

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Lincoln Borges Ferreira

**Um sistema para previsão de tempo em fila de
espera utilizando Lógica Fuzzy**

Monte Carmelo, Brasil

2017

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Lincoln Borges Ferreira

**Um sistema para previsão de tempo em fila de espera
utilizando Lógica Fuzzy**

Trabalho de conclusão de curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, Minas Gerais, como
requisito exigido parcial à obtenção do grau
de Bacharel em Sistemas de Informação.

Orientadora: Profa. MSc. Daniele Carvalho Oliveira

Co-Orientadora: Profa. Dra. Fernanda Maria da Cunha Santos

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Sistemas de Informação

Monte Carmelo, Brasil

2017

Lincoln Borges Ferreira

Um sistema para previsão de tempo em fila de espera utilizando Lógica Fuzzy

Trabalho de conclusão de curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, Minas Gerais, como
requisito exigido parcial à obtenção do grau
de Bacharel em Sistemas de Informação.

Trabalho aprovado. Monte Carmelo, Brasil, 21 de dezembro de 2017:

Profa. MSc. Daniele Carvalho Oliveira
Orientador

Profa. Dra. Fernanda Maria da Cunha
Co-Orientadora

Prof. Dr. Murillo Guimarães Carneiro

**Prof. Dr. Carlos Ramon Pantaleon
Dionísio**

Monte Carmelo, Brasil
2017

Dedico este trabalho aos meus pais que estiveram do meu lado desde o início e que acreditaram em mim.

Agradecimentos

À Deus por tudo por ter me proporcionado saúde e motivação ao longo de minha vida.

Aos meus pais pelo incentivo, confiança e amor incondicional.

Às minhas orientadoras Daniele Carvalho Oliveira e Fernanda Maria da Cunha Santos pela paciência e por todo o incentivo que me deram para concluir este trabalho.

Aos meus amigos que conheci durante o curso, que considero como irmãos, por toda ajuda e suporte.

À banca examinadora por aceitar fazer parte desse momento tão importante na minha vida.

“A felicidade não se resume na ausência de problemas, mas sim na sua capacidade de lidar com eles.” (Albert Einstein)

Resumo

Restaurantes muito procurados precisam lidar com uma longa fila de espera, e para evitar desconforto e insatisfação dos clientes costumam implementar sistemas que mostram uma previsão de tempo para atendimento. Essa previsão, no entanto, é difícil de ser calculada, pois precisa levar em consideração muitos fatores. Assim, frequentemente, o cálculo é feito através de análises estatísticas realizadas em cima de um histórico de atendimento. Este trabalho propõe a criação de um sistema de previsão de tempo em fila de espera por mesas de restaurantes utilizando a Lógica Fuzzy. Foi visto como resultado que a Lógica Fuzzy é ideal para resolver problemas de fila de espera, no qual pode se ter um tempo de espera muito preciso se for feito um estudo de caso em um restaurante.

Palavras-chave: Lógica Fuzzy, previsão, fila de espera, restaurante.

Lista de ilustrações

Figura 1 – Modelo da lógica fuzzy.	16
Figura 2 – Função de pertinência Singleton.	18
Figura 3 – Gráfico da função de pertinência triangular.	18
Figura 4 – Função de pertinência trapezoidal.	19
Figura 5 – Etapas do Raciocínio Fuzzy	19
Figura 6 – Fórmula do Centro de Gravidade	20
Figura 7 – Gráfico criado pela biblioteca JFuzzyLogic com 2 variáveis de entrada e 1 resultado.	21
Figura 8 – Método de ativação utilizada na JFuzzyLogic	22
Figura 9 – Método de acumulação utilizada na JFuzzyLogic	22
Figura 10 – Representação da função Singleton da biblioteca JFuzzyLogic	23
Figura 11 – Representação da função Triangular da biblioteca JFuzzyLogic	24
Figura 12 – Representação da função Trapezoidal da biblioteca JFuzzyLogic	24
Figura 13 – Modelo de um Fuzzy Control Language	25
Figura 14 – Bloco de fuzzyficação	25
Figura 15 – Bloco de defuzzyficação	26
Figura 16 – Modelo de um bloco de regras.	26
Figura 17 – Exemplo da saída do sistema Fuzzy aplicado na previsão da execução de <i>queries</i>	27
Figura 18 – Diagrama de Arquitetura do Sistema	29
Figura 19 – Diagrama de Atividades do Projeto	32
Figura 20 – Diagrama de Atividades do Projeto	32
Figura 21 – Modelo do banco de dados	34
Figura 22 – Rotina da <i>procedure</i>	34
Figura 23 – Declaração das variáveis utilizadas no arquivo FCL	36
Figura 24 – Gráfico representando 7 pessoas na biblioteca JFuzzyLogic	37
Figura 25 – Gráfico representando a variável <i>Singleton</i> bebida com o valor 2	37
Figura 26 – Bloco de defuzzyficação da variável tempo.	38
Figura 27 – Exemplo das regras criadas no Bloco de Regras	38
Figura 28 – Código para execução da biblioteca JFuzzyLogic	39
Figura 29 – Código da interface do Webservice em Java	40
Figura 30 – Menu da aplicação cliente.	41
Figura 31 – Cadastro de funcionários da aplicação cliente.	41
Figura 32 – Listagem das mesas da aplicação cliente.	42
Figura 33 – Tela de ocupar mesas da aplicação cliente.	43
Figura 34 – Listagem das mesas para o cliente.	44

Figura 35 – Parte 1 do questionário realizado	48
Figura 36 – Parte 2 do questionário realizado	49
Figura 37 – Parte 3 do questionário realizado	49
Figura 38 – Respostas da primeira pergunta do questionário	50
Figura 39 – Respostas da segunda pergunta do questionário	51
Figura 40 – Respostas da terceira pergunta do questionário	51
Figura 41 – Respostas da quarta pergunta do questionário	52
Figura 42 – Respostas da quinta pergunta do questionário	52
Figura 43 – Respostas da sexta pergunta do questionário	53
Figura 44 – Respostas da sétima pergunta do questionário	53
Figura 45 – Respostas da oitava pergunta do questionário	54
Figura 46 – Regras definidas no Bloco de Regras	55

Lista de tabelas

Tabela 1	–	Requisitos Funcionais do Sistema	31
Tabela 2	–	Dicionário de dados da entidade eventos diários	33
Tabela 3	–	Dicionário de dados da entidade mesas	33
Tabela 4	–	Dicionário de dados da entidade usuário	33
Tabela 5	–	Dicionário de dados da entidade tempomesa	33
Tabela 6	–	Dicionário de dados da entidade tempocliente	34

Lista de abreviaturas e siglas

COG	Center of Gravity
FCL	Fuzzy Control Language
SOAP	Simple Object Access Protocol
RPC	Remote Procedure Call
URL	Uniform Resource Locator
XML	eXtensible Markup Language

Sumário

1	INTRODUÇÃO	13
2	LÓGICA FUZZY	15
2.1	Lógica Fuzzy	15
2.1.1	Conjuntos Nebulosos	16
2.1.2	Variáveis Linguísticas	17
2.1.3	Funções de Pertinência	17
2.1.3.1	Singleton	17
2.1.3.2	Triangular	18
2.1.3.3	Trapezoidal	18
2.1.4	Raciocínio Fuzzy	19
2.2	JFuzzyLogic	20
2.2.1	Bloco de Regras	20
2.2.2	Métodos do bloco de regras	21
2.2.3	Representação das funções do JFuzzyLogic	22
2.2.4	Elementos do Fuzzy Control Language	24
2.3	Trabalhos Relacionados	26
3	DESENVOLVIMENTO	29
3.1	Arquitetura do Sistema	29
3.2	Modelagem do Sistema	30
3.2.1	Análise de Requisitos	30
3.2.2	Diagrama de Atividades	30
3.2.3	Diagrama de Caso de Uso	31
3.3	Banco de Dados	33
3.4	Codificação	35
3.4.1	Criação do arquivo FCL	35
3.4.2	Criação do Servidor Fuzzy	39
3.4.3	Criação da aplicação cliente	40
4	CONCLUSÃO	45
4.1	Considerações Finais	45
4.2	Trabalhos Futuros	45
	REFERÊNCIAS	46

APÊNDICES	47
APÊNDICE A – QUESTIONÁRIO	48
APÊNDICE B – RESULTADOS OBTIDOS PELO QUESTIONÁRIO	50
APÊNDICE C – BLOCO DE REGRAS UTILIZADO NO ARQUIVO	
FCL	55

1 Introdução

A percepção do tempo de espera de um cliente em um restaurante pode ser crítica em relação à qualidade de serviço, uma vez que o tempo percebido pelo consumidor pode ser bem maior do que o analisado pelo dono do estabelecimento causando assim um desconforto para quem espera. No entanto, quanto maior o tempo de espera de um cliente, maior é a sua insatisfação com o serviço, e é típico em restaurantes grandes as pessoas terem que esperar um longo tempo na fila para serem atendidas sem ao menos saberem o tempo que gastarão esperando.

Este trabalho propõe a criação de uma aplicação, de forma a reduzir o impacto da espera do atendimento ao prever o tempo de espera que o cliente terá que esperar para ser atendido através da análise do uso das mesas e o comportamento dos clientes que estão em atendimento. O objetivo geral da aplicação consiste em calcular e exibir um tempo estimado para as pessoas que estão esperando para que possam ter uma noção do tempo que irão gastar esperando para serem atendidas.

O método a ser utilizado para o cálculo será a Lógica Fuzzy, no qual dado as variáveis de entrada, previamente estabelecidas, o sistema tentará prever, baseado em situações diferentes de cada consumidor, um tempo estimado que será gasto desde o momento que sentam na mesa, até o momento que saem da mesa. O cálculo será realizado usando a biblioteca JFuzzyLogic, disponível para a linguagem Java.

O sistema será desenvolvido como um WebService fornecendo serviço para dois tipos de clientes. De um lado, uma aplicação pelos funcionários do restaurante, que deverão inserir informações e as variáveis para o cálculo da estimativa de tempo. De outro lado, tem-se uma aplicação acessada pelos clientes dos restaurantes contendo a estimativa de espera. O lado servidor será responsável por calcular a estimativa, com base nas informações dadas pelos funcionários a respeito do cliente.

O objetivo deste trabalho é calcular o tempo que os clientes de um restaurante irão gastar do momento em que sentam em uma mesa até a saída do restaurante e desenvolver uma aplicação que consiga realizar esse cálculo do tempo de espera na fila do restaurante, baseada em Lógica Fuzzy, exibindo para os clientes que estão esperando, uma estimativa do tempo da próxima mesa a ser liberada.

Este trabalho foi organizado em 4 capítulos, sendo este a Introdução onde é apresentado o tema, com suas justificativas e objetivos. O segundo capítulo apresenta a fundamentação teórica necessária para o desenvolvimento deste trabalho como a Lógica Fuzzy, a biblioteca JFuzzyLogic e os trabalhos relacionados. O terceiro capítulo apresenta o desenvolvimento das aplicações cliente e servidor e como as duas se comunicam entre si para

envio e recebimento das variáveis. O último capítulo apresenta a conclusão do desenvolvimento deste trabalho.

2 Lógica Fuzzy

Lógica é definida por [Mendelson \(1987\)](#) como o raciocínio e validação de argumentos, não importando de fato se as afirmações são verdadeiras, mas sim com sua validade, como por exemplo:

Todos os limões são azuis

Maria é um limão

Então, Maria é azul

Neste exemplo, as afirmações não fazem muito sentido, mas a conclusão que Maria é azul, é uma consequência lógica das duas afirmações iniciais, denominadas premissas.

Expressões lógicas, que são criadas a partir de regras do mundo real, são descritas com o uso de operadores lógicos, como por exemplo:

1. *and*: $\wedge$
2. *or*: $\vee$
3. *not*: $\neg$
4. *implies*: $\rightarrow$
5. *iff*: $\iff$

Ou seja, a frase: *está chovendo e é segunda-feira*, pode ser transformada em uma expressão lógica da seguinte forma:

$C \wedge S$

no qual C representa o argumento **está chovendo** e o S representa o argumento **é segunda-feira**.

A lógica clássica é composta por dois valores, 1 para representar verdadeiro e 0 para o falso. A lógica polivalente pode abranger além dos valores da lógica clássica, o valor indeterminado, que é representado por um intervalo de números reais entre 0 e 1 mostrando o grau de verdade. ([COPPIN, 2004](#)) ([PILAN, 2011](#))

2.1 Lógica Fuzzy

A Lógica Fuzzy é baseada na teoria dos conjuntos fuzzy sendo diferente da lógica tradicional. Na lógica tradicional valores podem assumir verdadeiro (1) ou falso (0) e

na Lógica Fuzzy esses valores podem assumir uma série de valores finitos dentro de um intervalo. Na lógica binária os predicados são exatos (menor que, par, igual a), já na Lógica Fuzzy esses predicados são nebulosos (muito pouco, pouco, médio). Em relação aos quantificadores, a lógica tradicional usa somente quantificadores existenciais como ($\exists$) e universais ($\forall$), enquanto que na lógica nebulosa pode-se utilizar diversos (poucos, frequentemente, às vezes, em alguns dias). (GOMIDE; GUDWIN, 1994)(ZADEH, 1988)

Na Lógica Fuzzy os valores podem assumir um intervalo de intensidade (muito quente, quente, morno, frio, muito frio, ...), cada um desses termos são interpretados como um subconjunto Fuzzy. A Figura 1 exemplifica o modelo de um conjunto Fuzzy, com graus de intensidade de 0 a 1 das seguintes variáveis: "poor", "good", "excellent". Por exemplo: O ponto 0 tem intensidade máxima na variável "poor", e o ponto 1 tem intensidade mínima em "good", mas em relação a distância o ponto 1 possui um valor maior que o ponto 0 mesmo com intensidade menor. (GOMIDE; GUDWIN, 1994)

Na Lógica Fuzzy, como os valores são resultados de uma estimativa, o rigor aplicado no resultado final não é tão importante como na lógica tradicional, ou seja, a precisão do resultado é uma questão de grau.(ZADEH, 1988)



Figura 1 – Modelo da lógica fuzzy.

Fonte: Cingolani e Alcalá-Fdez (2012)

2.1.1 Conjuntos Nebulosos

A lógica fuzzy é utilizada para raciocinar sobre conjuntos nebulosos, que são contrastados com os conjuntos utilizados na teoria tradicional de conjuntos, conhecidos como conjuntos discretos (crisp). (COPPIN, 2004)

Um conjunto discreto é definido pelos valores que contém, ou seja está no conjunto, ou não. Por exemplo, um conjunto discreto de números naturais, valores como 1, 2, 3, 4, fazem parte desse conjunto, valores como: 1,23 e π , não fazem parte desse conjunto.

Dado um conjunto de pessoas velhas, uma pessoa x possui 80 anos, uma pessoa y possui 40 anos, e uma pessoa z possui 20 anos. A pessoa x pertence ao conjunto de pessoas velhas, possivelmente a pessoa y também seria colocada no conjunto, dificilmente a pessoa z pertenceria ao conjunto. Mas de certa forma, todas as pessoas pertencem em certo grau ao conjunto, e ao mesmo tempo não fazem parte do conjunto em algum grau. Isso mostra que a definição de conjuntos nebulosos é mais sobre o que as pessoas pensam sobre as coisas, pois é muito comum dizer que a pessoa é pouco velha, ou muito velha ao invés de dizer velha, ou não velha. (COPPIN, 2004)

2.1.2 Variáveis Linguísticas

Na Lógica Fuzzy as variáveis linguísticas são definidas por variáveis cujo valores são nomes de conjuntos Fuzzy como por exemplo valores relacionados à *quantidade de pessoas* em um dado processo que podem variar, por exemplo, entre *poucas*, *poucas mas não muito poucas*, *muitas*, *muitas mas não multidão*, *multidão*, e seus valores são descritos por intermédios de conjuntos Fuzzy. Nesse exemplo, as variáveis são formadas a partir dos nomes *poucas*, *muitas* e *multidão*, da negação *não*, do conectivo *mas* e do modificador *muito*. (COPPIN, 2004) (LEE, 1990)

Então temos quatro categorias representadas para valores compostos, como:

1. Termos Primários: por exemplo as palavras *poucas* e *multidão*.
2. Conectivos: podem ser definidos pelos termos *não* e *e*.
3. Modificadores: como por exemplo, *muito* e *bastante*.
4. Delimitadores: definidos por parênteses, por exemplo.

2.1.3 Funções de Pertinência

Conjuntos Fuzzy são caracterizados pelas funções de pertinência. Nessas funções são representadas o quanto um elemento pertence ao conjunto. (ZIMMERMANN, 2011) Existem diferentes tipos de funções para representar.

2.1.3.1 Singleton

A função de pertinência Singleton pode ser representada por 2 valores diferentes. A Figura 2 exibe como é representado a função definida pela fórmula a seguir:

$$Singleton(x) = \begin{cases} 0, & x \neq m \\ 1, & x = m \end{cases} \quad (2.1)$$

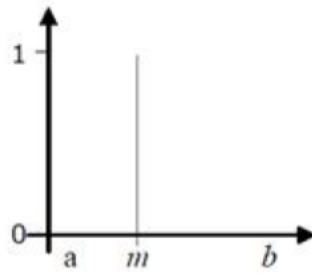


Figura 2 – Função de pertinência Singleton.

Fonte: Adaptado de [Zadeh \(1988\)](#)

2.1.3.2 Triangular

Definido pelo limite inferior a , um limite superior b e o valor modal m , com a regra que $a < m < b$ é dado a representação dessa função de pertinência na Figura 3 dada pela fórmula a seguir:

$$Triangular(x) = \begin{cases} 0, & x \leq a \\ (x - a)/(m - a), & x \in (a, m] \\ (b - x)/(b - m), & x \in (m, b) \\ 0, & x \geq b \end{cases} \quad (2.2)$$

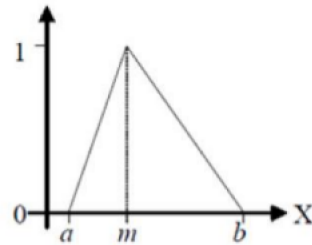


Figura 3 – Gráfico da função de pertinência triangular.

Fonte: Adaptado de [Zadeh \(1988\)](#)

2.1.3.3 Trapezoidal

Definido pelo limite inferior a o limite superior d e seus limites inferiores do núcleo b e c respectivamente. A Figura 4 mostra sua representação dada pela fórmula a seguir:

$$Trapezoidal(x) = \begin{cases} 0, & (x \leq a) \\ (x - a)/(b - a), & x \in (a, b] \\ 1, & x \in (b, c) \\ (d - x)/(d - c), & x \in (c, d) \\ 0, & (x \geq d) \end{cases} \quad (2.3)$$

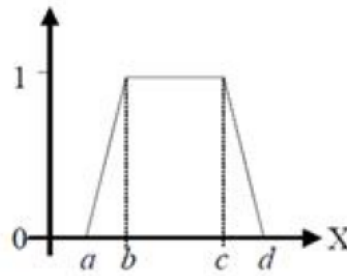


Figura 4 – Função de pertinência trapezoidal.

Fonte: Adaptado de [Zadeh \(1988\)](#)

2.1.4 Raciocínio Fuzzy

Segundo o autor [Medeiros \(2011\)](#) o raciocínio Fuzzy é composto de 3 etapas, sendo elas a Fuzzyficação, a Inferência e a Defuzzyficação, como mostra a Figura 5.

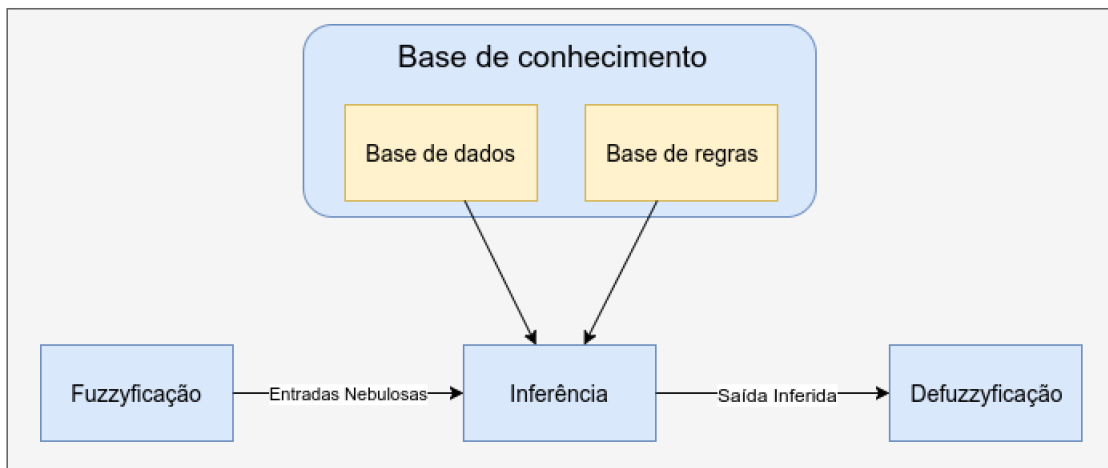


Figura 5 – Etapas do Raciocínio Fuzzy

Fonte: Do autor

As entradas são compostas de valores numéricos que são transformadas em variáveis linguísticas que serão utilizadas no processo de Inferência. A saída é dada pelo resultado da inferência das regras que compõem o bloco de regras sobre as entradas.

Segundo o autor [Mendel \(1995\)](#) as regras são compostas por sentenças *Se x então y*, como por exemplo: *Se chove então a rua está molhada*. Essas regras são conectadas através de operadores lógicos como *And* e *Or*.

O processo de Fuzzyficação é explicado pelos autores [Simoes e Shaw \(2007\)](#) como um processo de atribuir valores linguísticos definidos pelas funções de pertinência às variáveis de entrada.

A Inferência tem o papel de combinar as regras definidas criando uma base para tomadas de decisões. ([MENDEL, 1995](#))

Segundo os autores [Simoes e Shaw \(2007\)](#) o defuzzyficador tem o trabalho de decodificar as variáveis linguísticas de saída depois de serem inferidas pelas regras estabelecidas no Bloco de Regras para valores reais, um modo de realizar essa defuzzyficação é através da fórmula do Centro de Gravidade ou Média dos Máximos, que é dada pela fórmula que está na Figura 6.

$$COG = \frac{\sum_{x=a}^b \mu(x) \cdot x}{\sum_{x=a}^b \mu(x)}$$

Figura 6 – Fórmula do Centro de Gravidade

Fonte: [Coppin \(2004\)](#)

2.2 JFuzzyLogic

JFuzzyLogic é uma biblioteca *open source* escrita em Java utilizada para realizar cálculos baseados em Lógica Fuzzy. Essa biblioteca consegue interpretar as regras e variáveis escritas em FCL (Fuzzy Control Language) e realizar o cálculo a partir das funções e regras definidas e exibindo em gráficos os resultados, como mostra a Figura 7. ([CINGOLANI; ALCALA-FDEZ, 2012](#))

A inferência criada é composta por um ou mais blocos de função (*Function Blocks*) contendo variáveis de entrada, saída e de instância, fuzzyficadores, defuzzyficadores, além de possuir também os blocos de regras chamados de *Rule Blocks*. ([PASSINO; YURKOVICH; REINFRANK, 1998](#))

2.2.1 Bloco de Regras

O bloco de regras, segundo o autor [Cingolani e Alcala-Fdez \(2012\)](#), é composto por uma parte antecedente (IF) e a parte consequente (THEN), como por exemplo "IF

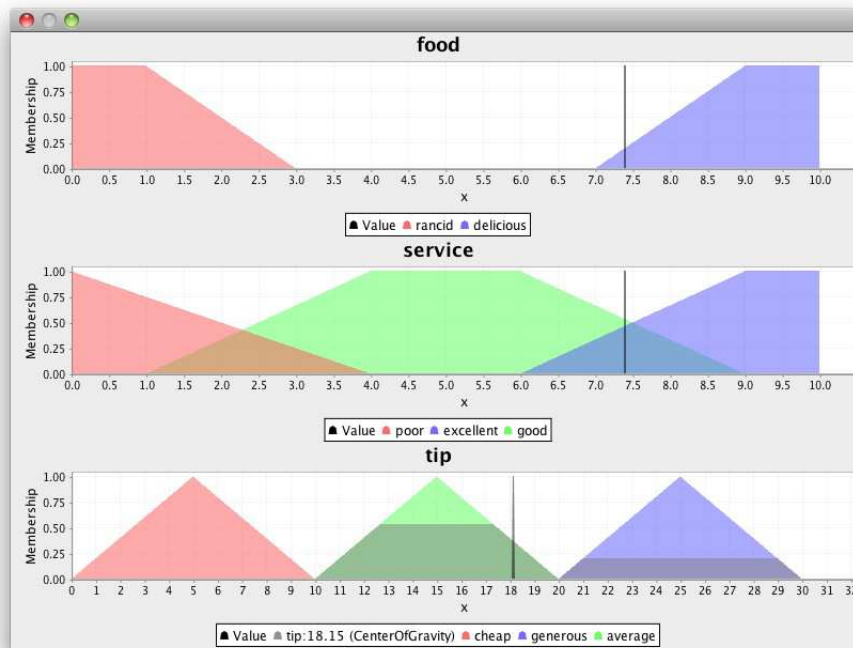


Figura 7 – Gráfico criado pela biblioteca JFuzzyLogic com 2 variáveis de entrada e 1 resultado.

Fonte: Cingolani e Alcalá-Fdez (2012)

```
service IS poor OR food IS rancid THEN tip IS cheap;"
```

A parte consequente é uma coleção da classe *RuleTerms*, por exemplo a parte "tip IS cheap". A parte antecedente é definida como uma classe *RuleExpression* que são dois termos que estão conectados por alguns destes conectores: *AND*, *OR* e *NOT*, por exemplo: service IS poor OR food IS rancid.

Cada *RuleTerm* é definido por uma variável declarada e um *LinguisticTermName* como por exemplo: "service IS poor" onde "service" é a variável e "poor" o *LinguisticTermName*, possuindo o conector *IS*, e também pode ser negada como "service IS NOT poor".

2.2.2 Métodos do bloco de regras

Segundo o autor Passino, Yurkovich e Reinfrank (1998) cada bloco de função possui regras que utilizam diferentes métodos como:

1. Agregação: utiliza as regras dos operadores *AND*, *OR* e *NOT* que satisfazem a lei de De Morgan.
2. Ativação: é definida pelo fato de como uma regra anterior modifica a regra seguinte,

por exemplo, se a parte IF da regra foi executada, de qual forma implicará na parte THEN da regra, usado para calcular o Mínimo ou Produto, como na Figura 8.

3. Acumulação: define como os consequentes de várias regras são combinadas dentro de um bloco de regras, como por exemplo se precisar calcular uma Soma, como na Figura 9.



Figura 8 – Método de ativação utilizada na JFuzzyLogic

Fonte: Cingolani e Alcala-Fdez (2012)

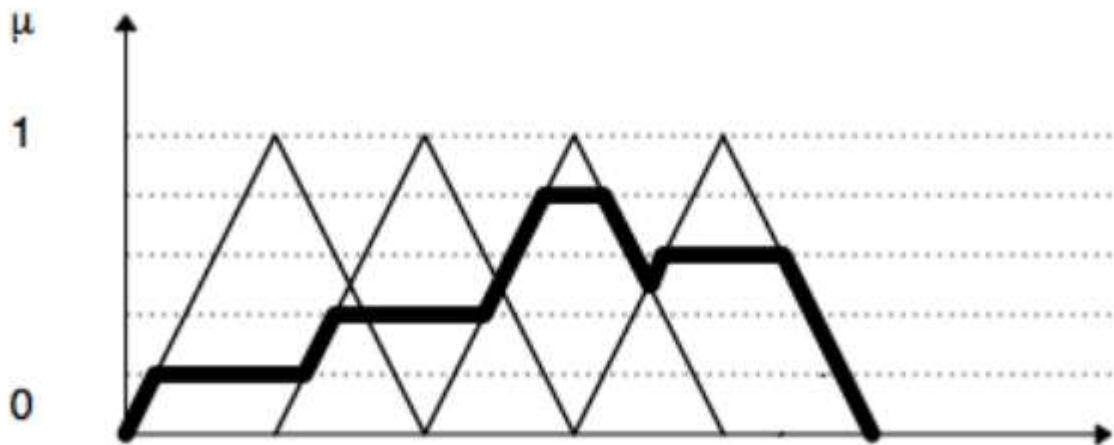


Figura 9 – Método de acumulação utilizada na JFuzzyLogic

Fonte: Cingolani e Alcala-Fdez (2012)

A biblioteca necessita das variáveis de entrada com seus valores e as regras que irão delimitar qual a consequência dada, dependendo do valor de entrada das variáveis. Calculado isso, então o programa retorna o valor da variável resposta. (CINGOLANI; ALCALA-FDEZ, 2012)

2.2.3 Representação das funções do JFuzzyLogic

A biblioteca possui várias formas na representação de funções separadas em contínuas e discretas, as mais importantes para este trabalho são as: *Singleton*, Triangular e

Trapezoidal, mas ainda possui algumas outras como: *Gauss*, e Sigmoide. (CINGOLANI; ALCALA-FDEZ, 2012)

A *Singleton* é a representação de um valor único para alguma variável. A Figura 10 mostra o gráfico da função. Para se declarar um tipo *Singleton* não é necessário o uso de uma *keyword* como as outras duas funções acima, então apenas escreve-se o valor correspondente, como por exemplo:

```
TERM poor := 2;
TERM good := 5;
TERM excellent := 8;
```

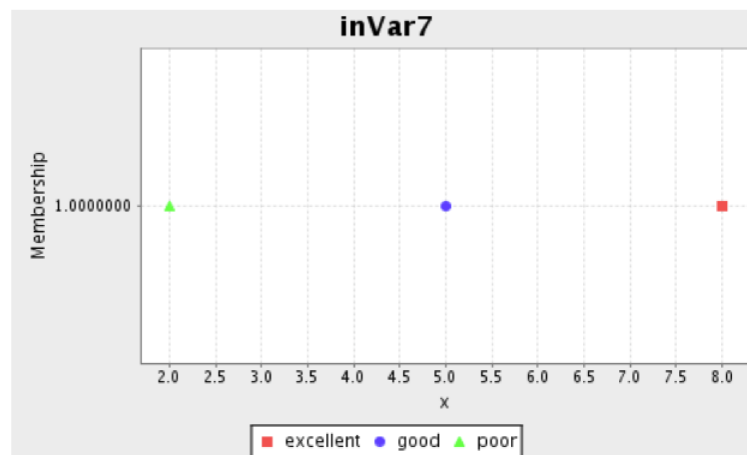


Figura 10 – Representação da função Singleton da biblioteca JFuzzyLogic

Fonte: Cingolani e Alcala-Fdez (2012)

A Triangular, como mostrada na Figura 11, é definida três valores para cada variável representando respectivamente o valor mínimo, médio e máximo, como por exemplo:

```
TERM poor := trian 0 2.5 5;
TERM good := trian 2.5 5 7.5;
TERM excellent := trian 5 7.5 10;
```

A Trapezoidal, como mostrada na Figura 12, é composta de quatro valores, representando o valor mínimo, médio-baixo, médio-alto e máximo, como por exemplo:

```
TERM poor := trape 0 2 3 4;
TERM good := trape 3 4 5 6;
TERM excellent := trape 5 6 7 8;
```

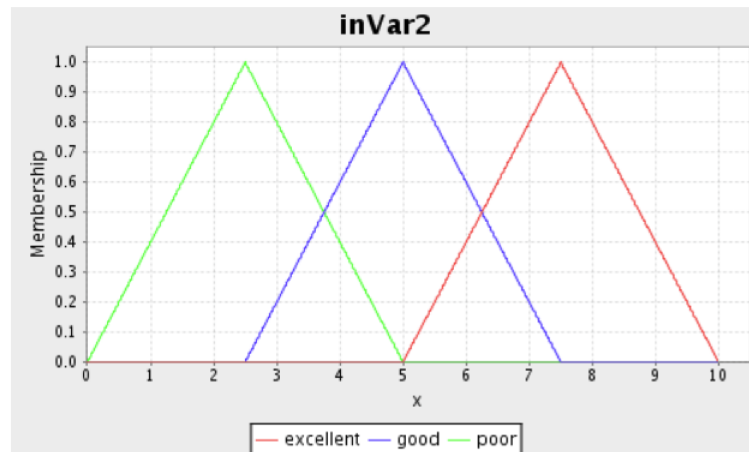


Figura 11 – Representação da função Triangular da biblioteca JFuzzyLogic

Fonte: [Cingolani e Alcala-Fdez \(2012\)](#)

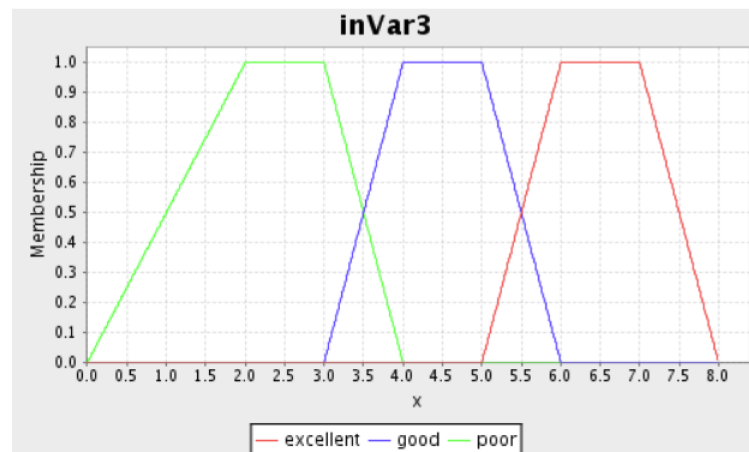


Figura 12 – Representação da função Trapezoidal da biblioteca JFuzzyLogic

Fonte: [Cingolani e Alcala-Fdez \(2012\)](#)

2.2.4 Elementos do Fuzzy Control Language

O arquivo FCL é composto por vários elementos que se unem para construir um arquivo de regras, que será interpretado pela biblioteca. A Figura 13 mostra como é composto o arquivo FCL e segundo o autor [Cingolani e Alcala-Fdez \(2012\)](#) os elementos são:

1. *FUNCTION BLOCK*: descreve o nome da função, engloba todo o arquivo FCL, delimitando o início e o fim do bloco.
2. *VAR INPUT*: descreve quais variáveis de entrada serão utilizadas para realizar o processo de Fuzzyficação. Só é implementado na versão utilizada variáveis tipos REAL.

```

FUNCTION_BLOCK functionName
VAR_INPUT
...
END_VAR

VAR_OUTPUT
...
END_VAR

FUZZIFY inputVariable
...
END_FUZZIFY

DEFUZZIFY outputVariable
...
END_DEFUZZIFY

RULEBLOCK No1
...
END_RULEBLOCK

END_FUNCTION_BLOCK

```

Figura 13 – Modelo de um Fuzzy Control Language

Fonte: [Cingolani e Alcala-Fdez \(2012\)](#)

3. *VAR OUTPUT*: contém a variável que terá o valor de saída do processo de defuzzyficação, também só é implementado variáveis do tipo REAL.
4. *FUZZIFY*: este bloco contém como cada variável em *VAR INPUT* é fuzzyficada, podendo ter vários termos (*LinguisticTerms*) e cada um desses termos possui uma função, transformando-os em instâncias de variáveis linguísticas. ([GOMIDE; GUDWIN, 1994](#)) A Figura 14 mostra como é organizado este bloco.

```

FUZZIFY food
  TERM rancid := (0, 1) (1, 1) (3,0) ;
  TERM delicious := (7,0) (9,1);
END_FUZZIFY

```

Figura 14 – Bloco de fuzzyficação

Fonte: [Cingolani e Alcala-Fdez \(2012\)](#)

5. *DEFUZZIFY*: É onde ficam armazenadas as variáveis que serão defuzzyficadas efetuando um escalonamento para retornar valores normalizados retornando um valor

real, e também é definido os termos linguísticos, tão como a função. (GOMIDE; GUDWIN, 1994) A Figura 15 mostra o exemplo de composição deste bloco.

```
DEFUZZIFY tip
  TERM cheap := (0,0) (5,1) (10,0);
  TERM average := (10,0) (15,1) (20,0);
  TERM generous := (20,0) (25,1) (30,0);
```

Figura 15 – Bloco de defuzzyficação

Fonte: Cingolani e Alcala-Fdez (2012)

6. *RULEBLOCK*: este bloco contém as regras que serão aplicadas no processo de fuzzyficação e defuzzyficação. Pode ser composto de vários blocos de regras. Primeiramente define-se alguns parâmetros como regras utilizadas, por exemplo : *DeMorgan's*, Método de Ativação, Método de Acumulação. Por fim é definido qual tipo de método será utilizado. A Figura 16 mostra como é composto esse bloco de regras.

```
RULE 1 : IF service IS poor OR food IS rancid THEN tip IS cheap;
RULE 2 : IF service IS good THEN tip IS average;
RULE 3 : IF service IS excellent AND food IS delicious THEN tip is generous;
END_RULEBLOCK
```

Figura 16 – Modelo de um bloco de regras.

Fonte: Cingolani e Alcala-Fdez (2012)

2.3 Trabalhos Relacionados

A aplicação da lógica Fuzzy é utilizada em várias áreas para previsão de tempo de espera, e esta seção, visa explorar os trabalhos relacionados à lógica Fuzzy e sistemas de controle de restaurantes.

No artigo "Lógica Fuzzy Aplicada ao Problema de Previsão do Tempo de Execução de Queries", Escovedo et al. (2012) propõe aplicar a Lógica Fuzzy para analisar o tempo de execução de uma *query* de uma ferramenta chamada SAP BW. O autor explica que não existe uma forma para calcular o tempo que será gasto para este cálculo, então o autor implementa um sistema de Lógica Fuzzy para realizar o cálculo a partir de certas variáveis.

O contexto da aplicação do sistema é em uma empresa brasileira no setor petrolífero que possui cerca de 8 mil usuários, executando mensalmente cerca de 150 mil *queries*.

A necessidade de prever o tempo de execução é que certas *queries* demoram consideravelmente para serem executadas uma vez que trabalham com um volume de informações muito grande. Não tendo uma forma de prever o tempo de execução, a Lógica Fuzzy permitirá decidir qual será o melhor tempo para executar aquela *query*.(ESCOVEDO et al., 2012)

Uma *query*, pode conter diversos filtros de entrada, linhas e colunas específicas que o usuário pode alterar, além de trabalhar com milhões de registros. O problema é que quanto mais usuários estiverem executando essas *queries* no servidor, maior será o tempo de execução de cada uma, e consequentemente o tempo de resposta será maior. Uma *query* executada em momentos distintos tende a ter resultados diferentes, dependendo do tempo de resposta do servidor, pois certos dados podem ser alterados se o servidor estiver recebendo muitas requisições, ou seja, a *query* irá demorar para ser executada e pode trazer resultados imprecisos.(ESCOVEDO et al., 2012)

Para criar esse Sistema Fuzzy, foi necessário saber quais variáveis utilizar para poder estabelecer o tempo previsto para execução, e com a experiência de especialistas na área, o autor conseguiu ver algumas variáveis que poderia alterar o tempo da execução da *query*, como: período do dia, volume de dados do cubo origem e volume de dados trazidos pelas *queries* como está mostrado na Figura 17. (ESCOVEDO et al., 2012)

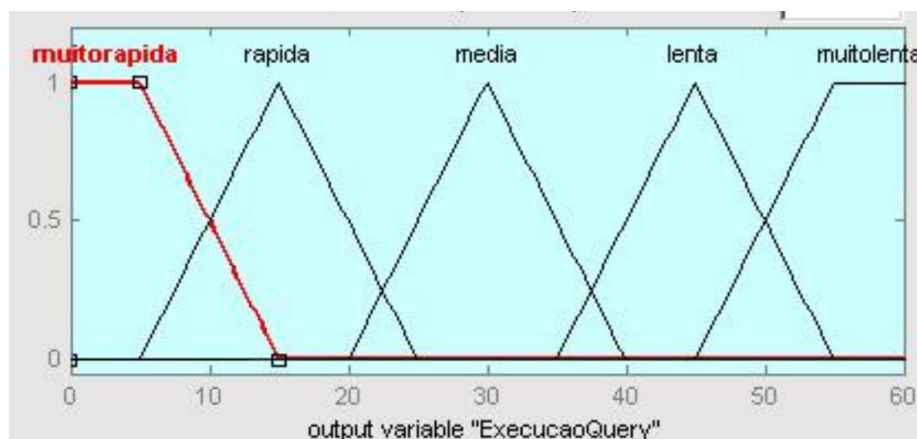


Figura 17 – Exemplo da saída do sistema Fuzzy aplicado na previsão da execução de *queries*

Fonte: Escovedo et al. (2012)

No artigo *Mobile Ticket Dispenser System with Waiting Time Prediction* os autores Lin e Lin (2014) criaram uma aplicação móvel que realiza pedidos num restaurante e mostra o tempo que o cliente terá que esperar até o pedido ficar pronto e poder pegá-lo. O cliente entra no aplicativo e marca seu lugar na fila e realiza seu pedido. A regra do sistema estabelece que o primeiro a pedir, é o primeiro a ser atendido.

O cliente consegue realizar o seu pedido sem estar no restaurante e baseado no

tempo de preparação da refeição, o cliente consegue saber quando ele vai ter que se deslocar para buscar seu pedido.(LIN; LIN, 2014)

A cada atualização do pedido o cliente recebe no próprio smartphone o tempo de espera atualizado, caso haja alguma mudança. Isso serve para periodicamente melhorar a precisão do tempo estimado. O problema da fila de espera e da boa experiência do cliente em ter que esperar é resolvido usando aplicações móveis e wireless, permitindo que mesmo não estando no restaurante ele consiga ter uma estimativa de tempo.(LIN; LIN, 2014)

Um dos problemas que ainda pode acontecer é o cliente chegar atrasado ou o sistema marcar um tempo maior do que o necessário para a comida ficar pronta, fazendo assim o pedido não estar mais na temperatura ideal. Então, é muito importante que o tempo seja o mais preciso possível.(LIN; LIN, 2014)

3 Desenvolvimento

O capítulo 3 apresenta a modelagem do software desenvolvido para o cálculo e apresentação do tempo de espera em filas de restaurante proposto neste trabalho e detalha cada etapa que o compõe. Para o servidor foi utilizado a linguagem Java junto com a biblioteca JFuzzyLogic, e a aplicação cliente foi desenvolvida utilizando as linguagens PHP, JavaScript e HTML. As duas aplicações se comunicam através do protocolo SOAP no qual o cliente envia os valores das variáveis relacionadas ao cálculo do tempo e o servidor retorna o tempo de espera estimado utilizando a Lógica Fuzzy.

3.1 Arquitetura do Sistema

Esta seção apresenta uma visão geral da arquitetura do software proposto neste trabalho, para o cálculo e previsão de tempo em fila de espera em restaurantes. A Figura 18 esquematiza a arquitetura do sistema.

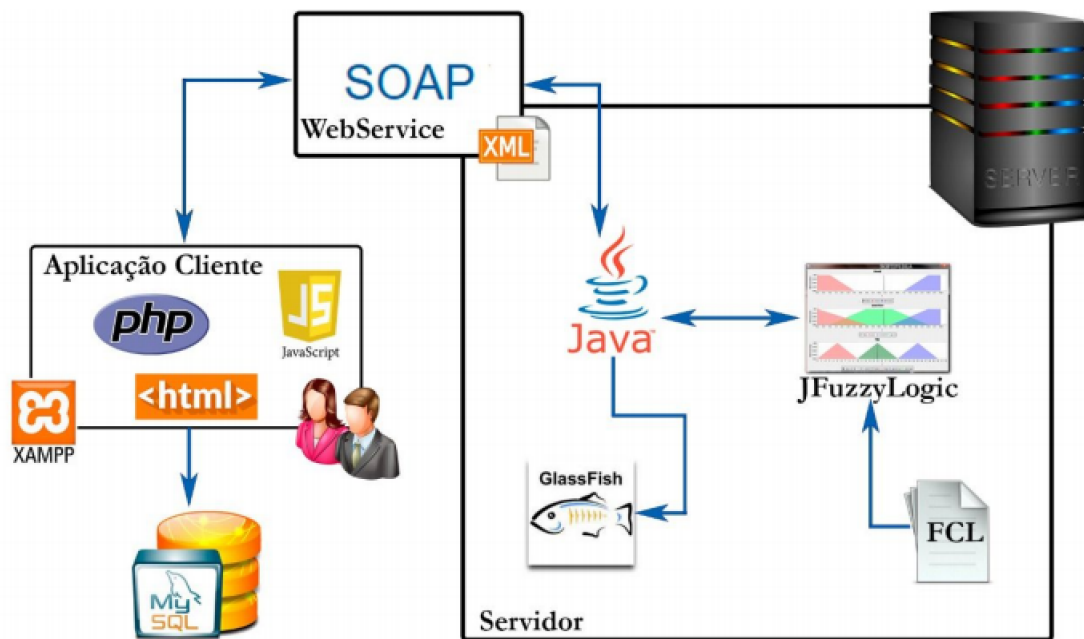


Figura 18 – Diagrama de Arquitetura do Sistema

Fonte: Do autor

A aplicação cliente foi desenvolvida utilizando as linguagens PHP, HTML e JavaScript. Para fins de testes, a aplicação foi implantada em um servidor XAMPP, que conta com a máquina virtual PHP e o servidor de Banco de Dados MySQL. No servidor de banco de dados foi armazenado todas as informações essenciais do sistema, como os

dados das mesas cadastradas, funcionários e o tempo que cada cliente gastará em cada mesa. A aplicação é dividida em dois sistemas, um gerenciador e um sistema de usuário:

1. Aplicação Gerenciadora utilizado pelos funcionários do restaurante, controla todo o sistema do restaurante como as mesas, funcionários e variáveis de entrada para o Sistema Fuzzy.
2. Aplicação para Cliente, exibe o tempo de espera do cliente até ser atendido no restaurante.

A aplicação servidora foi implantada no servidor *GlassFish Server*, como um Web-Service desenvolvido em Java. O desenvolvimento do servidor conta com a biblioteca JFuzzyLogic, que possui todas os métodos que serão utilizados no processo do cálculo dos valores relacionados a Lógica Fuzzy, e acessa o arquivo de regras em FCL para pegar os dados relacionados ao Sistema Fuzzy, como as regras e os valores das variáveis que serão utilizadas.

A comunicação entre o servidor e o cliente é realizada através do protocolo de comunicação SOAP, permitindo que o cliente envie informações das variáveis dadas pelos funcionários e o servidor retorne o valor após ter realizado todo o cálculo necessário, essa comunicação é feita através do envio e recebimento de arquivos XML.

3.2 Modelagem do Sistema

Esta seção mostra os requisitos funcionais do sistema, que são necessários para o funcionamento completo, além de mostrar o diagrama de atividades e o diagrama de caso de uso.

3.2.1 Análise de Requisitos

Os requisitos funcionais detalham o que o sistema pode fazer, ou seja, o que ele deve possuir. A Tabela 1 mostra todos os requisitos funcionais implementados no sistema.

3.2.2 Diagrama de Atividades

Os diagramas de atividades mostram o comportamento do sistema, ilustrando graficamente o funcionamento do software e seu fluxo.

A Figura 19 ilustra o diagrama de atividades que englobam o sistema, onde o usuário tem acesso a todas as funções do sistema depois de efetuar o login. A única função que não tem restrição de acesso é a de listagem do tempo das mesas pois a funcionalidade

Tabela 1 – Requisitos Funcionais do Sistema

Nº	Nome	Descrição
RF01	Login	Permite o usuário logar no sistema, de maneira a ter acesso às funcionalidades.
RF02	Gerenciamento de Funcionários	Permite o usuário cadastrar funcionários para acesso ao sistema.
RF03	Gerenciamento de Mesas	Permite o usuário cadastrar mesas.
RF04	Listagem de Funcionários	Permite o usuário listar todos os funcionários e excluir.
RF05	Listagem de Mesas	Permite o usuário listar todas as mesas e excluir.
RF06	Ocupar Mesa	Permite o usuário ocupar uma mesa, colocando como parâmetros a quantidade de pessoas sentadas, e se estão bebendo.
RF07	Desocupar Mesa	Permite o usuário colocar uma mesa como desocupada.
RF08	Atualizar Evento Diário	Permite o usuário atualizar o evento diário, que é a variável que implica no cálculo do tempo de todas as mesas do restaurante
RF09	Calcular Tempo das Mesas	Permite o usuário a chamar o Webservice passando as variáveis de cada mesa salvando no banco de dados o tempo estimado de cada mesa.
RF10	Listar Tempo das Mesas	Pega as informações do banco de dados referente ao tempo das mesas e lista o tempo restante para cada mesa ficar disponível.

destinada aos clientes não precisam ter login, na qual só é exibido informações a respeito do tempo das mesas.

3.2.3 Diagrama de Caso de Uso

Os diagramas de caso de uso descrevem os requisitos que devem ser atendidos pelo sistema mostrando quais atores têm acesso a qual função específica no sistema.

A Figura 20 ilustra os casos de uso do sistema, que contam com dois atores que utilizarão o programa, o funcionário e o cliente. O funcionário têm acesso a todas as funções do sistema, incluindo o cadastro de mesas e funcionários, gerenciamento das mesas, atualização de eventos diários, listagem das mesas e funcionários, cálculo do tempo das mesas e também a listagem do tempo de espera das mesas. Para o cliente é só importante a visualização das mesas com o tempo restante para serem desocupadas, então é um acesso irrestrito, não precisando de estar logado no sistema para visualizar essa função.

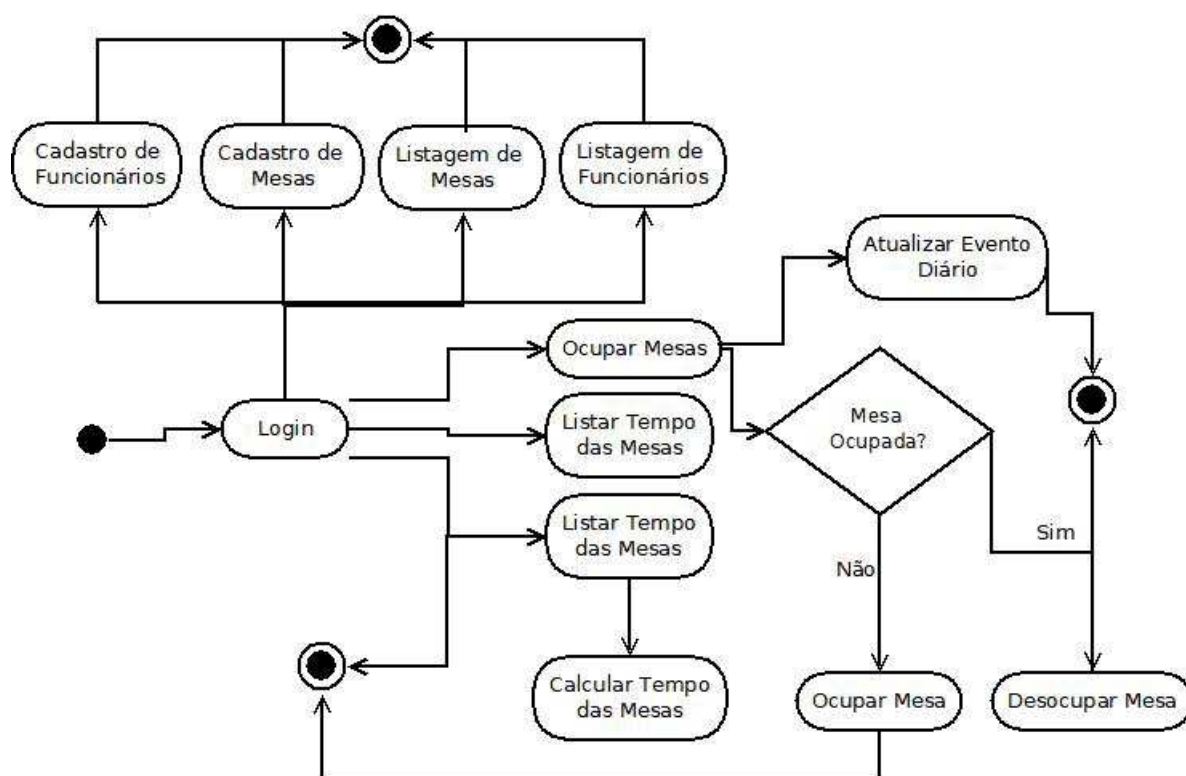


Figura 19 – Diagrama de Atividades do Projeto

Fonte: Do autor

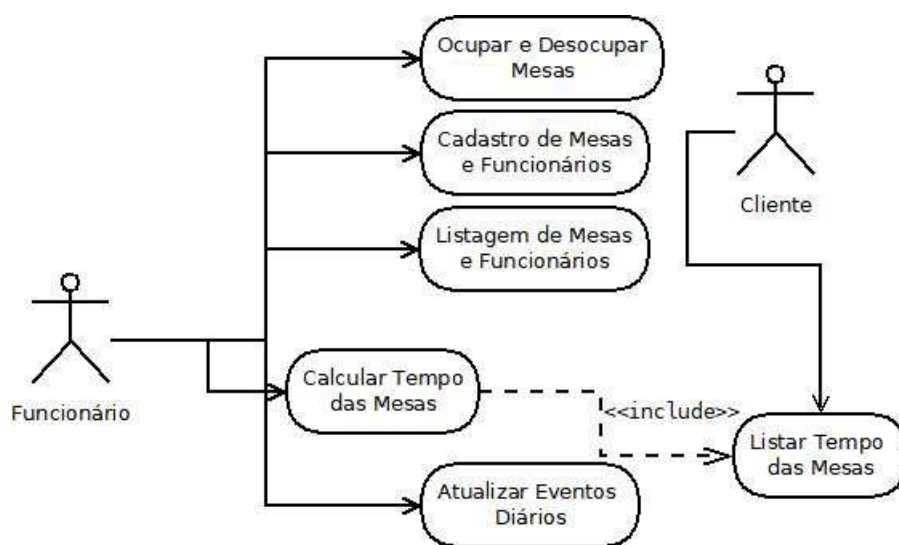


Figura 20 – Diagrama de Atividades do Projeto

Fonte: Do autor

3.3 Banco de Dados

O banco de dados foi criado para o funcionamento mínimo do sistema, apenas para armazenamento das mesas, funcionários informações relacionadas às variáveis de cada mesa ocupada e o tempo de espera de cada cliente . A Figura 21 mostra o modelo entidade-relacionamento do banco de dados, que é composto das Tabelas 2, 3, 4, 5 e 6

Tabela 2 – Dicionário de dados da entidade eventos diários

Chave	Atributo	Tipo	Descrição
PK	id_evento	int	Identificador único de um evento
	com_chuva	int	Armazena o valor se está chovendo ou não, onde valores alternam entre 0 e 1
	com_show	int	Armazena o valor se está havendo show ao vivo ou não, onde valores alternam entre 0 e 1

Tabela 3 – Dicionário de dados da entidade mesas

Chave	Atributo	Tipo	Descrição
PK	id	int	Identificador único de uma mesa
	numeromesa	int	Armazena o número respectivo da mesa cadastrada
	capacidade	int	Número máximo de pessoas que é comportado na mesa
	ocupada	varchar	Mostra se a mesa está ocupada ou não.

Tabela 4 – Dicionário de dados da entidade usuário

Chave	Atributo	Tipo	Descrição
PK	id	int	Identificador único de um usuário
	nome	varchar	Nome do usuário
	senha	varchar	Senha para login do usuário
	cargo	varchar	Cargo do usuário no restaurante
	cpf	varchar	Número do CPF do usuário

Tabela 5 – Dicionário de dados da entidade tempomesa

Chave	Atributo	Tipo	Descrição
PK	id	int	Identificador único de um tempomesa
	quantidadedepessoas	int	Quantidade de pessoas sentadas na mesa
	combebida	int	Representa se as pessoas pediram bebida ou não
	com chuva	int	Valor pegado da tabela eventos_diarios
	com show	int	Valor pegado da tabela eventos_diarios
FK	idmesa	int	Chave estrangeira da coluda id da tabela mesas

Uma *procedure* e uma *trigger* controlam do tempo passado em uma mesa, fazendo que o tempo armazenado na coluna **tempo** da tabela **tempocliente** seja decrementado em 1 a cada minuto passado, quando esse tempo chega a 0, o status da mesa é trocado de Ocupado para Liberando. A Figura 22 mostra o corpo da rotina dessa *procedure*.

Tabela 6 – Dicionário de dados da entidade tempocliente

Chave	Atributo	Tipo	Descrição
PK	id	int	Identificador único de um tempocliente
	tempo	int	Valor retornado pela biblioteca JFuzzyLogic do tempo restante para a mesa ser liberada
FK	idmesa	int	Chave estrangeira da coluda id da tabela mesas
	status	varchar	Alterna entre valores Ocupado e Liberando dependendo da situação da mesa.

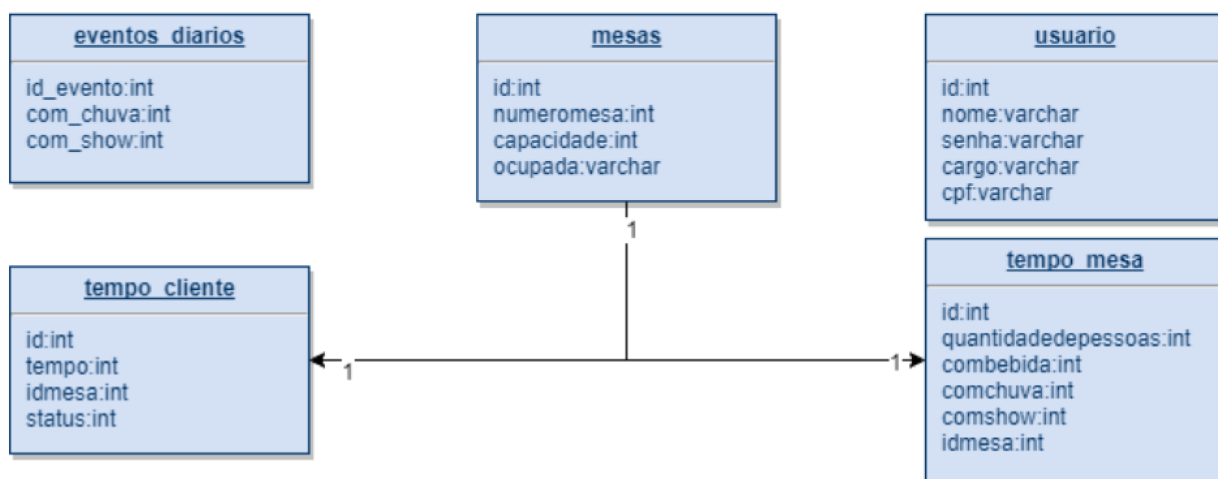


Figura 21 – Modelo do banco de dados

Fonte: Do autor

```

BEGIN
    UPDATE tempo_cliente set tempo = tempo - 1
    WHERE status = 'Ocupada'

    UPDATE tempo_cliente set status = 'Liberando'
    WHERE tempo <= 0;
END

```

Figura 22 – Rotina da *procedure*

Fonte: Do autor

3.4 Codificação

Esta seção fala sobre o código desenvolvido neste trabalho, incluindo o desenvolvimento da aplicação cliente utilizando PHP, HTML e JavaScript, a criação do arquivo FCL que é necessário para o Sistema Fuzzy, e também o desenvolvimento do servidor e mostrando como funciona a comunicação entre as aplicações e a comunicação com o cliente.

3.4.1 Criação do arquivo FCL

O arquivo FCL contém todos os dados necessários para realizar o processo de fuzzyficação e defuzzyficação. Para este trabalho foram utilizadas 4 variáveis de entrada diferentes, sendo 3 do tipo *Singleton* e uma do tipo *Trape*. A Figura 23 mostra a declaração dessas variáveis de entradas.

No bloco `VAR_INPUT` são declaradas das quatro variáveis, do tipo `REAL`, em seguida no `VAR_OUTPUT` declara a variável `tempo` que conterá o resultado do processo de defuzzyficação.

Para as variáveis de entrada tem-se:

1. `bebida`: *Singleton* alternando entre 1 se não há bebida na mesa e 2 se pediram bebida.
2. `chuva`: *Singleton* possui o valor 1 se não está chovendo e 2 se está chovendo.
3. `qnt`: são compostos por figuras geométricas em 6 intensidades para representar a quantidade de pessoas no cálculo. A Figura 24 exemplifica como fica o gráfico no exemplo de 7 pessoas, composto pelo triângulo formado por (6,0) (7,1) (8,0) com valor de intensidade 0 no valor 6, intensidade 1 no valor 7 e intensidade 0 no valor 8. O limite colocado para quantidade de pessoas nesse trabalho é 10. Os valores dessa variável foram definidos empiricamente.
4. `show`: *Singleton* com o valor 1 quando não há show ao vivo e 2 se está ocorrendo show no momento.

As variáveis *Singleton* declaradas, possuem o valor 1 em um ponto, e 0 em todos os outros pontos, como mostra a Figura 25 que exemplifica o valor da variável `bebida` valendo 1 em intensidade 2 e 0 para os pontos restantes.

Para a parte de defuzzyficação, a variável `tempo` é declarada no bloco `DEFUZZIFY` com seus valores. A Figura 26 mostra a declaração desse bloco. Os valores para a criação desse bloco foram extraídos e adaptados do questionário realizado no curso de Sistemas de Informação da Universidade Federal de Uberlândia, com 27 respostas, que se encontra no

```
FUNCTION_BLOCK calc_tempo

VAR_INPUT

    bebida : REAL;
    chuva  : REAL;
    qnt     : REAL;
    show    : REAL;

END_VAR

VAR_OUTPUT
    tempo : REAL;
END_VAR

FUZZIFY show

    TERM nao := 1;
    TERM sim := 2;

END_FUZZIFY

FUZZIFY bebida

    TERM nao := 1;
    TERM sim := 2;

END_FUZZIFY

FUZZIFY chuva

    TERM sem_chuva := 1;
    TERM com_chuva := 2;

END_FUZZIFY

FUZZIFY qnt

    TERM int1 := (0, 1) (1, 1) (2, 0);
    TERM int2 := (1, 0) (2, 1) (3, 0);
    TERM int3 := (2, 0) (4, 1) (5, 0);
    TERM int4 := (4, 0) (6, 1) (7, 0);
    TERM int5 := (6, 0) (7, 1) (8, 0);
    TERM int6 := (7, 0) (11, 1);

END_FUZZIFY
```

Figura 23 – Declaração das variáveis utilizadas no arquivo FCL

Fonte: Do autor

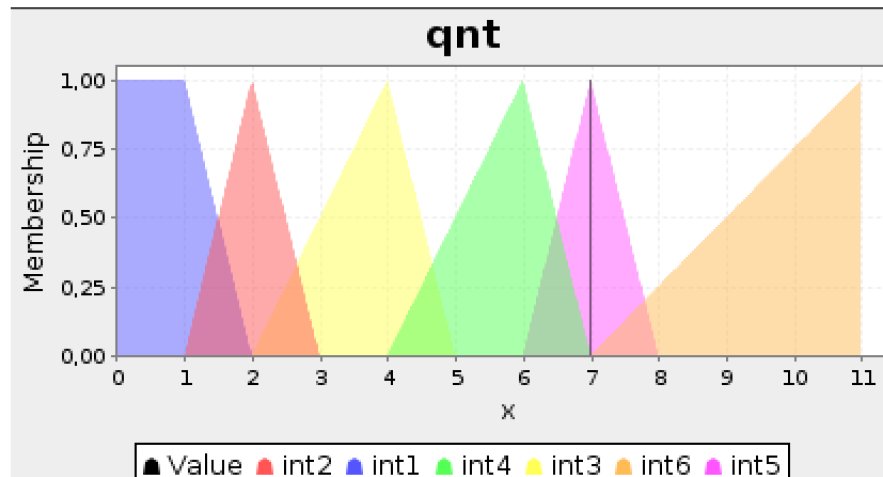


Figura 24 – Gráfico representando 7 pessoas na biblioteca JFuzzyLogic

Fonte: Do autor

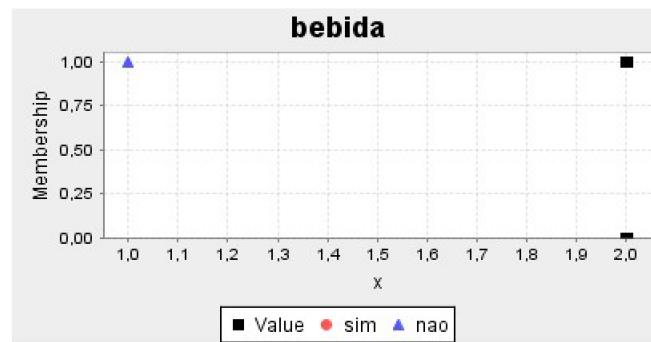


Figura 25 – Gráfico representando a variável *Singleton* bebida com o valor 2

Fonte: Do autor

Apêndice deste trabalho. Para os valores de `tint1` e `tint2` foi criado uma regra específica pois é um caso especial para 1 ou 2 pessoas. Para a criação das outras variáveis foi incrementado o valor de cada `trape` em 15 para se adaptar ao questionário.

No bloco `DEFUZZIFY` foram declarados 12 blocos de intensidade, definidos por uma `Trape`, possuindo 4 valores cada uma, representando o valor mínimo, médio-mínimo, médio-máximo e máximo. O método utilizado para defuzzificação é o `COG` (*Center of Gravity*) que é equivalente ao Centroide. Esses valores foram definidos para aumentar o valor do tempo conforme a regra dos Bloco de Regras. O valor `Default` foi colocado em 0 para caso não ative nenhuma regra.

O Bloco de Regras foi criado baseado no questionário realizado que está no Apêndice deste trabalho e é composto por 48 regras, onde se encaixa em todas as probabilidades, exemplificado pela Figura 27. Por exemplo, a regra: `if show is nao and bebida is nao and chuva is sem_chuva and qnt is int4 then tempo is tint4`; mostra que se o valor da variável `show` é 1, `bebida` é 1, `sem_chuva` é 1 e a quantidade de pessoas

```

DEFUZZIFY tempo
  TERM tint1 := trape 20 25 30 35;
  TERM tint2 := trape 50 55 60 65;
  TERM tint3 := trape 45 55 60 65;
  TERM tint4 := trape 65 70 75 80;
  TERM tint5 := trape 80 85 90 95;
  TERM tint6 := trape 95 100 105 110;
  TERM tint7 := trape 110 115 120 125;
  TERM tint8 := trape 125 130 135 140;
  TERM tint9 := trape 140 150 155 160;
  TERM tint10:= trape 160 165 170 175;
  TERM tint11:= trape 175 180 190 200;
  TERM tint12:= trape 200 210 215 220;
  METHOD : COG;
  DEFAULT := 0;
END_DEFUZZIFY

```

Figura 26 – Bloco de defuzzyficação da variável tempo.

Fonte: Do autor

entra na regra da `int4` então o valor da defuzzyficação da variável `tempo` será aplicado pelo termo `tint4` que equivale a `trape 65 70 75 80` em que o valor resultante será entre o intervalo de 65 a 80 minutos para desocupar a mesa. O restante das regras utilizadas estão definidas no Apêndice deste trabalho.

O número de regras foi baseado na quantidade de valores das variáveis de entrada, no qual foi multiplicado a quantidade de valores de cada variável, como:

$$2.2.2.6 = 48 \quad (3.1)$$

Então a partir desse número de regras, temos uma combinação de todas as regras de forma que se encaixe em todas as possibilidades. Os valores definidos na parte `then` de cada regra foram criados com base no questionário.

```

RULE 1: if show is nao and bebida is nao and chuva is sem_chuva and qnt is int1 then tempo is tint1;
RULE 2: if show is nao and bebida is sim and chuva is sem_chuva and qnt is int1 then tempo is tint3;
RULE 3: if show is nao and bebida is nao and chuva is com_chuva and qnt is int1 then tempo is tint2;
RULE 4: if show is nao and bebida is sim and chuva is com_chuva and qnt is int1 then tempo is tint4;
RULE 5: if show is sim and bebida is nao and chuva is sem_chuva and qnt is int1 then tempo is tint3;
RULE 6: if show is sim and bebida is sim and chuva is sem_chuva and qnt is int1 then tempo is tint5;

```

Figura 27 – Exemplo das regras criadas no Bloco de Regras

Fonte: Do autor

3.4.2 Criação do Servidor Fuzzy

Para garantir a perfeita comunicação entre as aplicações cliente e o servidor Java, rodando o Sistema Fuzzy com a biblioteca JFuzzyLogic, foi utilizada a tecnologia do Webservice. O servidor foi desenvolvido em Java e o cliente como aplicação Web PHP.

```
@WebService(endpointInterface = "com.lincoln.jfuzzy.FuzzyInterface")
public class Fuzzy implements FuzzyInterface{
    @Override
    public double getTime (int bebida, int chuva, int qnt, int show){
        final String filename = "/home/lincoln/dados2.fcl";
        OpenFile open = new OpenFile();
        FIS fis = open.open(filename);
        Variaveis v = new Variaveis();
        v.setBebida(bebida);
        v.setChuva (chuva);
        v.setQnt (qnt);
        v.setShow (show);
        fis.setVariable("bebida", v.getBebida());
        fis.setVariable("chuva", v.getChuva());
        fis.setVariable("qnt", v.getQnt());
        fis.setVariable("show", v.getShow());
        fis.evaluate();
        fis.getVariable("tempo").chartDefuzzifier(true);
        return fis.getVariable("tempo").defuzzify();
    }
}
```

Figura 28 – Código para execução da biblioteca JFuzzyLogic

Fonte: Do autor

No código da Figura 28 vemos que é necessário o arquivo FCL que contém todas as regras e variáveis para execução do Sistema Fuzzy. O arquivo FCL é carregado no Sistema Fuzzy através do comando `FIS.load(path)` onde o `path` é a localização do arquivo FCL no computador.

São colocados valores para as variáveis através dos métodos `fis.setVariable()` que pega o valor da variável bebida, chuva, e quantidade de pessoas. Para gerar o gráfico de cada variável inserida no `FUNCTION_BLOCK` do arquivo FCL com os valores passados no código, é necessário o uso do método `fis.chart()`, depois disso é necessário o uso do método `fis.evaluate()`, que de fato vai realizar todo o cálculo da Lógica Fuzzy.

O método `fis.getVariable("tempo").defuzzify()` exibe no console do servidor qual o valor obtido no processo de defuzzificação, no caso o valor da variável tempo.

Logo após o teste das variáveis e da biblioteca, vem a implementação do código para que o sistema em Java se torne um *Webservice*. Primeiramente no arquivo `main.java`

precisa-se de um Endpoint que é a URL do WebService que vai ser acessado pelo cliente, a declaração se dá por:

```
Endpoint.publish("http://127.0.0.1:9090/fuzzy", new Fuzzy());
```

Para o uso do protocolo SOAP foi necessário usar uma interface que vai conter todos os métodos que poderão ser invocados pelo cliente que quiser acessar o *WebService*, então foi criado a interface *FuzzyInterface* que contém apenas o método público `getTime()` que recebe os valores das variáveis bebida, chuva e quantidade de pessoas. A Figura 29 mostra o conteúdo composto pela interface.

```
@WebService
@SOAPBinding(style=Style.RPC)
public interface FuzzyInterface {

    @WebMethod public double getTime (int bebida,
                                     int chuva,
                                     int qnt,
                                     int show);
}
```

Figura 29 – Código da interface do WebService em Java

Fonte: Do autor

Nessa interface é declarado o estilo da comunicação, que é o RPC, marca a interface como um *WebService*, e o método declarado é definido como um *WebMethod*.

Tendo todas essas classes definidas, o servidor estará pronto para implantar e publicar, podendo ser acessado localmente por clientes através do endereço definido.

3.4.3 Criação da aplicação cliente

A aplicação cliente foi desenvolvida utilizando as linguagens PHP e JavaScript. Essa parte do programa possui toda a interface gráfica do sistema, tanto para o lado desenvolvido para o cliente que chega no restaurante, tanto para os funcionários que irão manipular o sistema para realizar cadastros e ocupação das mesas. A Figura 30 mostra o menu da aplicação com suas funcionalidades.

Na área de Cadastros, está a opção de inserir novas mesas e funcionários. A Figura 31 mostra como está criado a parte de cadastro de funcionários.

Em listagens, são mostradas as mesas e funcionários, na qual o usuário tem a opção de ver as informações relacionadas a mesas e funcionários. A Figura 32 mostra como está organizado a tela da listagem das mesas.

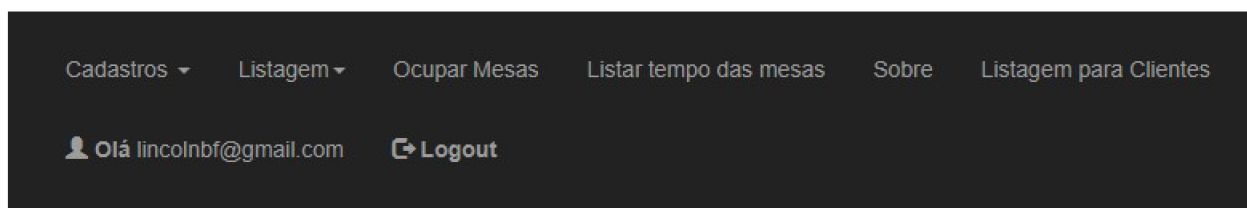


Figura 30 – Menu da aplicação cliente.

Fonte: Do autor

The form is titled 'Cadastro de Funcionários' in a large, bold font. Below the title is a section header 'Funcionários'. The form contains four input fields, each with a label above it: 'Nome do Funcionário', 'CPF do Funcionário', 'Cargo do Funcionário', and 'Senha do Funcionário'. Each input field has a light gray placeholder text matching its label. At the bottom of the form is a green button with the text 'Cadastrar Funcionário' in white.

Figura 31 – Cadastro de funcionários da aplicação cliente.

Fonte: Do autor

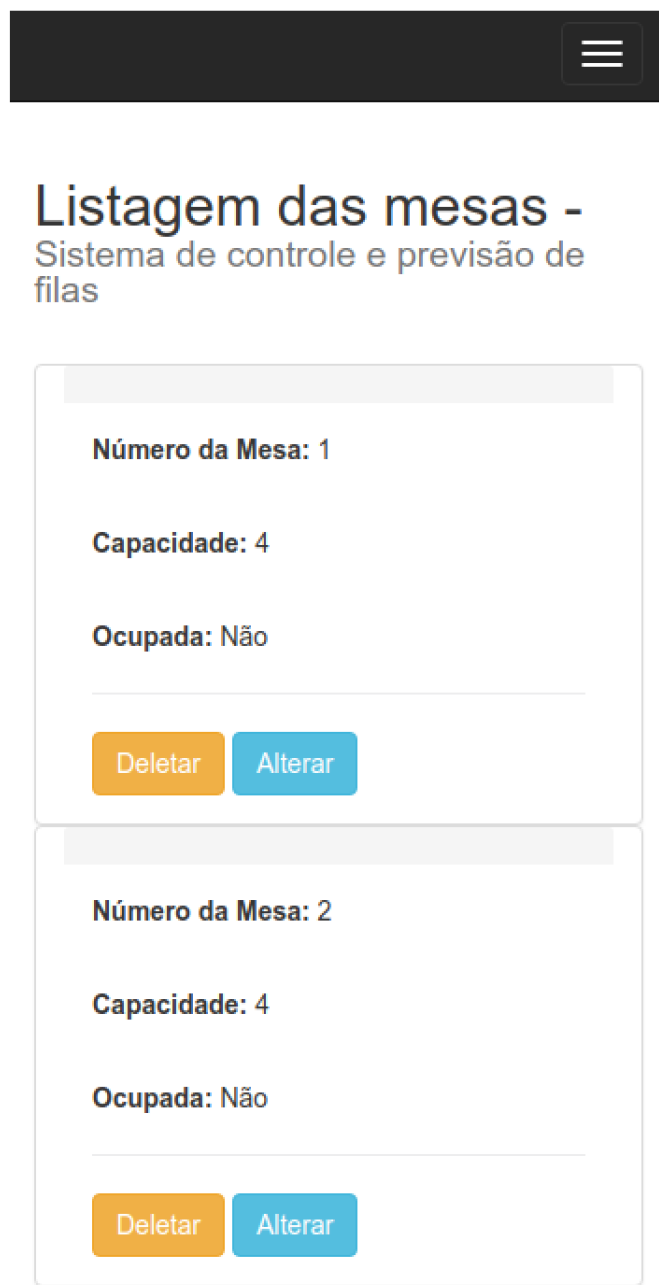


Figura 32 – Listagem das mesas da aplicação cliente.

Fonte: Do autor

Em ocupar mesas, está a opção de inserir o evento diário como chuva e show ao vivo, e também ocupar e desocupar uma mesa, como mostra a Figura 33.

Ocupar Mesas - Sistema de controle e previsão de filas

Evento diário

Com chuva? ☐ Não ☒ Sim

Com show ao vivo? ☐ Não ☒ Sim

Atualizar Evento Diário

Mesa 1	Mesa 2
Número da Mesa: 1	Número da Mesa: 2
Capacidade: 4	Capacidade: 4
Ocupada: Não	Ocupada: Sim
Quantidade de pessoas: 1	Desocupar Mesa
Com bebida? ☒ Não ☐ Sim	
Ocupar Mesa	

Figura 33 – Tela de ocupar mesas da aplicação cliente.

Fonte: Do autor

A ultima parte, mais importante do trabalho, é a listagem, que fica na aplicação do funcionário e na do cliente. Essa tela mostra o número da mesa, o status dela, e se ela está ocupada ou liberando. A Figura 34 mostra como será a listagem dessas mesas.

Na comunicação da aplicação cliente com o servidor Java, classe no cliente `GerarTempo.php` requisita o WebService através do código:

```
$client = new SoapClient('http://127.0.0.1:9000/fuzzy?wsdl');
```

Para chamar o SOAP, é passado 3 variáveis: a função do servidor que deverá ser chamada, os argumentos que serão passados para o servidor, e a localização do servidor. A função é chamada por: `$function = "getTime";`. Os argumentos são definidos por: `$arguments = array($bebida, $chuva, $qnt, $show);`. A localização do servidor é: `$options = array('location' => 'http://127.0.0.1:9000/fuzzy');`.



Mesa: 1 Tempo estimado para ficar livre: 33 minutos. Status da mesa: ocupada
Mesa: 2 Tempo estimado para ficar livre: 48 minutos. Status da mesa: ocupada
Mesa: 3 Tempo estimado para ficar livre: 48 minutos. Status da mesa: ocupada
Mesa: 4 Tempo estimado para ficar livre: 38 minutos. Status da mesa: ocupada

Figura 34 – Listagem das mesas para o cliente.

Fonte: Do autor

Com essas variáveis preenchidas é realizada uma chamada SOAP, essa função tem como retorno o valor calculado pelo servidor em relação as variáveis passadas a respeito das mesas, que logo após é inserido no banco de dados para visualização do cliente, através da função:

```
$client->__soapCall($function, $arguments, $options);
```

4 Conclusão

Este trabalho aborda o problema da estimativa do tempo de espera por mesas em restaurantes. Para isso foi utilizado a Lógica Fuzzy em um Webservice utilizando a biblioteca JFuzzyLogic para calcular a estimativa do tempo que uma pessoa demora dentro do restaurantes através de algumas variáveis. Para controlar isso foi utilizado uma aplicação cliente que repassa os dados dessas variáveis para o servidor e recebe de volta o tempo estimado.

4.1 Considerações Finais

A Lógica Fuzzy é boa para estimativas, pois difere da regra tradicional onde valores podem assumir somente 0 ou 1, então para este trabalho a Lógica Fuzzy foi uma boa escolha para tentar resolver o problema do tempo de espera em filas, uma vez que o problema tem a complexidade da análise de muitas variáveis que podem mudar de acordo com o restaurante ao qual é feita a análise. Para este trabalho foram levantadas algumas das variáveis que influenciam as pessoas a permanecer no restaurante, e os valores dessas variáveis foram estimadas a partir de um questionário realizado com algumas pessoas com o intuito de testar se o sistema poderia ser aplicado em uma situação real.

4.2 Trabalhos Futuros

Sugere-se como trabalho futuro modificar o sistema Fuzzy para que consiga ser auto-alimentável de forma que os usuários possam adicionar novas regras e modificar seus valores caso haja uma mudança dos clientes do restaurante, ou utilizar Redes Neurais Auto-Organizáveis para analisar a precisão e comparar com a Lógica Fuzzy.

Referências

- CINGOLANI, P.; ALCALA-FDEZ, J. jfuzzylogic: a robust and flexible fuzzy-logic inference system language implementation. In: IEEE. *Fuzzy Systems (FUZZ-IEEE)*, 2012 IEEE International Conference on. [S.l.], 2012. p. 1–8. Citado 8 vezes nas páginas 16, 20, 21, 22, 23, 24, 25 e 26.
- COPPIN, B. *Artificial intelligence illuminated*. [S.l.]: Jones & Bartlett Learning, 2004. Citado 4 vezes nas páginas 15, 16, 17 e 20.
- ESCOVEDO, T. et al. Lógica fuzzy aplicada ao problema de previsao do tempo de execucao de queries. 2012. Citado 2 vezes nas páginas 26 e 27.
- GOMIDE, F. A. C.; GUDWIN, R. R. Modelagem, controle, sistemas e lógica fuzzy. *SBA Controle & Automação*, v. 4, n. 3, p. 97–115, 1994. Citado 3 vezes nas páginas 16, 25 e 26.
- LEE, C.-C. Fuzzy logic in control systems: fuzzy logic controller. i. *IEEE Transactions on systems, man, and cybernetics*, IEEE, v. 20, n. 2, p. 404–418, 1990. Citado na página 17.
- LIN, Y.-W.; LIN, Y.-B. Mobile ticket dispenser system with waiting time prediction. In: IEEE. *Network Operations and Management Symposium (APNOMS)*, 2014 16th Asia-Pacific. [S.l.], 2014. p. 1–6. Citado 2 vezes nas páginas 27 e 28.
- MEDEIROS, S. F. de. Estudo de lógica fuzzy aplicada na classificação de águas industriais e sintonia pid. 2011. Citado na página 19.
- MENDEL, J. M. Fuzzy logic systems for engineering: a tutorial. *Proceedings of the IEEE*, IEEE, v. 83, n. 3, p. 345–377, 1995. Citado na página 20.
- MENDELSON, E. Introduction. In: *Introduction to Mathematical Logic*. [S.l.]: Springer, 1987. p. 1–9. Citado na página 15.
- PASSINO, K. M.; YURKOVICH, S.; REINFRANK, M. *Fuzzy control*. [S.l.]: Addison-wesley Reading, MA, 1998. v. 20. Citado 2 vezes nas páginas 20 e 21.
- PILAN, F. C. Aproximações entre o pragmatismo de john dewey e o desenvolvimento das lógicas não-clássicas. *Intuitio*, v. 4, n. 1, p. 51–60, 2011. Citado na página 15.
- SIMÕES, M. G.; SHAW, I. S. Controle e modelagem fuzzy. *Blucher: FAPESP, São Paulo, BRA*, 2007. Citado na página 20.
- ZADEH, L. A. Fuzzy logic. *Computer*, IEEE, v. 21, n. 4, p. 83–93, 1988. Citado 3 vezes nas páginas 16, 18 e 19.
- ZIMMERMANN, H.-J. *Fuzzy set theory—and its applications*. [S.l.]: Springer Science & Business Media, 2011. Citado na página 17.

Apêndices

APÊNDICE A – Questionário

Foi realizado um questionário para os estudantes de Sistemas de Informação no Campus Monte Carmelo e para outras pessoas selecionadas.

TCC - Sistema de Restaurante Utilizando Lógica Fuzzy.

Meu trabalho é coletar variáveis que afetam o tempo que a pessoa gasta dentro de um restaurante, dependendo da quantidade das pessoas, se está chovendo ou não, se está bebendo ou não e se está tendo show ao vivo ou não.

* Required

1. Qual fator é mais relevante para que fique mais tempo em um restaurante? *

Check all that apply.

- ☐ Se estou bebendo
- ☐ Se está chovendo
- ☐ Se está tendo música ao vivo
- ☐ Other: _____

2. Quanto tempo é acrescentado no tempo que você gasta em um restaurante se estiver bebendo? *

Mark only one oval.

- ☐ Nenhum
- ☐ Entre 10 a 20 minutos
- ☐ Entre 20 a 40 minutos
- ☐ Mais de 40 minutos
- ☐ Other: _____

3. Quanto tempo é acrescentado no tempo que você gasta em um restaurante se estiver chovendo? *

Mark only one oval.

- ☐ Nenhum
- ☐ Entre 10 a 20 minutos
- ☐ Entre 20 a 40 minutos
- ☐ Mais de 40 minutos
- ☐ Other: _____

Figura 35 – Parte 1 do questionário realizado

Fonte: Do autor

4. Quanto tempo é acrescentado no tempo que você gasta em um restaurante se estiver tendo show ao vivo?

Mark only one oval.

- ☐ Nenhum
- ☐ Entre 10 a 20 minutos
- ☐ Entre 20 a 40 minutos
- ☐ Mais de 40 minutos
- ☐ Other: _____

5. Se você estiver sozinho, você costuma ficar quanto tempo em um restaurante?

Mark only one oval.

- ☐ Menos de 30 minutos
- ☐ Entre 30 minutos a 50 minutos
- ☐ Entre 50 minutos a 1 hora e meia
- ☐ Mais de uma hora e meia
- ☐ Other: _____

6. Se você estiver com outra pessoa, você costuma ficar quanto tempo em um restaurante?

Mark only one oval.

- ☐ Menos de 30 minutos
- ☐ Entre 30 minutos a 50 minutos
- ☐ Entre 50 minutos a 1 hora e meia
- ☐ Mais de 1 hora e meia
- ☐ Other: _____

7. Se você estiver com um grupo entre 3 a 6 pessoas, você costuma ficar quanto tempo em um restaurante?

Mark only one oval.

- ☐ Menos de 30 minutos
- ☐ Entre 30 minutos a 50 minutos
- ☐ Entre 50 minutos a 1 hora e meia
- ☐ Mais de 1 hora e meia
- ☐ Other: _____

Figura 36 – Parte 2 do questionário realizado

Fonte: Do autor

8. Se você estiver com um grupo com mais de 6 pessoas, você costuma ficar quanto tempo em um restaurante?

Mark only one oval.

- ☐ Menos de 30 minutos
- ☐ Entre 30 minutos a 50 minutos
- ☐ Entre 50 minutos a 1 hora e meia
- ☐ Mais de 1 hora e meia
- ☐ Other: _____

Figura 37 – Parte 3 do questionário realizado

Fonte: Do autor

APÊNDICE B – Resultados obtidos pelo questionário

Qual fator é mais relevante para que fique mais tempo em um restaurante?

27 responses

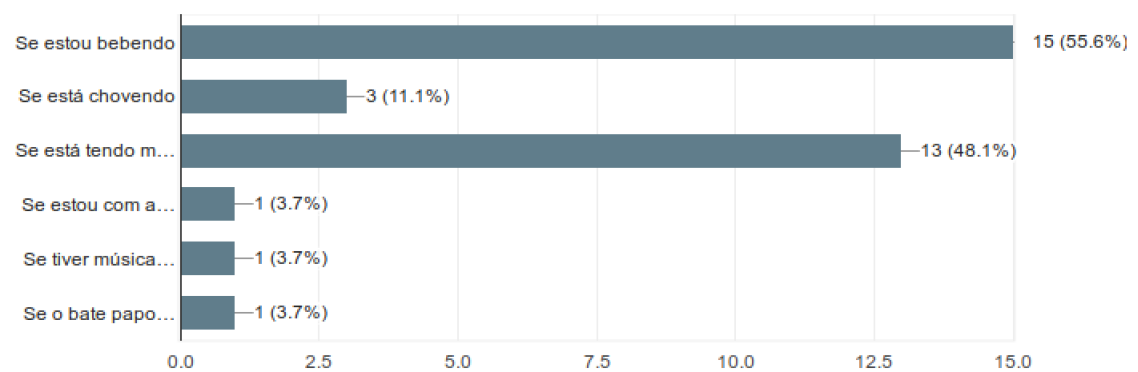


Figura 38 – Respostas da primeira pergunta do questionário

Fonte: Do autor

Quanto tempo é acrescido no tempo que você gasta em um restaurante se estiver bebendo?

27 responses

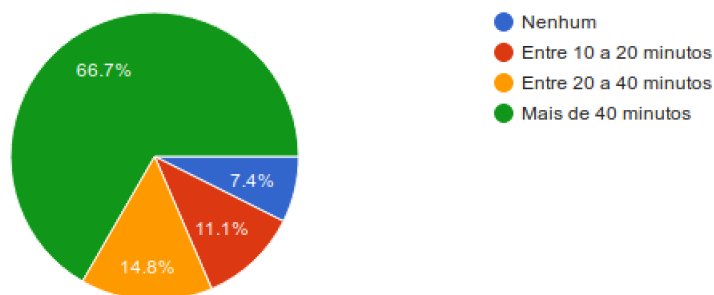


Figura 39 – Respostas da segunda pergunta do questionário

Fonte: Do autor

Quanto tempo é acrescido no tempo que você gasta em um restaurante se estiver chovendo?

27 responses

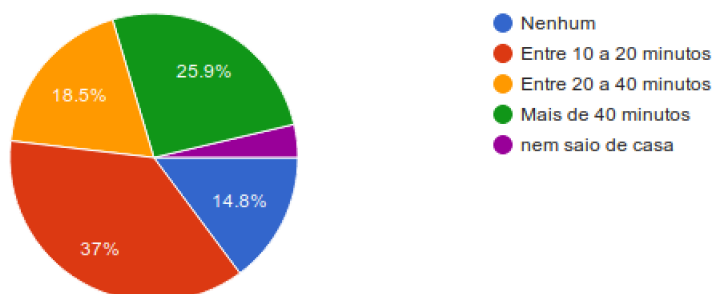


Figura 40 – Respostas da terceira pergunta do questionário

Fonte: Do autor

Quanto tempo é acrescido no tempo que você gasta em um restaurante se estiver tendo show ao vivo?

27 responses

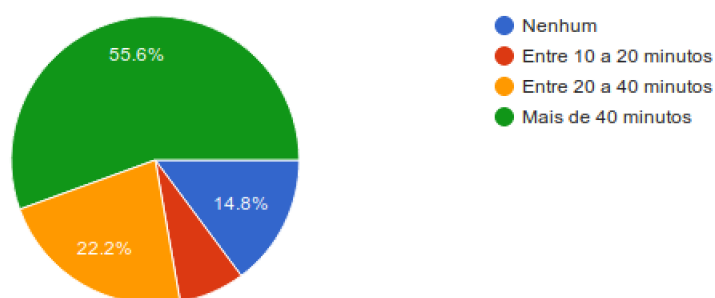


Figura 41 – Respostas da quarta pergunta do questionário

Fonte: Do autor

Se você estiver sozinho, você costuma ficar quanto tempo em um restaurante?

26 responses

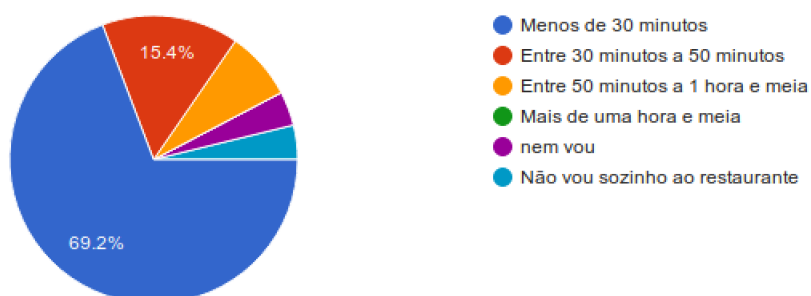


Figura 42 – Respostas da quinta pergunta do questionário

Fonte: Do autor

Se você estiver com outra pessoa, você costuma ficar quanto tempo em um restaurante?

27 responses

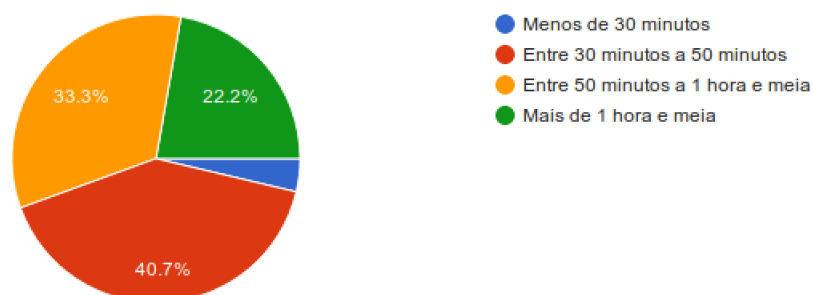


Figura 43 – Respostas da sexta pergunta do questionário

Fonte: Do autor

Se você estiver com um grupo entre 3 a 6 pessoas, você costuma ficar quanto tempo em um restaurante?

27 responses

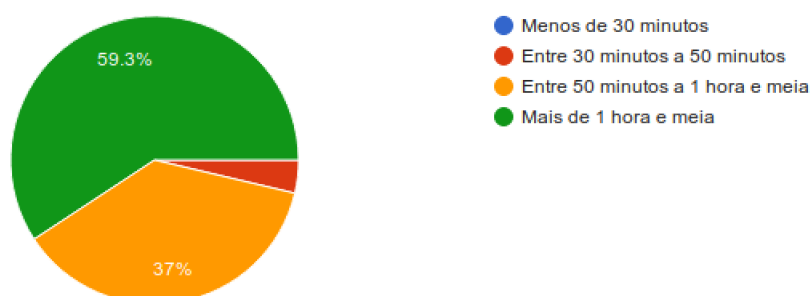


Figura 44 – Respostas da sétima pergunta do questionário

Fonte: Do autor

Se você estiver com um grupo com mais de 6 pessoas, você costuma ficar quanto tempo em um restaurante?

27 responses

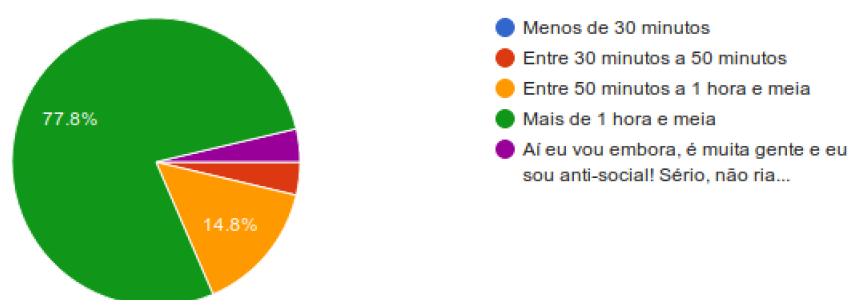


Figura 45 – Respostas da oitava pergunta do questionário

Fonte: Do autor

APÊNDICE C – Bloco de Regras Utilizado no Arquivo FCL

```

RULEBLOCK No1
  AND : MIN;
  ACT : MIN;
  ACCU : MAX;

  RULE 1: if show is nao and bebida is nao and chuva is sem_chuva and qnt is int1 then tempo is tint1;
  RULE 2: if show is nao and bebida is sim and chuva is sem_chuva and qnt is int1 then tempo is tint3;
  RULE 3: if show is nao and bebida is nao and chuva is com_chuva and qnt is int1 then tempo is tint2;
  RULE 4: if show is nao and bebida is sim and chuva is com_chuva and qnt is int1 then tempo is tint4;
  RULE 5: if show is sim and bebida is nao and chuva is sem_chuva and qnt is int1 then tempo is tint3;
  RULE 6: if show is sim and bebida is sim and chuva is sem_chuva and qnt is int1 then tempo is tint5;
  RULE 7: if show is sim and bebida is nao and chuva is com_chuva and qnt is int1 then tempo is tint4;
  RULE 8: if show is sim and bebida is sim and chuva is com_chuva and qnt is int1 then tempo is tint6;
  RULE 9: if show is nao and bebida is nao and chuva is sem_chuva and qnt is int2 then tempo is tint2;
  RULE 10: if show is nao and bebida is sim and chuva is sem_chuva and qnt is int2 then tempo is tint4;
  RULE 11: if show is nao and bebida is nao and chuva is com_chuva and qnt is int2 then tempo is tint3;
  RULE 12: if show is nao and bebida is sim and chuva is com_chuva and qnt is int2 then tempo is tint5;
  RULE 13: if show is sim and bebida is nao and chuva is sem_chuva and qnt is int2 then tempo is tint4;
  RULE 14: if show is sim and bebida is sim and chuva is sem_chuva and qnt is int2 then tempo is tint6;
  RULE 15: if show is sim and bebida is nao and chuva is com_chuva and qnt is int2 then tempo is tint5;
  RULE 16: if show is sim and bebida is sim and chuva is com_chuva and qnt is int2 then tempo is tint7;
  RULE 17: if show is nao and bebida is nao and chuva is sem_chuva and qnt is int3 then tempo is tint3;
  RULE 18: if show is nao and bebida is sim and chuva is sem_chuva and qnt is int3 then tempo is tint5;
  RULE 19: if show is nao and bebida is nao and chuva is com_chuva and qnt is int3 then tempo is tint4;
  RULE 20: if show is nao and bebida is sim and chuva is com_chuva and qnt is int3 then tempo is tint6;
  RULE 21: if show is sim and bebida is nao and chuva is sem_chuva and qnt is int3 then tempo is tint5;
  RULE 22: if show is sim and bebida is sim and chuva is sem_chuva and qnt is int3 then tempo is tint7;
  RULE 23: if show is sim and bebida is nao and chuva is com_chuva and qnt is int3 then tempo is tint6;
  RULE 24: if show is sim and bebida is sim and chuva is com_chuva and qnt is int3 then tempo is tint8;
  RULE 25: if show is nao and bebida is nao and chuva is sem_chuva and qnt is int4 then tempo is tint4;
  RULE 26: if show is nao and bebida is sim and chuva is sem_chuva and qnt is int4 then tempo is tint6;
  RULE 27: if show is nao and bebida is nao and chuva is com_chuva and qnt is int4 then tempo is tint5;
  RULE 28: if show is nao and bebida is sim and chuva is com_chuva and qnt is int4 then tempo is tint7;
  RULE 29: if show is sim and bebida is nao and chuva is sem_chuva and qnt is int4 then tempo is tint6;
  RULE 30: if show is sim and bebida is sim and chuva is sem_chuva and qnt is int4 then tempo is tint8;
  RULE 31: if show is sim and bebida is nao and chuva is com_chuva and qnt is int4 then tempo is tint7;
  RULE 32: if show is sim and bebida is sim and chuva is com_chuva and qnt is int4 then tempo is tint9;
  RULE 33: if show is nao and bebida is nao and chuva is sem_chuva and qnt is int5 then tempo is tint5;
  RULE 34: if show is nao and bebida is sim and chuva is sem_chuva and qnt is int5 then tempo is tint7;
  RULE 35: if show is nao and bebida is nao and chuva is com_chuva and qnt is int5 then tempo is tint6;
  RULE 36: if show is nao and bebida is sim and chuva is com_chuva and qnt is int5 then tempo is tint8;
  RULE 37: if show is sim and bebida is nao and chuva is sem_chuva and qnt is int5 then tempo is tint7;
  RULE 38: if show is sim and bebida is sim and chuva is sem_chuva and qnt is int5 then tempo is tint9;
  RULE 39: if show is sim and bebida is nao and chuva is com_chuva and qnt is int5 then tempo is tint8;
  RULE 40: if show is sim and bebida is sim and chuva is com_chuva and qnt is int5 then tempo is tint10;
  RULE 41: if show is nao and bebida is nao and chuva is sem_chuva and qnt is int6 then tempo is tint6;
  RULE 42: if show is nao and bebida is sim and chuva is sem_chuva and qnt is int6 then tempo is tint8;
  RULE 43: if show is nao and bebida is nao and chuva is com_chuva and qnt is int6 then tempo is tint7;
  RULE 44: if show is nao and bebida is sim and chuva is com_chuva and qnt is int6 then tempo is tint9;
  RULE 45: if show is sim and bebida is nao and chuva is sem_chuva and qnt is int6 then tempo is tint8;
  RULE 46: if show is sim and bebida is sim and chuva is sem_chuva and qnt is int6 then tempo is tint10;
  RULE 47: if show is sim and bebida is nao and chuva is com_chuva and qnt is int6 then tempo is tint9;
  RULE 48: if show is sim and bebida is sim and chuva is com_chuva and qnt is int6 then tempo is tint11;
END_RULEBLOCK

```

Figura 46 – Regras definidas no Bloco de Regras

Fonte: Do autor