

**UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE – UERN  
FACULDADE DE CIÊNCIAS EXATAS E NATURAIS – FANAT  
DEPARTAMENTO DE INFORMÁTICA – DI**

**ÍCARO ANDRÉ VIANA DE AZEVEDO**

**ECONOFERRO: APLICATIVO MÓVEL PARA O PROBLEMA DO CORTE DE  
VERGALHÕES COM SOBRAS APROVEITÁVEIS NA CONSTRUÇÃO CIVIL**

**MOSSORÓ - RN  
2019**

**ÍCARO ANDRÉ VIANA DE AZEVEDO**

**ECONOFERRO: APLICATIVO MÓVEL PARA O PROBLEMA DO CORTE DE  
VERGALHÕES COM SOBRAS APROVEITÁVEIS NA CONSTRUÇÃO CIVIL**

**Monografia apresentada à Universidade do Estado do Rio Grande do Norte, como um dos pré-requisitos para obtenção do grau de bacharel em Ciência da Computação, sob orientação do Prof. Dr. Dario José Aloise e co-orientadora Profa. Dra. Cicilia Raquel Maia Leite.**

**MOSSORÓ - RN**

**2019**

© Todos os direitos estão reservados a Universidade do Estado do Rio Grande do Norte. O conteúdo desta obra é de inteira responsabilidade do(a) autor(a), sendo o mesmo, passível de sanções administrativas ou penais, caso sejam infringidas as leis que regulamentam a Propriedade Intelectual, respectivamente, Patentes: Lei nº 9.279/1996 e Direitos Autorais: Lei nº 9.610/1998. A mesma poderá servir de base literária para novas pesquisas, desde que a obra e seu(a) respectivo(a) autor(a) sejam devidamente citados e mencionados os seus créditos bibliográficos.

**Catálogo da Publicação na Fonte.  
Universidade do Estado do Rio Grande do Norte.**

A994e Azevedo, Ícaro André Viana de  
ECONOFERRO: APLICATIVO MÓVEL PARA O  
PROBLEMA DO CORTE DE VERGALHÕES COM  
SOBRAS APROVEITÁVEIS NA CONSTRUÇÃO CIVIL. /  
Ícaro André Viana de Azevedo. - Mossoró, 2019.  
63p.

Orientador(a): Prof. Dr. Dario José Aloise.  
Coorientador(a): Profa. Dra. Círcia Raquel Maia Leite.  
Monografia (Graduação em Ciência da Computação).  
Universidade do Estado do Rio Grande do Norte.

1. Ciência da Computação. 2. Heurísticas. 3.  
Otimização. 4. Aplicativo. 5. Econoferro. I. Aloise, Dario  
José. II. Universidade do Estado do Rio Grande do Norte.  
III. Título.

O serviço de Geração Automática de Ficha Catalográfica para Trabalhos de Conclusão de Curso (TCC's) foi desenvolvido pela Diretoria de Informatização (DINF), sob orientação dos bibliotecários do SIB-UERN, para ser adaptado às necessidades da comunidade acadêmica UERN.

ÍCARO ANDRÉ VIANA DE AZEVEDO

ECONOFERRO: APLICATIVO MÓVEL PARA O PROBLEMA DO CORTE DE VERGALHÕES COM SOBRAS APROVEITÁVEIS NA CONSTRUÇÃO CIVIL

Monografia apresentada como pré-requisito para a obtenção do título de Bacharel em Ciência da Computação da Universidade do Estado do Rio Grande do Norte – UERN, submetida à aprovação da banca examinadora composta pelos seguintes membros:

Aprovada em: 03/05/2019

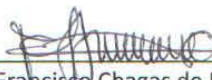
#### Banca Examinadora



\_\_\_\_\_  
Prof. Dr. Dario José Aloise  
Universidade do Estado do Rio Grande do Norte - UERN



\_\_\_\_\_  
Profa. Dra. Cíllia Raquel Maia Leite  
Universidade do Estado do Rio Grande do Norte - UERN



\_\_\_\_\_  
Prof. Dr. Francisco Chagas de Lima Júnior  
Universidade do Estado do Rio Grande do Norte - UERN



\_\_\_\_\_  
Prof. Dr. Carlos Heitor Pereira Liberalino  
Universidade do Estado do Rio Grande do Norte - UERN

*Dedico este trabalho aos meus pais e a  
minha esposa, que sempre acreditaram  
em mim e me apoiaram nos momentos  
mais difíceis.*

## **AGRADECIMENTOS**

A Deus, primeiramente.

Aos meus pais, José Jeová de Azevedo e Suely Carneiro Viana de Azevedo, por sempre ter acreditado em mim e ter apoiado toda minha trajetória no curso.

A minha esposa, Liza Alexandra Viera Magalhães de Azevedo, desde o momento em que plantou a ideia do conhecimento. Ela que sempre incentivou na minha trajetória e apoiou em todos os momentos difíceis.

Ao professor orientador Dr. Dario José Aloise, pelo vasto conhecimento na área abordada por este trabalho, que me proporcionou aprendizado e segurança para realizá-lo a contento.

À professora Co-orientadora Dra. Círcia Raquel Maia Leite, por total suporte no desenvolvimento deste trabalho.

Aos professores do Departamento de informática UERN do Campus Central, por ter proporcionado uma vasta visão da computação e ótimos conselhos ao decorrer do curso.

*“O que é invisível para nós é também  
crucial para o nosso bem-estar”.*

(Desconhecido)

## RESUMO

Nas obras da construção civil, quando há a necessidade de cortar vergalhões de aço durante a montagem de uma estrutura, como por exemplo: vigas, estacas e pilares. Com isso, surgem dificuldades na hora de planejar os cortes em função da quantidade de vergalhões necessários para atender a demanda das estruturas. E o problema aumenta cada vez em que há o acréscimo de cortes de vergalhões para mais estruturas. Desta forma, o objetivo desse trabalho é o desenvolvimento do aplicativo móvel Android, intitulado de Econoferro, que visa a otimização do plano de corte de vergalhões com sobras aproveitáveis, por meio de procedimentos heurísticos que fornece uma solução satisfatória com mínima perda de material, facilitando ao usuário final a geração de planos de corte das estruturas através do seu dispositivo Android.

**Palavras-Chave:** Plano de corte; Vergalhões de aço; Aplicativo Econoferro; Otimização; Heurística.



## **ABSTRACT**

In the construction works, when there is a need to cut steel rebar during assembly of a structure, such as beams, stakes and pillars. This leads to difficulties in planning cuts due to the amount of rebar required to meet the demand of the structures. And the problem increases every time there is the addition of rebar cuts to more structures. In this way the objective of this work is the development of the Android mobile application titled Econoferro, which aims to optimize the plan of cutting rebar with usable leftovers, through heuristic procedures that provides a satisfactory solution with minimum loss of material, facilitating the end user generating plans to cut structures through their Android device.

**Keywords:** Cutting plan; Steel Rebar; App Econoferro; Optimization; Heuristics.

## LISTA DE SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
BF	Best Fit
BFD	Best Fit Decreasing
FF	First Fit
FFD	First Fit Decreasing
IDE	Integrated Development Environment
INPI	Instituto Nacional da Propriedade Industrial
LI	Limite Inferior
MMD	Maiores Menores Decrescentes
OPT	Optimal
PCVSA	Problema de Corte de Vergalhões com Sobras Aproveitáveis
UML	Unified Modeling Language

## LISTA DE FIGURAS

Figura 1 – Exemplo de empacotamento ótimo .....	15
Figura 2 – Demanda de Itens desordenados, Corte de Vergalhões Padrão..	16
Figura 3 – Quantidade de bins (Vergalhões Padrão) requeridos para o FF. .	16
Figura 4 – Quantidade de bins requeridos para o BF. ....	17
Figura 5 - Demanda de Itens (Cortes Vergalhões) em ordem decrescente..	18
Figura 6 – Quantidade de bins requeridos para o FFD.....	19
Figura 7 – Quantidade de bins requeridos para o MMD. ....	20
Figura 8 – Visão geral do Econoferro .....	31
Figura 9 – Diagrama de Sequência, sucesso Login. ....	33
Figura 10 – Interface do Android Studio .....	34
Figura 11 – Interface de login .....	36
Figura 12 – Tela de Cadastro .....	36
Figura 13 – Tela principal .....	37
Figura 14 – Tela para adicionar obra.....	38
Figura 15 – Tela para adicionar uma nova estrutura civil .....	39
Figura 16 – Tela para gerar o plano de corte em .pdf.....	40

## LISTA DE TABELAS

Tabela 1 – Resultados obtidos com a demanda de 20 itens. ....	24
Tabela 2 – Resultados obtidos com a demanda de 50 itens. ....	25
Tabela 3 – Resultados obtidos com a demanda de 100 itens. ....	25
Tabela 4 – Resultados obtidos com a demanda de 1000 itens. ....	26
Tabela 5 – Resultados obtidos com a demanda de 5000 itens. ....	26
Tabela 6 – Resultados obtidos para 20 instâncias com 5 mil itens.....	26
Tabela 7 – Sobra Total em metros, 20 instâncias com 5 mil itens. ....	27
Tabela 8 – Quantidade de vergalhões para cada demanda por bitola. ....	41

## LISTA DE GRÁFICOS

Gráfico 1 – Quantidade de bins gerados em cada instância.....	29
Gráfico 2 – Sobra total em cada instância, quanto menor, melhor. ....	29

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO</b> .....	<b>12</b>
1.1	DEFINIÇÃO DO PROBLEMA.....	12
1.2	OBJETIVOS DA PESQUISA .....	13
<b>2</b>	<b>FUNDAMENTAÇÃO TEÓRICA</b> .....	<b>14</b>
2.1	PROBLEMAS DE CORTE E EMPACOTAMENTO (BIN-PACKING).....	14
2.2	HEURÍSTICAS PARA O PROBLEMA DE CORTE E EMPACOTAMENTO... 15	
<b>2.2.1</b>	<b>First Fit</b> .....	<b>15</b>
<b>2.2.2</b>	<b>Best Fit</b> .....	<b>17</b>
<b>2.2.3</b>	<b>First Fit Decreasing</b> .....	<b>17</b>
<b>2.2.4</b>	<b>Best Fit Decreasing</b> .....	<b>19</b>
<b>2.2.5</b>	<b>Maiores Menores Decrescentes</b> .....	<b>19</b>
2.3	APLICAÇÃO DA HEURÍSTICA MMD AO PCVSA .....	20
<b>2.3.1</b>	<b>Procedimentos Heurísticos</b> .....	<b>21</b>
<b>2.3.2</b>	<b>Construção do algoritmo MMD</b> .....	<b>22</b>
<b>3</b>	<b>TESTES COMPUTACIONAIS</b> .....	<b>24</b>
<b>4</b>	<b>DESENVOLVIMENTO DO APLICATIVO MÓVEL ANDROID</b> .....	<b>31</b>
4.1	VISÃO GERAL.....	31
4.2	ENGENHARIA DE SOFTWARE .....	31
<b>4.2.1</b>	<b>Engenharia de requisitos</b> .....	<b>32</b>
<b>4.2.2</b>	<b>Modelagem UML do aplicativo Econoferro</b> .....	<b>33</b>
<b>4.2.4</b>	<b>Verificação e Validação</b> .....	<b>40</b>
<b>5</b>	<b>CONSIDERAÇÕES FINAIS</b> .....	<b>42</b>
	<b>REFERÊNCIAS</b> .....	<b>43</b>
	<b>APÊNDICE A – Diagrama de classes ECONOFERRO</b> .....	<b>45</b>
	<b>APÊNDICE B – Documento de requisitos aplicativo ECONOFERRO</b> .....	<b>46</b>
	<b>ANEXO A – Lista de demandas laje</b> .....	<b>56</b>
	<b>ANEXO B – Lista de demandas vigas</b> .....	<b>57</b>

## 1 INTRODUÇÃO

Na construção civil é muito comum o uso de aço em estruturas de concreto na forma de vergalhões, ou seja, de barras cilíndricas, sejam elas lisas ou nervuradas. Esse é um dos componentes denominados concreto armado, utilizado nessa área. Quando existe a necessidade da utilização desse material nas estruturas, os engenheiros e técnicos assumem o papel de realizar o plano de corte do aço, que será desenvolvido conforme a leitura do projeto estrutural.

O Problema do Corte de Vergalhões com Sobras Aproveitáveis (PCVSA) na Construção Civil ou plano de corte de ferragem, como é chamado nas obras, nada mais é do que um mapeamento do corte do aço. Este visa a sua economia e o aproveitamento das sobras que ocasionalmente são geradas, tendo-se em vista que esse é um dos materiais mais caros de uma obra (MORAES,1997). Além disso, o objetivo do plano de corte é fornecer uma quantidade mínima do aço que deverá ser comprado, de forma a minimizar a perda ou que a mesma permaneça dentro da margem de desperdício, a fim de que não haja prejuízo para a empreiteira.

Entretanto, mesmo sob a orientação do projeto estrutural normatizada pela ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT) NBR 6118 de 2014, a que diz que a elaboração do plano de corte não é tão simples e, mesmo seguindo fielmente o projeto, ainda é comum a ocorrência de erros grosseiros, como o não aproveitamento correto das sobras.

Esses erros ocorrem quando engenheiros e técnicos procuram alternativas em planilhas eletrônicas, inserindo manualmente os dados e obtendo resultados variados, ocasionando em muitos casos na falta de material ou desperdício deste, em que a quantidade comprada não atende de forma eficiente a demanda requerida pelo projeto estrutural. Ou seja, o plano de corte de aço na construção civil ainda é feito de forma artesanal, manual, de forma empírica e em planilhas, gerando um desperdício de tempo e da ferragem, acarretando em custos acima do esperado (MORAES, 1997).

### 1.1 DEFINIÇÃO DO PROBLEMA

O problema de corte de vergalhões consiste na variação de diferentes padrões de corte para uma determinada categoria de aço, estabelecido pela ABNT NBR 7480

2007, são eles: CA-25, CA-50 e CA-60 com diferentes tipos de bitolas. O vergalhão do aço é vendido em barras cilíndricas de 12 metros e pode ser usado em diversos elementos estruturais, como: estacas, blocos, sapatas, pilares, vigas e lajes. Como o corte é feito apenas em um sentido, o problema é de uma única dimensão. Sendo assim, o problema é de corte e empacotamento unidimensional.

## 1.2 OBJETIVOS DA PESQUISA

Esta pesquisa consiste no trabalho de conclusão de curso e tem uma contribuição para a área da construção civil. Seu objetivo geral é desenvolver um aplicativo móvel *Android*, utilizando técnicas de otimização combinatória, adaptadas para otimizar o plano de corte de vergalhões com sobras aproveitáveis. Mais especificamente, este trabalho visa:

- Pesquisar e desenvolver um aplicativo que envolve o módulo de otimização do corte de vergalhões com sobras aproveitáveis;
- Estudar o problema *bin-packing* unidimensional e suas heurísticas;
- Implementar um algoritmo heurístico MMD, para compor o módulo otimizador do aplicativo ECONOFERRO.

Este trabalho está organizado da seguinte forma: No capítulo 2, são apresentados toda a fundamentação teórica do módulo otimizador e técnicas computacionais utilizadas. O capítulo 3 apresenta testes computacionais, com análise e comparações de desempenho entre a heurística MMD e as heurísticas FFD e a BFD. No capítulo 4 é descrito o desenvolvimento do aplicativo *Android* ECONOFERRO, demonstrando as tecnologias e técnicas utilizadas. No capítulo 5, são apresentadas as considerações finais.



## 2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo apresentar toda a base teórica para o Problema de Corte de Vergalhões com Sobras Aproveitáveis (PCVSA). A fundamentação é de importância para elaboração, análise e interpretação dos dados obtidos, além de ser a base para o desenvolvimento da programação em Java da heurística implementada.

### 2.1 PROBLEMAS DE CORTE E EMPACOTAMENTO (BIN-PACKING)

Em problemas de otimização, busca-se maximizar (ou minimizar) uma função específica a qual é chamada de função objetivo.

O problema de corte e empacotamento é considerado NP-completo (GAREY e JOHNSON, 1979), visto que não se sabe da existência de algoritmos exatos que resolvam esse tipo de problema em tempo polinomial.

Algoritmos exatos que resolvem problemas NP-completo, levam a uma solução ótima, entretanto, apresentam complexidade de tempo exponencial, e isso resulta em um custo computacional muito alto (CORMEN et al, 2012).

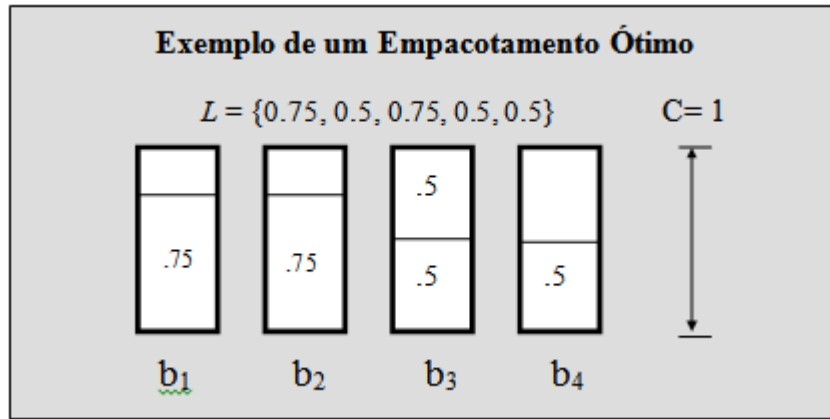
Por outro lado, as heurísticas não garantem soluções ótimas, porém garantem soluções sub-ótimas em menor tempo computacional, i.e., tempo polinomial (CORMEN et al, 2012).

O problema pode ser descrito como:

Seja  $L = \{a_1, a_2, \dots, a_n\}$  uma lista de itens (demanda de pedaços de vergalhões), onde  $0 < a_i \leq 1$  (vergalhões de tamanho 1. Está normalizado, ou seja, tudo dividido por 12 que é o tamanho padrão dos vergalhões); para  $1 \leq i \leq n$ . O problema *bin-packing* é empacotar (ou, cortar em pedaços de tamanho)  $a_1, a_2, \dots, a_n$  dentro do menor número de bins (mochilas, ou o que é a mesma coisa, cortar vergalhões) quanto possível. Uma solução ótima pode ser encontrada considerando todas as maneiras de particionar um conjunto de  $n$  itens dentro de  $n$  ou menos subconjuntos, mas o número de possíveis partições é maior do que  $(n/2)^{n/2}$ .

Exemplo de uma solução ótima é apresentado na Figura 1.

**Figura 1** – Exemplo de empacotamento ótimo.



Fonte: Baase (1988).

## 2.2 HEURÍSTICAS PARA O PROBLEMA DE CORTE E EMPACOTAMENTO

Neste trabalho, foi realizado o estudo das seguintes heurísticas para o problema de *bin-packing*: *First-Fit* (FF), *Best-Fit* (BF), *First-Fit Decreasing* (FFD), *Best-Fit Decreasing* (BFD) e por fim a heurística Maiores e Menores Decrescentes (MMD) proposta por ALOISE (1992).

### 2.2.1 First Fit

Na heurística *First-Fit*, o algoritmo fornece uma solução de tempo computacional rápida, mas, por outro lado, não fornece soluções satisfatórias. O algoritmo insere o item na primeira posição que couber e assim para os demais itens até atender a demanda. Seu desempenho computacional é na ordem de  $O(n \log n)$  operações, em que  $n$  é o número de elementos a ser empacotados (JOHNSON, 1973).

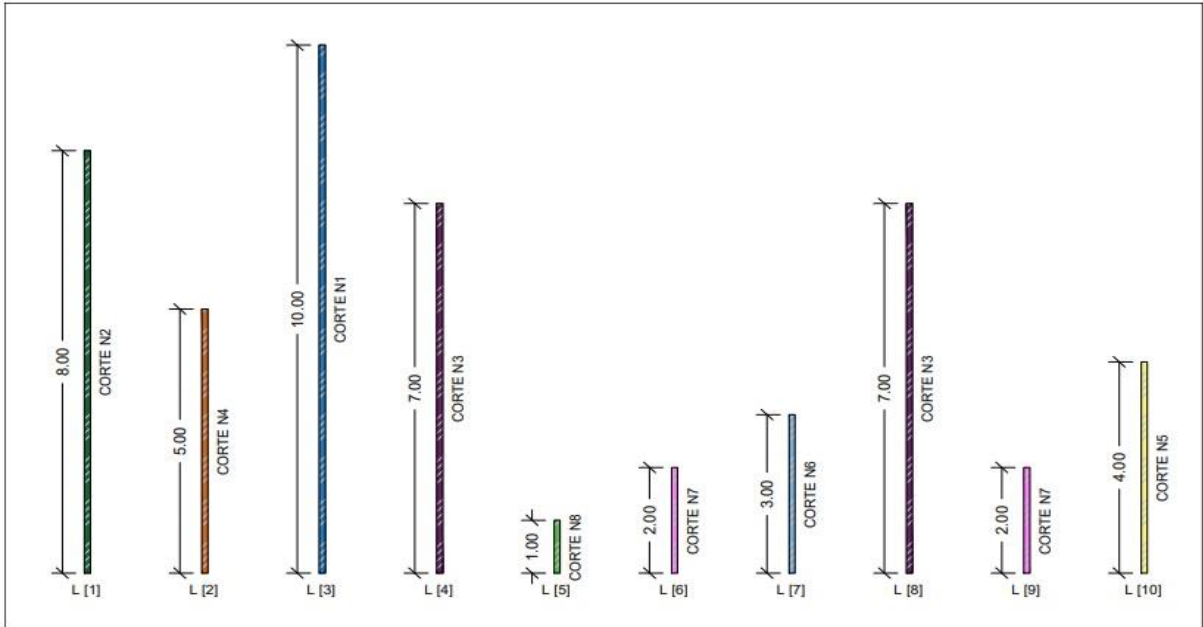
O algoritmo é classificado como guloso, em que processa os itens em ordem como chegam e para cada item verifica o espaço no primeiro pacote, que pode ser inserido. Caso não encontre nenhum espaço no pacote, verifica-se se existe outro pacote seguinte com espaço; caso contrário cria-se um pacote e adiciona o item.

JOHNSON et al (1974) apresenta a solução ótima no pior caso para o algoritmo FF é de:

$$FF(L) \leq \frac{17}{10} OPT(L) + 2 \text{ bins}$$

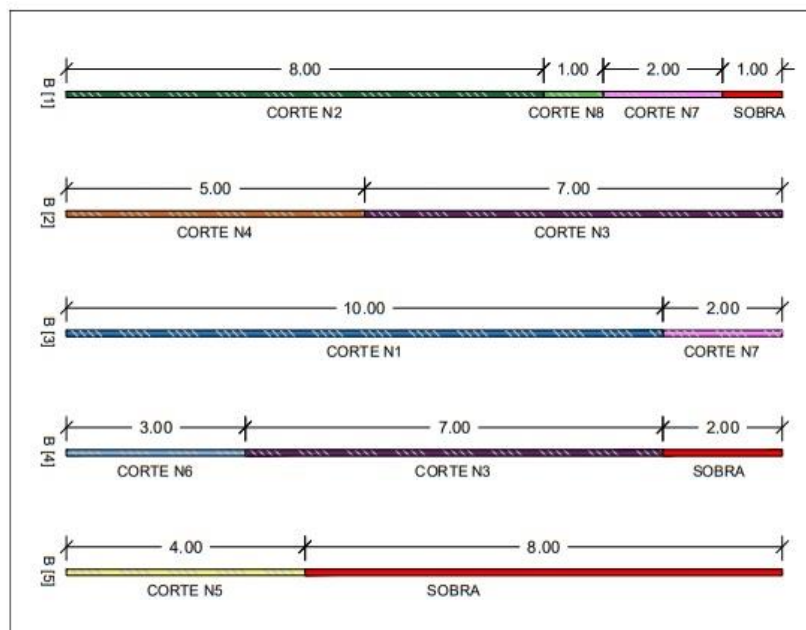
Uma ilustração da heurística *First-Fit*. Na Figura 2, tem-se a demanda de itens para serem cortados e na Figura 3, tem-se a quantidade de bins necessários e seus cortes, gerados a partir da heurística.

**Figura 2 – Demanda de Itens desordenados, Corte de Vergalhões Padrão.**



Fonte: Autoria Própria (2019).

**Figura 3 – Quantidade de bins (Vergalhões Padrão)**



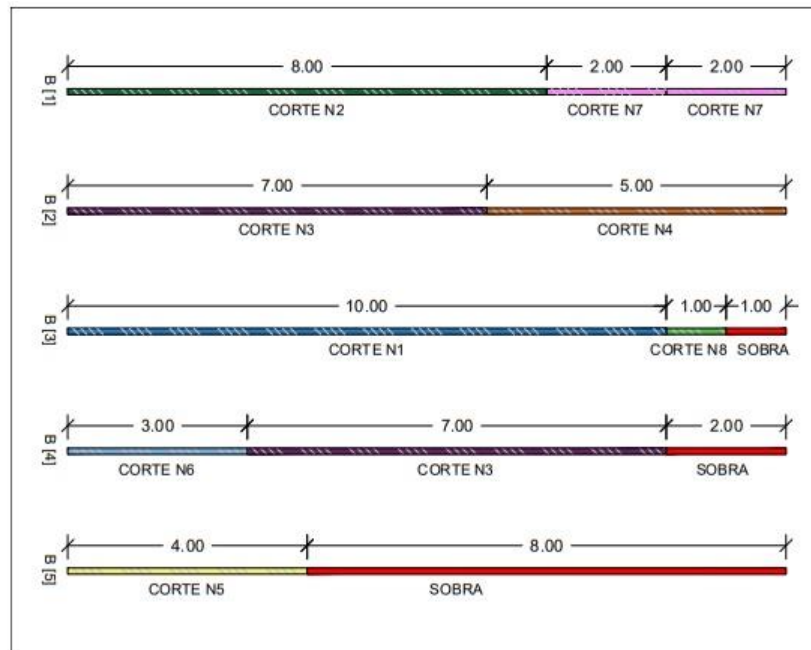
Fonte: Autoria Própria (2019).

### 2.2.2 Best Fit

Na heurística *Best-Fit*, o algoritmo funciona da seguinte forma: seleciona o item na ordem em que foi colocado e verifica em todos os pacotes, qual possui a menor sobra que caiba esse item. Sendo assim, o algoritmo apresenta um pior desempenho computacional, devido a verificação completa toda a vez que for adicionar um novo item no pacote (JOHNSON, 1973).

A solução ótima no pior caso segue a mesma fórmula matemática da heurística *First-Fit*. Na Figura 4, uma ilustração da aplicação da heurística *Best-Fit*, utilizando a mesma demanda de itens ilustrada anteriormente na Figura 2.

**Figura 4 – Quantidade de bins requeridos para o BF.**



Fonte: Autoria Própria (2019).

### 2.2.3 First Fit Decreasing

A heurística *First-Fit Decreasing* (FFD) é uma melhoria do algoritmo *First-Fit* (FF), em que foi acrescentando, no início do algoritmo, uma ordenação de ordem decrescente (JOHNSON, 1973). Seu tempo computacional difere do *First-Fit* (FF) devido a ordenação dos itens e essa ordenação pode variar com o algoritmo de ordenação de ordem de complexidade  $O(n \log n)$ , sejam eles: *MergeSort*, *QuickSort*

ou *HeapSort*. Na sequência, é apresentada a solução ótima no pior caso da heurística encontrada por (JOHNSON et al, 1974).

$$FFD(L) \leq \frac{11}{9} OPT(L) + 4 \text{ bins}$$

Ao longo dos anos, surgiram novas provas demonstrando a eficiência da heurística FFD, assim (MINYI, 1991) provou que a solução ótima no pior caso é:

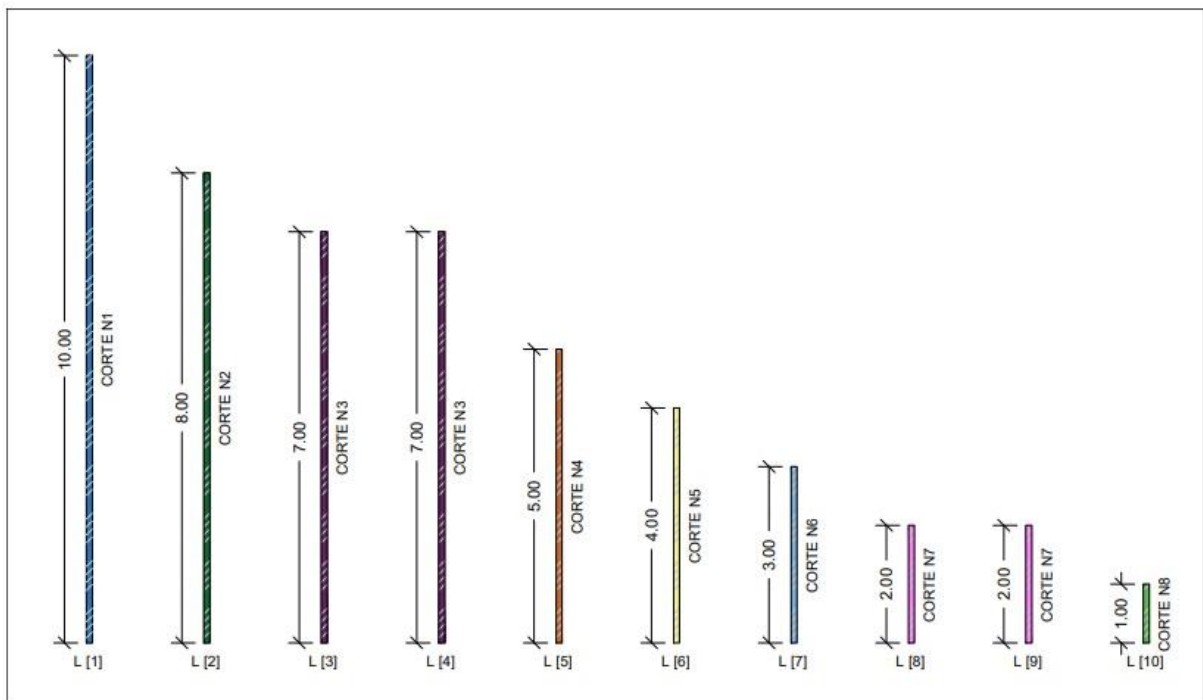
$$FFD(L) \leq \frac{11}{9} OPT(L) + 1 \text{ bins}$$

Por fim, Dósa (2007) provou que a solução ótima no pior caso é de:

$$FFD(L) \leq \frac{11}{9} OPT(L) + \frac{6}{9} \text{ bins}$$

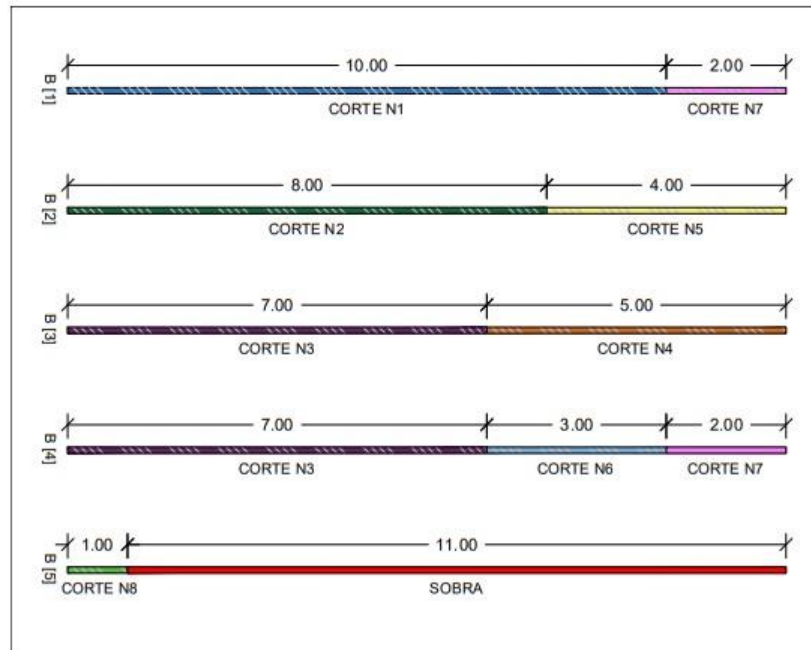
Uma ilustração da heurística *First-Fit Decreasing* (FFD), em que se utiliza da mesma demanda ilustrada anteriormente na Figura 2, porém em ordem decrescente. Na Figura 5, demanda de itens em ordem decrescente para melhor ilustração e na Figura 6, a quantidade de bins necessário e seus cortes gerados a partir da heurística.

**Figura 5 - Demanda de Itens (Cortes Vergalhões) em ordem decrescente.**



Fonte: Autoria Própria (2019).

**Figura 6** – Quantidade de bins requeridos para o FFD.



Fonte: Autoria Própria (2019).

#### 2.2.4 Best Fit Decreasing

A heurística *Best-Fit Decreasing* (BFD) é uma melhoria do algoritmo *Best-Fit* (BF), em que foi acrescentando no início do algoritmo uma ordenação de ordem decrescente. A solução ótima no pior caso também foi provada por (JOHNSON et al, 1974).

$$FFD(L) \leq \frac{11}{9} OPT(L) + 4 \text{ bins}$$

Utilizando da mesma demanda de itens em ordem decrescente ilustrada anteriormente na Figura 5, foi constatado o mesmo resultado encontrado.

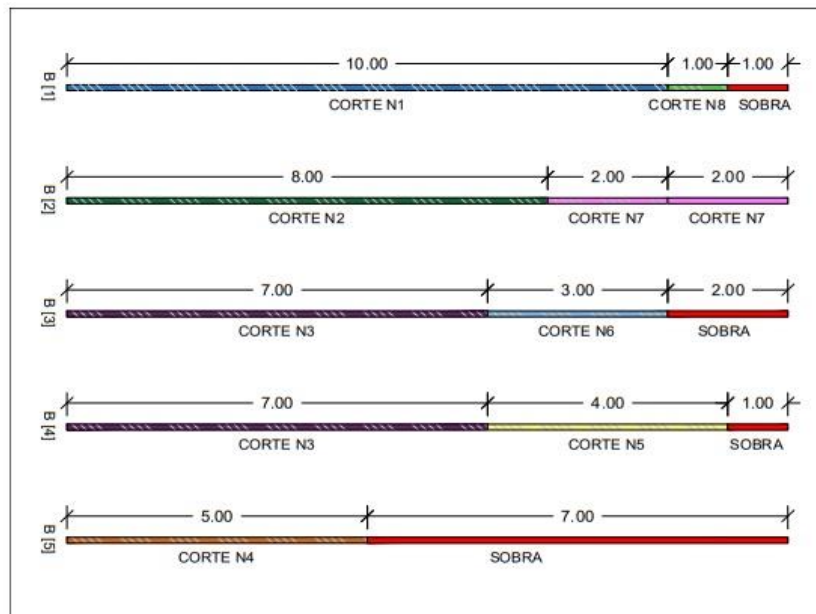
#### 2.2.5 Maiores Menores Decrescentes

A heurística Maiores e Menores Decrescentes (MMD) parte da ideia de, após a ordenação decrescente dos itens, criam-se 2 ponteiros, um para percorrer do maior para o menor e o segundo para percorrer do menor para o maior.

No próximo passo, é realizado um loop em que o primeiro ponteiro criado, verifica-se o item é  $\leq$  ao espaço restante do pacote; se sim, é adicionado o item maior no pacote (bin) e subtrai o item do espaço restante. Caso contrário, é realizado outro loop com o segundo ponteiro criado verificando-se o item menor é  $\leq$  espaço restante do pacote; se sim, adiciona o item no pacote (bin); se não, verifica-se a demanda dos itens foi atendida. Caso contrário, repete-se todo o procedimento inicial (ALOISE, 1992).

Uma ilustração da heurística MMD, em que se utiliza da mesma demanda de itens ilustrada, anteriormente, na Figura 5. Na Figura 7, a quantidade de bins necessário e seus cortes, gerados a partir da heurística.

**Figura 7** - Quantidade de bins requeridos para o MMD.



**Fonte:** Autoria Própria (2019).

### 2.3 APLICAÇÃO DA HEURÍSTICA MMD AO PCVSA

Nesta seção, aplicam-se os procedimentos heurísticos para a construção do algoritmo MMD.

### 2.3.1 Procedimentos Heurísticos

As heurísticas, para um problema de otimização combinatória, têm como objetivo encontrar uma boa solução; em alguns casos uma solução ótima, com bom tempo computacional (polinomial). As heurísticas podem ser construtivas, de melhoramento ou compostas (RIOS, 2017).

**Construtiva:** constrói uma solução em um processo iterativo, conforme um conjunto de condições ou regras, as quais são estabelecidas, por exemplo:

- Escolha do ponto inicial: inicialização;
- Critério de escolha do elemento seguinte a ser acrescentado à solução atual;
- Seleção da posição em que o novo elemento vai ser inserido.

**De melhoramento:** as heurísticas de melhoramento iniciam com uma solução sub ótima, na sequência realiza-se pequenas alterações iterativas para melhorar esta solução. O critério de parada dessas heurísticas é quando nenhuma outra alteração melhora o resultado atual; no entanto, não há garantias de que seja a melhor solução para o problema.

A solução inicial pode ser obtida:

- Aleatoriamente;
- Com uma heurística construtiva.

Se a solução viável for obtida a partir de uma heurística construtiva, estamos diante de uma heurística composta.

**Compostas:** tem uma fase construtiva e uma fase de melhoramento.

Em seguida, o modelo matemático para a heurística MMD, adaptado de Cherri (2006).

#### Dados de entrada

- $C$ : capacidade do Bin;
- $m$ : quantidade de tipos diferentes de itens;
- $L_i$ : comprimento do item do tipo  $i$ ,  $i = 1, \dots, m \quad \forall L \leq C$ ;
- $d_i$ : quantidade do item do tipo  $i$ ,  $i = 1, \dots, m$ .

#### Variáveis



- $LTD = \sum_{i=1}^m L_i d_i$  comprimento total das demandas;
- $D = \sum_{i=1}^m d_i$ , quantidade total da demanda;
- $b$ : quantidade de bins;
- $ER = C - L_i$ , espaço restante do Bin.

### Dados de saída

- $BG$ : quantidade total de bins gerados;
- $LB = \frac{LTD}{C}$ , limite inferior (*Lower Bounds*);
- $LSG$ : comprimento total das sobras geradas;
- $PC_{bLi}$ : plano de corte, para cada  $b$  de uma lista de comprimentos  $L_i$ .

### 2.3.2 Construção do algoritmo MMD

Nesta seção, é apresentado o pseudocódigo do algoritmo MMD e o detalhamento linha a linha.

```

Entrada: MMD (itens [], CAPACIDADE){
var numeroBins = 0; var itemMaior = 0; var itemMenor;
1 ordenaDecrescente(itens);
2 enquanto(itens.tamanho() > 0){
3     var espacoRestante [numeroBins] = CAPACIDADE;
4     enquanto(itens[itemMaior] <= espacoRestante[numeroBins]){
5         var novoEspaco = espacoRestante[numeroBins] - itens[itemMaior];
6         espacoRestante[numeroBins] = novoEspaco;
7         Itens.remove(itemMaior);
8         se(itens.tamanho() == 0)
9             retorne numeroBins++;
10    }
11    itemMenor = itens.tamanho() - 1;
12    enquanto(itens[itemMenor] <= espacoRestante[numeroBins]){
13        var novoEspaco = espacoRestante - itens[itemMenor];
14        espacoRestante[numeroBins] = novoEspaco;
15        Itens.remove(itemMenor);
16        se(itens.tamanho() == 0)
17            retorne numeroBins++;
18        senao
19            itemMenor--;
20    }
21    numeroBins++;
22 }

```

O algoritmo MMD recebe como parâmetro uma lista de itens e a capacidade do bin com o qual ele vai otimizar. Seguindo para as variáveis iniciais, tem-se o número de bins, em que ao final da otimização irá informar a quantidade total de bins utilizados para a determinada demanda de itens.

Na sequência, criam-se duas variáveis para controle dos índices dos itens maiores e menores. Na linha 1, chama-se uma função para ordenar os itens em ordem decrescente, com algoritmos de ordenação, sejam eles, *MergeSort*, *QuickSort* ou *HeapSort*. Na linha 2, tem-se o primeiro laço, em que a condição se torna falsa quando não há mais itens na lista.

Na linha 3, cria-se uma variável do tipo lista chamada espaço Restante, em que inicializa-se com a CAPACIDADE para cada bin criado. Na linha 4, o segundo laço, em que verifica-se o item maior é menor que o espaço restante, a condição se torna falsa quando não couber mais itens no bin.

Na linha 5, 6 e 7, realiza-se o cálculo para o novo espaço do bin e na sequência o item é removido da lista de itens. Vale destacar que não é necessário o incremento do item maior, visto que quando é removido, o próximo item torna-se o primeiro.

Na linha 8, verifica-se que a demanda de itens chegou a 0, pois pode ocorrer em alguns casos que só precisará percorrer do maior para o menor. Um exemplo seria de um único item na lista de demandas, caso aconteça o algoritmo encerra e retorna o incremento do número de bins. Assim, caso a demanda de itens não seja 0, na linha 11, a variável itemMenor atribui-se a quantidade total de itens -1.

Na linha 12 até 17, repete-se o mesmo laço da linha 4 com uma única diferença, em vez de percorrer os itens maiores, troca-se para a variável de itens menores. Na linha 18, utiliza-se para decrementar a variável itemMenor para percorrer os itens (demandas) do menor para o maior. Por fim, na linha 21, incrementa-se o valor do número de bins.

### 3 TESTES COMPUTACIONAIS

Neste capítulo, são apresentados os testes computacionais realizados para analisar o desempenho da heurística MMD para o PCVSA. Os códigos desenvolvidos para os testes computacionais foram desenvolvidos na IDE *IntelliJ IDEA* e utilizou-se da linguagem de programação Java, pois o desenvolvimento do aplicativo *Android* é baseado em Java.

Os testes foram realizados em um microcomputador com processador Intel(R) *Core i5-5200U CPU dual-core a 2,20 a 2.70 GHz* de frequência, com memória RAM de 8 GB DDR3, placa de vídeo Nvidia Geforce 920m com 2 GB de memória VRAM e sistema operacional Windows 10 versão 1809.

Na sequência, são apresentados os resultados obtidos pela heurística implementada MMD em comparação com as heurísticas mencionadas por este trabalho.

Para os testes, foram demandados uma lista com 20, 50, 100, 1000 e 5000 itens. Todos eles gerados de forma aleatória com números entre 1,0 a 11, 99, simulando itens reais de acordo com o comprimento dos vergalhões. Na sequência, são apresentadas as tabelas com resultados obtidos para as listas demandadas.

Na Tabela 1, uma demanda de 20 itens, a soma total do comprimento desses itens foi de 109,74 e seu Limite Inferior (LI) foi de 9,14 bins. Neste caso, pode-se arredondar para 10 bins.

**Tabela 1** – Resultados obtidos com a demanda de 20 itens.

<i>Heurísticas</i>	<i>Qtde. Bins</i>	<i>Sobra total</i>
<i>FFD</i>	10	10,19 m
<i>BFD</i>	10	10,24 m
<i>MMD</i>	10	10,14 m

**Fonte:** Autoria Própria (2019).

Vale destacar que as soluções obtidas, apresentadas na Tabela 1, por este caso específico, utilizaram-se da mesma quantidade de Bins em que o LI foi definido, sendo assim, uma solução ótima. Pode-se observar também, uma mínima vantagem da heurística implementada por este trabalho em relação a menor quantidade de sobras.

Na Tabela 2, uma demanda de 50 itens, a soma total do comprimento desses itens foi de 303,36 e seu LI é de 25,28 bins. Neste caso, pode-se arredondar para 26 bins.

**Tabela 2** - Resultados obtidos com a demanda de 50 itens.

<i>Heurísticas</i>	<i>Qtde. Bins</i>	<i>Sobra total</i>
<i>FFD</i>	27	20,59 m
<i>BFD</i>	27	20,64 m
<i>MMD</i>	27	20,49 m

**Fonte:** Autoria Própria (2019).

Na Tabela 3, uma demanda de 100 itens, a soma total do comprimento desses itens foi de 647,65 e seu LI é de 53,97 bins. Neste caso, pode-se arredondar para 54 bins.

**Tabela 3** - Resultados obtidos com a demanda de 100 itens.

<i>Heurísticas</i>	<i>Qtde. Bins</i>	<i>Sobra total</i>
<i>FFD</i>	58	48.13 m
<i>BFD</i>	58	48.33 m
<i>MMD</i>	58	47.94 m

**Fonte:** Autoria Própria (2019).

Na Tabela 4, uma demanda de 1000 itens, a soma total do comprimento desses itens foi de 6.550,29 e seu LI foi de 545,85 bins. Neste caso, pode-se arredondar para 546 bins.

**Tabela 4 – Resultados obtidos com a demanda de 1000 itens.**

<i>Heurísticas</i>	<i>Qtde. Bins</i>	<i>Sobra total</i>
<i>FFD</i>	557	132,12 m
<i>BFD</i>	557	133,71 m
<i>MMD</i>	557	130,09 m

Fonte: Autoria Própria (2019).

Na Tabela 5, uma demanda de 5000 itens, a soma total do comprimento desses itens foi de 32.826,78 e seu LI é de 2.735,56 bins. Neste caso, pode-se arredondar para 2.736 bins.

**Tabela 5 - Resultados obtidos com a demanda de 5000 itens.**

<i>Heurísticas</i>	<i>Qtde. Bins</i>	<i>Sobra total</i>
<i>FFD</i>	2797	730,70 m
<i>BFD</i>	2797	737,21 m
<i>MMD</i>	2797	722,50 m

Fonte: Autoria Própria (2019).

Em seguida, um teste com 20 instâncias com 5 mil itens cada, em que tem-se, na Tabela 6, a quantidade de Bins obtidos, a soma total do comprimento da demanda de cada instância e o LI. Na Tabela 7, a sobra total de cada instância.

**Tabela 6 – Resultados obtidos para 20 instâncias com 5 mil itens.**

<i>Classes</i>	<i>MMD</i>	<i>FFD</i>	<i>BFD</i>	<i>Soma Total</i> <i>(metros)</i>	<i>Limite Inferior (LI)</i>
<i>C1</i>	2745	2745	2745	32.524,12	2.710,34

C2	2749	2749	2749	32.534,52	2.711,21
C3	2772	2772	2772	32.838,63	2.736,55
C4	2734	2734	2734	32.340,70	2.695,05
C5	2759	2759	2759	32.568,53	2.714,04
C6	2730	<b>2728</b>	<b>2728</b>	32.363,79	2.696,98
C7	2800	2800	2800	32.741,82	2.728,48
C8	2759	2759	2759	32.613,87	2.717,82
C9	2769	2769	2769	32.734,57	2.727,88
C10	2735	2735	2735	32.356,69	2.696,39
C11	2778	2778	2778	32.708,65	2.725,72
C12	2750	2750	2750	32.495,92	2.707,99
C13	2744	2744	2744	32.471,37	2.705,94
C14	2757	2757	2757	32.612,84	2.717,73
C15	2835	2835	2835	33.115,29	2.759,60
C16	2755	2755	2755	32.586,27	2.715,52
C17	2786	2786	2786	32.797,45	2.733,12
C18	2771	2771	2771	32.798,94	2.733,24
C19	2751	<b>2750</b>	<b>2750</b>	32.571,01	2.714,25
C20	2762	2762	2762	32.643,89	2.720,32
<b>MÉDIA</b>	<b>2762,05</b>	<b>2761,9</b>	<b>2761,9</b>	<b>32.620,94</b>	<b>2.718,40</b>

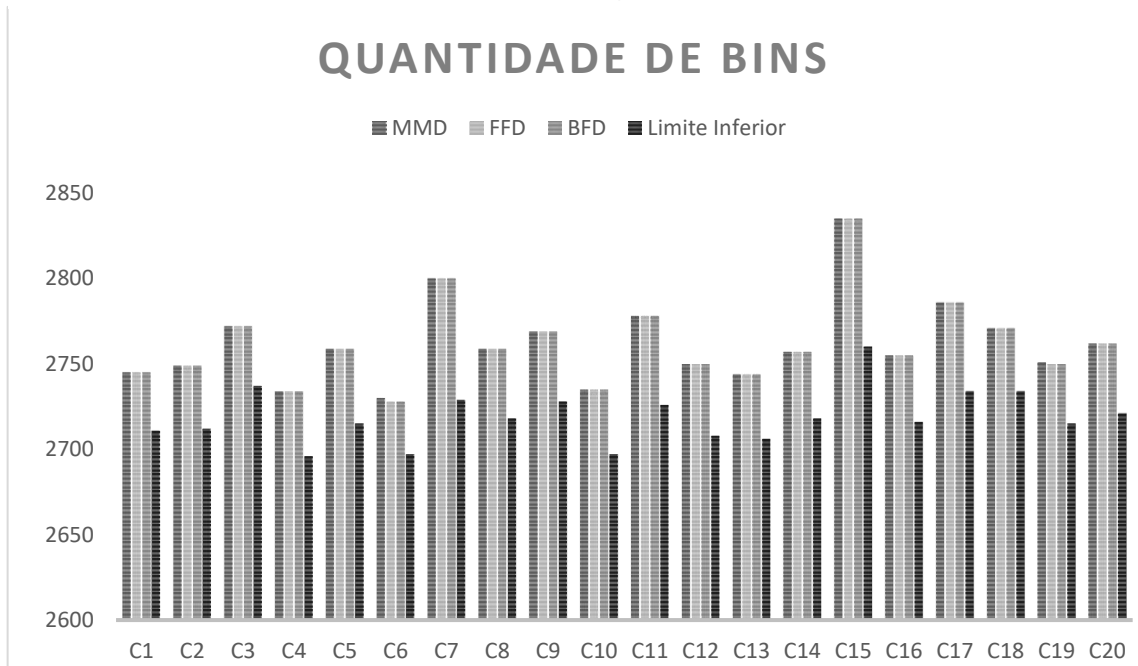
Fonte: Autoria Própria (2019).

**Tabela 7 – Sobra Total em metros, 20 instâncias com 5 mil itens.**

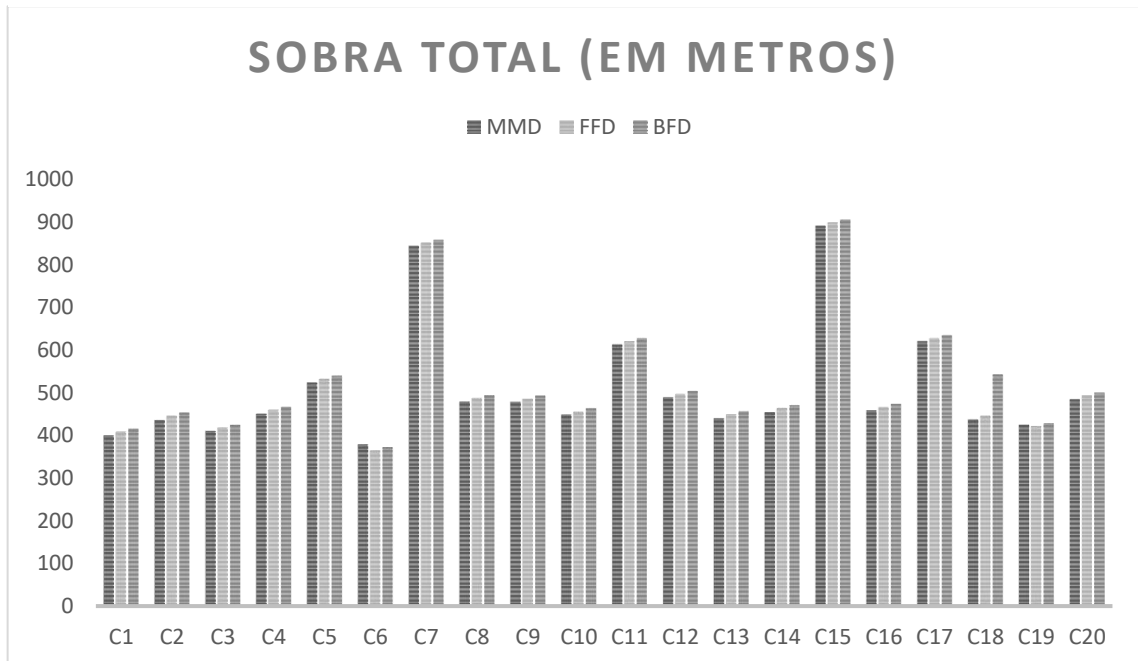
<i>Classes</i>	<i>MMD</i>	<i>FFD</i>	<i>BFD</i>
<hr/>			

C1	<b>400,49</b>	409,28	415,86
C2	<b>435,38</b>	446,51	453,46
C3	<b>410,39</b>	419,20	425,36
C4	<b>450,46</b>	460,63	467,28
C5	<b>523,68</b>	533,07	539,46
C6	379,29	<b>365,28</b>	372,19
C7	<b>842,68</b>	851,60	858,16
C8	<b>478,78</b>	487,78	494,12
C9	<b>478,27</b>	486,88	493,42
C10	<b>448,14</b>	456,79	463,29
C11	<b>612,22</b>	620,91	627,33
C12	<b>488,76</b>	497,58	504,06
C13	<b>440,35</b>	449,87	456,62
C14	<b>454,57</b>	464,53	471,14
C15	<b>890,05</b>	899,02	904,70
C16	<b>458,57</b>	467,11	473,71
C17	<b>619,86</b>	628,16	634,54
C18	<b>437,34</b>	446,57	543,04
C19	424,79	<b>422,37</b>	428,97
C20	<b>484,28</b>	493,82	500,09
<b>MÉDIA</b>	<b>507,91</b>	515,35	526,34

Fonte: Autoria Própria (2019).

**Gráfico 1 – Quantidade de bins gerados em cada instância.**

Fonte: Autoria Própria (2019).

**Gráfico 2 – Sobra total em cada instância, quanto menor, melhor.**

Fonte: Autoria Própria (2019).



Pode-se observar no Gráfico 1, que a heurística MMD igualou-se em praticamente todas as instâncias a quantidade de bins gerados, exceto no C6 e C19. No Gráfico 2, a representação total da sobra acumulada, isto é, cada pedaço que não foi utilizado no plano de corte para cada instância testada, pois quanto menor for este índice, melhor foi aproveitado. Destaca-se a heurística MMD, uma vez que teve menores perdas comparados com as heurísticas testadas, exceto na classe de instância C6 e C19.

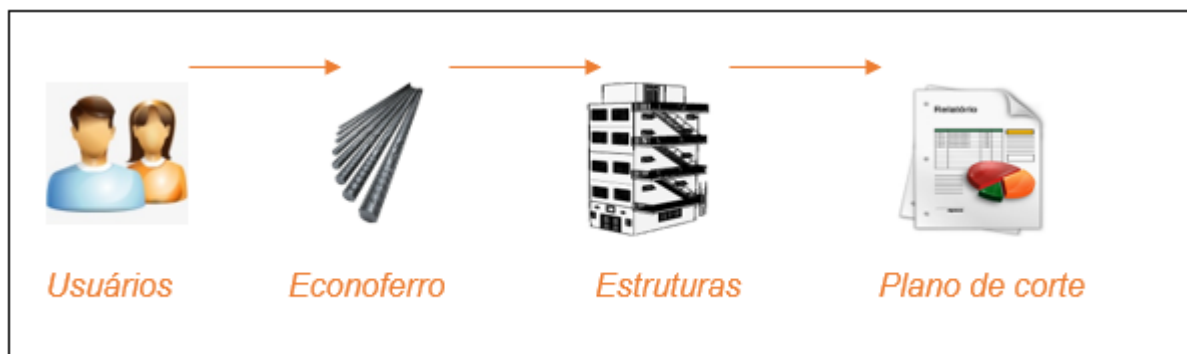
## 4 DESENVOLVIMENTO DO APLICATIVO MÓVEL ANDROID

### 4.1 VISÃO GERAL

O aplicativo Econoferro visa otimizar o Plano de Corte de Vergalhões com Sobras Aproveitáveis (PCVSA) para obter uma solução satisfatória em tempo computacional eficiente (polinomial) e reduzir as perdas de materiais, facilitando ao usuário final a geração de planos de corte das estruturas. Assim, o aplicativo visa também a boa usabilidade e facilidade para inserção dos dados, trazendo benefícios como aprendizado rápido, tempo e dinheiro.

O aplicativo mantém sua base de dados em nuvem, possibilitando, assim, segurança e disponibilidade dos dados. A Figura 8, apresenta uma visão geral do aplicativo Econoferro.

**Figura 8 – Visão geral do Econoferro.**



Fonte: Autoria própria (2019).

### 4.2 ENGENHARIA DE SOFTWARE

A engenharia de *software* é um campo (disciplina) da computação em que estudam-se todos os aspectos da produção de *software*, sejam eles especificação, desenvolvimento, validação e evolução de *software* (Sommerville, 2011).

Engenharia de *software* está profundamente incorporado em praticamente todos os aspectos de nossas vidas e, conseqüentemente, o número de pessoas interessadas nos recursos e nas funções oferecidas por determinada aplicação (Pressman e Maxim, 2016).

Um processo de *software* é importante para desenvolver aplicações sem ambiguidade ou surpresas no resultado da aplicação. Pressman e Maxim (2016) definiu processo de *software* da seguinte forma “Processo é um conjunto de atividades, ações e tarefas realizadas na criação de algum artefato”.

Outro fator importante na engenharia de *software* é a metodologia do processo, em que estabelece uma base para um processo de engenharia de *software* por meio da identificação de um pequeno número de atividades metodológicas aplicáveis a todos os projetos de *software*, independentemente de tamanho ou complexidade (Pressman e Maxim, 2016).

#### **4.2.1 Engenharia de requisitos**

A engenharia de requisitos é a descrição do que o sistema deve fazer, os serviços que oferecem e as restrições a seu funcionamento, Sommerville (2011) definiu da seguinte forma, “O processo de descobrir, analisar, documentar e verificar serviços e restrições é chamado de engenharia de requisitos”

Assim como um sistema, o aplicativo Econoferro desenvolvido por este trabalho, utilizou-se da engenharia de *software* para seu desenvolvimento. Para isso, o primeiro passo foi utilizar a engenharia de requisitos para o processo de descobrir, analisar, documentar e verificar serviços e restrições.

##### Requisitos funcionais (casos de uso)

São declarações de serviços ou funções que o aplicativo deve fornecer, de como deve reagir a entradas específicas e de como deve se comportar em determinadas situações. Para verificar o detalhamento dos requisitos funcionais levantados utilizados, consultar o APÊNDICE B – documento de requisitos Aplicativo Econoferro.

##### Requisitos não funcionais

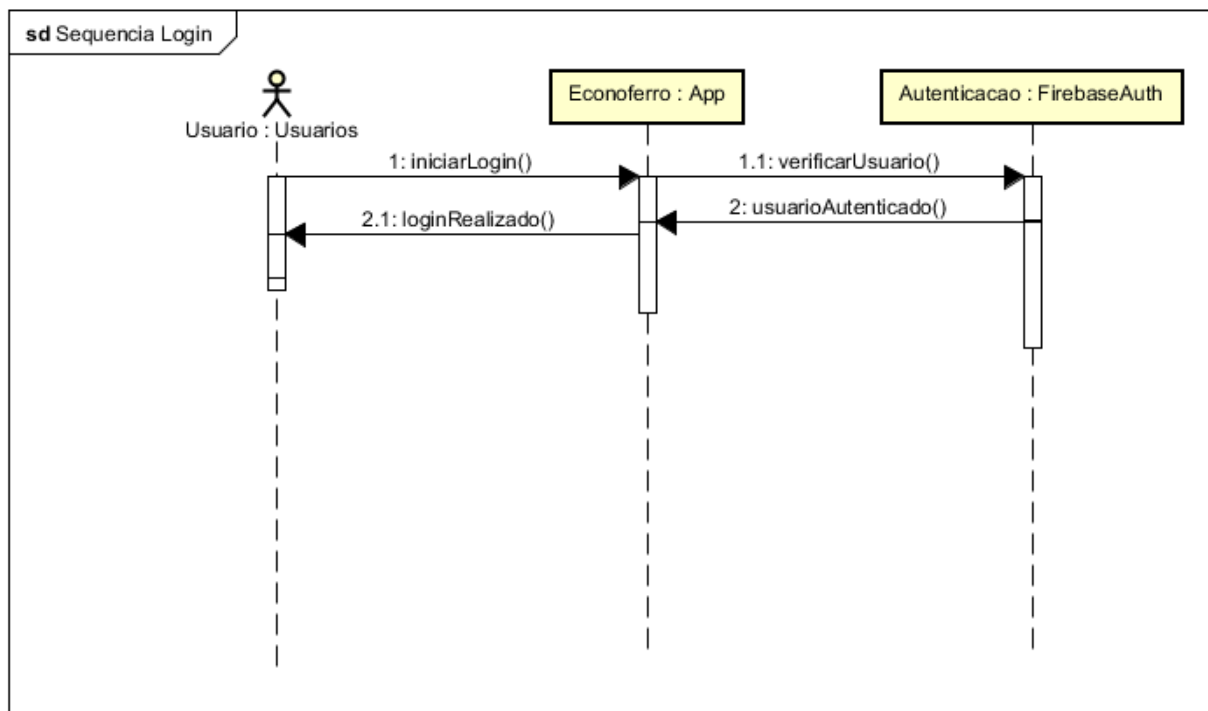
São restrições aos serviços ou funções oferecidas pelo aplicativo. Podem ser restrições de tempo, disponibilidade, normas e arquiteturas computacionais. Para verificar os requisitos não funcionais levantados, consultar o APÊNDICE B – documento de requisitos Aplicativo Econoferro.

#### 4.2.2 Modelagem UML do aplicativo Econoferro

Para a modelagem do aplicativo Econoferro, foi utilizado a *Unified Modeling Language* (UML), em que se utilizou diagramas para melhor compreensão do projeto como um todo.

O objetivo da modelagem UML é fornecer diversas visões do sistema ou aplicativo a ser modelado, analisando-o e modelando-o sob diversos aspectos. Sendo assim, visualizando o sistema como um todo, permitindo que cada diagrama complemente os outros (GUEDES, 2009). Na Figura 9, uma ilustração de um diagrama de seqüência, em que o usuário realiza o login com sucesso.

**Figura 9** – Diagrama de Sequência, sucesso Login.



**Fonte:** Autoria Própria (2019).

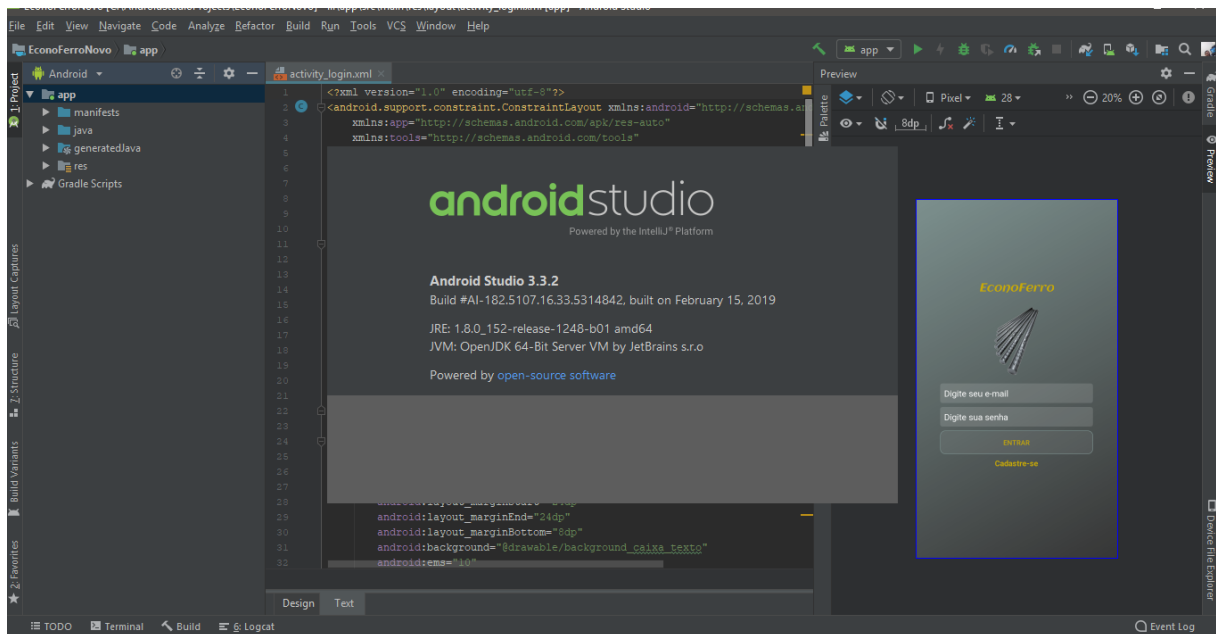
Para verificar a modelagem de classes do aplicativo Econoferro, consultar o APÊNDICE A – Diagrama de classes Econoferro.

### 4.2.3 Implementação

Para o desenvolvimento do aplicativo, foi utilizado o *Android Studio* na sua versão 3.3.2, um ambiente de desenvolvimento integrado, do inglês *Integrated Development Environment (IDE)*, que aumenta a produtividade do programador através de ferramentas e recursos de suma importância, como por exemplo um emulador *Android* integrado. Outro recurso importante é o plugin do *GitHub* para versionamento do aplicativo desenvolvido, trazendo a automação do *Git* para armazenar o projeto na nuvem.

Na Figura 10, é apresentado a interface *IDE Android Studio*.

**Figura 10** – Interface do Android Studio



**Fonte:** Autoria própria (2019).

Para a base de dados, foi utilizado o *Firebase*, uma plataforma de desenvolvimento de aplicativos para dispositivos móveis e web, em que existe um conjunto de tecnologias e serviços; como por exemplo, o banco de dados em tempo real utilizado neste trabalho e autenticação de usuários.

Para a codificação do Econoferro, foi utilizada a *IDE Android Studio* com a linguagem de programação Java. A interface gráfica da aplicação foi desenvolvida através do recurso arraste e solta do *Android Studio*. O Java foi utilizado na elaboração

da lógica de funcionamento, como a utilização da heurística MMD para o módulo otimizador, bem como serviços e conexões com os servidores do *Firebase*.

O Econoferro utilizou o banco de dados orientado a documentos em real time do *Firebase*, onde foram criadas 6 coleções principais, a saber:

- *Usuários*: coleção para armazenar e manipular os dados dos usuários;
- *Obras*: coleção utilizada para melhor organização das estruturas;
- *Estruturas*: coleção utilizada para gravar as estruturas que irão utilizar os vergalhões;
- *Demandas*: coleção para armazenar os itens (vergalhões) e suas devidas quantidades, separados por categoria.
- *Planos de corte*: coleção utilizada para armazenar os diferentes planos de corte obtidos pela heurística MMD;
- *Estoque Vergalhões*: coleção que tem como armazenamento, os diferentes tipos de retalhos gerados pelo corte da heurística MMD.

A aplicação Econoferro tem como característica sua interface simplista, intuitiva e ágil, em que o profissional da construção civil terá acesso fácil ao plano de corte através da geração de arquivo em formato pdf.

A Figura 11 é apresentado a tela de login de usuário, em que deverão ser inserido os dados corretamente, para que assim a autenticação do usuário seja validada pelo servidor de autenticação da plataforma *Firebase*. Caso o usuário ainda não tenha uma conta no aplicativo, clicar no nome *cadastre-se*, em seguida o usuário é redirecionado para tela de cadastro ilustrado na Figura 12, em que é necessário o nome do usuário, e-mail e senha para ter acesso ao aplicativo.

**Figura 11** – Interface de login.

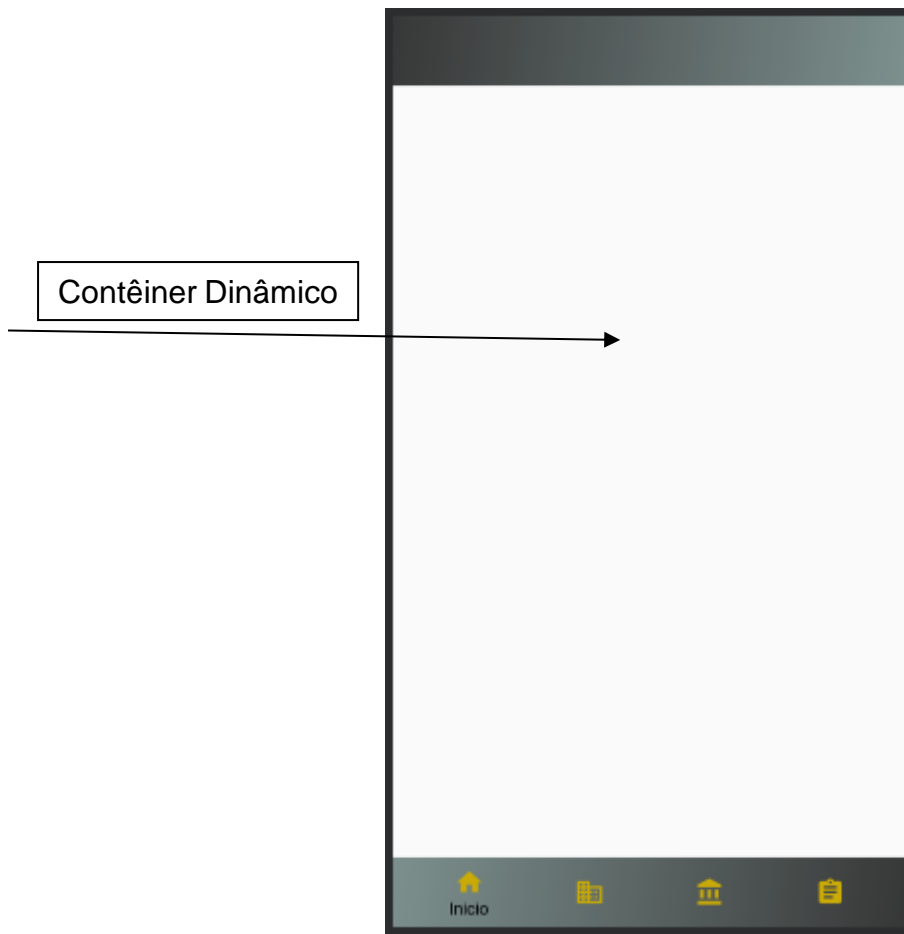
**Fonte:** Autoria própria (2019).

**Figura 12** – Tela de Cadastro.

**Fonte:** Autoria própria (2019).

Na Figura 13, é apresentada a tela principal, logo após o usuário realizar o login no aplicativo. Assim, apresenta-se um menu com as seguintes opções:

- *Início:* em que o usuário visualizará as estruturas e os planos de cortes adicionadas recentemente;
- *Obra:* Usuário adiciona obras que serão utilizadas para melhor organização do plano de corte.
- *Estrutura:* função essencial para geração do plano de corte, em que o usuário irá inserir os dados corretamente para não ocorrer anomalias na geração deste plano;
- *Plano de corte:* em que o usuário selecionará a obra que vai ser utilizada para o plano de corte e suas respectivas estruturas demandas, para que seja possível obter o plano em formato *pdf*.

**Figura 13 – Tela principal.**

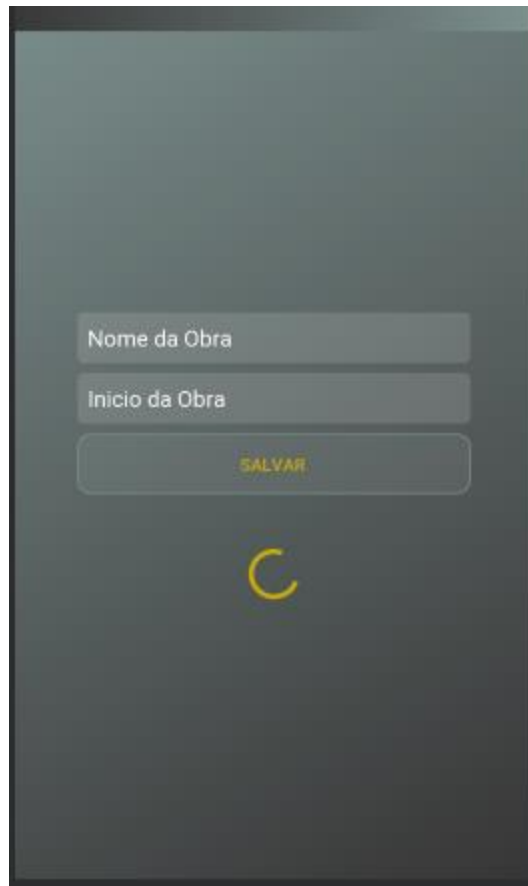
**Fonte:** Autoria própria (2019).

Na tela principal, em que observa-se uma grande tela branca, é intitulado de contêiner de informações dinâmicas. Cada menu será invocado dinamicamente dentro desse contêiner. Os dados obtidos são recuperados do banco de dados *Firebase*.

Na Figura 14, é apresentada a tela para adicionar uma nova obra, em que preencherá os campos *Nome da Obra* e a data do *Início da Obra*. Vale ressaltar que a inserção do elemento Obra é de suma importância para adquirir o plano de corte, visto que o aplicativo requer uma obra com demandas estruturais para realizar o mapeamento do corte.



**Figura 14** – Tela para adicionar obra.

A imagem mostra uma interface de usuário para adicionar uma obra. O fundo é escuro cinza. No topo, há um campo de texto cinza claro com o rótulo "Nome da Obra". Abaixo dele, há outro campo de texto cinza claro com o rótulo "Inicio da Obra". Logo abaixo dos campos, há um botão retangular com o texto "SALVAR" em letras maiúsculas e cor amarela. No centro da tela, há um ícone de uma seta curva amarela, indicando uma ação de confirmação ou avanço.

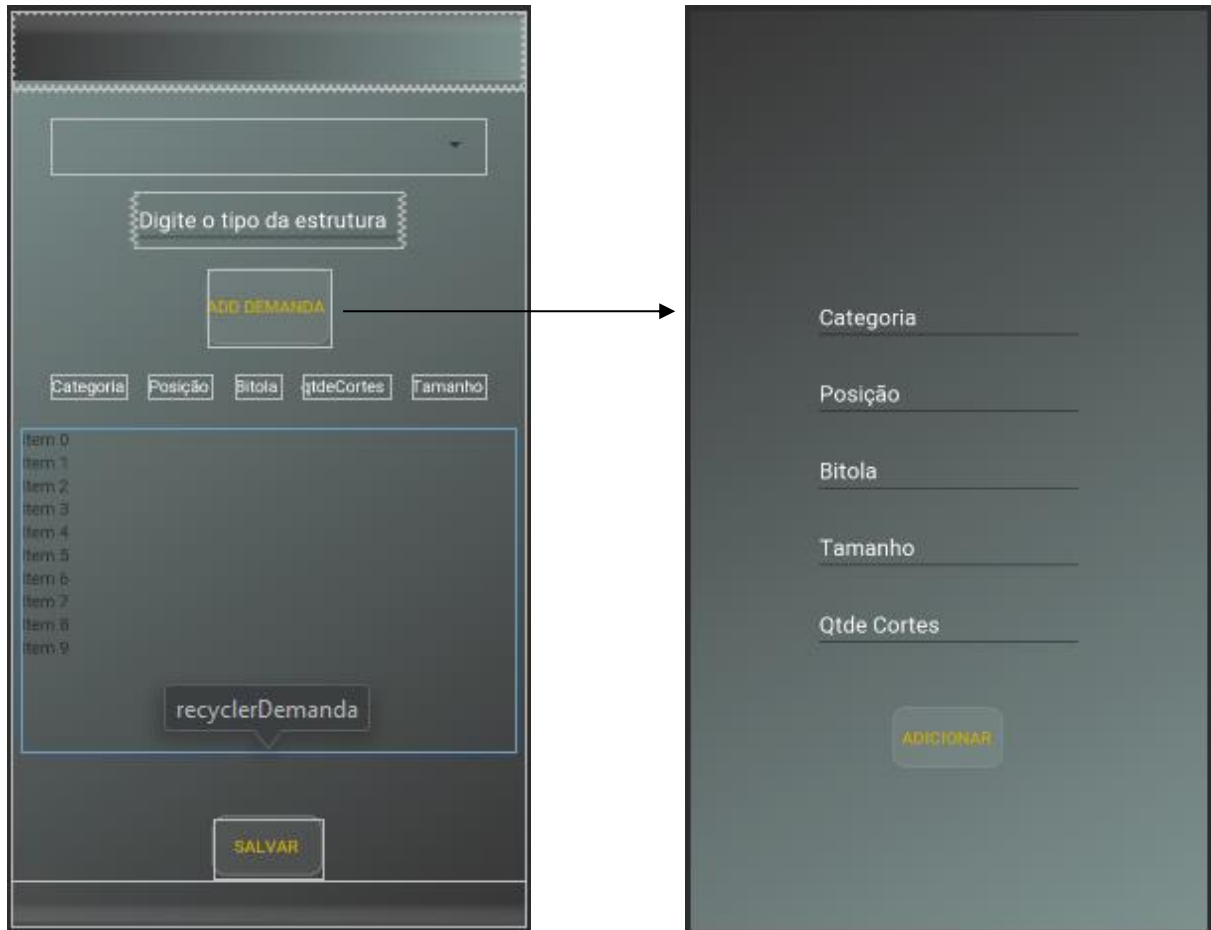
**Fonte:** Autoria Própria (2019).

Na Figura 15, é apresentada ao usuário a tela para adicionar uma nova estrutura, sendo a função crucial para determinar o plano de corte, inserir os dados com devida atenção, para não provocar anomalias no plano de corte.

No primeiro campo, temos um selecionador de obras, em que o usuário irá selecionar a obra em que a estrutura vai ser adicionada. Na sequência, o tipo da estrutura exemplo: viga, coluna, laje etc.

Ao finalizar esses passos, clicar no botão ADD DEMANDA, o usuário terá sua tela alterada para tela cheia para a inserção da demanda, em que será necessário inserir os seguintes campos: Categoria, Posição, Bitola, Quantidade de Cortes e o Tamanho. Assim que o usuário adicionar essa demanda, pode-se observar a inserção de uma lista de itens. Ao finalizar o processo de inserção de demandas da estrutura escolhida, o usuário poderá salvar a estrutura.

**Figura 15** – Tela para adicionar uma nova estrutura civil.



**Fonte:** Autoria própria (2019).

Ao finalizar a inserção das estruturas e demandas de vergalhões da obra, o passo seguinte é abrir a função de plano de corte, em que o usuário seleciona a obra que foram inseridas as estruturas com suas demandas de corte.

Na Figura 16, é apresentada a tela para determinar o plano de corte. O usuário seleciona a obra em qual as demandas foram inseridas e em seguida clica no botão GERAR PLANO DE CORTE. Neste momento, a Heurística MMD realiza os procedimentos para otimizar e mapear o plano de corte das estruturas, retornando para o usuário um arquivo pdf com as quantidades de vergalhões necessárias para atender as demandas, as sobras (perda) e o cortes necessários em cada vergalhão.

**Figura 16** - Tela para gerar o plano de corte em .pdf.



**Fonte:** Autoria própria (2019).

#### **4.2.4 Verificação e Validação**

A verificação do aplicativo foi realizada através de um checklist no documento dos requisitos, ou seja, o que foi planejado, foi realmente implementado.

A validação do aplicativo foi realizado em um teste real, em que o plano de corte foi obtido a partir de uma lista de demandas de um projeto estrutural de um dos pavimentos com lajes e vigas cedidos por uma empresa privada.

Para verificar as demandas, consultar o ANEXO A – Lista de demandas LAJE e ANEXO B – Lista de demandas Vigas.

Na Tabela 8, são apresentados os resultados obtidos pelo aplicativo Econoferro em comparativo com os métodos adotados pela empresa.

**Tabela 8** – Quantidade de vergalhões para cada demanda por bitola.

<i>BITOLAS</i>	<i>DEMANDAS</i>							
	5.0	6.3	8.0	10.0	12.5	16.0	20.0	25.0
<i>ECONOFERRO</i>	125	70	2565	124	913	251	170	42
<i>EMPRESA</i>	122	69	2661	132	871	229	152	42

**Fonte:** Aatoria Própria (2019).

Pode-se observar que a quantidade de vergalhões das bitolas 5.0 e 6.3 ficaram próximas, com vantagem para o método adotado pela empresa; já nas bitolas 8.0 e 10.0 têm-se uma boa vantagem do aplicativo Econoferro, chegando a uma economia de 96 vergalhões na bitola 8.0 e 8 vergalhões na bitola 10.0. Nas bitolas 12.5, 16.0 e 20.0 têm-se vantagem para o método adotado pela empresa.

A vantagem do aplicativo Econoferro está no mapeamento do plano de corte em relação a empresa, pois ainda pratica de forma artesanal o mapeamento do plano de corte, o que leva muito tempo.

Através do aplicativo Econoferro é possível visualizar todo plano de corte gerado a partir das demandas cedidas, conforme exemplo disponível através do link: <<https://drive.google.com/open?id=1783td5kCKx4tkwu9t77OBD4NQMTCLA-I>>.

Acesso em: 18 abr. 2019.

## 5 CONSIDERAÇÕES FINAIS

Neste trabalho, foi realizado um estudo sobre o problema de corte e empacotamento de uma única dimensão e suas heurísticas para resolução do problema, de forma que se obtenha o melhor aproveitamento das sobras. Foi implementada a heurística MMD na linguagem de programação Java, em que os procedimentos utilizados tiveram como objetivo a minimização das perdas e o mapeamento do plano de corte.

Após a realização dos testes, observou-se vantagem da heurística implementada quando comparado com outras heurísticas existentes na literatura, tendo em vista que a heurística utilizada aproveitou da melhor forma as sobras. Vale ressaltar que não foi necessário a comparação das heurísticas em termos de tempo computacional, visto que todas as heurísticas testadas nas classes de 5 mil itens obtiveram em tempo computacional abaixo de 1 segundo, tornando o tempo irrelevante para comparações.

Foram realizados dois tipos de testes, sendo um em ambiente simulado e o outro em ambiente real. Para os testes em ambiente simulado, observaram-se bons resultados em comparação com as heurísticas estudadas da literatura. Já para os testes em ambiente real, foi apresentado o aplicativo Econoferro ao setor técnico de uma empresa privada de construção civil, em que foram inseridos os dados de vigas e lajes de um dos pavimentos, conforme projeto cedido. Neste último, o aplicativo foi avaliado positivamente pela facilidade de uso, tempo computacional e bons resultados obtidos.

Vale mencionar que o aplicativo encontra-se em fase de teste no presente momento deste trabalho e ainda não foi disponibilizado na loja de aplicativos GooglePlay, pois falta realizar o registro junto ao INPI.

Como trabalhos futuros, o aperfeiçoamento do aplicativo Econoferro, nos aspectos design melhorando a usabilidade, no núcleo de otimização do plano de corte, visando melhorias na heurística MMD para o reaproveitamento das sobras.

## REFERÊNCIAS

- ALOISE, D. J. **Contribuição à solução do problema bin-packing**: formulações, relaxações e novos algoritmos aproximativos. 165 p. Tese (Doutorado Ciências em Engenharia de Sistemas e Computação) – Universidade Federal do Rio de Janeiro, Rio de Janeiro, 1992.
- ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6118**: Projeto de estruturas de concreto – Procedimento. 3. Ed. Rio de Janeiro, 2014.
- \_\_\_\_\_. **NBR 7480**: Aço destinado a armaduras para estruturas de concreto armado – Especificação. 2. ed. Rio de Janeiro, 2007.
- BAASE, S. **Computer Algorithms**: introduction to design and analysis. 2. ed. Boston: Addison-Wesley Longman, 1988.
- CHERRI, A. C. **O problema de corte de estoque com reaproveitamento das sobras de material**. 121 p. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2006.
- CORMEN, T. H. et al. **Algoritmos teoria e prática**. 3. ed. Rio de Janeiro: Elsevier, 2012.
- DÓSA G. **The Tight Bound of First Fit Decreasing Bin-Packing Algorithm Is  $FFD(I) \leq 11/9 OPT(I) + 6/9$** . Department of Mathematics, University of Pannonia, Veszprém, Hungary, 2007.
- GAREY, M. R.; JOHNSON, D.S. **Computers and intractability**: a guide to theory of NP – completeness. WH Free. Co. São Francisco, 1979.
- GUEDES, G. T. A. **UML 2**: uma abordagem prática. São Paulo: Novatec Editora, 2009.
- JOHNSON, D.S. **Near-Optimal Bin-Packing Algorithms**. Doctoral thesis, M.I.T, Cambridge, Mass., 1973.
- \_\_\_\_\_. et. al. WORST-CASE PERFORMANCE BOUNDS FOR SIMPLE ONE-DIMENSIONAL PACKING ALGORITHMS. **SIAM J. Comput**, Vol.3, 1974.

MINYI Y. **A SIMPLE PROOF OF THE INEQUALITY  $FFD(L) \leq 11/9 OPT(L) + 1, \forall L$  FOR THE FFD BIN-PACKING ALGORITHM.** Institute of Applied Mathematics, Academia Sinica, Beijing, 1991.

MORAES, M. C. B. **As perdas na construção civil: gestão do desperdício estudo de caso condomínio costa esmeralda.** 233 p. Dissertação (Mestrado em Engenharia Civil) – Universidade Federal de Santa Catarina, Florianópolis, 1997.

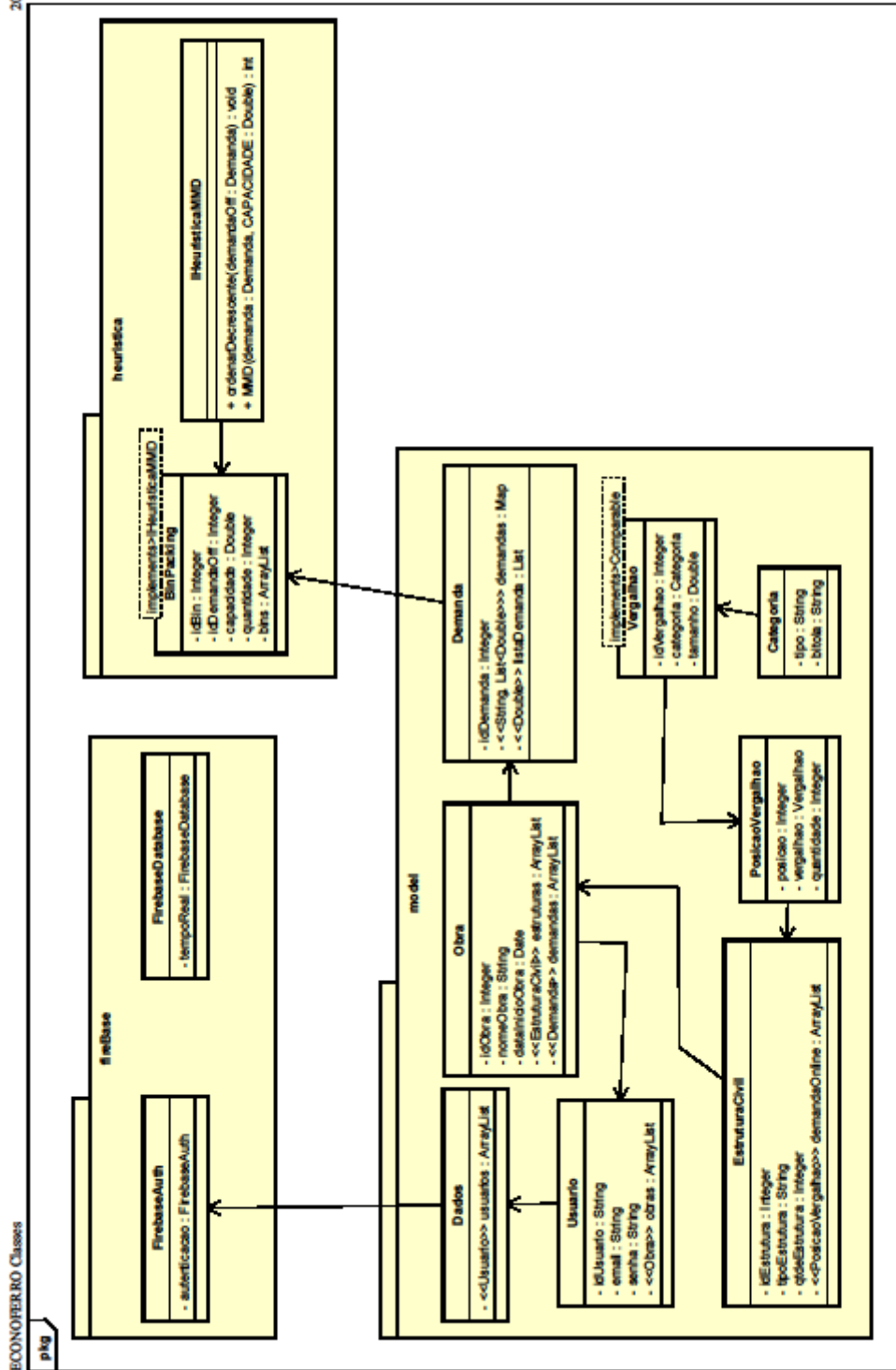
PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software: uma abordagem profissional.** 8. ed. São Paulo: AMGH Editora Ltda, 2016.

RIOS, A. R. **Problema de corte de estoque unidimensional com aproveitamento de sobras: resolução via meta-heurística GRASP.** 79 p. Dissertação (Mestrado em Matemática Aplicada) – Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas, São Paulo, 2017.

SOMMERVILLE, I. **Engenharia de Software.** 9. ed. São Paulo: Editora Pearson Education, 2011.

## APÊNDICE A – Diagrama de classes ECONOFERRO

2019/04/04





## APÊNDICE B – Documento de requisitos aplicativo ECONOFERRO

---

APLICATIVO ECONOFERRO – Trabalho de Graduação  
Graduando: ÍCARO ANDRÉ VIANA DE AZEVEDO  
Orientador: DARIO JOSÉ ALOISE  
Co-Orientadora: CÍCILIA RAQUEL MAIA LEITE

---

**Documento de Requisitos do Aplicativo  
*ECONOFERRO***

**Versão 1.0**



## Conteúdo

<b>1. INTRODUÇÃO</b>	<b>4</b>
1.1 VISÃO GERAL DO DOCUMENTO	4
1.2 CONVENÇÕES, TERMOS E ABREVIações	4
1.2.1 Identificação dos requisitos	4
1.2.2 Prioridades dos requisitos	4
<b>2. DESCRIÇÃO GERAL DO APLICATIVO</b>	<b>5</b>
2.1 ABRANGÊNCIA E APLICATIVOS RELACIONADOS	5
<b>3. REQUISITOS FUNCIONAIS (CASOS DE USO)</b>	<b>5</b>
3.1 CADASTRO	5
[RF001] Criar Obra	5
[RF002] Alterar Obra	5
[RF003] Excluir Obra	5
[RF004] Listar Obras	6
[RF005] Criar Estrutura	6
[RF006] Alterar Estrutura	6
[RF007] Excluir Estrutura	6
[RF008] Listar Estruturas	7
[RF009] Criar Demandas	7
[RF010] Alterar Demanda	7
[RF011] Excluir Demanda	8
[RF012] Listar Demandas	8
3.2 INTERFACE	8
[RF001] Plano de Corte	8
[RF002] Visualizar Plano de Corte	9
[RF003] Excluir Plano de Corte	9
<b>4. REQUISITOS NÃO-FUNCIONAIS</b>	<b>9</b>
[NF001] Usabilidade	9
[NF002] Desempenho	9
[NF003] Hardware e Software	10
<b>5. REFERÊNCIAS</b>	<b>10</b>

## 1. Introdução

Este documento especifica os requisitos do aplicativo *ECONOFERRO*, fornecendo aos desenvolvedores as informações necessárias para o projeto e implementação, assim como para a realização dos testes e homologação do aplicativo.

### 1.1 Visão geral do documento

Além desta seção introdutória, as seções seguintes estão organizadas como descrito abaixo.

1. **Seção 2 – Descrição geral do aplicativo:** apresenta uma visão geral do aplicativo, caracterizando qual é o seu escopo e descrevendo seus usuários.
2. **Seção 3 – Requisitos funcionais (casos de uso):** especifica todos os casos de uso do aplicativo, descrevendo os fluxos de eventos, prioridades, atores, entradas e saídas de cada caso de uso a ser implementado.
3. **Seção 4 – Requisitos não-funcionais:** especifica todos os requisitos não funcionais do aplicativo, divididos em requisitos de usabilidade, confiabilidade, desempenho, segurança, distribuição, adequação a padrões e requisitos de hardware e software.
4. **Seção 5 – Referências:** apresenta referências para outros documentos utilizados para a confecção deste documento.

### 1.2 Convenções, termos e abreviações

A correta interpretação deste documento exige o conhecimento de algumas convenções e termos específicos, que são descritos a seguir.

#### 1.2.1 Identificação dos requisitos

Por convenção, a referência a requisitos é feita através do nome da subseção onde eles estão descritos, seguidos do identificador do requisito, de acordo com a especificação a seguir:

*[nome da subseção. identificador do requisito]*

Por exemplo, o requisito funcional [Recuperação de dados.RF016] deve estar descrito em uma subseção chamada “Recuperação de dados”, em um bloco identificado pelo número [RF016]. Já o requisito não-funcional [Confiabilidade.NF008] deve estar descrito na seção de requisitos não-funcionais de Confiabilidade, em um bloco identificado por [NF008].

Os requisitos devem ser identificados com um identificador único. A numeração inicia com o identificador [RF001] ou [NF001] e prossegue sendo incrementada à medida que forem surgindo novos requisitos.

#### 1.2.2 Prioridades dos requisitos

Para estabelecer a prioridade dos requisitos, nas seções 4 e 5, foram adotadas as denominações “essencial”, “importante” e “desejável”.

5. **Essencial** é o requisito sem o qual o aplicativo não entra em funcionamento. Requisitos essenciais são requisitos imprescindíveis, que têm que ser implementados impreterivelmente.
6. **Importante** é o requisito sem o qual o aplicativo entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas, se não forem, o aplicativo poderá ser implantado e usado mesmo assim.
7. **Desejável** é o requisito que não compromete as funcionalidades básicas do aplicativo, isto é, o aplicativo pode funcionar de forma satisfatória sem ele. Requisitos desejáveis podem ser deixados para versões posteriores do aplicativo, caso não haja tempo hábil para implementá-los na versão que está sendo especificada.

## 2. Descrição geral do aplicativo

### 2.1 Abrangência e aplicativos relacionados

O aplicativo *Econoferro* realiza a otimização do plano de corte de vergalhões na construção civil onde fornece uma interface simples, intuitiva e eficiente para otimizar e organizar de forma segura os dados adquiridos.

## 3. Requisitos funcionais (casos de uso)

### 3.1 Cadastro

#### [RF001] Criar Obra

Descrição do caso de uso: Este caso de uso permite que o usuário crie e armazene uma nova obra no aplicativo.

Prioridade:  Essencial  Importante  Desejável

Entradas e pré-condições: não tem.

Saídas e pós-condição: uma obra é cadastrada no aplicativo.

#### [RF002] Alterar Obra

Descrição do caso de uso: Este caso de uso permite que o usuário altere uma obra do cadastro de obras do aplicativo.

Prioridade:  Essencial  Importante  Desejável

Entradas e pré-condições: recebe como entrada a obra em que se deseja alterar.

Saídas e pós-condição: o usuário consegue alterar a obra em que deseja.

#### [RF003] Excluir Obra

Descrição do caso de uso: Este caso de uso permite que o usuário exclua uma obra do cadastro de obras do aplicativo.

Prioridade:  Essencial  Importante  Desejável

**Entradas e pré-condições:** recebe como entrada a obra em que se deseja excluir.

**Saídas e pós-condição:** o usuário consegue excluir a obra em que deseja.

#### [RF004] Listar Obras

**Descrição do caso de uso:** Este caso de uso permite que o usuário veja na tela as obras já cadastradas.

**Prioridade:**  Essencial  Importante  Desejável

**Entradas e pré-condições:** recebe como entrada n obras, n é o total de obras cadastradas.

**Saídas e pós-condição:** todas as obras cadastradas são listadas na tela do aplicativo.

#### [RF005] Criar Estrutura

**Descrição do caso de uso:** Este caso de uso permite que o usuário crie e armazene uma nova estrutura no aplicativo.

**Prioridade:**  Essencial  Importante  Desejável

**Entradas e pré-condições:** Existir uma obra cadastrada.

**Saídas e pós-condição:** uma estrutura é cadastrada no aplicativo e vinculada a obra selecionada.

#### [RF006] Alterar Estrutura

**Descrição do caso de uso:** Este caso de uso permite que o usuário altere uma estrutura do cadastro de obras do aplicativo.

**Prioridade:**  Essencial  Importante  Desejável

**Entradas e pré-condições:** recebe como entrada a obra e a estrutura em que se deseja alterar.

**Saídas e pós-condição:** o usuário consegue alterar a estrutura em que deseja.

#### [RF007] Excluir Estrutura

**Descrição do caso de uso:** Este caso de uso permite que o usuário exclua uma estrutura do cadastro de obras do aplicativo.

Prioridade:  Essencial  Importante  Desejável

Entradas e pré-condições: recebe como entrada a obra e a estrutura em que se deseja excluir.

Saídas e pós-condição: o usuário consegue excluir a estrutura em que deseja.

#### [RF008] Listar Estruturas

Descrição do caso de uso: Este caso de uso permite que o usuário veja na tela as estruturas cadastradas, por obra selecionada já cadastradas.

Prioridade:  Essencial  Importante  Desejável

Entradas e pré-condições: recebe como entrada uma obra cadastrada e n estruturas, onde n é o total de estruturas cadastradas da obra selecionada.

Saídas e pós-condição: todas as estruturas vinculadas a obra selecionada, são listadas na tela do aplicativo.

#### [RF009] Criar Demandas

Descrição do caso de uso: Este caso de uso permite que o usuário crie e armazene n demandas vinculadas a uma estrutura já cadastrada no aplicativo.

Prioridade:  Essencial  Importante  Desejável

Entradas e pré-condições: Existir uma obra cadastrada e estrutura.

Saídas e pós-condição: as demandas são adicionadas e vinculadas a estrutura de uma determinada obra no aplicativo.

#### [RF010] Alterar Demanda

Descrição do caso de uso: Este caso de uso permite que o usuário altere uma demanda da lista de demandas de uma determinada estrutura do aplicativo.

Prioridade:  Essencial  Importante  Desejável

Entradas e pré-condições: recebe como entrada a obra e a estrutura em que se deseja alterar a demanda.

Saídas e pós-condição: o usuário consegue alterar a demanda em que deseja.

<b>[RF011] Excluir Demanda</b>
--------------------------------

**Descrição do caso de uso:** Este caso de uso permite que o usuário exclua uma demanda da lista de demandas de uma determinada estrutura do aplicativo.

**Prioridade:**      Essencial                     Importante                     Desejável

**Entradas e pré-condições:** recebe como entrada a obra e a estrutura em que se deseja excluir a demanda.

**Saídas e pós-condição:** o usuário consegue excluir a demanda em que deseja.

<b>[RF012] Listar Demandas</b>
--------------------------------

**Descrição do caso de uso:** Este caso de uso permite que o usuário veja na tela as demandas inseridas na estrutura.

**Prioridade:**      Essencial                     Importante                     Desejável

**Entradas e pré-condições:** recebe como entrada uma obra cadastrada e n estruturas, onde n é o total de estruturas cadastradas da obra selecionada.

**Saídas e pós-condição:** todas as demandas são listadas na ordem em que são inseridas na tela do aplicativo.

## 3.2 Interface

<b>[RF001] Plano de Corte</b>
-------------------------------

**Descrição do caso de uso:** Este caso de uso permite que o usuário selecione uma obra para gerar o plano de corte em formato pdf.

**Prioridade:**      Essencial                     Importante                     Desejável

**Entradas e pré-condições:** deve receber como entrada uma obra em que se deseja otimizar com suas devidas estruturas e demandas.

**Saídas e pós-condição:** gera um arquivo pdf com o plano de corte otimizado.



#### [RF002] Visualizar Plano de Corte

**Descrição do caso de uso:** Este caso de uso permite que o usuário visualize os planos de cortes gerados, através de um leitor de pdf.

**Prioridade:**  Essencial  Importante  Desejável

**Entradas e pré-condições:** recebe como entrada o plano de corte.

**Saídas e pós-condição:** o usuário consegue visualizar o plano de corte em que deseja.

#### [RF003] Excluir Plano de Corte

**Descrição do caso de uso:** Este caso de uso permite que o usuário exclua um plano de corte do aplicativo.

**Prioridade:**  Essencial  Importante  Desejável

**Entradas e pré-condições:** recebe como entrada o plano de corte em que se deseja excluir.

**Saídas e pós-condição:** o usuário consegue excluir o plano de corte em que deseja.

### 4. Requisitos não-funcionais

#### [NF001] Usabilidade

A interface com o usuário é de vital importância para o sucesso do aplicativo. Principalmente por ser um aplicativo que não será utilizado diariamente, o usuário não possui tempo disponível para aprender como utilizar o aplicativo.

O aplicativo terá uma interface amigável ao usuário primário sem se tornar cansativa aos usuários mais experientes.

**Prioridade:**  Essencial  Importante  Desejável

#### [NF002] Desempenho

Embora não seja um requisito essencial ao aplicativo, deve ser considerada por corresponder a um fator de qualidade de software.

**Prioridade:**  Essencial  Importante  Desejável

**[NF003] Hardware e Software**

Visando criar um produto com maior extensibilidade, mobilidade e disponibilidade, deve ser adotado compatibilidade com a API 19 (KitKat) até a API 29 (Q) do sistema android, onde atingirá cerca de 95,3 % dos aparelhos android.

Prioridade:     Essencial             Importante             Desejável

**5. Referências**

1. Furlan, J. D. **Modelagem de Objetos através da UML**. São Paulo, Makron Books, 1998.
2. Kruchten, P. **The Rational Unified Process – An introduction**. Addison-Wesley, 1998.
3. Página da disciplina Análise e Especificação de Requisitos. [www.cin.ufpe.br/~ifl19](http://www.cin.ufpe.br/~ifl19).
4. Página da disciplina Metodologia e Desenvolvimento de Software [www.cin.ufpe.br/~mds](http://www.cin.ufpe.br/~mds).
5. Página da empresa Rational Software Corporation [www.rational.com](http://www.rational.com).
6. Página do projeto de instanciação de ambientes de desenvolvimento de software convencionais e orientados a domínios (visitada em 18/04/2019) [www.cos.uftj.br/~taba](http://www.cos.uftj.br/~taba).

## ANEXO A – Lista de demandas laje

Quantidade de aço CA-50 para Lajes do 3º Pav. - COMPRAR EM KG									
BITOLA	ELEMENTO ESTRUTURAL	POSIÇÃO	QUANT.	COMPR. DO FERRO (M)	TOTAL (M)	PESO UNIT (KG/M)	PESO TOTAL (KG)	PESO + 10% (KG)	TOTAL GERAL
Ø 8.0	POS. VERTICAL	N18	94	1,34	125,96	0,395	49,75	54,73	917,11
	POS. VERTICAL	N26	30	1,20	36,00	0,395	14,22	15,64	
	CISALHAMENTO	-	466	3,86	1796,76	0,395	710,51	781,56	
	NEG. HORIZONTAL	N31	50	3,00	150,00	0,395	59,25	65,18	
Ø 10.0	NEG. HORIZONTAL	N35	6	2,70	16,20	0,617	10,00	10,99	512,27
	NEG. HORIZONTAL	N36	6	3,88	23,28	0,617	14,36	15,80	
	NEG. HORIZONTAL	N32	6	3,90	23,40	0,617	14,44	15,88	
	POS. HORIZONTAL	N8	5	3,98	19,90	0,617	12,28	13,51	
	NEG. HORIZONTAL	N46	148	4,00	592,00	0,617	365,26	401,79	
	NEG. HORIZONTAL	N47	4	5,50	22,00	0,617	13,57	14,93	
	NEG. HORIZONTAL	N44	2	9,00	18,00	0,617	11,11	12,22	
	NEG. HORIZONTAL	N40	4	10,00	40,00	0,617	24,68	27,15	
Ø 12.5	POS. VERTICAL	N23	1	2,48	2,48	0,963	2,39	2,63	8413,23
	POS. HORIZONTAL	N5	3	2,90	8,70	0,963	8,38	9,22	
	POS. HORIZONTAL	N4	56	3,60	201,60	0,963	194,14	213,55	
	NEG. HORIZONTAL	N30	778	3,90	3034,20	0,963	2921,93	3214,13	
	POS. HORIZONTAL	N10	2	3,98	7,96	0,963	7,67	8,43	
	NEG. HORIZONTAL	N48	3	4,00	12,00	0,963	11,56	12,71	
	POS. HORIZONTAL	N6 E N11	13	4,50	58,50	0,963	56,34	61,97	
	POS. VERTICAL	N19	2	4,75	9,50	0,963	9,15	10,06	
	POS. HORIZONTAL	N7	6	4,88	29,28	0,963	28,20	31,02	
	POS. VERTICAL	N21	3	4,95	14,85	0,963	14,30	15,73	
	NEG. HORIZONTAL	N33 E N38	14	5,00	70,00	0,963	67,41	74,15	
	POS. HORIZONTAL	N9	6	5,25	31,50	0,963	30,33	33,37	
	NEG. HORIZONTAL	N34	8	6,00	48,00	0,963	46,22	50,85	
	NEG. HORIZONTAL	N39	2	6,10	12,20	0,963	11,75	12,92	
	NEG. HORIZONTAL	N29	112	6,40	716,80	0,963	690,28	759,31	
	NEG. HORIZONTAL	N37	31	7,00	217,00	0,963	208,97	229,87	
	POS. HORIZONTAL	N13	2	7,05	14,10	0,963	13,58	14,94	
	POS. HORIZONTAL	N14	4	7,52	30,08	0,963	28,97	31,86	
	NEG. VERTICAL	N45	3	8,00	24,00	0,963	23,11	25,42	
	POS. HORIZONTAL	N15	2	8,40	16,80	0,963	16,18	17,80	
	POS. HORIZONTAL	N1	331	8,60	2846,60	0,963	2741,28	3015,40	
	POS. VERTICAL	N24	3	8,70	26,10	0,963	25,13	27,65	
	POS. HORIZONTAL	N3	56	8,75	490,00	0,963	471,87	519,06	
NEG. VERTICAL	N49	2	10,00	20,00	0,963	19,26	21,19		
Ø 16.0	POS. VERTICAL	N26	1	2,96	2,96	1,578	4,67	5,14	2626,59
	POS. VERTICAL	N27	2	3,00	6,00	1,578	9,47	10,41	
	POS. VERTICAL	N22	1	5,00	5,00	1,578	7,89	8,68	
	POS. HORIZONTAL	N12	50	8,45	422,50	1,578	666,71	733,38	
	POS. VERTICAL	N20	4	8,50	34,00	1,578	53,65	59,02	
	POS. HORIZONTAL	N2	10	8,60	86,00	1,578	135,71	149,28	
	POS. VERTICAL	N25	20	9,10	182,00	1,578	287,20	315,92	
	POS. VERTICAL	N16	7	10,00	70,00	1,578	110,46	121,51	
	POS. VERTICAL	N17	56	12,00	696,00	1,578	1098,29	1098,29	

## ANEXO B – Lista de demandas vigas

Quantidade de aço CA-50 para vigas do 3º Pav. - COMPRAR EM KG									
BITOLA	ELEMENTO ESTRUTURAL	POSIÇÃO	QUANT.	COMPR. DO FERRO (M)	TOTAL (M)	PESO UNIT (KG/M)	PESO TOTAL (KG)	PESO + 10% (KG)	TOTAL GERAL
Ø 5.0	V1a	N11	205	1,52	448,40	0,154	69,05	75,96	224,32
	V1b = V1d	N3	14	1,52	21,28	0,154	3,28	3,60	
	V1c	N6	91	1,52	138,32	0,154	21,30	23,43	
	V1e	N6	101	1,52	153,52	0,154	23,64	26,01	
	V15	N3	17	1,52	25,84	0,154	3,98	4,38	
	V10	N4	147	1,52	223,44	0,154	34,41	37,85	
	V8	N9	2	3,95	7,90	0,154	1,22	1,34	
	V9	N12	201	1,52	305,52	0,154	47,05	51,76	
Ø 6.3	V1a	N3	8	3,00	24,00	0,245	5,88	6,47	202,83
	V1e	N1	4	3,50	14,00	0,245	3,43	3,77	
	V16 = V25	N13	8	2,40	19,20	0,245	4,70	5,17	
	V24	N13	8	1,80	14,40	0,245	3,53	3,88	
	V3 = V4 = V18	N2	12	0,65	7,80	0,245	1,91	2,10	
	V3 = V4 = V18	N3	36	1,52	54,72	0,245	13,41	14,75	
	V10	N3	2	0,85	1,70	0,245	0,42	0,46	
	V11 = V12 = V13 = V14	N11	32	2,40	76,80	0,245	18,82	20,70	
	V19	N2	24	1,52	36,48	0,245	8,94	9,83	
	V20	N11	4	2,40	9,60	0,245	2,35	2,59	
	V21 = V23	N2	8	0,65	5,20	0,245	1,27	1,40	
	V21 = V23	N3	26	1,52	39,52	0,245	9,68	10,65	
	V26	N13	1	1,35	1,35	0,245	0,33	0,36	
	V5 = V7	N4	22	1,52	33,44	0,245	8,19	9,01	
	V8	N3	2	6,05	12,10	0,245	2,98	3,28	
	V8	N8	2	3,80	7,60	0,245	1,86	2,05	
	V8	N21	2	1,35	2,70	0,245	0,66	0,73	
V8	N22	221	1,52	335,92	0,245	82,30	90,53		
V9	N1	4	3,00	12,00	0,245	2,94	3,23		
V22	N2	20	1,52	30,40	0,245	7,45	8,19		
V28	N2	9	1,52	13,68	0,245	3,35	3,69		
	V1a	N12	6	3,00	18,00	0,395	7,11	7,82	
	V1a	N13	30	8,00	240,00	0,395	94,80	104,28	
	V1a	N14	6	1,61	9,66	0,395	3,82	4,20	
	V1b = V1d	N4	8	1,36	10,88	0,395	4,30	4,73	
	V1c	N7	6	6,60	39,60	0,395	15,64	17,21	
	V1c	N8	6	7,90	47,40	0,395	18,72	20,60	
	V1e	N7	6	7,81	46,86	0,395	18,51	20,36	
	V1e	N8	6	8,00	48,00	0,395	18,96	20,86	
	V15	N4	6	2,93	17,58	0,395	6,94	7,64	
	V16 = V25	N14	318	4,50	1422,00	0,395	561,69	617,86	
	V16 = V25	N15	632	2,28	1440,96	0,395	569,18	626,10	
	V24	N14	158	4,50	711,00	0,395	280,85	308,93	
	V24	N15	316	2,28	720,48	0,395	284,59	313,05	
	V24	N18	18	2,20	39,60	0,395	15,64	17,21	
	V3 = V4 = V18	N4	18	2,28	40,68	0,395	16,07	17,68	
	V10	N5	12	6,32	75,84	0,395	29,96	32,95	
	V10	N6	6	11,40	68,40	0,395	27,02	29,72	

Ø 8.0	V11 = V12 = V13 = V14	N12	720	4,50	3240,00	0,395	1279,80	1407,78	11696,36
	V11 = V12 = V13 = V14	N13	1440	2,28	3283,20	0,395	1296,86	1426,55	
	VIGAS CHATAS	ESTRIBOS	2628	2,28	5991,84	0,395	2366,78	2603,45	
	VIGAS CHATAS	ESTRIBOS	1314	4,50	5913,00	0,395	2335,64	2589,20	
	V17	N10	4	2,40	9,60	0,395	3,79	4,17	
	V17	N11	4	2,75	11,00	0,395	4,35	4,78	
	V17	N12	126	4,50	567,00	0,395	223,97	246,36	
	V17	N13	252	2,28	574,56	0,395	226,95	249,85	
	V19	N3	6	4,16	24,96	0,395	9,86	10,85	
	V20	N12	137	4,50	616,50	0,395	243,52	267,87	
	V20	N13	274	2,28	624,72	0,395	246,76	271,44	
	V21 = V23	N4	12	2,41	28,92	0,395	11,42	12,57	
	V26	N1	2	3,50	7,00	0,395	2,77	3,04	
	V26	N3	2	4,70	9,40	0,395	3,71	4,08	
	V26	N7	6	1,80	10,80	0,395	4,27	4,69	
	V26	N14	109	1,54	167,86	0,395	66,30	72,94	
	V26	N15	6	7,13	42,78	0,395	16,90	18,59	
	V26	N16	6	10,33	61,98	0,395	24,48	26,93	
	V5 = V7	N3	4	0,65	2,60	0,395	1,03	1,13	
	V5 = V7	N5	12	2,32	27,84	0,395	11,00	12,10	
	V6	N4	45	1,52	68,40	0,395	27,02	29,72	
	V6	N5	6	2,49	14,94	0,395	5,90	6,49	
	V6	N6	6	3,17	19,02	0,395	7,51	8,26	
	V8	N23	6	3,66	21,96	0,395	8,67	9,54	
	V8	N24	18	7,68	138,24	0,395	54,60	60,07	
	V8	N25	6	11,53	69,18	0,395	27,33	30,06	
	V8	N26	6	3,89	23,34	0,395	9,22	10,14	
	V9	N13	6	2,80	16,80	0,395	6,64	7,30	
	V9	N14	6	7,55	45,30	0,395	17,89	19,68	
	V9	N15	6	7,40	44,40	0,395	17,54	19,29	
	V9	N16	6	7,53	45,18	0,395	17,85	19,63	
	V9	N17	6	4,03	24,18	0,395	9,55	10,51	
	V22	N3	6	3,41	20,46	0,395	8,08	8,89	
	V27	N3	16	4,50	72,00	0,395	28,44	31,28	
	V27	N4	32	2,28	72,96	0,395	28,82	31,70	
	V28	N3	6	1,71	10,26	0,395	4,05	4,46	
	V1a	N7	3	3,00	9,00	0,617	5,55	6,11	
	V1a	N8	3	7,55	22,65	0,617	13,98	15,37	
	V1a	N9	12	7,40	88,80	0,617	54,79	60,27	
	V1a	N10	3	1,61	4,83	0,617	2,98	3,28	
V1c	N4	3	6,40	19,20	0,617	11,85	13,03		
V1c	N5	3	7,80	23,40	0,617	14,44	15,88		
V1e	N4	3	7,70	23,10	0,617	14,25	15,68		
V16 = V25	N9	18	1,25	22,50	0,617	13,88	15,27		
V24	N9	9	1,25	11,25	0,617	6,94	7,64		
V11 = V12 = V13 = V14	N6	36	1,25	45,00	0,617	27,77	30,54		
V11 = V12 = V13 = V14	N8	36	1,25	45,00	0,617	27,77	30,54		
V11 = V12 = V13 = V14	N10	44	2,20	96,80	0,617	59,73	65,70		
V17	N6	9	1,25	11,25	0,617	6,94	7,64		
V20	N6	18	1,25	22,50	0,617	13,88	15,27		

Ø 10.0	V26	N10	2	2,10	4,20	0,617	2,59	2,85	458,45
	V5 = V7	N2	6	2,54	15,24	0,617	9,40	10,34	
	V8	N10	2	5,30	10,60	0,617	6,54	7,19	
	V8	N11	2	2,65	5,30	0,617	3,27	3,60	
	V8	N12	3	2,06	6,18	0,617	3,81	4,19	
	V8	N13	2	4,11	8,22	0,617	5,07	5,58	
	V8	N14	2	7,75	15,50	0,617	9,56	10,52	
	V8	N15	1	4,40	4,40	0,617	2,71	2,99	
	V8	N18	3	11,25	33,75	0,617	20,82	22,91	
	V8	N19	3	7,86	23,58	0,617	14,55	16,00	
	V8	N20	1	5,85	5,85	0,617	3,61	3,97	
	V9	N6	2	3,03	6,06	0,617	3,74	4,11	
	V9	N7	2	7,65	15,30	0,617	9,44	10,38	
	V9	N8	1	4,55	4,55	0,617	2,81	3,09	
	V9	N9	6	7,50	45,00	0,617	27,77	30,54	
	V9	N10	2	4,27	8,54	0,617	5,27	5,80	
	V9	N11	1	1,85	1,85	0,617	1,14	1,26	
	V27	N2	8	2,01	16,08	0,617	9,92	10,91	
	Ø 12.5	V1a	N1	2	6,00	12,00	0,963	11,56	
V1a		N2	1	4,00	4,00	0,963	3,85	4,24	
V1a		N4	8	5,90	47,20	0,963	45,45	50,00	
V1b = V1d		N1	4	2,50	10,00	0,963	9,63	10,59	
V1b = V1d		N2	4	1,48	5,92	0,963	5,70	6,27	
V1c		N1	1	4,00	4,00	0,963	3,85	4,24	
V1c		N2	2	5,30	10,60	0,963	10,21	11,23	
V1c		N3	2	10,00	20,00	0,963	19,26	21,19	
V1e		N2	4	2,80	11,20	0,963	10,79	11,86	
V1e		N3	2	5,90	11,80	0,963	11,36	12,50	
V1e		N5	3	8,00	24,00	0,963	23,11	25,42	
V15		N1	2	4,00	8,00	0,963	7,70	8,47	
V15		N2	2	3,25	6,50	0,963	6,26	6,89	
V16 = V25		N1	12	3,00	36,00	0,963	34,67	38,13	
V16 = V25		N3	12	4,75	57,00	0,963	54,89	60,38	
V16 = V25		N12	22	2,20	48,40	0,963	46,81	51,27	
V24		N1	6	3,00	18,00	0,963	17,33	19,07	
V24		N3	6	4,75	28,50	0,963	27,45	30,19	
V24		N12	4	2,20	8,80	0,963	8,47	9,32	
V24		N17	7	3,20	22,40	0,963	21,57	23,73	
V3 = V4 = V18		N1	12	2,58	30,96	0,963	29,81	32,80	
V10		N1	8	7,00	56,00	0,963	53,93	59,32	
V10		N2	4	12,00	48,00	0,963	46,22	50,85	
V11 = V12 = V13 = V14		N2	24	5,50	132,00	0,963	127,12	139,83	
V11 = V12 = V13 = V14		N7	44	10,20	448,80	0,963	432,19	475,41	
V17		N1	6	5,20	31,20	0,963	30,05	33,05	
V17		N5	11	4,00	44,00	0,963	42,37	46,61	
V17		N9	11	2,30	25,30	0,963	24,36	26,80	
V19		N1	8	4,48	35,84	0,963	34,51	37,97	
V20		N1	6	5,20	31,20	0,963	30,05	33,05	
V20	N7	11	4,00	44,00	0,963	42,37	46,61		
V20	N10	11	3,25	35,75	0,963	34,43	37,87		





	V26	N6	2	5,35	10,70	2,466	26,39	29,02	
	V26	N11	2	6,00	12,00	2,466	29,59	32,55	
	V26	N12	2	10,95	21,90	2,466	54,01	59,41	
	V8	N1	3	7,70	23,10	2,466	56,96	62,66	
	V8	N7	2	4,50	9,00	2,466	22,19	24,41	
	V8	N1	2	7,00	14,00	2,466	34,52	37,98	
	V8	N2	1	4,00	4,00	2,466	9,86	10,85	
	V8	N4	2	7,90	15,80	2,466	38,96	42,86	
	V8	N5	2	3,00	6,00	2,466	14,80	16,28	
	V8	N6	2	9,00	18,00	2,466	44,39	48,83	
	V8	N7	1	2,30	2,30	2,466	5,67	6,24	
	V8	N16	2	6,00	12,00	2,466	29,59	32,55	
	V8	N17	3	11,80	34,80	2,466	85,82	94,40	
	V9	N5	2	9,60	19,20	2,466	47,35	52,08	
Ø 25.0	V16 = V25	N10	12	11,35	136,20	3,853	524,78	554,83	1941,91
	V24	N10	6	11,35	68,10	3,853	262,39	277,42	
	V11 = V12 = V13 = V14	N9	24	11,35	272,40	3,853	1049,56	1109,66	