking in a Smart Kitchen. Conference on Human Factors in Computing Systems. San Jose, CA, USA, 2007. Disponível em: <[http://mll.csie.ntu.edu.tw/papers/WiP309\\_chi.pdf](http://mll.csie.ntu.edu.tw/papers/WiP309_chi.pdf)>. Acesso em: 26 jan. 2008.

CUGNASCA, C. E. Técnicas de Desenvolvimento de Projetos Baseados em Microprocessadores. Laboratório de Processadores I. Departamento de Engenharia de Computação e Sistemas Digitais. Escola Politécnica da Universidade de São Paulo. São Paulo, 2006. Disponível em: <<http://www.pcs.usp.br/~pcs2497/>>. Acesso em: 20 ago. 2009.

EASY RFID – Site da comunidade destinada a aplicações que utilizam RFID em aplicações Java. Disponível em: < <http://www.smartlab.deusto.es/java-rfid/> > Acesso em 15/11/09.

LANGHEINRICH, M.; MATTERN, F.; RÖMER, K.; VO

GT, H. First Steps Towards an Event-Based Infrastructure for Smart Things. Distributed System Group, Department of Computer Science. ETH Zurich, Swiss Federal Institute of Technology. Disponível em: <<http://www.vs.inf.ethz.ch/publ/papers/firststeps.pdf>>. Acesso em: 10 jan. 2008.

LUBLINSKY, B. Defining SOA as an architectural style. Align your business model with technology. IBM, 2007. Disponível em: <<http://www.ibm.com/developerworks/library/ar-soastyle/>>. Acesso em: 02 dez. 2009.

MARKS, E., A.; BELL, M. **Service Oriented Architecture**. A Planning and Implementation Guide for Business and Technology. New Jersey: John Wiley & Sons, Inc., 2006. 376 p.

OPEN GROUP (SOA Definition Team of the SOA Working Group). Definition of SOA. 2006. Disponível em: <<http://opengroup.org/projects/soa/doc.tpl?gdid=10632>>. Acessado em: 03 dez. 2009.

OSGI. Site do OSGi™ Alliance. OSGi Technology. 2009. Disponível em: <<http://www.osgi.org/About/Technology>>. Acesso em: 03 dez. 2009.

PERVASIVE COMPUTING RESEARCH GROUP. Site do grupo de pesquisa sobre computação pervasiva da Universidade Federal de Santa Catarina. Disponível em: <[http://www.inf.ufsc.br/~rcampiol/pervasive\\_computing/](http://www.inf.ufsc.br/~rcampiol/pervasive_computing/)>. Acesso em: 02 nov. 2009.

SIERRA, K.; BATES, B. SCJP Sun® Certified Programmer for Java™ 5 Study Guide (Exam 310-055). Emeryville, California : McGraw-Hill/Osborne, 2006. 825p.

SOYLU, A.; DE CAUSMAECKER, P. ;DESMET, P. Context and Adaptivity in Pervasive Computing Environments: Links with Software Engineering and Ontological Engineering. Artigo do Journal of Software, Vol 4, No 9, Novembro 2009

WEISER, M. The Computer for the 21st Century. **Scientific American Special Issue on Communications, Computers, and Networks**, Sep. 1991. Disponível em: <http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html>>. Acesso em: 20 nov. 2009.

WEISER, M. Ubiquitous Computing. Computer Sciences Laboratory, Xerox PARC, 1996. Disponível em <http://sandbox.xerox.com/ubicomp/>>. Acesso em 10 ago. 2009.

## 9 Referências complementares

ACUPROX. Leitor de proximidade AP 09

COUNTER INTELLIGENCE – MIT MEDIA LAB. Projeto de cozinha inteligente do MIT Media Lab. Disponível em: <<http://www.media.mit.edu/ci/>>. Acesso em: 22 jan. 2008.

RÖMER, K.; SCHOCH, T. Infrastructure Concepts for Tag-Based Ubiquitous Computing Applications. Department of Computer Science. ETH Zurich, Swiss Federal Institute of Technology. Disponível em: <<http://www.vs.inf.ethz.ch/res/papers/infrastructure-concepts.pdf>>. Acesso em: 20 jan. 2008.

## GLOSSÁRIO

**SOA** – *Service Oriented Architecture*. Arquitetura de *software* na qual a funcionalidade do sistema é decomposta em diversos serviços simples realizados por diferentes componentes.

**Computação ubíqua** – paradigma de computação que explora a crescente capacidade de processamento e armazenamento de dispositivos móveis, propondo a invisibilidade do sistema na óptica do usuário, utilizando-se de interfaces naturais.

**Amigo** – *middleware de código aberto* que integra dispositivos e aplicativos de um sistema de computação ubíqua, realizado por um consórcio de empresas e universidades europeias.

**Middleware** – *software* que gerencia a interligação de dois ou mais aplicativos em diferentes máquinas, possivelmente escritos em diferentes linguagens.

**RFID** – *Radio Frequency Identification*. Sistema (tecnologia e padrão) de identificação eletrônica que usa dispositivos acoplados a objetos para transmissão de dados sem fio.

## APÊNDICE A – Plano de testes funcionais



### PLANO DE TESTES

Projeto: TCC Geladeira

Autor: Anderson

#### 1 Suíte de Teste: Módulo Computador Pessoal

##### 1.1 Suíte de Teste: Gerenciamento de Receitas

<b>Caso de Teste TCC-1: Inserção de Receita</b>
Inserir receita na base de dados
<b>Caso de Teste TCC-2: Remoção de Receita</b>
Remover receita da base de dados
<b>Caso de Teste TCC-3: Determinar receitas possíveis</b>
Determinar as receitas possíveis com os alimentos atualmente na geladeira
<b>Caso de Teste TCC-5: Modificar Receita</b>
O usuário faz uma modificação na configuração de uma receita e ela é guardada na base de dados.

## 1.2 Suíte de Teste : Gerenciamento do Estoque

### Caso de Teste TCC-4: Inserção e remoção de Ingredientes na geladeira

O usuário utiliza a etiqueta RFID dos alimentos para inserir ou remover os alimentos que estão na geladeira.

### Caso de Teste TCC-7: Lista de compras

Calcula uma lista de compras baseado nos ingredientes favoritos do utilizador

### Caso de Teste TCC-8: Lista de alimentos presentes na geladeira

Esta opção permite ao usuário visualizar uma lista com os alimentos que estão dentro da geladeira

### Caso de Teste TCC-9: Gerenciamento de Ingredientes favoritos

Permite ao usuário indicar quais seus ingredientes favoritos para a lista de compras.

### Caso de Teste TCC-11: Aviso sobre data de validade de alimentos

Um aviso é enviado automaticamente ao utilizador (por email ou por um aviso no desktop do computador pessoal) para avisar que alguns alimentos estão fora da data de validade. O utilizador pode configurar um intervalo de tempo antes da data de validade do produto para que a notificação de validade lhe seja enviada.

## 2 Suíte de Teste: Módulo Assistente Digital Pessoal

### Test Case TCC-10: Lista de compras

Permite ao usuário ter acesso remoto aos ingredientes em falta em sua geladeira.

## APÊNDICE B – Resultados dos testes funcionais

### 1 Suíte de Teste : Módulo Computador Pessoal

#### 1.1 Suíte de Testes: Gerenciamento de Receitas

Caso de Teste TCC-1: Inserção de Receita	
<u>Resultado:</u>	<b>Aprovado</b>
Testador	jun
Notas do Teste	Funciona corretamente
Caso de Teste TCC-2: Remoção de Receita	
<u>Resultado:</u>	<b>Aprovado</b>
Testador	jun
Notas do Teste	Funciona corretamente
Caso de Teste TCC-3: Determinar receitas possíveis	
<u>Resultado:</u>	<b>Aprovado</b>
Testador	jun
Notas do Teste	Funciona corretamente
Caso de Teste TCC-5: Modificar Receita	
<u>Resultado:</u>	<b>Falha</b>
Testador	jun
Notas do Teste	Não é possível modificar a receita. Para tanto, é preciso excluir a antiga e inserir uma nova

#### 1.2 Suíte de Teste : Gerenciamento do Estoque

Caso de Teste TCC-4: Inserção e remoção de Ingredientes na geladeira	
<u>Resultado:</u>	<b>Aprovado</b>
Testador	Anderson
Notas do Teste	Funciona corretamente. Entretanto em algumas ocasiões a etiqueta não é lida corretamente e é necessário passá-la novamente pelo leitor.
Caso de Teste TCC-7: Lista de compras	
<u>Resultado:</u>	<b>Aprovado</b>

Testador	Anderson
Notas do Teste	Funciona corretamente
<b>Caso de Teste TCC-8: Lista de alimentos presentes na geladeira</b>	
<u>Resultado:</u>	<b>Aprovado</b>
Testador	Anderson
Notas do Teste	Funciona corretamente. Uma ressalva: não são apresentadas as informações nutricionais dos ingredientes.
<b>Caso de Teste TCC-9: Gerenciamento de Ingredientes favoritos</b>	
<u>Resultado:</u>	<b>Aprovado</b>
Testador	Anderson
Notas do Teste	Funciona corretamente. A interface pode ser melhorada para facilitar a seleção dos ingredientes favoritos.
<b>Caso de Teste TCC-11: Aviso sobre data de validade de alimentos</b>	
<u>Resultado:</u>	<b>Falha</b>
Testador	Anderson
Notas do Teste	Esta função não foi implementada.

## 2 Suíte de Teste : Módulo Assistente Digital Pessoal

<b>Caso de Teste TCC-10: Lista de compras</b>	
<u>Resultado:</u>	<b>Falha</b>
Testador	Anderson
Notas do Teste	Esta funcionalidade não foi implementada.

## APÊNDICE C – Detalhes do cálculo do orçamento de projeto

Orçamento do projeto:

Item	Quantidade	Valor Unitário	Total
Computadores Pessoais	3	R\$ 1 500,00	R\$ 4 500,00
Leitor RFID	1	R\$ 700,00	R\$ 700,00
Etiquetas RFID	15	R\$ 0,50	R\$ 7,50
Transporte (gasolina e pedágio)	-	-	R\$ 1 500,00
Papel e tinta para impressão	-	-	R\$ 50,00
CD	1	R\$ 3,00	R\$ 3,00
<b>TOTAL</b>			<b>R\$ 6 760,50</b>

**Tabela 6 – Tabela do orçamento destinado ao projeto**

Dos custos apresentados, apenas o leitor RFID, as etiquetas, as impressões e o CD seriam custos reais do projeto, já que os computadores são itens pessoais e as despesas com transportes não seriam evitadas, pois era necessário de qualquer forma ir até a Cidade Universitária para outros compromissos.