UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

SABRINA AGUIAR DA SILVA

DESENVOLVIMENTO DE SOFTWARE PARA ANÁLISE, VERIFICAÇÃO E ARMAZENAMENTO DE RESULTADOS DE ENSAIOS MARSHALL

CAMPO MOURÃO

2018

SABRINA AGUIAR DA SILVA

DESENVOLVIMENTO DE SOFTWARE PARA ANÁLISE, VERIFICAÇÃO E ARMAZENAMENTO DE RESULTADOS DE ENSAIOS MARSHALL

Trabalho de Conclusão de Curso de Graduação apresentado à Disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior em Engenharia Civil do Departamento Acadêmico de Construção Civil – DACOC – da Universidade Tecnológica Federal do Paraná – UTFPR, para obtenção do título de bacharel em engenharia civil.

Orientador: Prof. Dr. Ewerton Clayton Alves da Fonseca.

CAMPO MOURÃO

2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Campo Mourão
Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Construção Civil
Coordenação de Engenharia Civil

TERMO DE APROVAÇÃO

Trabalho de Conclusão de Curso

DESENVOLVIMENTO DE SOFTWARE PARA ANÁLISE, VERIFICAÇÃO E ARMAZENAMENTO DE RESULTADOS DE ENSAIOS MARSHALL

por

Sabrina Aguiar da Silva

Este Trabalho de Conclusão de Curso foi apresentado às 16h00min do dia 26 de Fevereirode 2018como requisito parcial para a obtenção do título de ENGENHEIRO CIVIL, pela Universidade Tecnológica Federal do Paraná. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Dr. Leandro Waidemam
(UTFPR)

Prof. Me. Paulo Henrique Rodrigues
(UTFPR)

Prof. Me. Lucas Lauer Verdade
(UTFPR)
Coorientador

Prof. Dr.Ewerton Clayton Alves da Fonseca
(UTFPR)
Orientador

Responsável pelo TCC: **Prof. Me. Valdomiro Lubachevski Kurta**

Coordenador do Curso de Engenharia Civil: **Prof. Dr. Ronaldo Rigobello**

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

Dedico aos meus pais, irmão e avó por acreditarem em mim.
Em memória do meu querido avô, Pedro José de Aguiar (*in memorian*) e dos meus queridos e saudososamigos Willian Petsche Higor Leonardo (*in memorian*).

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus pelo dom da vida, por estar presente em meu dia-a-dia, por me dar força e coragem nos momentos que mais precisei e por nunca me abandonar.

Agradeço aos meus pais Arnaldo Francisco da Silva e Simone Aguiar da Silvae à minha segunda mãe, Santa Nunes de Aguiar por nunca desistirem de mim e me apoiarem em todas as decisões, além de serem meus grandes exemplos. Também agradeço aos meus avós Maria Jandira da Silva e José Francisco da Silva. Se sou essa pessoa hoje, devo tudo a vocês. A meu irmão Diego Aguiar da Silva por me ajudar sendo exemplo de persistência, superação e dedicação.

Aos meus amigos, Bianca Mendonça, Matheus Korczovei, Heliberto Gonçalves, Henriquy Aguiar, Nicolle Kozielski, Thais Ribeiro, Leda Batelo, Patricia Davi, Gabriel Mendonça, Bruna Aoki, Dayane Omura, Gilberto Henrique, Beatriz Tomeix, José Vítor, Lucas Senger, Marcus Vinícius, Luiz Henrique, Vanessa Santos, Elijaine Denardo e Junior da Rocha, pessoas as quais contribuíram consideravelmente para minha formação pessoal e que de alguma forma me motivaram a continuar buscando pela realização deste sonho. Obrigada pelas risadas, pela amizade sincera, e por trazerem calmaria em momentos difíceis. Considero vocês parte de minha família.

Ao meu orientador, prof. Dr.Ewerton Clayton Alves da Fonseca, pelos ensinamentos e paciência durante todo o desenvolvimento deste trabalho, colocando-se sempre disposto a ensinar e a ajudar sem medir esforços. Ao meu coorientador, Prof. Me. Lucas Lauer Verdade, por todo apoio, conhecimento, ensinamentos e sugestões proporcionados para a conclusão deste trabalho.

Um agradecimento especial também aos professores: Prof. Paulo Henrique Rodrigues, Prof. Dr. Leandro Waidemame Prof. Sérgio Roberto Oberhauser Quintanilha Braga, que contribuíram significativamente para a realização deste trabalho.

A UTFPR, todos os professores e profissionais que diretamente ou indiretamente fizeram parte da minha graduação e que me ajudaram para que eu chegasse até aqui.

Enfim, a todos, meu sincero muito obrigada!

RESUMO

Os laboratórios de pavimentação asfáltica geralmente não possuem ferramentas adequadas (“softwares”) para realizaras análises e verificações dos dados obtidos a partir dos ensaios executados. Assim sendo, o armazenamento destes dados pode ser feito de forma inapropriada, uma vez que, na prática, é comum que as análises sejam feitas por meio de planilhas Excel, que quando não utilizadas de maneira apropriada, permitem ao usuário editar os dados que não se enquadraram no projeto da mistura asfáltica, atitude esta que pode interferir no desempenho do pavimento. Devido a estes fatores, novas ferramentas devem ser idealizadas e implementadas para suprir as necessidades dos escritórios de pavimentação e agilizar os processos de análises e verificaçõesde dados. Portanto, neste trabalho, um “software” capaz de analisar e verificar dados obtidos mediante a realização de ensaios Marshall foi desenvolvido a partir das recomendações da DNER 043 (1995), já que, além de acelerar os processos de análises e verificações, ferramentas computacionais como esta, podem reduzir significativamente a ocorrência de manipulação de dados por parte dos usuários. O código fonte foi desenvolvido com a utilização das linguagens Java e SQL. Esta segunda foi útil para armazenar os resultados de ensaios em banco de dados para posterior utilização em medições mensais, as quais são obrigatórias em obras de pavimentação. Para ser utilizado na prática de pavimentação de modo a suprir a necessidade de realização de demais tipos de ensaios, devem ser realizadas implementações no código já existente, para que a análise completa dos parâmetros da mistura asfáltica seja efetuada.

Palavras-chave: Ensaio Marshall, Software, Java, SQL.

ABSTRACT

Asphalt paving laboratories usually do not have suitable tools (softwares) to carry out analyzes and verifications of the data obtained from tests performed. Thus, the storage of this data can be done in an inappropriate manner, giving that, in practice, it is common for these analyzes to be done using Excel spreadsheets, that when not used correctly, allow the user to edit the date that do not fit in the project for asphalt mix, this attitude can interfere in the performance of the paving. Due to these factors, new tools must be idealized and implemented to satisfy the needs of the paving offices and speed the processes if analysis and verification of data. Therefore, in this paper, a software capable of analyze and verify data obtained by the perform of Marshall testing was developed in according with the recommendations of DNER 043 (1995), since, besides accelerating the processes of analysis and verifications, computational tools like this, can significantly reduce the occurrence of data manipulation by the users. The source code was developed utilizing the languages Java and SQL. The second was useful to storage the results of tests in database for future use in monthly measurement, these being obligatory in paving works. To be used in paving practice and satisfy the needs of performance of the remaining types tests, there must be implementations in the existing code, so that the complete analysis of the parameters of the asphalt mixture can be executed.

Keywords: Marshall Test, Software, Java, SQL.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

SQL	Structured Query Language (Linguagem de Consulta Estruturada)
MVC	Model View Controller (Modelo Visão Controlador)
JPA	Java Persistence API
DMT	Densidade Máxima Teórica
ATR	Afundamento em Trilha de Roda
DNER	Departamento Nacional de Estradas de Rodagem
ASTM	American Society for Testing and Materials
CPF	Cadastro de Pessoa Física
CBUQ	Concreto Betuminoso Usinado a Quente
ID	Identidade

SUMÁRIO

1 INTRODUÇÃO

Com o desenvolvimento tecnológico, cada vez mais as empresas buscam realizar suas atividades e funções por meio do uso de novas tecnologias. Devido a isto, as inovações tecnológicas são elaboradas, geralmente por empreendedores, com o objetivo de buscar por soluções, desde as mais simples e rápidas até as mais complexas. Uma forma de obter estas soluções é por meio de sistemas de informação, que são utilizados para auxiliar no desenvolvimento de atividades das empresas.

Os sistemas de informação podem ser desenvolvidos mediante uso da linguagem Java, uma linguagem de programação orientada a objetos. Esta linguagem atualmente é a base para diversos tipos de aplicações em rede e é o padrão global para o desenvolvimento e distribuição de aplicações móveis, jogos e conteúdo baseado na “web” e “softwares” corporativos.

Além de ser necessário fazer uso de uma linguagem de programação, quando sistemas de informação necessitam armazenar quantidades consideráveis de dados, é comum fazer uso do MySQL, um sistema de gerenciamento de banco de dados, o qual utiliza a linguagem SQL (linguagem de consulta estruturada) como interface.

Este trabalho visa desenvolver um “software”, utilizando-se as linguagens Java e SQL, para análise e verificação de resultados de ensaios Marshall, além do armazenamento dos dados obtidos por meio destes ensaios.

O ensaio Marshall, o principal ensaio presente neste trabalho, serve para realizar a determinação do ótimo teor de ligante a ser utilizado em misturas asfálticas usinadas à quente para que estas se enquadrem dentro de especificações que são definidas com o objetivo de evitar desagregação da mistura por falta de ligante, ou superfícies escorregadias e deformáveis, por excesso do mesmo.

Atualmente os laboratórios de empresas de pavimentação realizam ensaios Marshall diariamente e não possuem o tipo de ferramentadesenvolvida neste trabalho. Por isto, as mesmas acabam por fazer uso de planilhas criadas no editor Microsoft Office Excel, com a finalidade de analisar, verificar e armazenar os resultados oriundos da realização de ensaios Marshall. Neste caso, o operador da planilha possui acesso irrestrito ao seu conteúdo, o que permite a edição das mesmas, o que pode vir a favorecer a existência erros e/ou equívocos.

2 OBJETIVOS

2.1 Objetivo Geral

Desenvolver um “software” de simples manuseio, capaz de analisar, verificar e armazenar os resultados obtidos a partir da realização de ensaios Marshall.

2.2 Objetivos Específicos

- Definir o algoritmode elaboração do código fonte do “software”;
- Determinar e classificar o código fonte do “software” em classes, objetos, métodos, entidades e atributos;
- Utilizar o “Model View Controller” (MVC) dentro da linguagem Java 8 para trabalhar com classes, variáveis, informações, atributos e definir o que e como será disponibilizado aos usuários;
- Fazer uso do “Framework Hibernate” da linguagem JPA para armazenar automaticamente os dados necessários em banco de dados por meio do sistema de gerenciamento de banco de dados MySQL 5.7.4;
- Elaborarum “software” com plataforma simples e de fácil manuseio por meio do uso da linguagem Java 8 e do sistema de gerenciamento de banco de dados MySQL 5.7.4, de modo que determinados dados sejam informados pelo usuário e retornados ao mesmo, conforme solicitado.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Ensaio Marshall

O Ensaio Marshall tem se baseia na obtenção de três principais resultados: Determinação do teor de ligante asfáltico presente na mistura asfáltica, determinação da estabilidade do corpo de prova do Ensaio Marshall e determinação da deformação apresentada pelo corpo de prova no momento em que este é rompido na prensa.

3.1.1 Densidades aparente e real de misturas asfálticas

Segundo Bernucci *et al*. (2008),a densidade aparente da mistura é calculada pela relação entre a massa da mistura asfáltica ao ar suscetível ao ensaio Marshall e o volume do corpo de prova. Este último é obtido pela diferença entre a massa do corpo de prova medida ao ar e a massa do corpo de prova medida quando imersa em água, a partir do princípio de Arquimedes, conforme a Equação01:

$$V=Ms-Mssub \qquad \text{Eq. 01}$$

Onde:
V = Volume do corpo de prova;
Ms = Massa do corpo de prova seco;
Mssub = Massa do corpo de prova submerso em água.

Segundo Santos (2011), a densidade aparente (densidade bruta) corresponde ao volume ocupado pela mistura asfáltica, incluindo-se a porosidade da mesma, e pode ser calculada por meio da Equação 02:

$$Gmb = \frac{Ms}{V} \qquad \text{Eq. 02}$$

Onde:
Gmb = Massa específica aparente da mistura asfáltica;

Ms = Massa do corpo de prova seco;
V = Volume do corpo de prova.

Ainda segundo Santos (2011), a Densidade Real corresponde ao real volume que determinado sólido ocupa, ou seja, desconsiderando sua porosidade. Portanto, neste caso, a massa da mistura asfáltica para cálculo da densidade real é obtida quando todos os poros de sua superfície estão preenchidos com água e é obtida por meio da Equação 03:

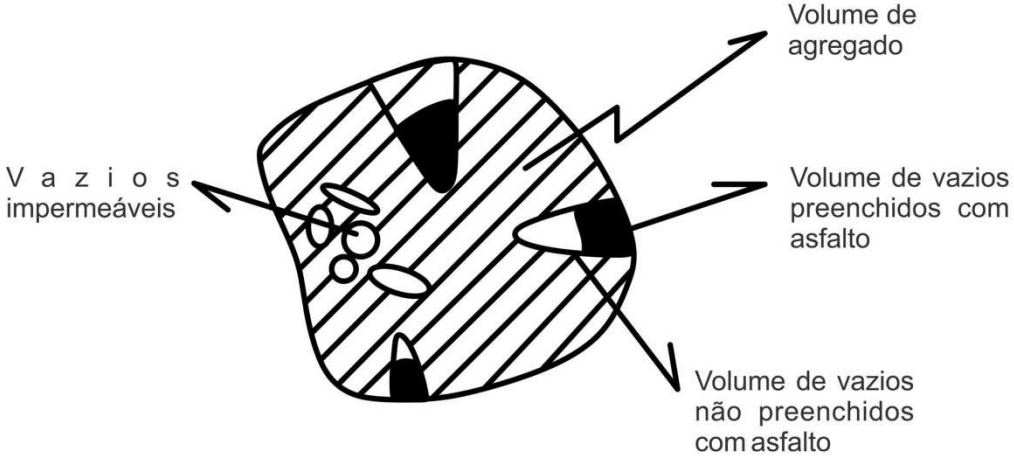
$$Gsa = \frac{M}{V} \qquad \text{Eq. 03}$$

Onde:
Gsa = Massa específica real da mistura asfáltica;
M = Massa do corpo de prova úmido;
V = Volume do corpo de prova.

3.1.2 Densidade máxima teórica de misturas asfálticas

Segundo Roberts *et al*. (1996), a densidade máxima teórica é numericamente igual à razão entre a massa do agregado mais ligante asfáltico e a soma dos volumes de agregado, vazios impermeáveis e vazios permeáveis não preenchidos com asfalto, de acordo com a ilustração mostrada na Figura 01.

Figura 1 - Ilustração da Densidade Máxima Teórica (DMT) para misturas asfálticas



Fonte: Bernucci *et al*. (2008).

Roberts *et al*. (1996) afirma que o conhecimento da DMT é necessário para os cálculos dos parâmetros de percentual de vazios de misturas asfálticas compactadas, absorção de ligante pelos agregados, massa específica efetiva do agregado, teor de asfalto efetivo da mistura asfáltica e, ainda, para fornecer valores alvo para a compactação de misturas asfálticas,mediante uso do compactador giratório. Outra utilização da DMT é na determinação da massa específica de misturas asfálticas já compactadas em campo. Juntamente com a espessura do pavimento, a DMT é necessária para que se estime a massa específica da mistura, sem extração de corposde prova, por meio do método nuclear, por exemplo.

Segundo Soares *et al*. (2003), a determinação da Densidade Máxima Teórica é realizada por meio de uma ponderação das densidades reais dos materiais que compõem a mistura asfáltica. Finalmente, o valor correspondente a Densidade Máxima Teórica (DMT) é obtida por meio da Equação 04:

$$DMT = \frac{Pcap+P1+P2+\ldots+Pn}{\left(\frac{Pcap}{Gcap}\right)+\left(\frac{P1}{G1}\right)+\left(\frac{P2}{G2}\right)+\ldots+\left(\frac{Pn}{Gn}\right)} \qquad \text{Eq. 04}$$

Onde:
DMT = Densidade máxima teórica da mistura asfáltica;
P = Peso do material constituinte da mistura asfáltica;
G = Densidade real do material constituinte da mistura asfáltica.

3.1.3 Volume de vazios presente na mistura asfáltica

O volume de vazios é a característica volumétrica mais importante do concreto asfáltico. Sempre são necessários vazios de ar dentro da mistura compactada para permitir a expansão térmica dos ligantes e suportar a leve compactação causada pelo tráfego. Volume de vazios baixo (menores do que 3%) compromete o desempenho de misturas quanto ao afundamento em trilha de rodas (ATR), enquanto que volume de vazios alto (maiores do que 8%) comprometem a durabilidade (Instituto do Asfalto, 1998).

Segundo Bernucci *et al*. (2008), o valor dovolume de vazios presente na mistura asfáltica é obtido por meio da Equação 05:

$$Vv = \frac{DMT - Gmb}{DMT} \qquad \text{Eq. 05}$$

Onde:
Vv = Porosidade ou volume de vazios da mistura asfáltica;
DMT = Densidade máxima teórica da mistura asfáltica;
Gmb = Massa específica aparente da mistura asfáltica.

3.1.4 Importância da estabilidade Marshall

Conforme as especificações da norma DNER-ME 043/1995, a estabilidade Marshall é a resistência máxima à compressão diametral, apresentada pelo corpo de prova, quando moldado e ensaiado de acordo com o processo estabelecido no método de ensaio Marshall. A estabilidade Marshall calculada é expressa em MPa.

De acordo com o que explica Resende (2016), o Ensaio Marshall é essencial para realizar a determinação da quantidade ótima de ligante a ser utilizada em misturas asfálticas usinadas a quente, destinadas à pavimentação de vias. Com ele é possível também determinar a estabilidade, que é a resistência máxima à compressão radialapresentada pelo corpo de prova. A realização deste último ensaio (estabilidade) permite a obtenção de uma relação entre os resultados obtidos em laboratório e a vida útil que o pavimento poderá apresentar após a aplicação da mistura asfáltica ensaiada.

Segundo Resende (2016), para a determinação da estabilidade Marshall, primeiramente a carga necessária para produzir o rompimento do corpo de prova é anotada como “estabilidade lida”. Este valor é corrigido para a espessura do corpo de prova ensaiado, multiplicando-se por um fator que é função da espessura do corpo de prova, calculado por meio da Equação 06:

$$f = 927{,}23 * h^{-1{,}64} \qquad \text{Eq. 06}$$

Onde:
f = Fator de correção;
h = Espessura do corpo de prova.

Vale ressaltar que o fator de correção (f) também pode ser obtido por meio da Tabela 1, conforme a espessura do corpo de prova, que será apresentada mais adiante, no Item 3.1.6.

3.1.5 Deformação

Segundo Nilson (2006), um corpo sólido se deforma quando sujeito a mudanças de temperatura ou a ação de uma força externa. Por exemplo, num ensaio de corpo de prova, quando aplicada uma força externa, o corpo de prova apresenta mudanças em seu comprimento. A relação entre o comprimento final e inicial apresentados pelo corpo de prova é denominada deformação. Esta deformação pode ser classificada em elástica ou plástica.

A deformação elástica é proporcioinal ao esforço aplicado e é uma deformação reversível, ou seja, quando a tensão que provocou a deformação elástica é cessada, o material volta ao seu estado inicial. Já a deformação plástica não é proporcional ao esforço aplicado, sendo assim, quando a tensão que provocou esta deformação é cessada, o material não volta ao seu estado inicial e apresenta uma deformação permanente, classificada como deformação plástica (Nilson, 2006).

Conforme explica Bernucci *et al*. (2008),a mistura asfáltica não é considerada um material elástico, sendo o uso da teoria da elasticidade uma aproximação. Portanto, o comportamento de alguns materiais de pavimentação pode ser aproximado e classificado como elástico não-linear.

3.1.6 Procedimentos para realização do ensaio Marshall

Os procedimentos a seguir descritos são baseados nas especificações da norma DNER-ME 043/1995.

Primeiramente, no mínimo três corpos de prova devem ser preparados para cada dosagem Marshall, conhecendo-se as porcentagens (em massa) em que os agregados e o ligante betuminoso serão misturados. Calcula-se, então, a quantidade de cada um deles capaz de compor um corpo de prova.

Os agregados devem ser secos em estufa, a temperaturas entre 105℃ a 110℃ até constância de massa. Após isto, devem ser separados nas seguintes frações:

I. 25 a 19 mm;
II. 19 a 9,5 mm;
III. 9,5 a 4,8 mm;
IV. 4,8 a 2,0 mm;
V. Passantes na peneira de 2,0 mm.

É necessário então, determinar a massa dos agregados para compor um corpo de prova de cada vez, em recipientes separados. Após serem misturados, o ligante e os agregados devem compor corpos de prova com aproximadamente 1200 g.

Após obter os três recipientes com as devidas composições, estes devem ser colocados em placa quente ou estufa e aquecidos entre 10℃ e 15℃ acima da temperatura de aquecimento do ligante, não devendo ultrapassar os 177℃. A temperatura de aquecimento do ligante para ser misturado aos agregados deve ser aquela na qual apresente uma viscosidade de (170 ± 20) cSt (centistokes) ou a viscosidade específica Engler de 25 ± 3 para alcatrão.

Os agregados de cada recipiente devem ser aquecidos, e em cada recipiente deve ser aberta uma cratera para receber o ligante (a massa do ligante deve ser medida neste momento). A mistura deve ser feita rapidamente, dentro de 2 a 3 minutos até completa cobertura dos agregados.

Após a realização da composição das misturas, dá-se início ao processo de compactação dos corpos de prova. O molde de compactação deve ser colocado no suporte de compactação e nele deve ser introduzida uma folha de papel-filtro, cortada conforme a seção do molde. A mistura deve ser introduzida no molde de compressão e o anel superior deve ser então removido e a mistura alisada com uma colher ligeiramente aquecida.

O anel superior deve ser recolocado. É necessário aplicar 75 golpes no corpo de prova, com altura de queda livre do soquete compactador de 47,72 cm. Deve-se então remover o anel superior, inverter o anel inferior e forçar com o soquete a mistura até atingir a placa-base e aplicar o mesmo número de golpes no corpo de prova invertido.

Após a compactação, deve-se retirar o corpo de prova do anel inferior, colocá-lo cuidadosamente numa superfície lisa e plana e deixá-lo em repouso durante, no mínimo, 12 horas à temperatura ambiente.

A altura do corpo de prova deve ser obtida por meio da média aritmética das medidas realizadas com paquímetro, em quatro posições diametralmente opostas.

Para dar início ao processo de determinação da estabilidade e fluência é necessário imergir os corpos de prova em água (60°C ± 1°C), para misturas com cimento asfáltico ou a (38°C ± 1°C), para misturas com alcatrão, por um período de 30 a 40 minutos. Os corpos de prova também podem ser colocados em estufa nas mesmas temperaturas pelo período de duas horas.

Primeiramente as massas de todos os corpos de prova “ao ar” devem ser aferidas. Sequentemente estes mesmos corpos de prova devem ter suas massas medidas quando imersos em água (massa do corpo de prova mais o empuxo da água) para que os seguintes resultados possam ser obtidos: porcentagem de vazios, porcentagem de vazios cheios de betume, porcentagem de vazios do agregado mineral, e a relação dos vazios cheios de betume com os vazios do agregado mineral.

Em seguida, cada corpo de prova deve ser colocado no molde de compressão. Este molde é posicionado na prensa e o medidor de deformação é colocado e ajustado na posição de ensaio. A prensa é operada para que seu êmbolo se eleve a uma velocidade de 5 cm/min., até a ruptura do corpo de prova. O valor da “força” aplicada no corpo de prova no momento em que o mesmo rompe deve ser aferido e convertido para kgf.

Finalmente, a estabilidade (N) é a carga, em kgf, necessária para romper o corpo de prova. Este valor deverá ser corrigido para a espessura do corpo de prova ensaiado, multiplicando-se o valor da estabilidade pelo fator de correção que depende da espessura do corpo de prova e que pode ser observado na Tabela 01.

Tabela 1 - Correção da estabilidade em função da espessura do corpo de prova

Espessura (mm)	Fator	Espessura (mm)	Fator	Espessura (mm)	Fator
50,8	1,47	56,3	1,22	64,3	0,98
51,0	1,45	56,6	1,21	64,7	0,97
51,2	1,44	56,8	1,20	65,1	0,96
51,6	1,43	57,1	1,19	65,6	0,95
51,8	1,42	57,4	1,18	66,1	0,94
52,0	1,41	57,7	1,17	66,7	0,93
52,2	1,40	58,1	1,16	67,1	0,92
52,4	1,39	58,4	1,15	67,5	0,91
52,6	1,38	58,7	1,14	67,9	0,90
52,9	1,37	59,0	1,13	68,3	0,89
53,1	1,36	59,3	1,12	68,8	0,88
53,3	1,35	59,7	1,11	69,3	0,87
53,5	1,34	60,0	1,10	69,9	0,86
53,8	1,33	60,3	1,09	70,3	0,85
54,0	1,32	60,6	1,08	70,8	0,84
54,2	1,31	60,9	1,07	71,4	0,83
54,5	1,30	61,1	1,06	72,2	0,82
54,7	1,29	61,4	1,05	73,0	0,81
54,9	1,28	61,9	1,04	73,5	0,80
55,1	1,27	62,3	1,03	74,0	0,79
55,4	1,26	62,7	1,02	74,6	0,78
55,6	1,25	63,1	1,01	75,4	0,77
55,8	1,24	63,5	1,00	76,2	0,76
56,1	1,23	63,9	0,99		

Fonte: DNER-ME 043/95; p.07/11.

Portanto, conforme explica a norma DNER-ME 043/1995, o valor da deformação do corpo de prova é determinado simultaneamente ao da estabilidade. Durante a aplicação da carga, a luva-guia do medidor de deformação será firmada contra o topo do segmento superior do molde de compressão, diretamente sobre um dos pinos-guia. A pressão da mão sobre a luva do medidor de deformação deve ser relaxada, no momento em que se der o rompimento do corpo de prova, momento este em que será lido e anotado o valor da deformação.

Na Figura 02 é possível observar a ilustração de uma prensa que permite o registro automático da carga e da deformação.

Figura 2–Equipamento: Realização de ensaio de estabilidade Marshall

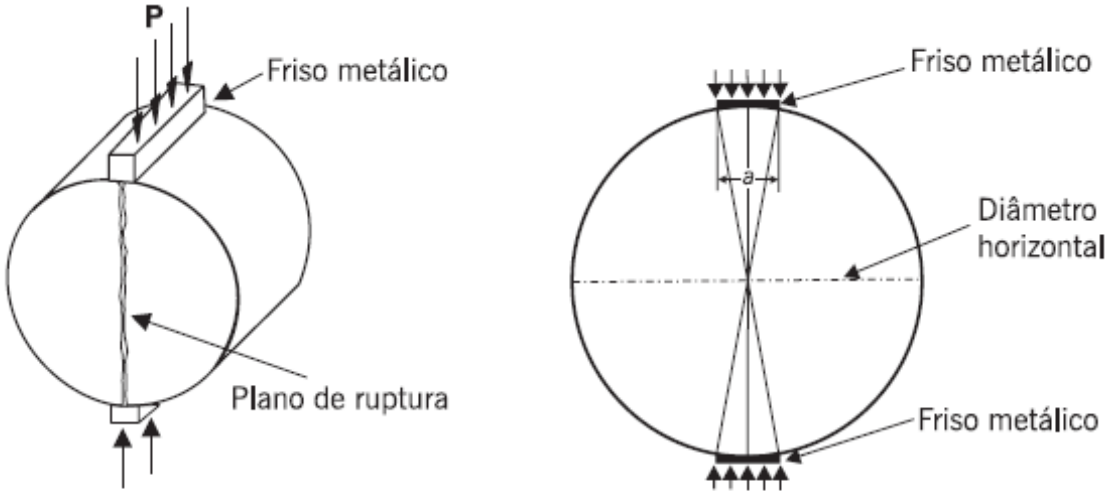
Fonte: Bernucci *et al*. (2008).

3.1.7 Ensaio de resistência à tração por compressão diametral

Segundo Bernucci *et al*. (2008), a configuração do ensaio para determinação da resistência à tração por compressão diametral considera a aplicação de forças de compressão, diametralmente opostas, em um corpo de prova cilíndrico, as quais geram forças de tração consideradas uniformes e que atuam perpendicularmente a direção das forças de compressão.

A aplicação das forças se dá por meio de frisos metálicos de 12,7 mm de largura com curvatura adequada ao corpo de prova cilíndrico (Figura 03). A ASTM D 4123-82 (1982) e o DNER (1994) não consideram a influência destes frisos no cálculo da Resistência à Tração por Compressão Diametral. De acordo com a expressão usada por estas entidades, assume-se que o comportamento do material é elástico durante o ensaio e a ruptura do corpo de prova é devida às tensões de tração geradas.

Figura 3 - Esquema do ensaio de resistência à tração por compressão diametral.



a – Corda do friso (12,7mm)
P – Carga aplicada

Fonte: Bernucci *et al*. (2008).

3.2Desenvolvimento do "Software"

A seguir serão explanadas as literaturas necessárias para compreensão dos materiais e métodos necessários e utilizados para o desenvolvimento do "software".

3.2.1Sistema de Gerenciamento de Banco de Dados "MySQL" 5.7.4

A seguir serão contempladas, de maneira sucinta, as principais definições de Sistema de Gerenciamento de Banco de Dados, segundo o manual de referência MySQL 5.7. Neste caso, as definições estarão principalmente relacionadas ao sistema MySQL 5.7.4, a ser utilizado para desenvolver o trabalho proposto.

Banco de dados é uma coleção de dados estruturados. Ele pode ser desde uma simples lista de compras a uma galeria de imagens ou a grande quantidade de informação de uma rede coorporativa. Para adicionar, acessar, e processar dados armazenados em um banco de dados de um computador, é necessário um sistema de gerenciamento de bancos de dados como o Servidor MySQL. Como os computadores apresentam significativo desempenho ao lidar com grandes quantidades de dados, o gerenciamento de bancos de dados funciona como a engrenagem central na computação, seja como utilitários independentes ou como partes de outras aplicações (Oracle Corporation, 2017).

O MySQL é um sistema de gerenciamento de bancos de dados relacional, ou seja, esse tipo de banco de dados armazena dados em tabelas separadas em vez de colocar todos os dados um só local. Isto proporciona velocidade e flexibilidade ao sistema. Além disto, este sistema de gerenciamento de banco de dados é classificado como um sistema Open Source, ou seja, é possível que o programa seja utilizado e/ou modificado por qualquer pessoa, de forma gratuita (Oracle Corporation, 2017).

Portanto, o Programa de Banco de Dados MySQL é um sistema cliente/servidor que consiste de um servidor SQL multitarefa que suporta diferentes acessos, diversos programas, clientes, bibliotecas, ferramentas administrativas e diversas interfaces de programação (Oracle Corporation, 2017).

3.2.2Linguagem de programação Java 8

Para a definição mais clara da linguagem de programação Java e suas extensões e aplicações, primeiramente será necessária a conceituação de algoritmo.

3.2.2.1 Conceituação de Algoritmo

Segundo Buffoni (2003), algoritmo é a descrição, de forma lógica, dos passos a serem executados no cumprimento de determinada tarefa. Ou seja, um algoritmo é formalmente uma sequência finita de passos que levam a execução de uma tarefa. Estas tarefas não podem ser redundantes nem subjetivas na sua definição e devem ser claras e precisas. Como exemplos de algoritmos, existem os algoritmos das operações básicas (adição, subtração, multiplicação e divisão) de números reais decimais.

Em geral, o algoritmo está associado ao processamento eletrônico de dados, onde representa o rascunho para programas (“software”). Como sua linguagem é intermediária à linguagem humana e às linguagens de programação, servem como modelos para desenvolvimento de programas, sendo então, uma boa ferramenta na validação da lógica de tarefas a serem automatizadas. Sendo assim, os algoritmos, apesar de servirem para representar a solução de qualquer problema, no caso do processamento de dados, eles devem seguir as regras básicas de programação para que sejam compatíveis com as linguagens de programação.(Buffoni, 2003).

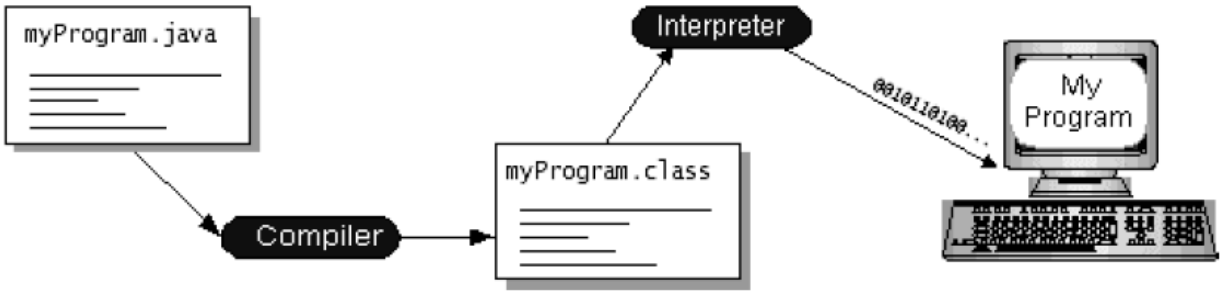
3.2.2.2 Linguagem de programação Java

Segundo Pereira (2009), originalmente desenvolvido por uma equipe de desenvolvedores liderada por James Gosling na “Sun Microsystems” (atualmente de propriedade da Oracle) e lançado em 1995, Java é uma linguagem de programação orientada a objetos que atualmente faz parte do núcleo da Plataforma Java.

A orientação a objetos é um tipo de paradigma de análise, para a programação de sistemas no qual todos os elementos inseridos são objetos. O desenvolvedor é responsável por modelar o papel desempenhado pelos objetos e a interação entre eles. Por exemplo, em um sistema desenvolvido para uma padaria, existiriam objetos do tipo "cliente" e objetos que simulam as ações que um cliente pode realizar (Pereira, 2009).

Segundo Mengue (2002), na maioria das linguagens de programação, é necessário compilar ou interpretar um programa para que ele seja executado. Ao fazer-se uso da linguagem Java, é possível tanto compilar quanto interpretar os programas. Com o compilador, inicialmente o programa é transformado em uma linguagem intermediária, conhecida por “bytecode”, que é independente de plataforma e posteriormente é interpretado pelo interpretador Java. Portanto, a compilação ocorre apenas uma vez, enquanto que a interpretação acontece todas as vezes em que o programa é executado. Na Figura 04 é possível observar como isto ocorre.

Figura 4- Representação da compilação einterpretação: Linguagem Java



Fonte: Mengue (2002).

3.2.2.3 Relações entre Classe, Objeto, Método, Atributo e Entidade

Segundo Ricarte (2000), uma classe é um conjunto de objetos, enquanto que uma entidade é um conjunto de atributos. O autor explica que uma classe é um gabarito para a definição de objetos. Por meio dela descreve-se que propriedades ou atributos o objeto terá. Além disso, a classe define qual o comportamento de objetos da classe. Esses comportamentos são descritos por meio de métodos, que são procedimentos ou funções, com a restrição de manipular apenas suas variáveis locais e os atributos que foram definidos para a classe.

Já segundo Sanches (2005), uma entidade é um objeto distinguível dos outros objetos. Por exemplo, uma pessoa com número de CPF 123.456.789-00 é uma entidade, visto que isso identifica unicamente uma pessoa particular do universo. Assim a conta número 40167-9 na agência Lapa é uma entidade que identifica unicamente uma conta corrente particular. Uma entidade pode ser concreta, como uma pessoa ou um livro, ou pode ser abstrata, como um feriado ou um conceito. Portanto, uma entidade pode ser classificada com um grupo de atributos.

3.2.2.4 “Model View Controller” (MVC)

Segundo Figueiredo (2015), o padrão MVC (“Model View Controller”), é constituído por três “aplicações”: “Model” (Modelo), “View” (Visão) e “Controller” (Controlador). Modelo, é tudo aquilo destinado à lógica da aplicação, ou seja, onde se localizam as classes, variáveis, informações e atributos. Visão é qualquer tipo de retorno de dado para uma interface qualquer (o navegador, por exemplo) é responsabilidade da Visão. Finalmente, o controlador é o responsável por definir quando as coisas devem acontecer e é usado para intermediar o modelo e a visão.

3.2.2.5Linguagem JPA (“Java Persistence API”)e “Framework Hibernate”

Segundo Filitto (2015), o JPA é uma coleção de classes e métodos voltados para armazenar persistentemente as vastas quantidades de dados em um banco de dados. Com base no JPA,o desenvolvimento de Frameworks (como o Hibernate) é feito com o objetivo de proporcionar uma interação com um banco de dados

relacional, de modo a evitar que o desenvolvedor gastetempo com o desenvolvimento de códigos voltados para a manipulação dos dados presentes no banco de dados. Sendo assim, o JPA proporciona meios de armazenar os dados presentes nos objetos implementados no “software” desenvolvido dentro das entidades no banco de dados.

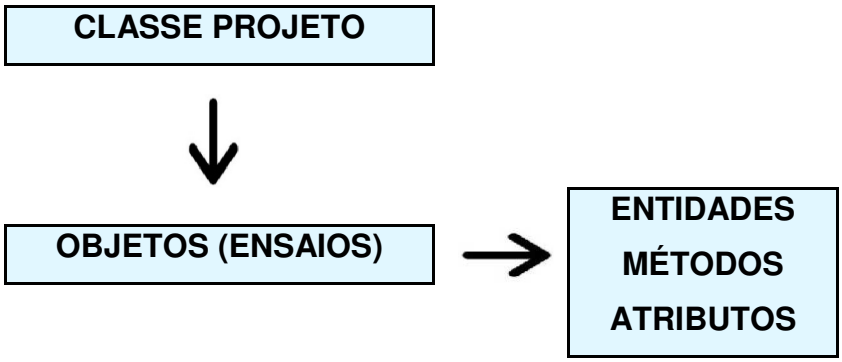
4 MATERIAIS E MÉTODOS EMPREGADOS

4.1 Materiais

4.1.1 Ferramentas utilizadas para o desenvolvimento do código do “software”

Por meio da linguagem Java 8, classes, entidades e objetos foram utilizados para o desenvolvimento do código do software. Os objetos estão contidos na classe denominada Projeto e a cada um dos objetos existentes (ensaios) estão atribuídos entidades, métodos e atributos, como representado na Figura 05.

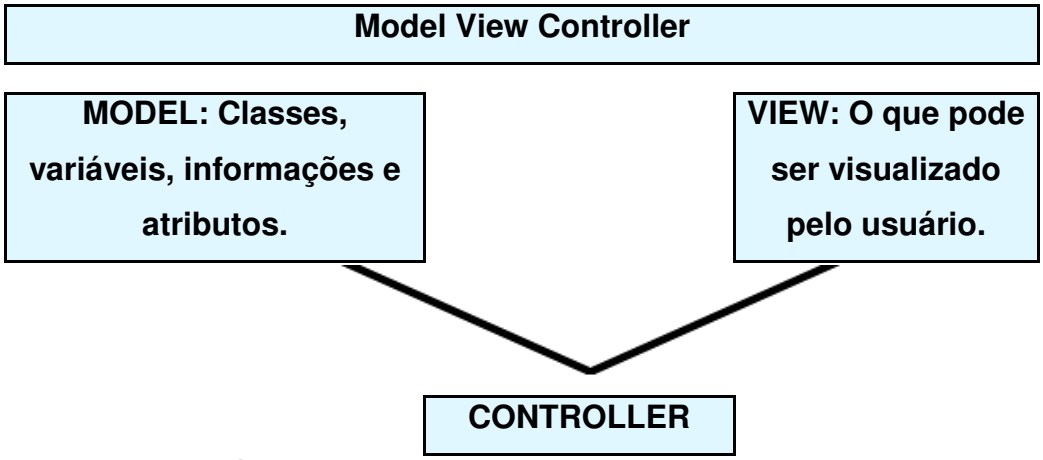
Figura 5 - Relação entre classes, objetos e entidades



Fonte: Autoria própria (2017).

Para trabalhar com as entidades dispostas dentro das classes, foi utilizado o “Model View Controller” (MVC), detalhado no item 3.2.2.4 e representado na Figura 06.

Figura 6 - Model View Controller

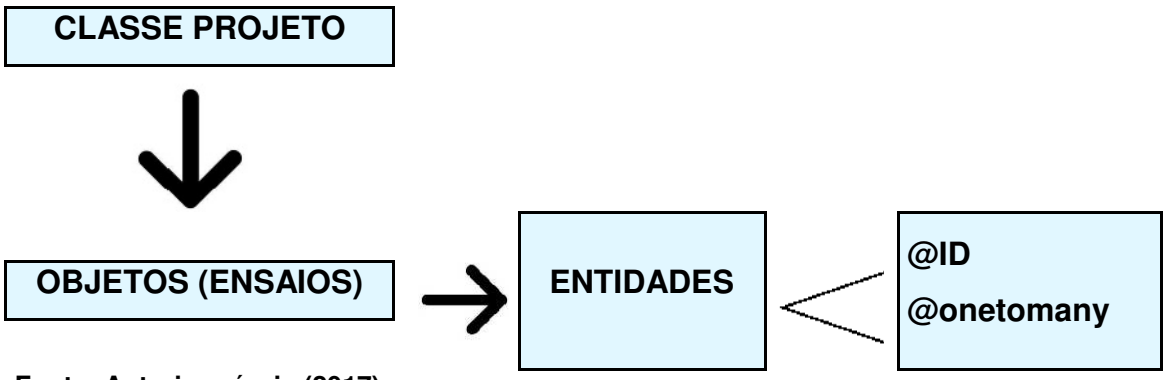


Fonte: Autoria própria (2017).

As classes, variáveis, informações e atributos foram atribuídos ao “Model”. Tudo o que pode ser visualizado pelo usuário foi atribuído à “View” e ao “Controller” realiza a interação entre o Model e a View.

Já para o armazenamento das entidades em banco de dados, foi utilizado o “Framework Hibernate”, disponível na linguagem JPA. Este “Framework” têm a capacidade de mapear as classes presentes no “software” automaticamente para o banco de dados. Cada entidade possui um ID e um código (@onetomany) para armazenamento, por meio do sistema de gerenciamento de banco de dados MySQL. A representação da utilização deste “framework” é mostrada na Figura 07.

Figura 7 - Armazenamento de entidades em banco de dados



Fonte: Autoria própria (2017).

4.1.2 Ferramentas necessárias para o funcionamento do “software”

A faixa de trabalho inicialmente será inserida no “software”e poderá ser editada caso a especificação dos materiais a serem utilizados seja alterada. Estes dados estão dispostos na norma DNER-ME 043/1995 e são descritos no Quadro 1.

Quadro 1 - Dados de entrada para o "software": definição da faixa de trabalho

Definição da faixa de trabalho: valores máximos e mínimos (limites)
Percentuais de agregados passantes em cada peneira
Teor de asfalto (betume) presente na mistura asfáltica
Estabilidade (kgf) que os corpos de prova ensaiados devem apresentar
Deformação que os corpos de prova ensaiados devem apresentar
Resistência à tração (MPa) dos corpos de prova
Densidade aparente (kg/cm³) da mistura asfáltica
Densidade real (kg/cm³) da mistura asfáltica
Porosidade da mistura asfáltica
Percentual de poros preenchidos com betume na mistura asfáltica
Percentual de poros presente no agregado mineral da mistura asfáltica
Relação betume/poros presentes na mistura asfáltica

Fonte: Autoria própria (2017).

Após a obtenção de todos os limites listados no Quadro 1, por meio do software é possível que os usuários façam a edição dos valores ou informem uma faixa de trabalho distinta, caso o projeto da mistura asfáltica seja alterado, ou seja necessário trabalhar com outro tipo de mistura asfáltica.

Com a faixa de trabalho pré-estabelecida inserida na memória do "software", é possível iniciar os processos de análise e verificação, para armazenamento de dados e resultados oriundos da realização de ensaios Marshall. Portanto, será necessário que os usuários informem ao "software" os dados descritos no Quadro 2.

Quadro 2 - Dados de entrada para o "software": ensaio Marshall

Dados de entrada para obtenção de resultados de ensaios Marshall
Rodovia na qual a mistura asfáltica será empregada
Data de realização dos ensaios
Informações sobre a granulometria dos agregados da mistura asfáltica utilizados na composição dos corpos de prova do ensaio Marshall (massas, antes e depois da realização da granulometria e as massas retidas nas peneiras)
Massas das amostras (g), medidas ao ar e quando imersas em água
Ruptura por estabilidade: alturas (cm) dos corpos de prova; força (N) suportada pelo corpo de prova na ruptura e deformação (%) devido à aplicação da força
Ruptura por tração: alturas (cm) dos corpos de prova; força (N) suportada pelo corpo de prova na ruptura e deformação(%) devido à aplicação da força

Fonte: Autoria própria (2017).

4.3 Métodos

4.3.1 Métodos utilizados para o desenvolvimento do “software”

Inicialmente foi criada a classe “Projeto” onde todas as especificações, métodos e cálculos de um ensaio Marshall estão armazenados. Dentro da linguagem de programação Java 8, os ensaios Marshall são chamados de “Objetos”, pois uma classe armazena um conjunto de objetos com características similares. Estes objetos são constituídos por “entidades”, responsáveis por tratar e armazenar dados. Ou seja, cada entidade é responsável por armazenar um tipo específico de dado da classe.

Além disto, todos os cálculos e métodos necessários são constituintes da classe “Projeto”, uma vez que os cálculos são realizados em cadeia. Portanto, conforme a necessidade do “software” ou conforme o que foi solicitado pelo usuário, dentro da classe “Projeto”, um método aciona o próximo método em “cadeia” para que o cálculo necessário possa ser realizado. Esse tipo de abordagem permite que os métodos trabalhem interativamente sem a necessidade de que todos os cálculos sejam realizados somente por um método, além de permitir que cada método possua um “papel específico” dentro do “software”.

O código do “software” foi desenvolvidopor meio do padrão MVC. Este tipo de padrão foi utilizado para que todas as classes, informações e atributos sejam de responsabilidade do “Model”, enquanto que tudo aquilo que for acessível ao usuário se enquadre nas especificações determinadas pelo “View”. Finalmente, tudo o que estiver contido em “Model” e “View” étratado pelo “Controller”, que é responsável por “unir” model e view e controlar as interações entre eles.

4.3.2 Armazenamento de dados em banco de dados

Todos os dados que foreminseridos no “software”serão armazenados por meio do uso de códigos desenvolvidos conforme a linguagem JPA, devido a esta possuir uma vasta coleção de classes e métodos voltados para o armazenamento de dados em banco de dados.

4.3.3Funcionamento do “software”

Inicialmente, o “software” armazena os parâmetros previamente estabelecidos no projeto da mistura asfáltica (ver Quadro 1). Estes parâmetros devem ser inseridos pelo usuário, como dados iniciais.

Após realização de ensaios Marshall e com a faixa de trabalho do projeto da mistura asfáltica armazenada no “software”, os usuários podem realizar o lançamento dos dados obtidos por meio deste ensaio. O “software”, portanto, é capaz de realizar os procedimentos de análise e verificar se os dados obtidos a partir dos ensaios Marshall são adequados para os limites da faixa de trabalho previamente estabelecidos.

Para os resultados que não estiverem de acordo com os limites da faixa de trabalho pré-estabelecidos, o ”software” é capaz de indicá-los para que o(s) problema(s) na mistura asfáltica seja(m) verificado(s) pelo órgão fiscalizador e posteriormente sanado(s) pela empresa responsável pela construção e/ou reconstrução asfáltica. No entanto, é necessária a realização de outro ensaio para assegurar a qualidade da mistura asfáltica, já que a edição e/ou exclusão de dados deensaios já lançados e armazenados não é permitida aos usuários.

A realização dos ensaios Marshall é responsabilidade da empresa contratada paraa construção e/ou reconstrução asfáltica. Já a verificação da conformidade dos resultados obtidos por meio dos ensaios com a faixa de trabalho do projeto da mistura asfáltica é responsabilidade do órgão fiscalizador. Portanto, o “software” desenvolvido neste trabalho é destinado aos fiscais, para que estes verifiquem os resultados de ensaios Marshall e tomem as medidas necessárias caso os ensaios não estejam em conformidade com o que foi pré-estabelecido.

Finalmente, todos os dados obtidos por meio de ensaios Marshall, inseridos pelos usuários, são armazenados e ficam disponíveis para pesquisas e consultas, mediante a base de dadosdo “software” desenvolvido.

5 APRESENTAÇÃO DOS RESULTADOS

Este capítulo será desenvolvido por meio da apresentação de dois exemplos de inserção de dados de resultados de ensaios Marshall no “software”. Um destes exemplos está enquadrado dentro dos limites da faixa de trabalho da mistura asfáltica inicialmente inserida no “software” enquanto que o outro não está enquadrado dentro destes limites.

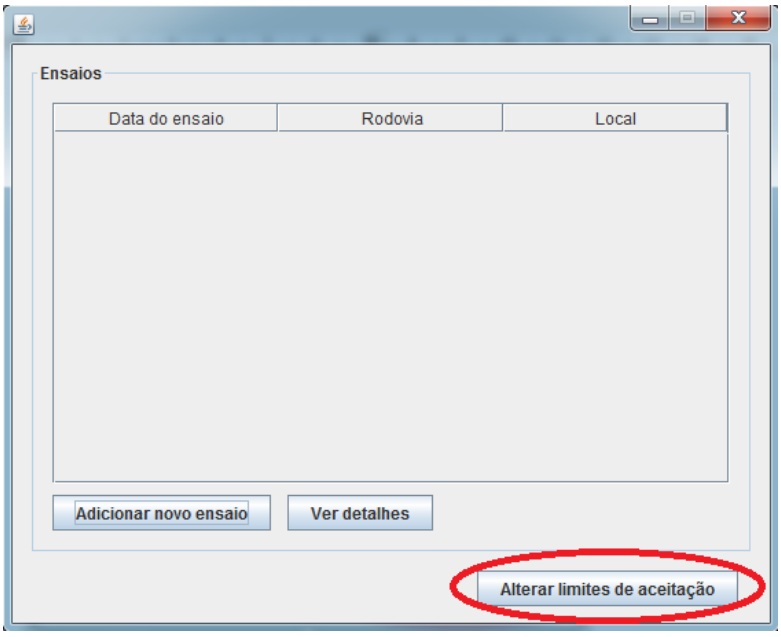
5.1 Dados de entrada para o “software”

5.1.1 Faixa de trabalho da mistura asfáltica

Antes de projetos de construção e/ou restauração de pavimentos asfálticos serem iniciados, para o uso do CBUQ (Concreto Betuminoso Usinado à Quente), é necessária a realização de um projeto da mistura asfáltica. Neste projeto, conforme os materiais a serem empregados, são estabelecidos limites (máximos e mínimos) para garantir qualidade e vida útil adequadas aos pavimentos.

A Figura 08 apresenta a tela inicial do “software”. Ao clicar em “Alterar limites de aceitação”, os dados referentes à faixa de trabalho da mistura asfáltica poderão ser inseridos no “software”.

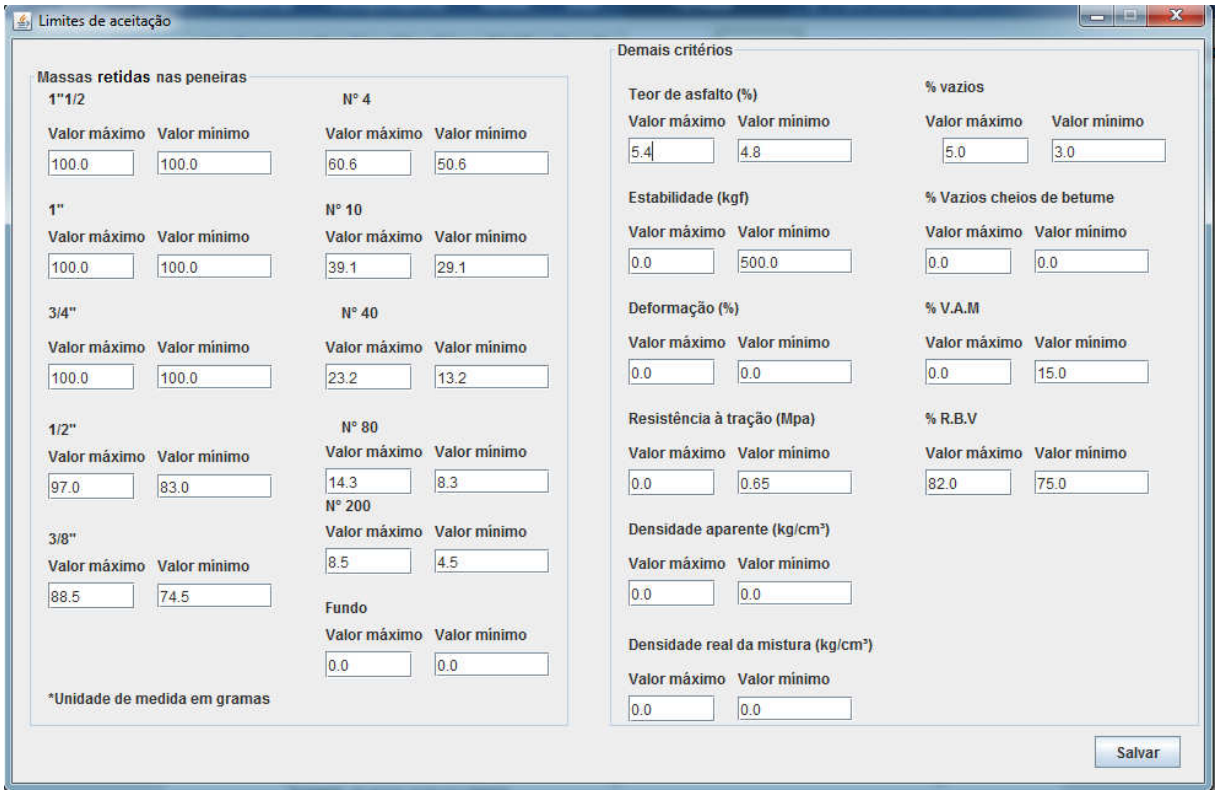
Figura 8 - Tela inicial do “software”: Alteração dos limites de aceitação



FONTE: Autoria própria (2017).

A Figura 09 representa a tela onde o usuário poderá fornecer os dados da faixa de trabalho oriunda do Projeto da Mistura Asfáltica. A faixa de trabalho representadaé fictícia e está disponível como anexo neste trabalho. Além disto, será utilizada para simular o uso de materiais similares ao deste projeto.

Figura 9– Inserção dos dados da faixa de trabalho da mistura asfáltica.



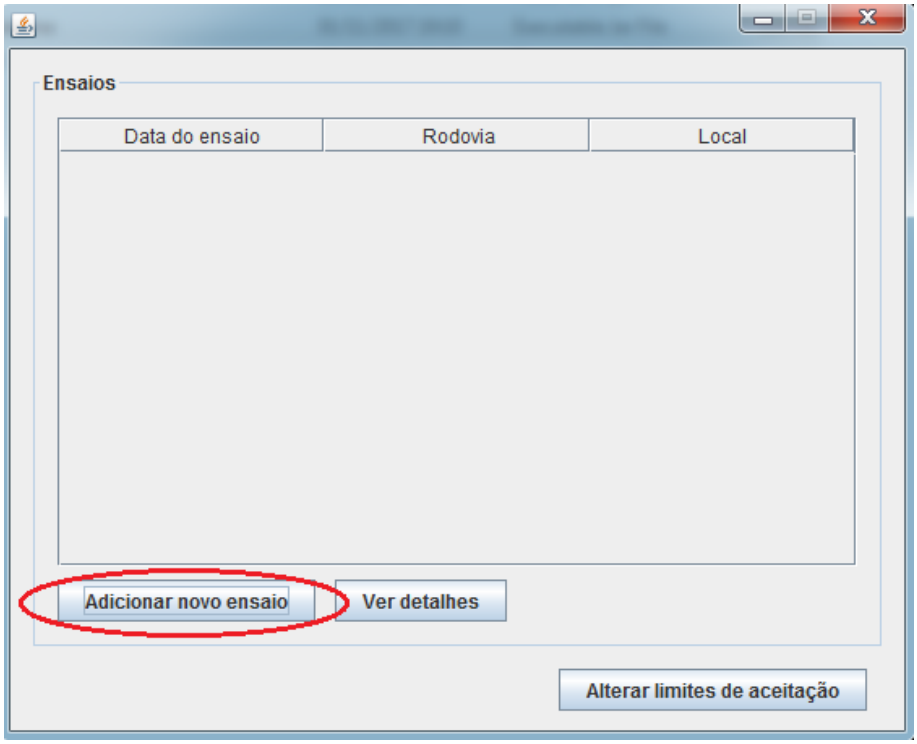
FONTE: Autoria própria (2017).

Após fornecer os dados da faixa de trabalho solicitados pelo “software”, é necessário que o usuário clique em “Salvar” para que estes dados sejam armazenados. Vale ressaltar que caso o Projeto da Mistura Asfáltica necessite ser alterado (mudança dos materiais empregados ou do tipo da mistura asfáltica), é possível que o usuário edite estas informações mesmoapós salvar os dados.

5.1.2 Armazenamento de novos ensaios Marshall no “software”

A Figura 10 apresenta a tela inicial do “software”, onde o usuário poderá inserir novos ensaios Marshall no banco de dados. Para isto, o usuário deve clicar no botão “Adicionar novo ensaio”.

Figura 10 - Tela inicial do "software":Adição de um novo ensaio



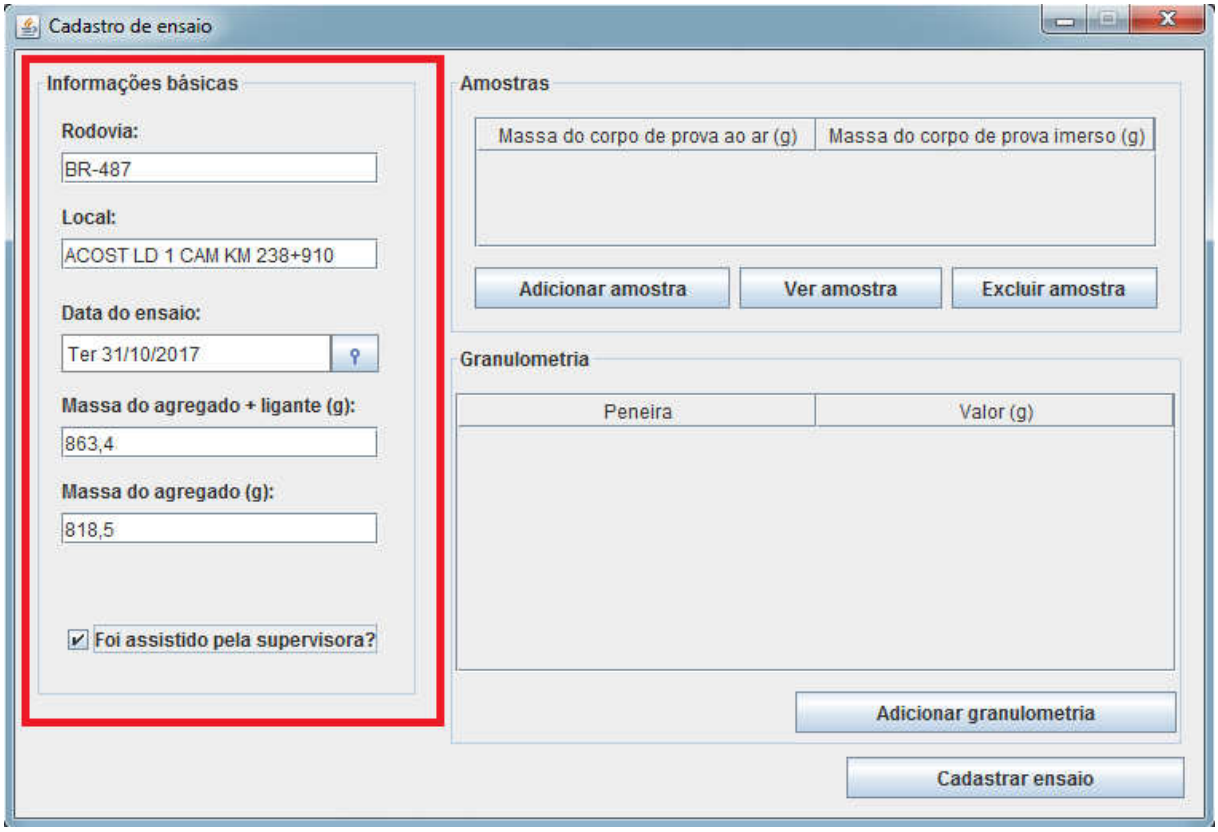
FONTE: Autoria própria (2017).

A inserção de novos Ensaios Marshall é dividida em três partes: Informações básicas, Amostras e Granulometria. A seguir, será demonstrada a inserção de um novo Ensaio Marshall.

5.1.2.1 Informações Básicas

A Figura 11 representa onde as informações básicas necessárias devem ser inseridas: Rodovia, Local, Data do Ensaio, Massa do Agregado mais ligante(g) e Massa do Agregado (g).

Figura 11 - Informações básicas do ensaio Marshall

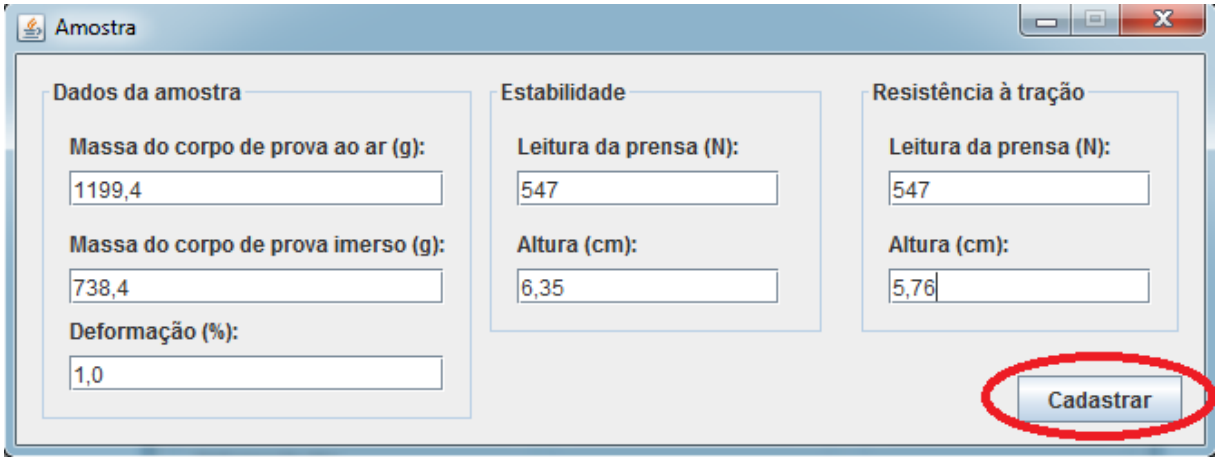


FONTE: Autoria própria (2017).

5.1.2.2 Amostras

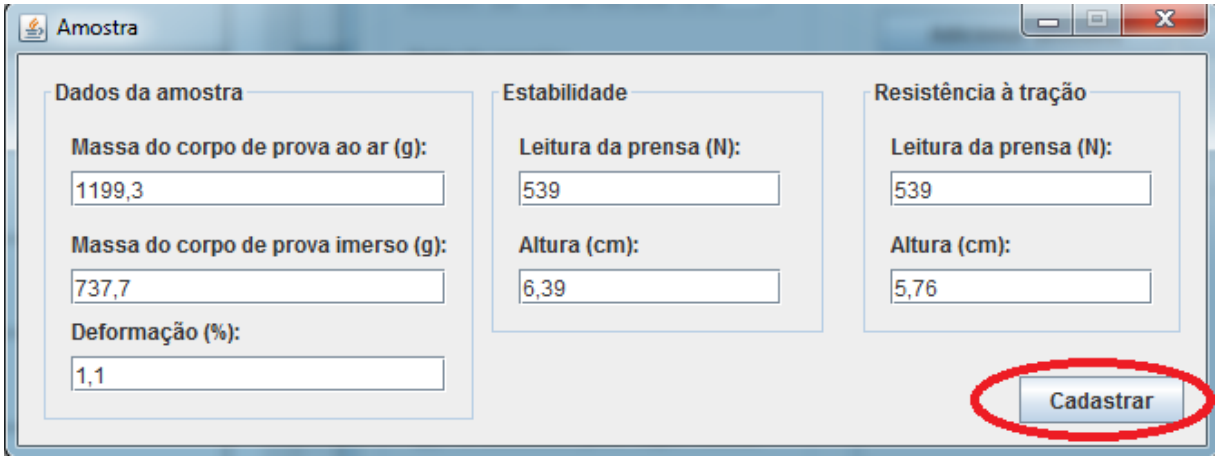
Nas Figuras 12, 13 e 14 estão representadas as inserções dos resultados de três amostras de ensaio Marshall, conforme a norma DNER-ME 043/1995. Esta tela é disponibilizada após o usuário clicar no botão “Adicionar amostra” na tela de cadastro de ensaio. Nesta categoria devem ser inseridas as massas do corpo de prova (g) pesado ao ar e imerso em água, além da Leitura da prensa (N) e das alturas do corpo de prova (cm) aferidos durante os rompimentos por estabilidade e por resistência à tração por compressão diametral. Após o término do preenchimento do formulário do “software” para cada amostra, é necessário que o usuário clique em “Cadastrar” e para que outra amostra seja adicionada em seguida, é necessário que o mesmo selecione o botão “Adicionar amostra”.

Figura 12 - Inserção dos dados da primeira amostra de ensaio Marshall



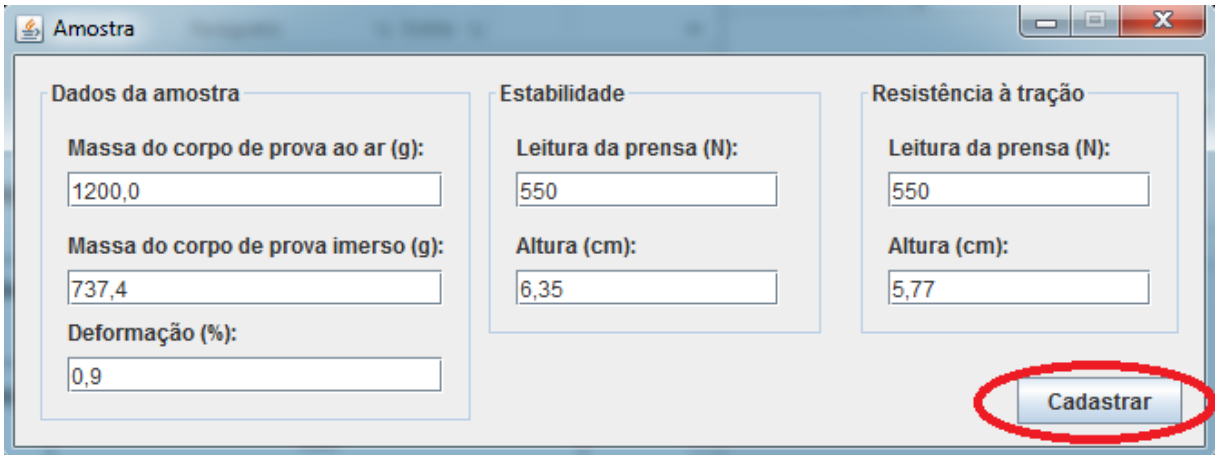
FONTE: Autoria própria (2017).

Figura 13 - Inserção dos dados da segunda amostra de ensaio Marshall



FONTE: Autoria própria (2017).

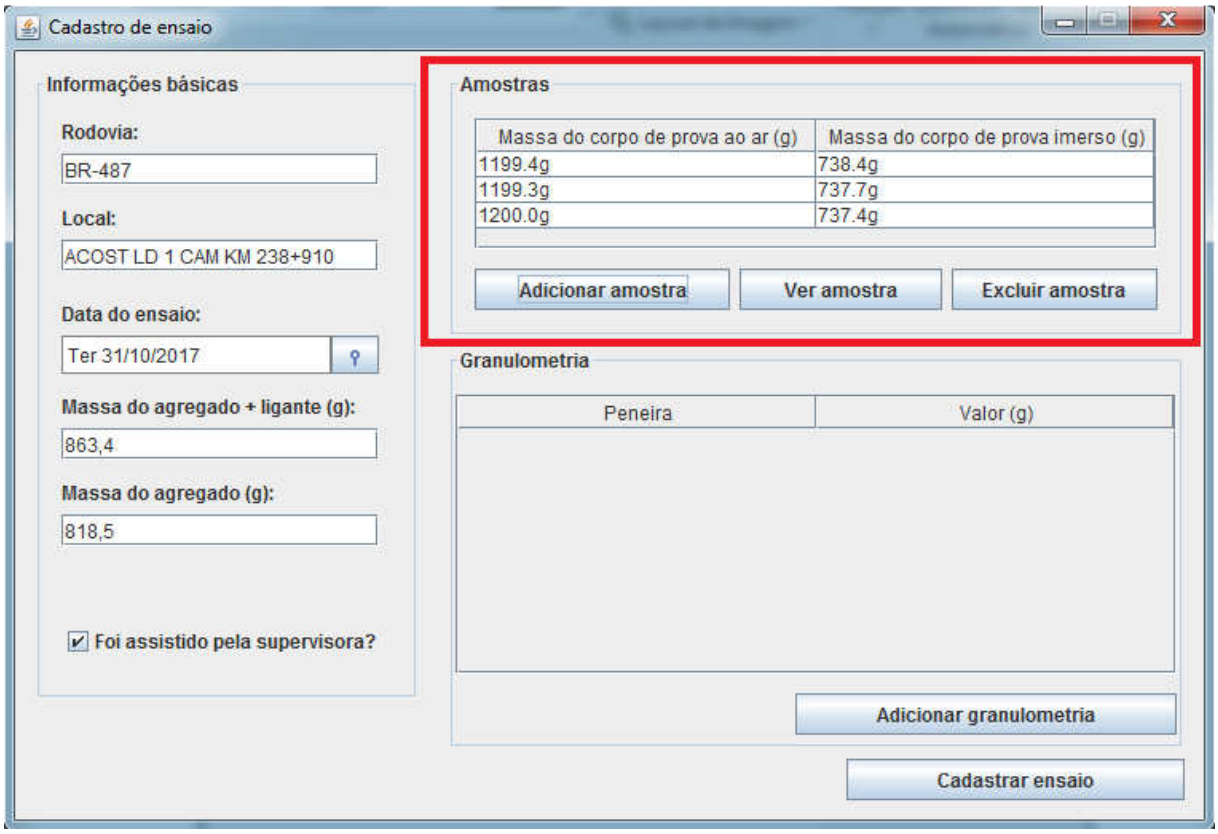
Figura 14 - Inserção dos dados da terceira amostra de ensaio Marshall



FONTE: Autoria própria (2017).

Na Figura 15 é possível observar o resumo das amostras inseridas com as massas dos corpos de prova ao ar(g) e imersas(g), gerado pelo “software”.

Figura 15 - Resumo das amostras inseridas no “software”

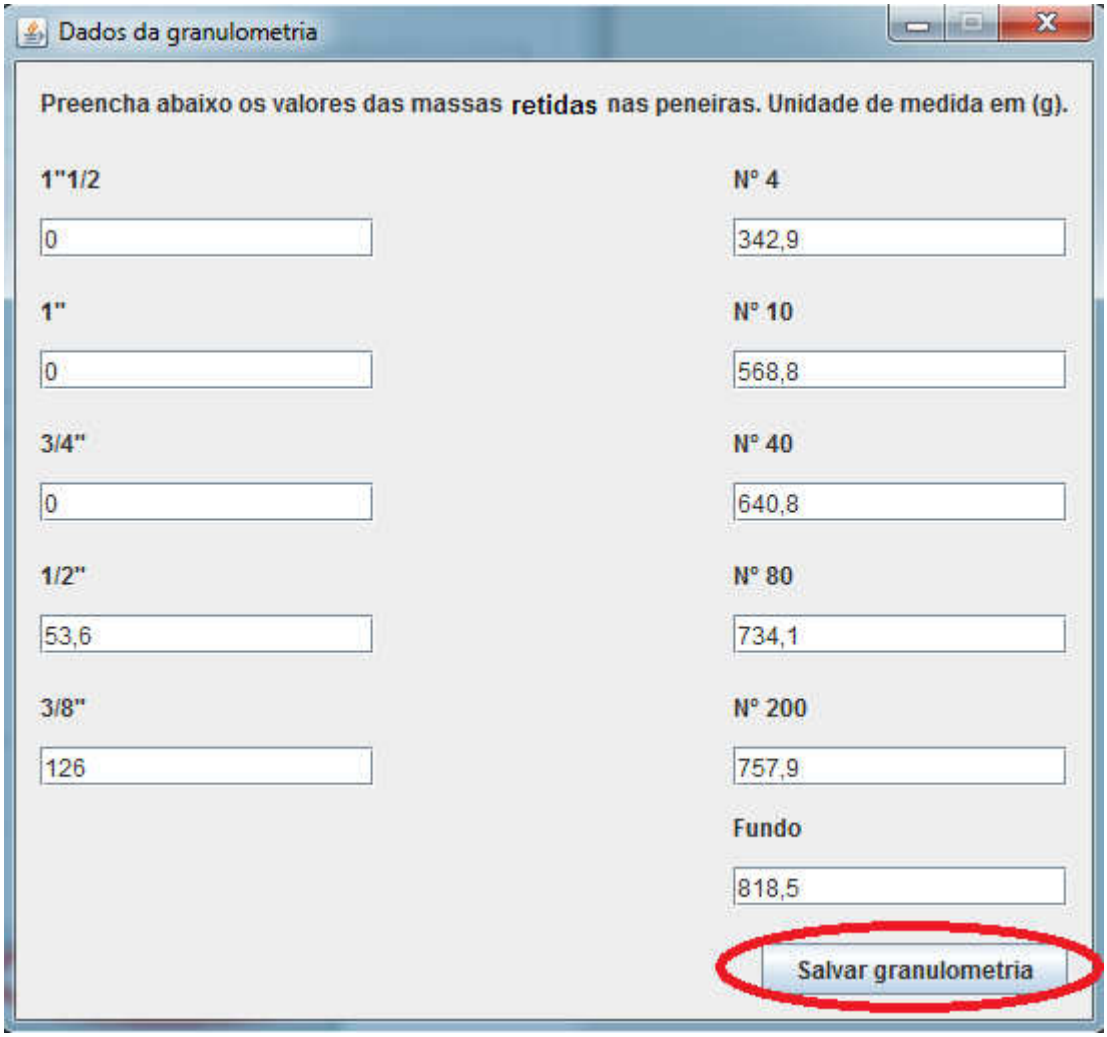


FONTE: Autoria própria (2017).

5.1.2.3 Granulometria

Na Figura 16 estão representados os dados da granulometria do ensaio Marshall a ser cadastrado, ou seja, a massa retida nas peneiras. Para que esta tela seja disponibilizada, é necessário que o usuário clique no botão “Adicionar granulometria”, e depois de inseridos todas as massas retidas nas peneiras, é necessário que o usuário clique no botão “Salvar granulometria”.

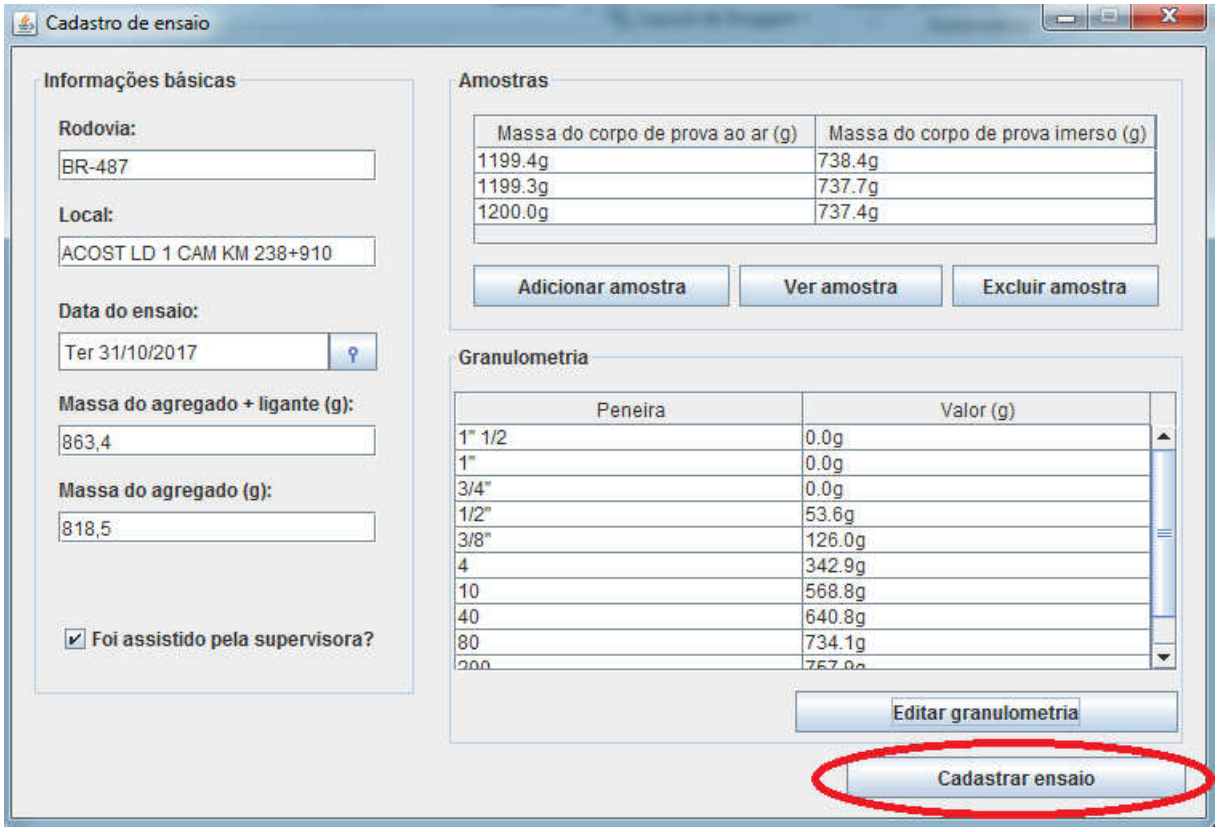
Figura 16 - Granulometria do ensaio Marshall



FONTE: Autoria própria (2017).

Na Figura 17 é possível observar a representação do resumo de todas as informações do ensaio Marshall inseridas no “software”. Este resumo permite ao usuário “conferir” todos os dados do Ensaio Marshall que foram inseridos, pois após clicar no botão “Cadastrar ensaio” o ensaio não poderá ser editado ou excluído.

Figura 17 - Resumo dos dados do primeiro ensaio Marshall cadastrado



FONTE: Autoria própria (2017).

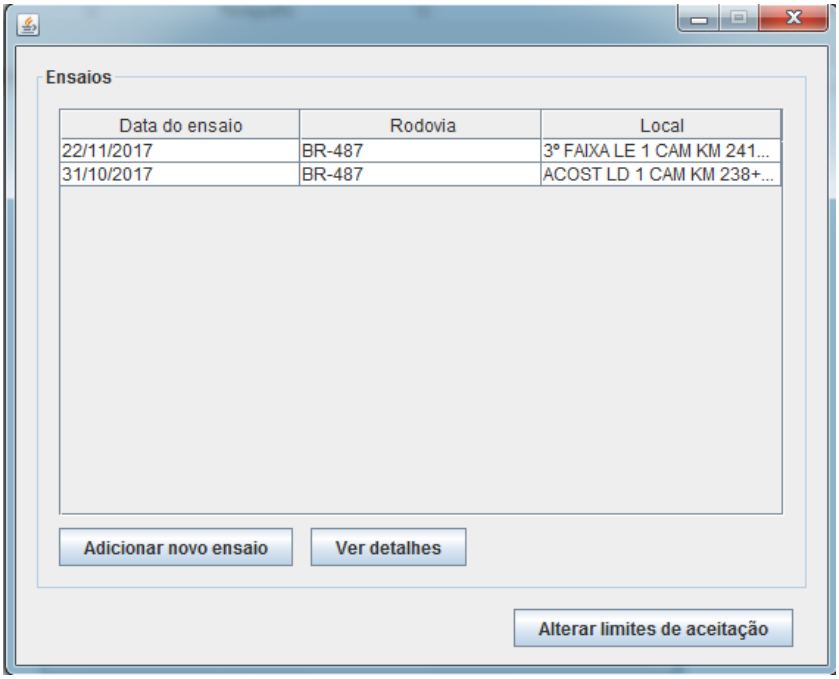
5.2 Resultados apresentados pelo “software”

Após inserir os dados obtidos a partir da realização dosensaios Marshall no “software”, estes se encontrarãoarmazenado no banco de dados deste e será possível então, constatar aceitação ou não para a faixa de trabalho do Projeto da Mistura Asfáltica inicialmente cadastrada.

Vale ressaltar aqui que um segundo ensaio (fora dos limites da faixa de trabalho do projeto da mistura asfáltica) foi cadastrado para fins comparativos da capacidade do “software”. O cadastro foi realizado conforme o passo a passo descrito no item 5.1.

Na Figura 18 é possível observar que os dois ensaios Marshall foram cadastrados no banco de dados do “software”. Ao clicar duas vezes sobre um dos ensaios ou clicar no botão “Ver detalhes” após selecionar um dos ensaios, é possível observar os resultados do ensaio cadastrado.

Figura 18 - Ensaios cadastrados no banco de dados do "software"



FONTE: Autoria própria (2017).

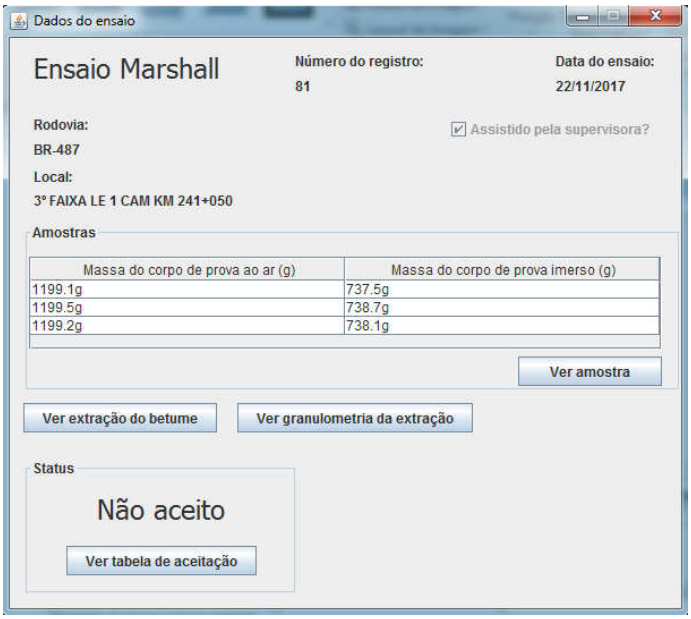
Na Figura 19 os resultados do primeiro ensaio Marshall inserido (Aceito) estão divididos em quatro partes: Ver amostra; Ver extração do betume; Ver granulometria da extração e Ver tabela de aceitação.

Figura 19 - Detalhes do primeiro ensaio cadastrado no "software"

FONTE: Autoria própria (2017).

Já na Figura 20 é possível observar os resultados do segundo ensaio Marshall inserido no “software” (Não aceito).

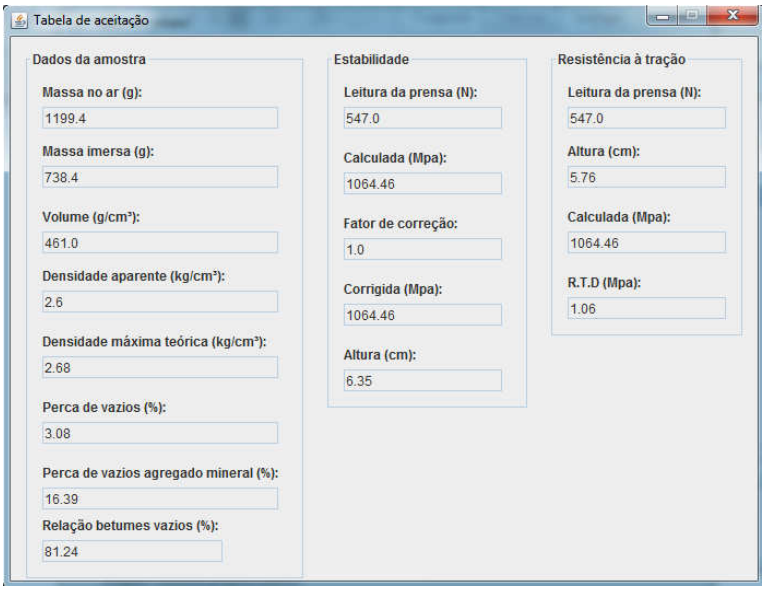
Figura 20 - Detalhes do segundo ensaio cadastrado no “software”



FONTE: Autoria própria (2017).

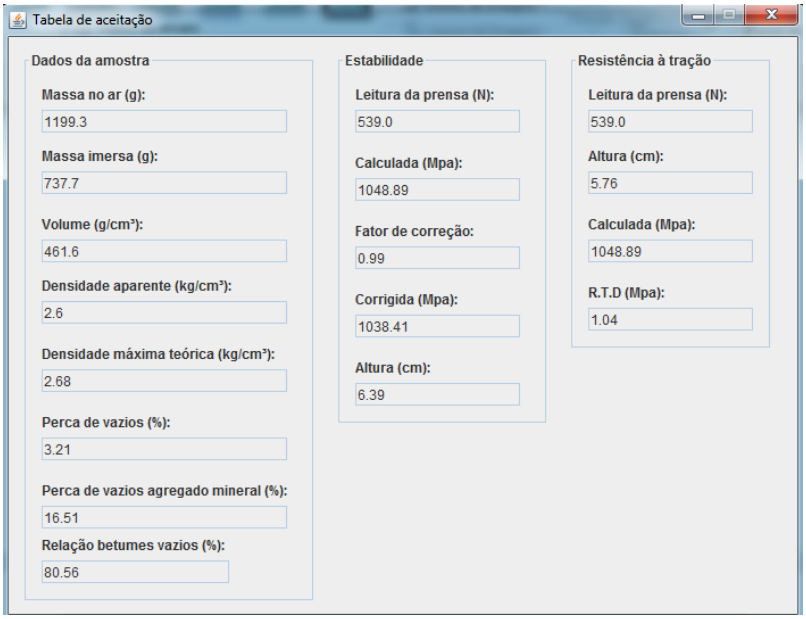
Nas Figuras 21, 22 e 23 é notável a representação dos resultados dos cálculos das três amostras do primeiro ensaio Marshall inseridas no “software”. Para isto, é necessário que o usuário selecione uma amostra e clique no botão “Ver amostra.

Figura 21 - Resultados referentes à primeira amostra do primeiro ensaio cadastrado



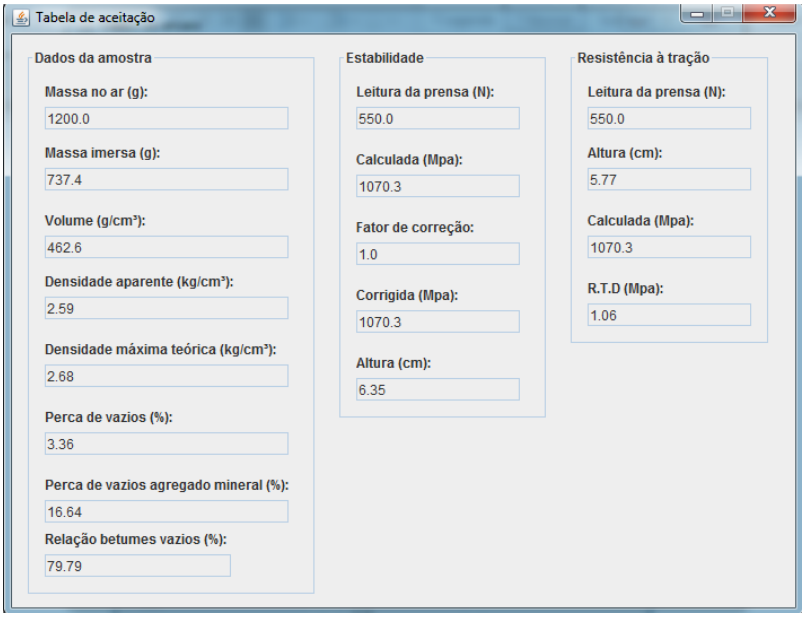
FONTE: Autoria própria (2017).

Figura 22 - Resultados referentes à segunda amostra do primeiro ensaio cadastrado



FONTE: Autoria própria (2017).

Figura 23 - Resultados referentes à terceira amostra do primeiro ensaio cadastrado



FONTE: Autoria própria (2017).

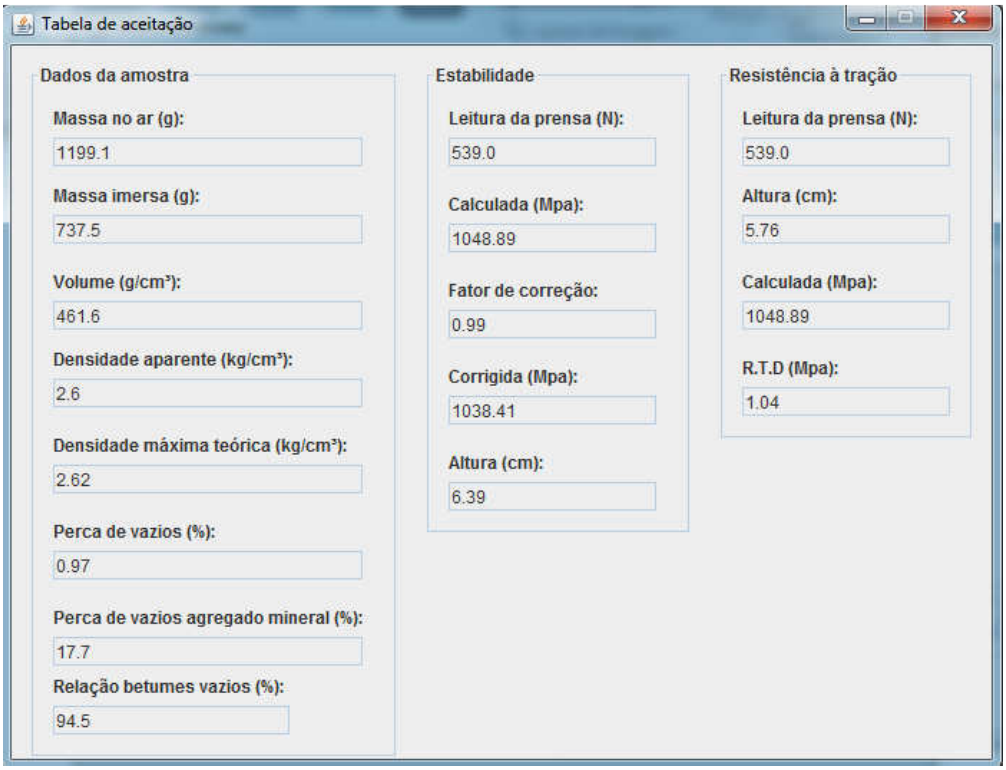
Nas Figuras 24, 25 e 26 é possível observar a representação dos resultados dos cálculos das três amostras do segundo ensaio Marshall inseridas no “software”.

Estes resultados se baseiam nos dados da amostra inicialmente inseridos. Com estes dados o “software” retorna ao usuário os seguintes parâmetros:

- Volume do corpo de prova (cm³);
- Densidade aparente do corpo de prova (kg/cm³);

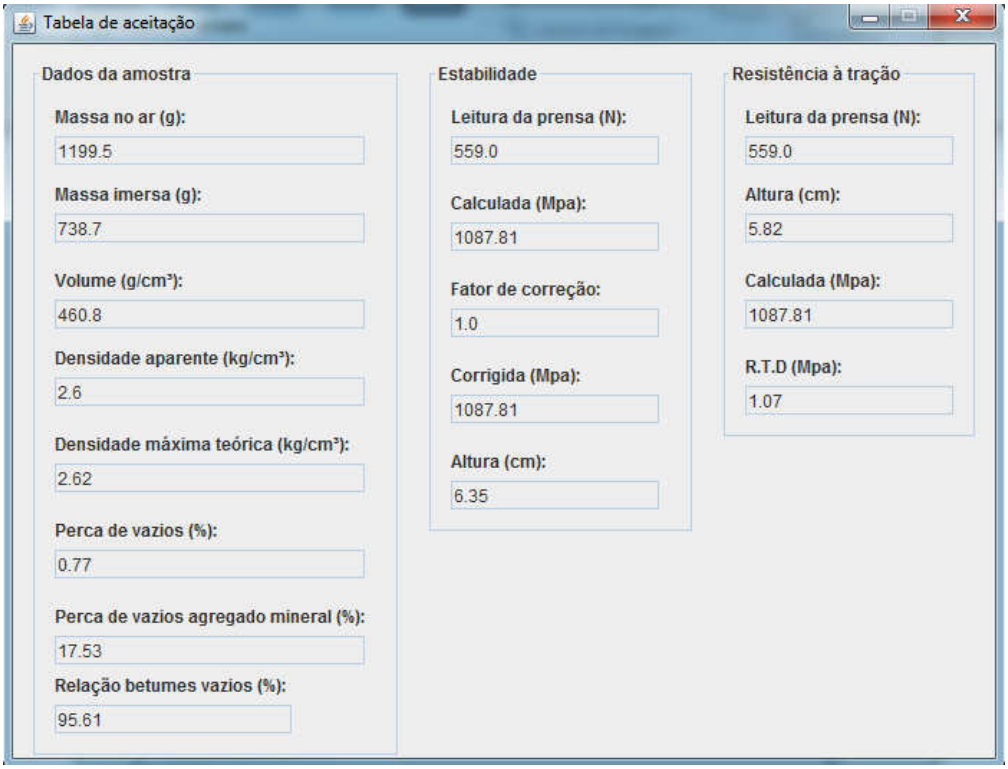
- Densidade máxima teórica do corpo de prova (kg/cm³);
- Perca de vazios do corpo de prova (%);
- Perca de vazios do agregado mineral do corpo de prova (%);
- Relação betume-vazios (%);
- Leitura da prensa (N) para estabilidade;
- Resistência calculada (MPa) para estabilidade;
- Fator de correção para estabilidade;
- Resistência corrigida (MPa) para estabilidade;
- Altura do corpo de prova (cm) para estabilidade;
- Leitura da prensa (N) para resistência à tração;
- Altura do corpo de prova (cm) para resistência à tração;
- Resistência calculada (MPa) para resistência à tração;
- Resistência à tração (MPa);

Figura 24 - Resultados referentes à primeira amostra do segundo ensaio cadastrado



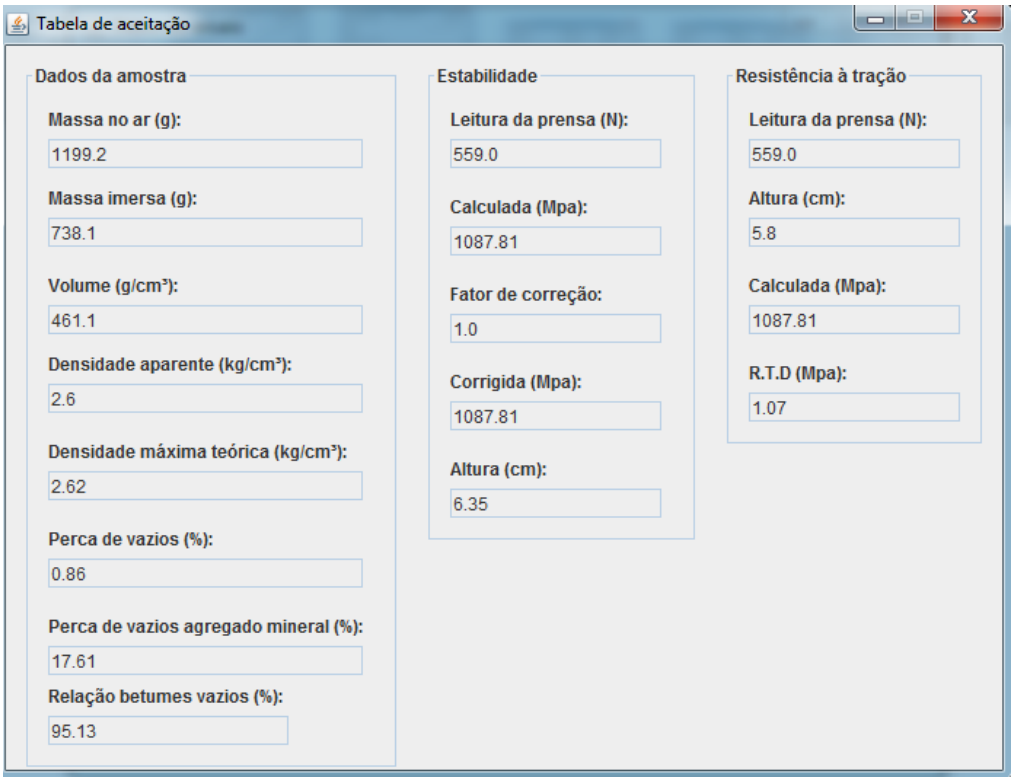
FONTE: Autoria própria (2017).

Figura 25 - Resultados referentes à segunda amostra do segundo ensaio cadastrado



FONTE: Autoria própria (2017).

Figura 26 - Resultados referentes à terceira amostra do segundo ensaio cadastrado



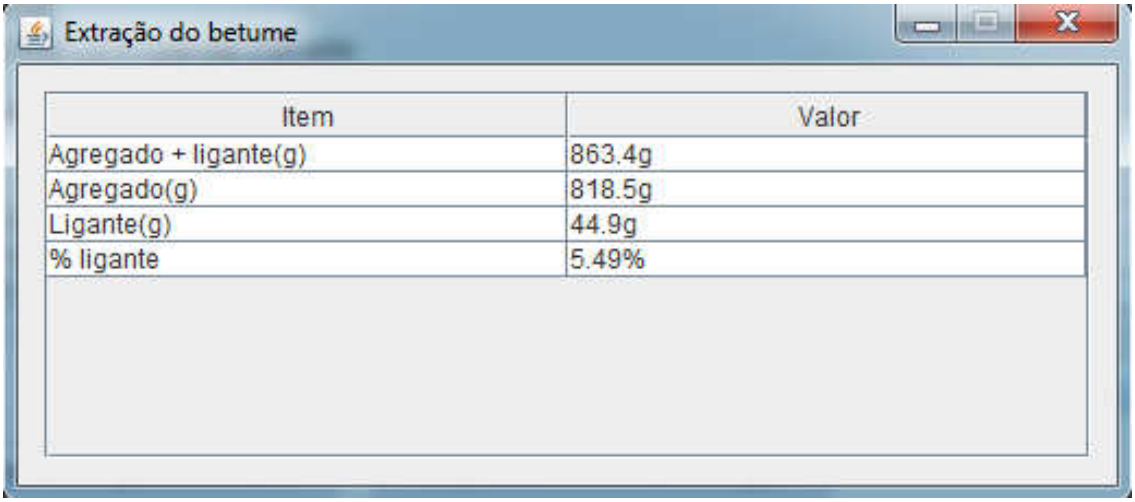
FONTE: Autoria própria (2017).

Na Figura 27 estão representados os resultados oriundos da extração do betume do primeiro ensaio cadastrado no “software”, enquanto que na Figura 28 estão representados os resultados oriundos da extração do betume do segundo ensaio cadastrado no “software”.

Esta tela é disponibilizada após o usuário clicar no botão “Ver extração do betume”. Os resultados são os seguintes:

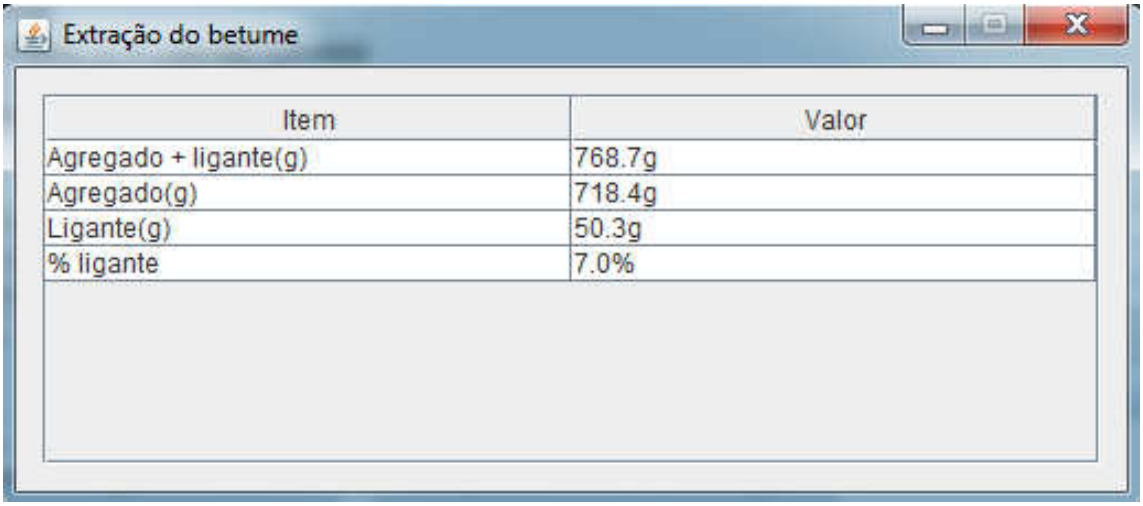
- Massa (g) do agregado mais o ligante;
- Massa (g) do agregado;
- Massa (g) do ligante constituinte da mistura asfáltica;
- Porcentagem de ligante presente na mistura asfáltica.

Figura 27 - Resultados referentes à Extração do betume do primeiro ensaio cadastrado



FONTE: Autoria própria (2017).

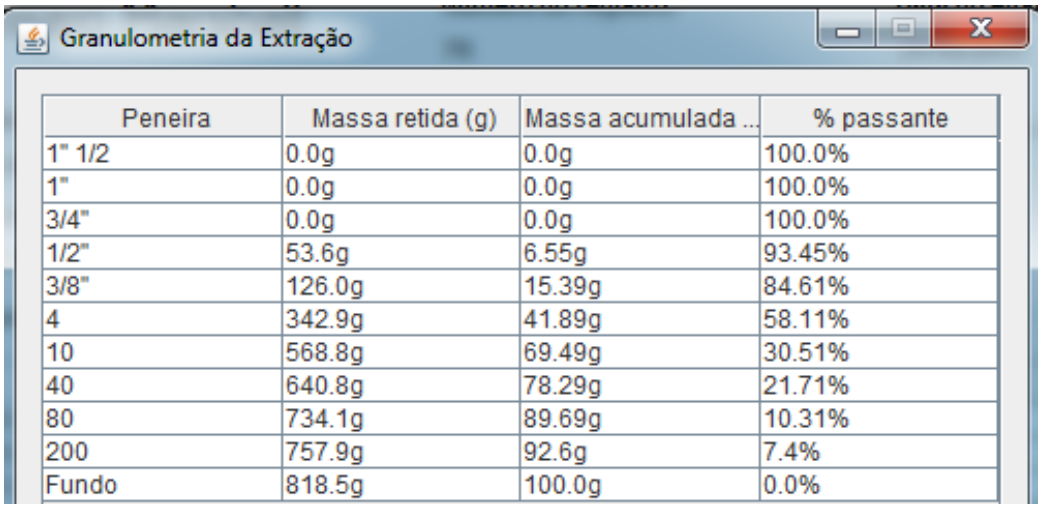
Figura 28 - Resultados referentes à Extração do betume do segundo ensaio cadastrado



FONTE: Autoria própria (2017).

Na Figura 29 estão representados os resultados da granulometria da extração do primeiro ensaio Marshall cadastrado no “software”. Esta tela é disponibilizada após o usuário selecionar o botão “Ver granulometria da extração”. São disponibilizadas as massas (g) retidas e acumuladas em cada peneira, além da porcentagem passante.

Figura 29 -Resultados referentes à Granulometria da Extração do betume do primeiro ensaio

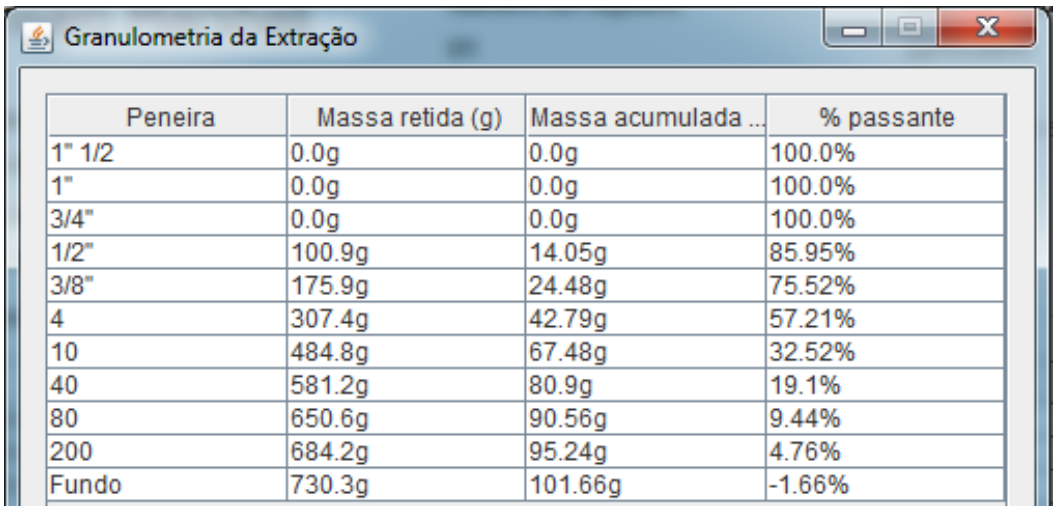


Peneira	Massa retida (g)	Massa acumulada ...	% passante
1" 1/2	0.0g	0.0g	100.0%
1"	0.0g	0.0g	100.0%
3/4"	0.0g	0.0g	100.0%
1/2"	53.6g	6.55g	93.45%
3/8"	126.0g	15.39g	84.61%
4	342.9g	41.89g	58.11%
10	568.8g	69.49g	30.51%
40	640.8g	78.29g	21.71%
80	734.1g	89.69g	10.31%
200	757.9g	92.6g	7.4%
Fundo	818.5g	100.0g	0.0%

FONTE: Autoria própria (2017).

Já na Figura 30 estão representados os resultados da granulometria da extração do segundo ensaio Marshall cadastrado no “software”.

Figura 30 - Resultados referentes à Granulometria da Extração do betume do segundo ensaio

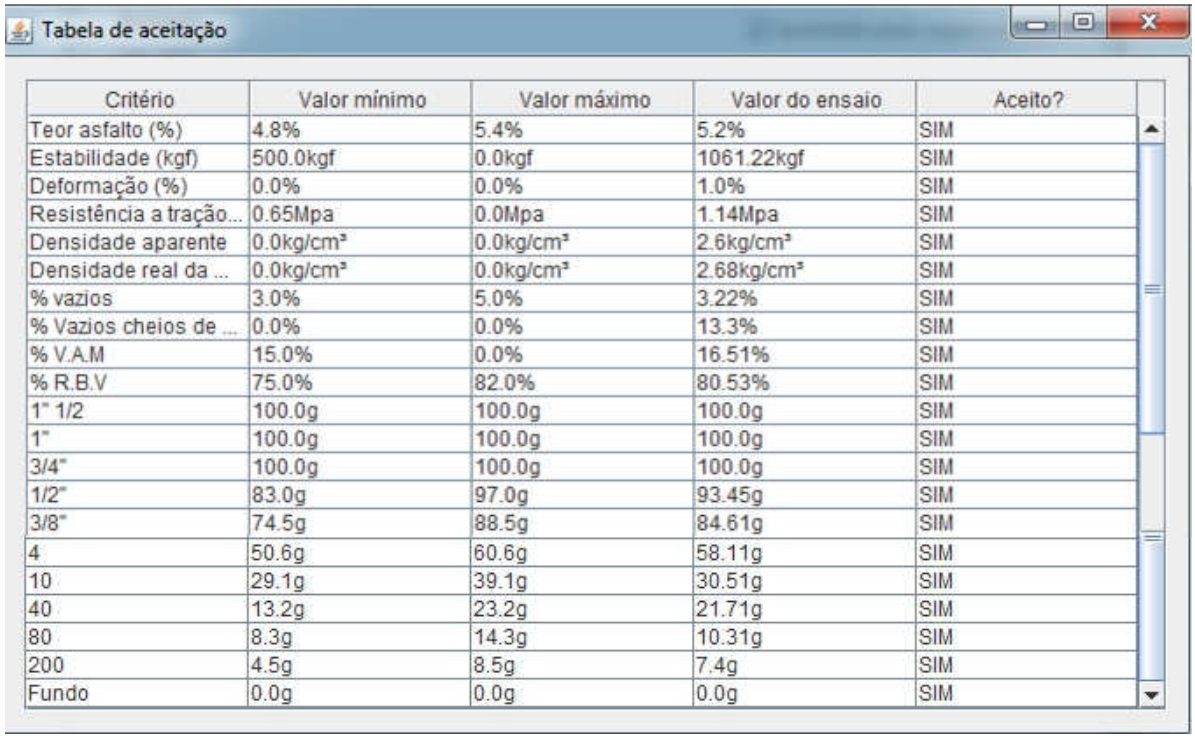


Peneira	Massa retida (g)	Massa acumulada ...	% passante
1" 1/2	0.0g	0.0g	100.0%
1"	0.0g	0.0g	100.0%
3/4"	0.0g	0.0g	100.0%
1/2"	100.9g	14.05g	85.95%
3/8"	175.9g	24.48g	75.52%
4	307.4g	42.79g	57.21%
10	484.8g	67.48g	32.52%
40	581.2g	80.9g	19.1%
80	650.6g	90.56g	9.44%
200	684.2g	95.24g	4.76%
Fundo	730.3g	101.66g	-1.66%

FONTE: Autoria própria (2017).

Na Figura 31 é possível observar a tabela de aceitação para o primeiro ensaio Marshall inicialmente cadastrado, e conforme observado anteriormente, o ensaio cadastrado foi aceito. Esta tabela de aceitação apresenta todos os parâmetros calculados para o ensaio Marshall, e caso o mesmo não tenha sido aceito, esta tela mostra quais parâmetros não estão de acordo com a faixa de trabalho do Projeto da Mistura Asfáltica.

Figura 31 - Tabela de aceitação do primeiro ensaio inserido no "software"

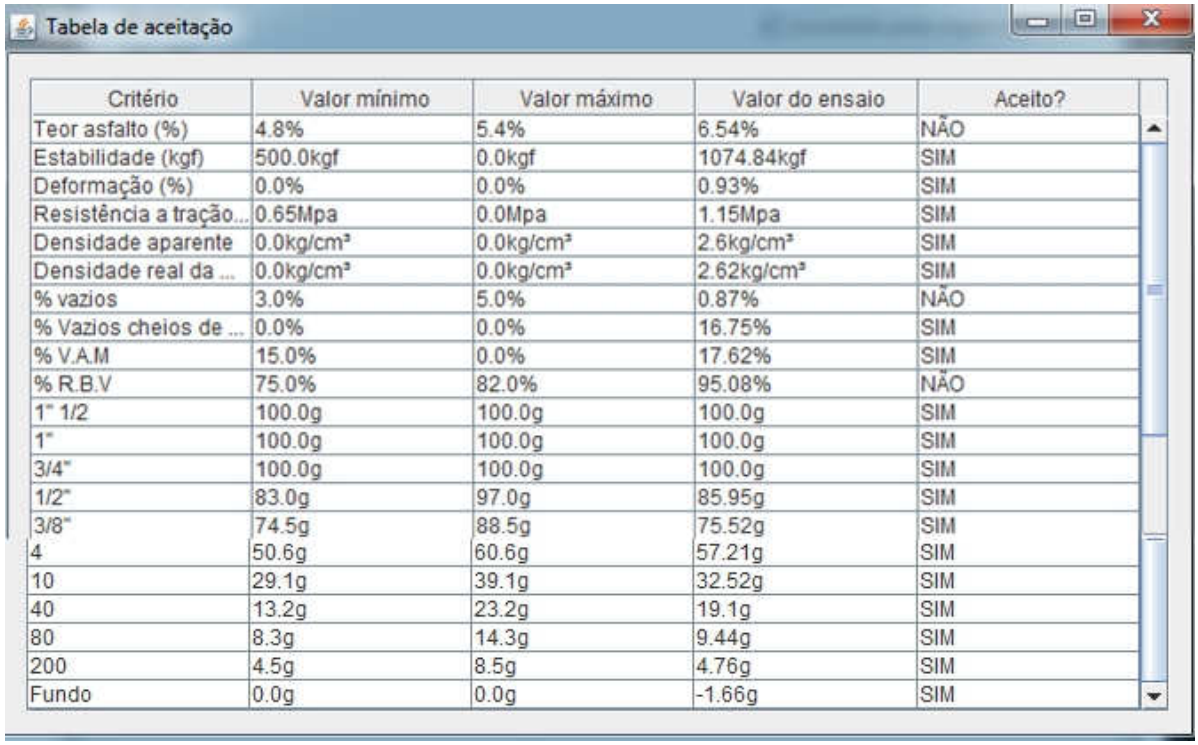


Critério	Valor mínimo	Valor máximo	Valor do ensaio	Aceito?
Teor asfalto (%)	4.8%	5.4%	5.2%	SIM
Estabilidade (kgf)	500.0kgf	0.0kgf	1061.22kgf	SIM
Deformação (%)	0.0%	0.0%	1.0%	SIM
Resistência a tração...	0.65Mpa	0.0Mpa	1.14Mpa	SIM
Densidade aparente	0.0kg/cm³	0.0kg/cm³	2.6kg/cm³	SIM
Densidade real da ...	0.0kg/cm³	0.0kg/cm³	2.68kg/cm³	SIM
% vazios	3.0%	5.0%	3.22%	SIM
% Vazios cheios de ...	0.0%	0.0%	13.3%	SIM
% V.A.M	15.0%	0.0%	16.51%	SIM
% R.B.V	75.0%	82.0%	80.53%	SIM
1" 1/2	100.0g	100.0g	100.0g	SIM
1"	100.0g	100.0g	100.0g	SIM
3/4"	100.0g	100.0g	100.0g	SIM
1/2"	83.0g	97.0g	93.45g	SIM
3/8"	74.5g	88.5g	84.61g	SIM
4	50.6g	60.6g	58.11g	SIM
10	29.1g	39.1g	30.51g	SIM
40	13.2g	23.2g	21.71g	SIM
80	8.3g	14.3g	10.31g	SIM
200	4.5g	8.5g	7.4g	SIM
Fundo	0.0g	0.0g	0.0g	SIM

FONTE: Autoria própria (2017).

Na Figura 32 é possível observar a tabela de aceitação para o segundo ensaio Marshall inicialmente cadastrado, e conforme observado anteriormente, este ensaio não foi aceito para a faixa de trabalho da mistura asfáltica previamente estabelecida.

Figura 32 - Tabela de aceitação do segundo ensaio inserido no "software"



Critério	Valor mínimo	Valor máximo	Valor do ensaio	Aceito?
Teor asfalto (%)	4.8%	5.4%	6.54%	NÃO
Estabilidade (kgf)	500.0kgf	0.0kgf	1074.84kgf	SIM
Deformação (%)	0.0%	0.0%	0.93%	SIM
Resistência a tração...	0.65Mpa	0.0Mpa	1.15Mpa	SIM
Densidade aparente	0.0kg/cm³	0.0kg/cm³	2.6kg/cm³	SIM
Densidade real da ...	0.0kg/cm³	0.0kg/cm³	2.62kg/cm³	SIM
% vazios	3.0%	5.0%	0.87%	NÃO
% Vazios cheios de ...	0.0%	0.0%	16.75%	SIM
% V.A.M	15.0%	0.0%	17.62%	SIM
% R.B.V	75.0%	82.0%	95.08%	NÃO
1" 1/2	100.0g	100.0g	100.0g	SIM
1"	100.0g	100.0g	100.0g	SIM
3/4"	100.0g	100.0g	100.0g	SIM
1/2"	83.0g	97.0g	85.95g	SIM
3/8"	74.5g	88.5g	75.52g	SIM
4	50.6g	60.6g	57.21g	SIM
10	29.1g	39.1g	32.52g	SIM
40	13.2g	23.2g	19.1g	SIM
80	8.3g	14.3g	9.44g	SIM
200	4.5g	8.5g	4.76g	SIM
Fundo	0.0g	0.0g	-1.66g	SIM

FONTE: Autoria própria (2017).

Portanto, os seguintes parâmetros do segundo ensaio Marshall inseridos no "software" estão fora dos limites pré-estabelecidos:

- Teor de Asfalto;
- Porcentagem de Vazios;
- RBV (Relação entre Volume de Vazios preenchidos com Asfalto e o Volume de Vazios Total dos Agregados Minerais).

Recomenda-se, neste caso, que o teor de asfalto (ligante) seja aumentado, devido à mistura asfáltica estar apresentando uma porcentagem de vazios muito alta. Se estes vazios passarem a ser preenchidos com ligante asfáltico, a mistura apresentará menor porosidade e se encaixará nos limites da faixa de trabalho estabelecida no projeto da mistura asfáltica.

7 CONCLUSÕES E SUGESTÕES PARA PESQUISAS FUTURAS

7.1 Conclusões

Neste trabalho, um “software” capaz de analisar e verificar dados oriundos da realização de ensaios Marshall foi desenvolvido. Além disto, o “software” também armazenar os resultados obtidos por meio das análises e verificações supracitadas. A seguir, estão listadas as principais conclusões obtidas neste trabalho.

- O MVC (“Model View Controller”) correspondeu às expectativas durante sua utilização, pois por meio dele foram realizadas as definições de como os resultados seriam apresentados aos usuários;
- O “Framework Hibernate” permitiu a diminuição da complexidade entre os programas Javae o banco de dados do modelo relacional (Sistema de Gerenciamento de Dados MySQL 5.7.4). Além disso, o emprego deste “Framework” possibilitou o desenvolvimento de consultas e atualizações dos dados do “software”, antes que os mesmos sejam salvos. Por meio dele também foi possível transformar as classes Java em tabelas de dados, tornando-as portáveis para qualquer banco de dados desenvolvido por meio da linguagem SQL. A desvantagem relacionada ao uso deste “Framework” foi o aumento do tempo de execução do “software”;
- Por meio de testes realizados com o “software” (inserção da faixa de trabalho do projeto da mistura asfáltica e de ensaios Marshall), observou-se que este foi capaz de analisar e verificar os dados de ensaios Marshall, de acordo com a DNER-ME 043 (1995). Além disso, o armazenamento dos resultados obtidos mediante as análises e verificações feitas pelo “software” foram satisfatórias;
- O “software” desenvolvido é capaz de dificultar a manipuação de dados por parte dos usuários, já que os resultados dos dados avaliados ficam salvos e, uma vez armazenados no sistema, não podem ser editados e nem apagados. Sendo assim, aferramenta implementada permite a fiscalização mais rigorosa por parte dos responsáveis pela garantia da qualidade da mistura asfáltica;
- Recomenda-se que o lançamento dos dados obtidos a partir de ensaios Marshall (o uso do “software”) seja feito pelo engenheiro responsável pelas

análises e verificações dos dados, a fim de garantir a qualidade da mistura asfáltica. Deste modo, sugere-se também que os laboratoristas apenas executem os ensaios e disponibilizem os dados obtidos ao engenheiro. Estas sugestões visam minimizar a manipulação de dados de ensaios, os quais são utilizados para avaliar a qualidade da mistura asfáltica a ser empregada no campo.

7.2 Sugestões para Pesquisas Futuras

A seguir estão listadas algumas sugestões para a melhoria contínua do "software" desenvolvido neste estudo.

- Inserir janelas de aviso aos usuários, com mensagens do tipo: "Tem certeza que deseja salvar os dados? Uma vez salvos, os mesmos não poderão ser editados ou excluídos";
- Inserir indicadores de aceitação para cada análise feita, a fim de que os ensaios aceitos e não aceitos possam ser evidenciados em forma de lista, sem que haja necessidade de acessá-los para saber se os mesmos foram ou não foram aceitos;
- Inserir ao código os algoritmos necessários para a realização de análises e verificações de dados, assim como o armazenamento de resultados, de outros ensaios comumente realizados em laboratórios de pavimentação, tais como os ensaios de extração de betume e granulometria individual, com a finalidade de avaliar a qualidade da mistura asfáltica de forma mais ampla.

REFERÊNCIAS BIBLIOGRÁFICAS

ASPHALT INSTITUTE. **The asphalt handbook. Manual Series, n. 4 (MS-4), College Park, 1956. p. 46-8.** Mix design methods for asphalt concrete and other hot-mix types: Manual series n. 2, 1995.

BERNUCCI, L.B.; MOTTA, L.M.G.; CERATTI, J.A.P.; SOARES, J.B..**Pavimentação Asfáltica: Formação básica para engenheiros.**Rio de Janeiro: PETROBRÁS/ABEDA, 2008.

BRASIL. DEPARTAMENTO NACIONAL DE ESTRADAS DE RODAGEM. **DNER-ME 043**: Misturas betuminosas a quente – Ensaio Marshall**.** Método de Ensaio. Rio de Janeiro: DNER, 1995.

BRASIL. DEPARTAMENTO NACIONAL DE ESTRADAS DE RODAGEM. **DNER-ME 083**: Agregados – análise granulométrica. Norma Rodoviária. Rio de Janeiro: DNER, 1998.

BUFFONI, S.. **Apostila de Algoritmo Estruturado- 4ª Edição.** Rio de Janeiro, 2003. Disponível em: <http://www.dainf.ct.utfpr.edu.br/~pbueno/Arquivos/Algoritmos.pdf>. Acesso em: 18/08/2017.

FIGUEIREDO, E.. **Entendendo o Padrão MVC na Prática.**Brasil, 2015. Disponível em: <https://tableless.com.br/entendendo-o-padrao-mvc-na-pratica/>. Acesso em: 29/08/2017.

FILITTO, D.. **JPA – O Que É? Para Que Serve? Como Implementar Um Sistema Silples?**São Paulo, 2015. Disponível em: <https://www.dfilitto.com.br/java/jpa-o-que-e-para-que-serve-como-implementar-um-sistema/>. Acesso em: 03/09/2017.

MASCIA, N.T.. **Teoria das Deformações.**Campinas, 2006. Disponível em: <http://www.fec.unicamp.br/~nilson/apostilas/Deformacoes.pdf>. Acesso em: 27/09/2017.

MENGUE, F.. **Curso de Java Básico.**Campinas, 2002. Disponível em: < ftp://ftp.unicamp.br/pub/apoio/treinamentos/linguagens/java_basico.pdf>. Acesso em: 15/10/2017.

ORACLE CORPORATION. **MySQL 5.7 Reference Manual.** USA/Canada, 2017. Disponível em: <https://dev.mysql.com/doc/refman/5.7/en/>. Acesso em: 10/08/2017.

PEREIRA, A.P.. **O que é Java?** Curitiba, 2009. Disponível em: <https://www.tecmundo.com.br/programacao/2710-o-que-e-java-.htm>. Acesso em: 24/08/2017.

RESENDE, M..**Ensaio Marshall.** Brasil, 2016. Disponível em: <https://pt.slideshare.net/MarlonRamosResende/ensaio-marshall>. Acesso em: 13/09/2017.

RICARTE, I.L.M.. **O que é uma Classe.**São Paulo, 2000. Disponível em: <http://www.dca.fee.unicamp.br/cursos/PooJava/classes/conceito.html>. Acesso em: 26/08/2017.

ROBERTS, F.L.; KANDHAL, P.S.; BROWN, E.R.; LEE, D-Y.; KENNEDY, T.W. **Hot Mix Asphalt Materials, Mixture, Design, and Construction. 2. Ed.** Lanham, Maryland: NAPA EDUCATIONS FOUNDATION, 1996.

SANCHES, A.R.. **Disciplina: Fundamentos de Armazenamento e Manipulação de Dados.**Brasil, 2005. Disponível em: <https://www.ime.usp.br/~andrers/aulas/bd2005-1/aula7.html>. Acesso em: 10/10/2017.

SANTOS, R.. **Ensaios com Agregados - Massa Unitária, Massa Específica Real e Umidade.** Brasil, 2017. Disponível em: <https://www.revistatransportes.org.br/anpet/article/viewFile/145/127>. Acesso em: 10/09/2017.

VASCONCELOS, K.L; SOARES, J.B; LEITE, L.M.**Efeito da Densidade Máxima Teórica na Dosagem e no Comportamento Mecânico de Mistura Asfáltica tipo CBUQ.** Ceará, 2017. Disponível em: <https://prezi.com/km1jsbvoepcm/ensaios-com-agregados-massa-unitaria-e-massa-especifica-re/>. Acesso em: 15/09/2017.

APÊNDICE A – CÓDIGOS DO “SOFTWARE” DESENVOLVIDO

Basicamente, como foi utilizado o padrão MVC para o desenvolvimento do “software”, os códigos estão divididos em “Model” e “View”.

Primeiramente, neste apêndice estão dispostos os códigos do “Model”, que abrange tudo o que está destinado a lógica da aplicação, ou seja, é onde estão localizadas as classes, as variáveis, as informações e os atributos.

- **Amostra**

```
package model;

import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToOne;
import model.dao.DAO;

@Entity
public class Amostra extends DAO<Ensaio> implements Serializable {
   @Id
   @GeneratedValue(strategy=GenerationType.AUTO)
private Integer id;

   private Double pesoAoAr;
private Double pesoImerso;

   private Double leituraPrensaEstabilidade;
   private Double alturaEstabilidade;
   private Double leituraPrensaResistenciaTracao;
   private Double alturaResistenciaTracao;
   private Double fluencia;
   @ManyToOne
     @JoinTable(name="ensaio_amostra",
   joinColumns={@JoinColumn(name="amostras_id")},
   inverseJoinColumns={@JoinColumn(name="Ensaio_id")})
   private Ensaio ensaio;
```

```
    public Integer getId() {
return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public Double getFluencia() {
        return fluencia;
    }

    public void setFluencia(Double fluencia) {
        this.fluencia = fluencia;
    }
    public Double getPesoAoAr() {
        return pesoAoAr;
    }

    public void setPesoAoAr(Double pesoAoAr) {
        this.pesoAoAr = pesoAoAr;
    }

    public Double getPesoImerso() {
        return pesoImerso;
    }

    public void setPesoImerso(Double pesoImerso) {
this.pesoImerso = pesoImerso;
    }

    public Double getLeituraPrensaEstabilidade() {
        return leituraPrensaEstabilidade;
    }

    public void setLeituraPrensaEstabilidade(Double leituraPrensaEstabilidade) {
        this.leituraPrensaEstabilidade = leituraPrensaEstabilidade;
    }

    public Double getAlturaEstabilidade() {
        return alturaEstabilidade;
    }

    public void setAlturaEstabilidade(Double alturaEstabilidade) {
```

```
        this.alturaEstabilidade = alturaEstabilidade;
    }

    public Double getLeituraPrensaResistenciaTracao() {
        return leituraPrensaResistenciaTracao;
    }

    public void setLeituraPrensaResistenciaTracao(Double
leituraPrensaResistenciaTracao) {
        this.leituraPrensaResistenciaTracao = leituraPrensaResistenciaTracao;
    }

    public Double getAlturaResistenciaTracao() {
        return alturaResistenciaTracao;
    }

    public void setAlturaResistenciaTracao(Double alturaResistenciaTracao) {
        this.alturaResistenciaTracao = alturaResistenciaTracao;
    }

    @Override
    public String toString() {
        return "Peso no ar: " + pesoAoAr + " | Peso imerso: " + pesoImerso;
}

    public Double getVolume(){
      return pesoAoAr - pesoImerso;
}
    public Double getPctPercaVazios(){
        return                                          ((getDdensidadeMaximaTeorica()-
getDdensidadeAparente())/getDdensidadeMaximaTeorica())*100;
    }
    public Double getVaziosCheiosBetume(){
        return
(getDdensidadeAparente()*getCimentoAsfaltico())/Constantes.densidadeCAPN50F70;
}
    public Double getPctVaziosAgregadoMineral(){
        return getPctPercaVazios()+getVaziosCheiosBetume();
    }

    public Double getRelacaoBetumesVazios(){
        return (getVaziosCheiosBetume()/getPctVaziosAgregadoMineral())*100;
    }
    public Double getDensidadeAparente(){
```

```
        return pesoAoAr / getVolume();
}

    public Double getDensidadeMaximaTeorica(){
        System.out.println(getCimentoAsfaltico());
        Double a = (100-getCimentoAsfaltico())/Constantes.densidadeAgregados;
        Double b = (getCimentoAsfaltico()/Constantes.densidadeCAPN50F70);
Double result = 100/(a+b);
        return result;
    }

    public Double getCimentoAsfaltico(){
        return getPctLigante();
    }

    public Double getPctLigante(){
        return (getLigante()/getEnsaio().getPeso())*100;
    }

    public Double getLigante(){

        return getEnsaio().getPeso() - getAgregado();
    }

    public Double getAgregado(){
return getEnsaio().getPesoAgregado();
    }
    public Double getEstabilidadeCalculada(){
      return leituraPrensaEstabilidade*Constantes.anel;
    }
    public Double getEstabilidadeFatorCorrecao(){
      return Constantes.fatorCorrecao(alturaEstabilidade);
    }
    public Double getEstabilidadeCorrigida(){
      return getEstabilidadeCalculada()*getEstabilidadeFatorCorrecao();
    }

    public Double getResistenciaTracaoCalculada(){
System.out.println(Constantes.anel);

return leituraPrensaResistenciaTracao * Constantes.anel;
    }
    public Double getRTDMpa(){
        Double a = 2.0 * getResistenciaTracaoCalculada();
```

```
        Double b = Math.PI*Constantes.diametroCP*alturaResistenciaTracao*11;
        return a/b;
    }
    public Ensaio getEnsaio(){
        return ensaio;
    }
}
```

- **Constantes**

```
package model;

import java.math.RoundingMode;
import java.text.DecimalFormat;

public class Constantes {
    public static Double anel = 1.946;
    public static Double diametroCP = 10.1;
    public static Double temperaturaLIG = 150.0;
    public static Double temperaturaAgreg = 160.0;
    public static Double temperaturaMist = 150.0;
    public static Double densidadeAgregados = 2.95;
    public static Double densidadeCAPN50F70 = 1.016;
    public static Double cap5070 = 5.10;

    public static Double fatorCorrecao(Double altura){
        DecimalFormat df = new DecimalFormat("#.##");
        df.setRoundingMode(RoundingMode.CEILING);
        altura = Double.parseDouble(df.format(altura).replace(",", "."));
        if(altura == 5.74) return 1.18;
        if(altura == 5.77) return 1.17;
        if(altura == 5.81) return 1.16;
        if(altura == 5.84) return 1.15;
        if(altura == 5.87) return 1.14;
        if(altura == 5.90) return 1.13;
        if(altura == 5.93) return 1.12;
        if(altura == 5.97) return 1.11;
        if(altura == 6.00) return 1.10;
        if(altura == 6.03) return 1.09;
        if(altura == 6.06) return 1.08;
        if(altura == 6.09) return 1.07;
        if(altura == 6.11) return 1.06;
        if(altura == 6.14) return 1.05;
        if(altura == 6.19) return 1.04;
```

```
        if(altura == 6.23) return 1.03;
        if(altura == 6.27) return 1.02;
        if(altura == 6.31) return 1.01;
        if(altura == 6.35) return 1.00;
        if(altura == 6.39) return 0.99;
        if(altura == 6.43) return 0.98;
        if(altura == 6.47) return 0.97;
        if(altura == 6.51) return 0.96;
        if(altura == 6.56) return 0.95;
        if(altura == 6.61) return 0.94;
        if(altura == 6.71) return 0.93;
return 0.0;
    }
}
```

- **Critério**

```
package model;

import java.io.Serializable;

public enum Criterio implements CriterioAceitacao, Serializable{
    TEOR_ASFALTO("Teor asfalto(%)", "%"),
    ESTABILIDADE("Estabilidade(kgf)", "kgf"),
    FLUENCIA("DeformaÃ§Ã£o(mm)", "mm"),
    RESISTENCIA_A_TRACAO("ResistÃªncia a traÃ§Ã£o Mpa", "Mpa"),
    DENSIDADE_APARENTE("Densidade aparente", "kg/cmÂ³"),
    DENSIDADE_REAL_MISTURA("Densidade real da mistura", "kg/cmÂ³"),
    PCT_VAZIOS("% vazios", "%"),
    VAZIOS_CHEIOS_BETUME("% Vazios cheios de betume", "%"),
    VAM("% V.A.M", "%"),
    RBV("% R.B.V", "%"),

    N1F1F2("1\" 1/2"),
    N1("1\""),
    N3F4("3/4\""),
    N1F2("1/2\""),
    N3F8("3/8\""),
    N4("4"),
    N10("10"),
    N40("40"),
    N80("80"),
    N200("200"),
    FUNDO("Fundo");
```

```
    private String nome;
    private Peneira equivalente;
    private String unidade;
    Criterio(String nome){
        this.nome = nome;
    }
    Criterio(Peneira equivalente){
        this.equivalente = equivalente;
    }
    Criterio(String nome, String unidade){
        this.nome = nome;
        this.unidade = unidade;
}
    public String toString(){
            return nome;
    }
    public String getUnidade(){
        if(this.isPeneira()){
            return "g";
        }else if(unidade != null){
return unidade;
        }else{
            return "";
        }
    }
    public Boolean isPeneira(){
        try{
            return Peneira.valueOf(this.name()) != null;
        }catch(IllegalArgumentException e){
            return false;
        }
    }
    public Peneira toPeneira(){
        try{
            return Peneira.valueOf(this.name());
        }catch(IllegalArgumentException e){
            return null;
}
    }
}
```

- **Critério de Aceitação**

```
package model;

import java.io.Serializable;
import javax.persistence.MappedSuperclass;

@MappedSuperclass

public interface CriterioAceitacao {

}
```

- **Datas**

```
package model;

import java.text.DateFormatSymbols;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class Datas {
    public static String[] diasDaSemana = new DateFormatSymbols().getWeekdays();

    public static Date HHmmToDate(String data){
        Date date;
        try{
            SimpleDateFormat sdf = new SimpleDateFormat("HH:mm");
            date = sdf.parse(data);

        }catch(Exception ex){
System.out.println("Erro: " + ex);
return null;
        }
        return date;
    }

    public static String DateToHHmm(Date data){
        try{
            SimpleDateFormat sdf = new SimpleDateFormat("HH:mm");
            return sdf.format(data);
```

```
        }catch(Exception ex){
System.out.println("Erro: " + ex);
return null;
        }
    }

    public static Date DateToTime(Date date){
        SimpleDateFormat pessego = new SimpleDateFormat("HH:mm:ss");
Date retorno = null;
        try {
            retorno = pessego.parse(pessego.format(date));
} catch (ParseException e) {
            return null;
        }
        return retorno;
    }
    public static Date Agora(){
        Date date = new Date();
        try{
            SimpleDateFormat sdf = new SimpleDateFormat("HH:mm");
            date = sdf.parse(DateToHHmm(date));

        }catch(Exception ex){
            System.out.println("Erro: " + ex);
            return null;
        }
        return date   ;
    }
    public static Date ddMMyyyyToDate(String data){
        Date date;
        try{
            SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
            date = sdf.parse(data);

        }catch(Exception ex){
System.out.println("Erro: " + ex);
return null;
        }
        return date;
    }

    public static String DateToddMMyyyy(Date data){
        try{
            SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
```

```
            return sdf.format(data);
        }catch(Exception ex){
System.out.println("Erro: " + ex);
return null;
        }
    }

    public static String DateToyyyyMMdd(Date data){
        try{
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
            return sdf.format(data);
        }catch(Exception ex){
System.out.println("Erro: " + ex);
return null;
        }
    }

    public static Date yyyyMMddToDate(String data){
        Date date;
        try{
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
            date = sdf.parse(data);

        }catch(Exception ex){
System.out.println("Erro: " + ex);
return null;
        }
        return date;
    }
    public static String diaSemana(Date data){
        Calendar cal = Calendar.getInstance();
        cal.setTime(data);
        int day = cal.get(Calendar.DAY_OF_WEEK);
        return diasDaSemana[day-1];
    }

    public static Boolean between(Date date, Date dateStart, Date dateEnd) {
        return date.compareTo(dateStart)>= 0 && date.compareTo(dateEnd) <=0;
    }

    public static Integer diferencaEmMinutos(Date date1, Date date2){
        long diferenca = Datas.DateToTime(date1).getTime() -
Datas.DateToTime(date2).getTime();
return (int) (diferenca / 1000L)/60;
```

```
    }

}
```

- **Double Statics**

```
package model;

import java.util.DoubleSummaryStatistics;
import java.util.stream.Collector;

public class DoubleStatistics extends DoubleSummaryStatistics {

    private double sumOfSquare = 0.0d;
    private double sumOfSquareCompensation; // Low order bits of sum
    private double simpleSumOfSquare; // Used to compute right sum for
                                      // non-finite inputs

    @Override
    public void accept(double value) {
        super.accept(value);
        double squareValue = value * value;
        simpleSumOfSquare += squareValue;
        sumOfSquareWithCompensation(squareValue);
    }

    public DoubleStatistics combine(DoubleStatistics other) {
        super.combine(other);
        simpleSumOfSquare += other.simpleSumOfSquare;
        sumOfSquareWithCompensation(other.sumOfSquare);
        sumOfSquareWithCompensation(other.sumOfSquareCompensation);
        return this;
    }

    private void sumOfSquareWithCompensation(double value) {
        double tmp = value - sumOfSquareCompensation;
        double velvel = sumOfSquare + tmp; // Little wolf of rounding error
        sumOfSquareCompensation = (velvel - sumOfSquare) - tmp;
        sumOfSquare = velvel;
}

    public double getSumOfSquare() {
        double tmp = sumOfSquare + sumOfSquareCompensation;
        if (Double.isNaN(tmp) && Double.isInfinite(simpleSumOfSquare)) {
```

```
            return simpleSumOfSquare;
        }
        return tmp;
    }

    public final double getStandardDeviation() {
        long count = getCount();
        double sumOfSquare = getSumOfSquare();
        double average = getAverage();
        return count > 0 ? Math.sqrt((sumOfSquare - count * Math.pow(average, 2)) /
(count - 1)) : 0.0d;
    }

    public static Collector<Double, ?, DoubleStatistics> collector() {
        return Collector.of(DoubleStatistics::new, DoubleStatistics::accept,
DoubleStatistics::combine);
}

}
```

- **Ensaio**

```
package model;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.List;
import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import javax.persistence.Temporal;
import model.dao.DAO;

@Entity
public class Ensaio extends DAO<Ensaio>{
    @Id
```

```
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Integer id;
    private String rodovia;
    private String lugar;

    @Temporal(javax.persistence.TemporalType.DATE)
    private Date dataEnsaio;

    private Double peso;

    private Boolean assistidoPelaSupervisora;

    private Double pesoAgregado;

    @OneToOne(cascade = CascadeType.PERSIST)
    private Granulometria granulometria;
    @OneToMany(cascade = CascadeType.PERSIST)
    private List<Amostra> amostras = new ArrayList<>();
    @ManyToOne
    private Projeto projeto;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }
    public String getLugar() {
        return lugar;
    }

    public void setLugar(String lugar) {
this.lugar = lugar;
    }
    public String getRodovia() {
        return rodovia;
    }

    public void setRodovia(String rodovia) {
        this.rodovia = rodovia;
    }

    public boolean addAmostra(Amostra amostra){
```

```
return this.amostras.add(amostra);
    }

    public Date getDataEnsaio() {
        return dataEnsaio;
}

    public void setDataEnsaio(Date dataEnsaio) {
        this.dataEnsaio = dataEnsaio;
}

    public Double getPeso() {
        return peso;
    }

    public void setPeso(Double peso) {
this.peso = peso;
    }

    public Boolean getAssistidoPelaSupervisora() {
        return assistidoPelaSupervisora;
    }


    public Granulometria granulometriaPorcentagemPassante(){
        if(granulometria != null){
            Granulometria retorno = new Granulometria();
            retorno.setPeneiraN1F1F2(100-
((granulometria.getPeneiraN1F1F2()/pesoAgregado)*100));
            retorno.setPeneiraN1(100-
((granulometria.getPeneiraN1()/pesoAgregado)*100));

            retorno.setPeneiraN10(100-
((granulometria.getPeneiraN10()/pesoAgregado)*100));
            retorno.setPeneiraN1F2(100-
((granulometria.getPeneiraN1F2()/pesoAgregado)*100));
            retorno.setPeneiraN200(100-
((granulometria.getPeneiraN200()/pesoAgregado)*100));
            retorno.setPeneiraN3F4(100-
((granulometria.getPeneiraN3F4()/pesoAgregado)*100));
            retorno.setPeneiraN3F8(100-
((granulometria.getPeneiraN3F8()/pesoAgregado)*100));
            retorno.setPeneiraN4(100-
((granulometria.getPeneiraN4()/pesoAgregado)*100));
```

```
            retorno.setPeneiraN40(100-
((granulometria.getPeneiraN40()/pesoAgregado)*100));
            retorno.setPeneiraN80(100-
((granulometria.getPeneiraN80()/pesoAgregado)*100));

            retorno.setPeneiraFundo(100-
((granulometria.getPeneiraFundo()/pesoAgregado)*100));
return retorno;
        }else{
            return null;
}
    }
    public void setAssistidoPelaSupervisora(Boolean assistidoPelaSupervisora) {
        this.assistidoPelaSupervisora = assistidoPelaSupervisora;
}

    public Double getPesoAgregado() {
        return pesoAgregado;
}

    public void setPesoAgregado(Double pesoAgregado) {
        this.pesoAgregado = pesoAgregado;
    }

    public Granulometria getGranulometria() {
        return granulometria;
    }

    public void setGranulometria(Granulometria granulometria) {
        this.granulometria = granulometria;
    }

    public List<Amostra> getAmostras() {
        return amostras;
    }

    public void setAmostras(List<Amostra> amostras) {
        this.amostras = amostras;
    }

    public Projeto getProjeto() {
        return Projeto.getInstance();
    }
```

```
    public void setProjeto(Projeto projeto) {
this.projeto = projeto;
    }

    public Double getRTMpa(){
Double rtmpa;
        Double                    mediaLeituraPrensaResistenciaTracao                    =
getMediaLeituraPrensaResistenciaTracao();
        Double mediaAlturaResistenciaTracao = getMediaAlturaResistenciaTracao();

        rtmpa = (2 * mediaLeituraPrensaResistenciaTracao * 9.81 * Constantes.anel);
        rtmpa  /=  (mediaAlturaResistenciaTracao  *  Constantes.diametroCP  *  Math.PI
*100);
return rtmpa;
    }


    private Double getMediaAlturaResistenciaTracao(){
        return amostras.stream()
.mapToDouble(o -> o.getAlturaResistenciaTracao())
                .average().getAsDouble();
    }

    private Double getMediaLeituraPrensaResistenciaTracao(){
        return amostras.stream()
                .mapToDouble(o -> o.getLeituraPrensaResistenciaTracao())
.average().getAsDouble();

    }
    public String getDataEnsaioString(){
        return Datas.DateToddMMyyyy(dataEnsaio);
    }

    public Boolean isAceito(){
        return getCriteriosReprovados().isEmpty();
    }
    public Boolean isAceito(Criterio criterio){
          Double maximo =  getProjeto().getToleranciaProjetoMax().get(criterio);
          Double minimo =  getProjeto().getToleranciaProjetoMin().get(criterio);
          if(maximo == 0.0){ maximo = Double.POSITIVE_INFINITY; }
if(minimo == 0.0){ minimo = Double.NEGATIVE_INFINITY; }

          if(getValorCriterio(criterio) == 0 ||
```

```
                getValorCriterio(criterio) <= maximo && getValorCriterio(criterio) >=
minimo){
return true;
          }
          return false;
    }
    public List<Criterio> getCriteriosReprovados(){
        List<Criterio> retorno= new ArrayList<>();
        for(Criterio criterio: Criterio.values()){
          if(!isAceito(criterio)){
              retorno.add(criterio);
          }
        }

        return retorno;
    }
    public Double getValorCriterio(Criterio criterio){
if(criterio.toPeneira() != null){
          return granulometriaPorcentagemPassante().getPeneira(criterio.toPeneira());
        }else{
           switch(criterio){
              case DENSIDADE_APARENTE:
                 return getDensidadeAparente();
              case ESTABILIDADE:
                 return getEstabilidade();
              case DENSIDADE_REAL_MISTURA:
return getDdensidadeRealMistura();
              case FLUENCIA:
                 return getFluencia();
              case PCT_VAZIOS:
                 return getPctVazios();
              case RBV:
                 return getRBV();
              case RESISTENCIA_A_TRACAO:
                 return getRTMpa();
              case TEOR_ASFALTO:
                 return getTeorAsfalto();
              case VAM:
                 return getVAM();
              case VAZIOS_CHEIOS_BETUME:
                 return getVaziosCheiosBetume();
           }
        }
        return 0.0;
```

```
    }
    public Double getVAM(){
        return getPctVazios() + getVaziosCheiosBetume();
    }
    public Double getRBV(){
        return getVaziosCheiosBetume()/getVAM()*100;
    }
    public Double getVaziosCheiosBetume(){
        return
getTeorAsfalto()*getDensidadeAparente()/Constantes.densidadeCAPN50F70;
    }
    public Double getPctVazios(){
return                                                    (getDensidadeRealMistura()-
getDensidadeAparente())/getDensidadeRealMistura()*100;
    }
    public Double getDensidadeRealMistura(){
        return                                                                100/((100-
getTeorAsfalto())/Constantes.densidadeAgregados+(getTeorAsfalto()/Constantes.de
nsidadeCAPN50F70));
    }
    public Double getTeorAsfalto(){
        return ((peso-pesoAgregado)/getPeso())*100;
    }
    public Double getEstabilidade(){
        return this.amostras.stream().mapToDouble(o -> o.getEstabilidadeCalculada())
.average().getAsDouble();
    }
    public Double getDensidadeAparente(){
        return this.amostras.stream().mapToDouble(o -> o.getDensidadeAparente())
                .average().getAsDouble();
    }
    public Double getFluencia(){
        return                    getAmostras().stream().mapToDouble(o                    ->
o.getFluencia()).average().getAsDouble();
    }
    public Double getResistenciaTracaoMPA(){
        return getRTMpa();
    }


    public Double getLigante(){
        return peso - pesoAgregado;
    }
    public Double getPctLigante(){
```

```
return (getLigante()/getPesoAgregado())*100;
    }

    public Double getPesoRetido(Peneira peneira){
        return this.granulometria.getPeneira(peneira);
    }

public Double getPesoAcumulado(Peneira peneira){
        return (getPesoRetido(peneira)/pesoAgregado)*100;
    }

    public Double getPctPassante(Peneira peneira){
        return 100-getPesoAcumulado(peneira);
    }

}
```

- **Entidade**

```
package model;

public interface Entidade {
    public void save();
    public void update();
    public void delete();
}
```

- **Granulometria**

```
package model;

import java.io.Serializable;
import java.util.HashMap;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import model.dao.DAO;

@Entity
public class Granulometria extends DAO<Granulometria> implements Serializable {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Integer id;
```

```
private Double peneiraN1F1F2;
private Double peneiraN3F4;
private Double peneiraN1F2;
private Double peneiraN3F8;
private Double peneiraN4;
private Double peneiraN10;
private Double peneiraN1;

public Double getPeneiraN1() {
    return peneiraN1;
}

public void setPeneiraN1(Double peneiraN1) {
    this.peneiraN1 = peneiraN1;
}

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

public Double getPeneiraN1F1F2() {
    return peneiraN1F1F2;
}

public void setPeneiraN1F1F2(Double peneiraN1F1F2) {
    this.peneiraN1F1F2 = peneiraN1F1F2;
}

public Double getPeneiraN3F4() {
    return peneiraN3F4;
}

public void setPeneiraN3F4(Double peneiraN3F4) {
    this.peneiraN3F4 = peneiraN3F4;
}

public Double getPeneiraN1F2() {
    return peneiraN1F2;
}
```

```
public void setPeneiraN1F2(Double peneiraN1F2) {
    this.peneiraN1F2 = peneiraN1F2;
}

public Double getPeneiraN3F8() {
    return peneiraN3F8;
}

public void setPeneiraN3F8(Double peneiraN3F8) {
    this.peneiraN3F8 = peneiraN3F8;
}

public Double getPeneiraN4() {
    return peneiraN4;
}

public void setPeneiraN4(Double peneiraN4) {
    this.peneiraN4 = peneiraN4;
}

public Double getPeneiraN10() {
    return peneiraN10;
}

public void setPeneiraN10(Double peneiraN10) {
    this.peneiraN10 = peneiraN10;
}

public Double getPeneiraN40() {
    return peneiraN40;
}

public void setPeneiraN40(Double peneiraN40) {
    this.peneiraN40 = peneiraN40;
}

public Double getPeneiraN80() {
    return peneiraN80;
}

public void setPeneiraN80(Double peneiraN80) {
    this.peneiraN80 = peneiraN80;
}
```

```
    public Double getPeneiraN200() {
        return peneiraN200;
    }

    public void setPeneiraN200(Double peneiraN200) {
        this.peneiraN200 = peneiraN200;
    }

    public Double getPeneiraFundo() {
        return peneiraFundo;
    }

    public void setPeneiraFundo(Double peneiraFundo) {
        this.peneiraFundo = peneiraFundo;
}
    private Double peneiraN40;
    private Double peneiraN80;
    private Double peneiraN200;
    private Double peneiraFundo;

    public Double getPorcentagemPassante(String rotulo){
        return 0.0;
    }

    public Double getPeneira(Peneira peneira){
        if(peneira == Peneira.N1) return peneiraN1;
        if(peneira == Peneira.N10) return peneiraN10;
        if(peneira == Peneira.N1F1F2) return peneiraN1F1F2;
        if(peneira == Peneira.N1F2) return peneiraN1F2;
        if(peneira == Peneira.N200) return peneiraN200;
        if(peneira == Peneira.N3F4) return peneiraN3F4;
        if(peneira == Peneira.N3F8) return peneiraN3F8;
        if(peneira == Peneira.N4) return peneiraN4;
        if(peneira == Peneira.N40) return peneiraN40;
        if(peneira == Peneira.N80) return peneiraN80;
if(peneira == Peneira.FUNDO) return peneiraFundo;

        return 0.0;
    }
}
```

- **Objectable**

```
package model;

public interface Objectable {


public Object[] toObjectArray();
}
```

- **Peneira**

```
package model;

import java.io.Serializable;

public enum Peneira implements Serializable {
    N1F1F2("1\" 1/2", Criterio.N1F1F2),
    N1("1\"", Criterio.N1),
    N3F4("3/4\"", Criterio.N3F4),
    N1F2("1/2\"", Criterio.N1F2),
    N3F8("3/8\"", Criterio.N3F8),
N4("4", Criterio.N4),
    N10("10", Criterio.N10),
    N40("40", Criterio.N40),
    N80("80", Criterio.N80),
    N200("200", Criterio.N200),
    FUNDO("Fundo", Criterio.FUNDO);

    private final String nome;
    private final Criterio equivalente;

    Peneira(String nome, Criterio criterio){
this.nome = nome;
this.equivalente = criterio;
    }

    public String toString(){
        return nome;
    }
    public Criterio toCriterio(){
                try{
        return Criterio.valueOf(this.name());
        }catch(IllegalArgumentException e){
```

```
            return null;
        }
    }
    public String getUnidade(){
        return "g";
}

}
```

- **Projeto**

```
package model;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.persistence.CascadeType;
import javax.persistence.CollectionTable;
import javax.persistence.Column;
import javax.persistence.ElementCollection;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.MapKey;
import javax.persistence.MapKeyEnumerated;
import javax.persistence.MapKeyJoinColumn;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.persistence.Transient;
import model.dao.DAO;

@Entity
public class Projeto extends DAO<Projeto> {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Integer id;
```

```
    @OneToMany(cascade = CascadeType.PERSIST)
private List<Ensaio> ensaios;

   // @ElementCollection
    @ElementCollection
private Map<Criterio, Double> toleranciaProjetoMax = new HashMap<>();
   // @ElementCollection
    @ElementCollection
    private Map<Criterio, Double> toleranciaProjetoMin = new HashMap<>();


    private static Projeto projeto;

    public Map<Criterio, Double> getToleranciaProjetoMax() {
        return toleranciaProjetoMax;
    }

    public void setToleranciaProjetoMax(Map<Criterio, Double> toleranciaProjetoMax)
{
        this.toleranciaProjetoMax = toleranciaProjetoMax;
    }

    public Map<Criterio, Double> getToleranciaProjetoMin() {
        return toleranciaProjetoMin;
    }

    public void setToleranciaProjetoMin(Map<Criterio, Double> toleranciaProjetoMin) {
        this.toleranciaProjetoMin = toleranciaProjetoMin;
    }

    public Boolean hasValoresPadroes(){
        return                    !getToleranciaProjetoMax().isEmpty()                    &&
!getToleranciaProjetoMin().isEmpty();
    }
    public Double getMaxPeneira(Peneira peneira){
        return ensaios.stream()
                .mapToDouble(ensaio                                                    ->
ensaio.getGranulometria().getPeneira(peneira)).max()
                .getAsDouble();
    }
    public Double getMinPeneira(Peneira peneira){
        return ensaios.stream()
.mapToDouble(ensaio -> ensaio.getGranulometria().getPeneira(peneira)).min()
.getAsDouble();
```

```
    }

    public Double getMediaPeneira(Peneira peneira){
        return ensaios.stream()
.mapToDouble(ensaio -> ensaio.getGranulometria().getPeneira(peneira)).average()
.getAsDouble();
    }
    public Double getDesvioPadraoPeneira(Peneira peneira){
          double collection[] = ensaios.stream()
.mapToDouble(ensaio -> ensaio.getGranulometria().getPeneira(peneira)).toArray();
return new Statistics(collection).getStdDev();
    }

    public Projeto() {
        this.toleranciaProjetoMin = new HashMap<>();
        this.toleranciaProjetoMin = new HashMap<>();
        ensaios = new ArrayList<>();
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }
    public List<Ensaio> getEnsaios() {
return ensaios;
    }

    public void setEnsaios(List<Ensaio> ensaios) {
        this.ensaios = ensaios;
    }
    public static Projeto getProjeto() {
        return projeto;
    }

    public static void setProjeto(Projeto projeto) {
        Projeto.projeto = projeto;
}

    public static Projeto getInstance(){
        if(projeto == null){
List<Projeto> projetos = new Projeto().all();
```

```
            if(projetos.isEmpty()){
                projeto = new Projeto();
                projeto.save();
projeto = new Projeto().all().get(0);
}else{
                projeto = projetos.get(0);
            }
        }
        return projeto;
    }

    public void addEnsaio(Ensaio ensaio){
        ensaios.add(ensaio);
    }

}
```

- **Statistics**

```
package model;

import java.util.Arrays;

public class Statistics
{
    double[] data;
    int size;

    public Statistics(double[] data)
    {
        this.data = data;
        size = data.length;
    }

    double getMean()
    {
        double sum = 0.0;
        for(double a : data)
            sum += a;
        return sum/size;
}

    double getVariance()
    {
```

```
        double mean = getMean();
        double temp = 0;
        for(double a :data)
            temp += (a-mean)*(a-mean);
        return temp/(size-1);
    }

    double getStdDev()
    {
        return Math.sqrt(getVariance());
    }

    public double median()
    {
       Arrays.sort(data);

       if (data.length % 2 == 0)
       {
          return (data[(data.length / 2) - 1] + data[data.length / 2]) / 2.0;
}
       return data[data.length / 2];
    }
}
```

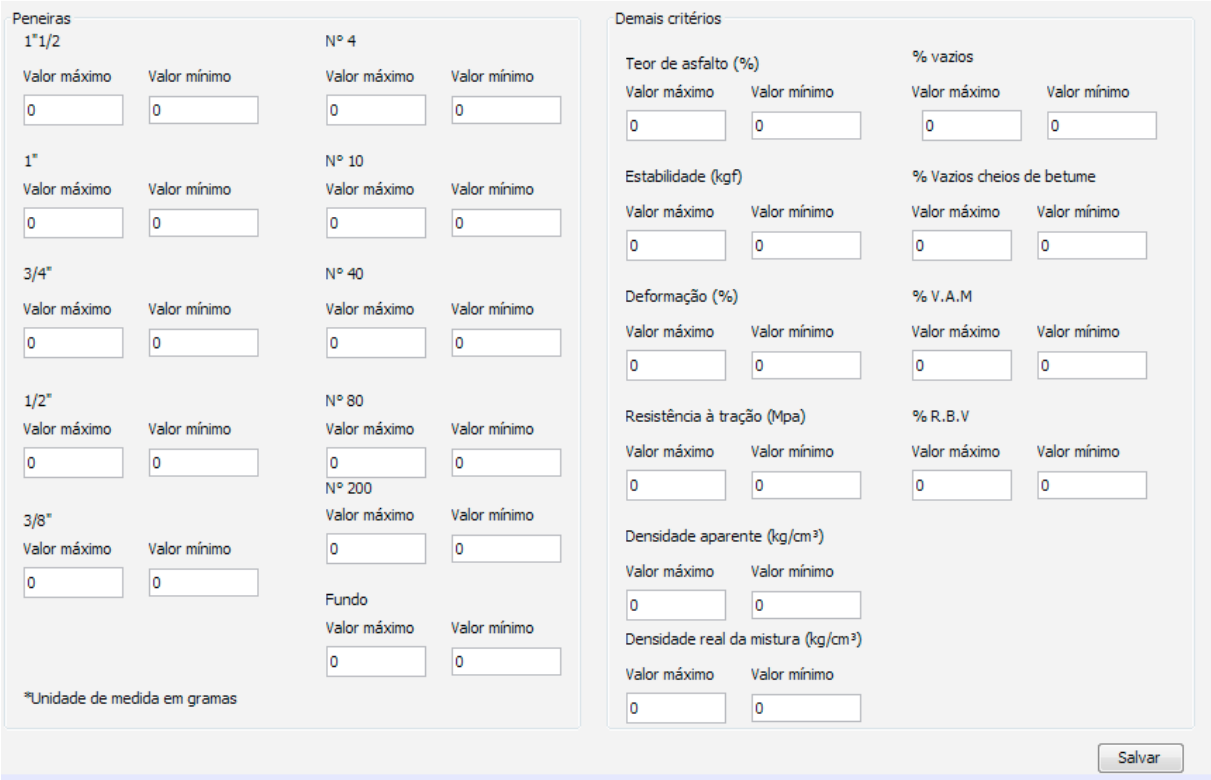
- **Utils**

```
package model;

public class Utils {
    public static Double formatDouble(Double n){
        double number = n;

        number = Math.round(number * 100);
number = number/100;
        return number;
    }
}
```

Aqui estarão dispostas as configurações do “View”, que abrange todo o tipo de retorno de dado para a interface (é tudo aqui que o “usuário vê”, ou seja, as telas). Neste caso, foi utilizada a ferramenta Java para determinar a interface das telas disponíveis ao usuário.

- **Configuração Inicial**

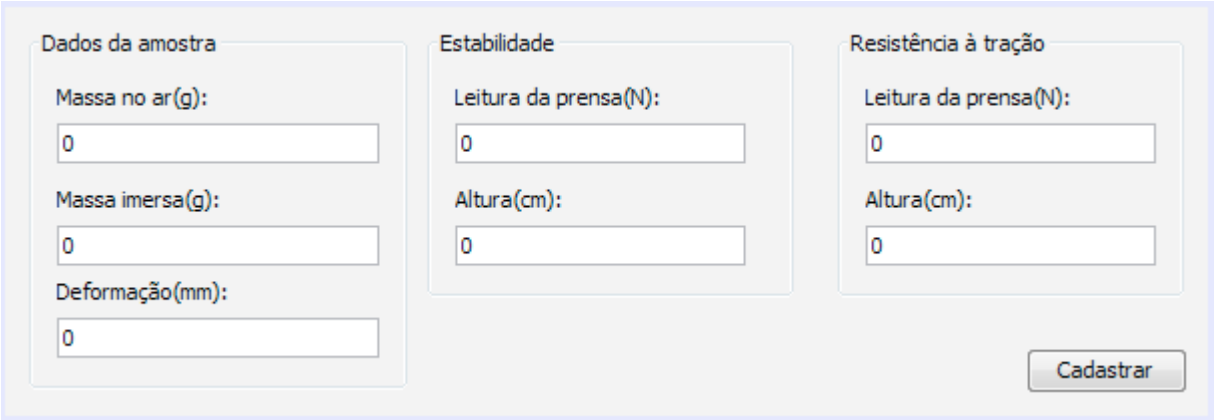
Figura 33 – Tela criada para a configuração inicial



FONTE: Autoria própria (2017).

- **Nova Amostra**

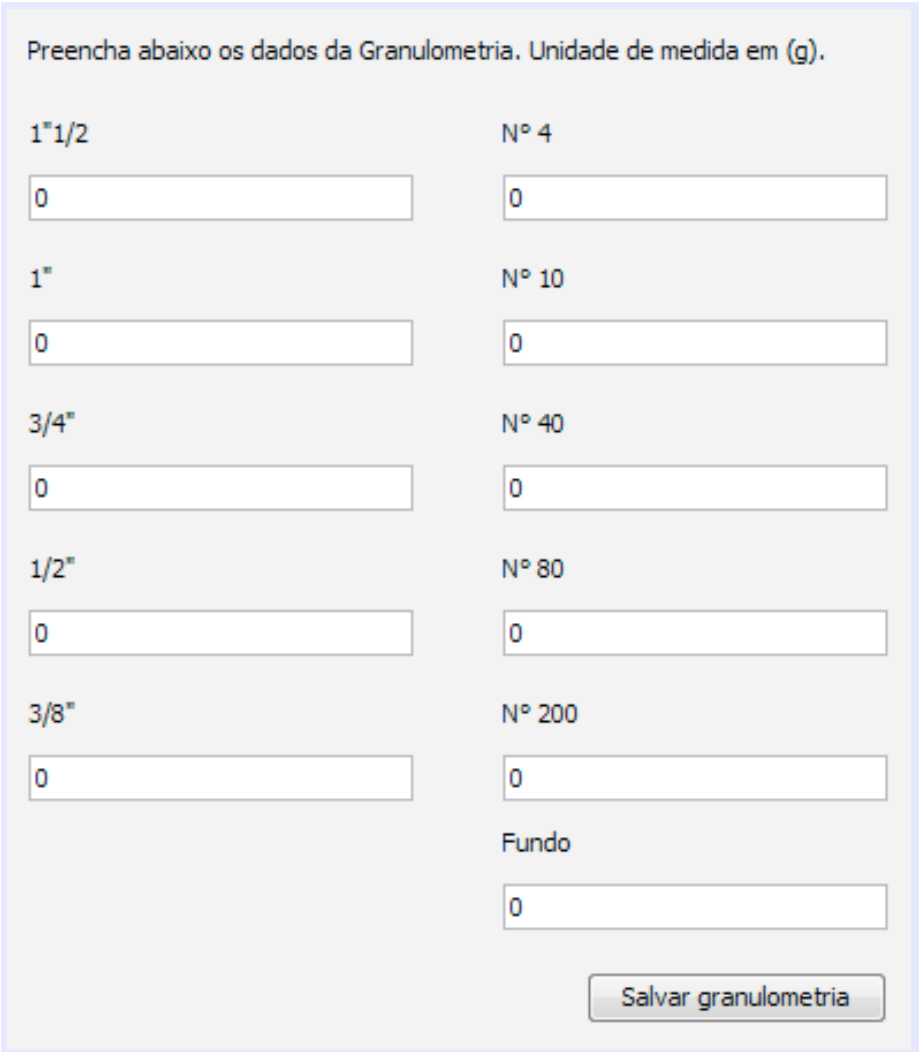
Figura 34 – Tela criada para adicionar uma nova amostra



FONTE: Autoria própria (2017).

- **Nova Granulometria**

Figura 35 - Tela criada para adicionar uma nova granulometria



FONTE: Autoria própria (2017).

- **Novo ensaio**

```
package view;

import controller.NovoEnsaioController;
import java.util.Arrays;
import java.util.List;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import model.Amostra;
import model.Ensaio;
import model.Granulometria;
import model.Peneira;
```

```
public class frmNovoEnsaio extends View<NovoEnsaioController> {
private Ensaio ensaio = new Ensaio();
private JTableWrapper<Amostra> wrapperAmostras;
    private JTableWrapper<Peneira> wrapperGranulometria;

    public frmNovoEnsaio() {
        initComponents();
        setController(new NovoEnsaioController(this));
wrapperAmostras = new JTableWrapper(tblAmostras);
        wrapperAmostras.setMapFuncion(o ->{
            Amostra amostra = (Amostra) o;
            Object[] linha = {amostra.getPesoAoAr().toString() + "g",
                amostra.getPesoImerso().toString() + "g"};
            return linha;
});
        wrapperAmostras.setOnDoubleClickListener((item)->                    new
frmNovaAmostra(item, this.ensaio).setVisible(true));

        wrapperGranulometria = new JTableWrapper(tblGranulometria);

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        rectanglePainter1 = new org.jdesktop.swingx.painter.RectanglePainter();
        jScrollPane3 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();
        jProgressBar1 = new javax.swing.JProgressBar();
        txtCadastrar = new javax.swing.JButton();
        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        txtDataEnsaio = new org.jdesktop.swingx.JXDatePicker();
        txtPeso = new javax.swing.JTextField();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        txtPesoAgregado = new javax.swing.JTextField();
        txtRodovia = new javax.swing.JTextField();
        jLabel4 = new javax.swing.JLabel();
        txtLocal = new javax.swing.JTextField();
        jLabel5 = new javax.swing.JLabel();
        checkAssistido = new javax.swing.JCheckBox();
        jPanel2 = new javax.swing.JPanel();
```

```
        btnAddAmostra = new javax.swing.JButton();
        btnVerAmostra = new javax.swing.JButton();
        btnExcluirAmostra = new javax.swing.JButton();
        jScrollPane4 = new javax.swing.JScrollPane();
        tblAmostras = new javax.swing.JTable();
        jPanel3 = new javax.swing.JPanel();
        btnAddGranulometria = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        tblGranulometria = new javax.swing.JTable();

        jTable1.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null}
            },
            new String [] {
                "Title 1", "Title 2", "Title 3", "Title 4"
            }
        ));
        jScrollPane3.setViewportView(jTable1);


setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
setTitle("Cadastro de ensaio");
        setResizable(false);
addWindowFocusListener(new java.awt.event.WindowFocusListener() {
            public void windowGainedFocus(java.awt.event.WindowEvent evt) {
                formWindowGainedFocus(evt);
            }
            public void windowLostFocus(java.awt.event.WindowEvent evt) {
            }
        });

        txtCadastrar.setText("Cadastrar ensaio");
        txtCadastrar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                txtCadastrarActionPerformed(evt);
            }
        });
```

```
jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("InformaÃ§Ãµes
bÃ¡sicas"));

jLabel1.setText("Data do ensaio:");

txtPeso.addActionListener(new java.awt.event.ActionListener() {
          public void actionPerformed(java.awt.event.ActionEvent evt) {
txtPesoActionPerformed(evt);
          }
       });

       jLabel2.setText("Massa do agregado + ligante(g):");

       jLabel3.setText("Massa do agregado(g):");

       jLabel4.setText("Rodovia:");

       jLabel5.setText("Local:");

       checkAssistido.setText("Foi assistido pela supervisora?");

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
       jPanel1.setLayout(jPanel1Layout);
       jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
          .addGroup(jPanel1Layout.createSequentialGroup()
             .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING, false)
                    .addComponent(jLabel5)
                    .addComponent(jLabel2)
                    .addComponent(txtPeso)
                    .addComponent(jLabel3)
                    .addComponent(txtPesoAgregado,
javax.swing.GroupLayout.DEFAULT_SIZE, 199, Short.MAX_VALUE)
                    .addComponent(jLabel4)
                    .addComponent(txtRodovia)
                    .addComponent(txtLocal)
```

```
                    .addComponent(jLabel1)
                    .addComponent(txtDataEnsaio,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addComponent(checkAssistido))
            .addContainerGap(18, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel4)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(txtRodovia,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jLabel5)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(txtLocal,     javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(txtDataEnsaio,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(txtPeso,     javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(txtPesoAgregado,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,       47,
Short.MAX_VALUE)
                .addComponent(checkAssistido)
                .addGap(20, 20, 20))
        );

        jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder("Amostras"));

        btnAddAmostra.setText("Adicionar amostra");
        btnAddAmostra.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnAddAmostraActionPerformed(evt);
            }
        });

        btnVerAmostra.setText("Ver amostra");
        btnVerAmostra.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
btnVerAmostraActionPerformed(evt);
            }
        });

        btnExcluirAmostra.setText("Excluir amostra");
btnExcluirAmostra.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnExcluirAmostraActionPerformed(evt);
            }
        });

        tblAmostras.setModel(new javax.swing.table.DefaultTableModel(
new Object [][] {

            },
            new String [] {
```

```
            "Massa ao ar(g)", "Massa imersa(g)"
}
        ) {
            boolean[] canEdit = new boolean [] {
                false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        jScrollPane4.setViewportView(tblAmostras);

        javax.swing.GroupLayout jPanel2Layout = new
javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
                    .addComponent(jScrollPane4,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addComponent(btnAddAmostra,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(btnVerAmostra,
javax.swing.GroupLayout.PREFERRED_SIZE, 127,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(btnExcluirAmostra,
javax.swing.GroupLayout.PREFERRED_SIZE, 129,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap())
        );
        jPanel2Layout.setVerticalGroup(
```

```
jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jScrollPane4,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    82,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
.addComponent(btnAddAmostra)
                    .addComponent(btnVerAmostra)
                    .addComponent(btnExcluirAmostra))
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );


jPanel3.setBorder(javax.swing.BorderFactory.createTitledBorder("Granulometria"));

        btnAddGranulometria.setText("Adicionar granulometria");
        btnAddGranulometria.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnAddGranulometriaActionPerformed(evt);
            }
        });

        tblGranulometria.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "Peneira", "Valor(g)"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
```

```
        jScrollPane1.setViewportView(tblGranulometria);

        javax.swing.GroupLayout jPanel3Layout = new
javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addGap(0, 0, Short.MAX_VALUE)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
                    .addComponent(jScrollPane1,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 452,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btnAddGranulometria,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 239,
javax.swing.GroupLayout.PREFERRED_SIZE)))
        );
        jPanel3Layout.setVerticalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 175,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(btnAddGranulometria)
                .addGap(171, 171, 171))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
```

```
            .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G)
                .addGroup(layout.createSequentialGroup()
                    .addGap(0, 0, Short.MAX_VALUE)
                    .addComponent(txtCadastrar,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    212,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
                        .addComponent(jPanel3,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jPanel2,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))
            .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
                .addComponent(jPanel1,  javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jPanel2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

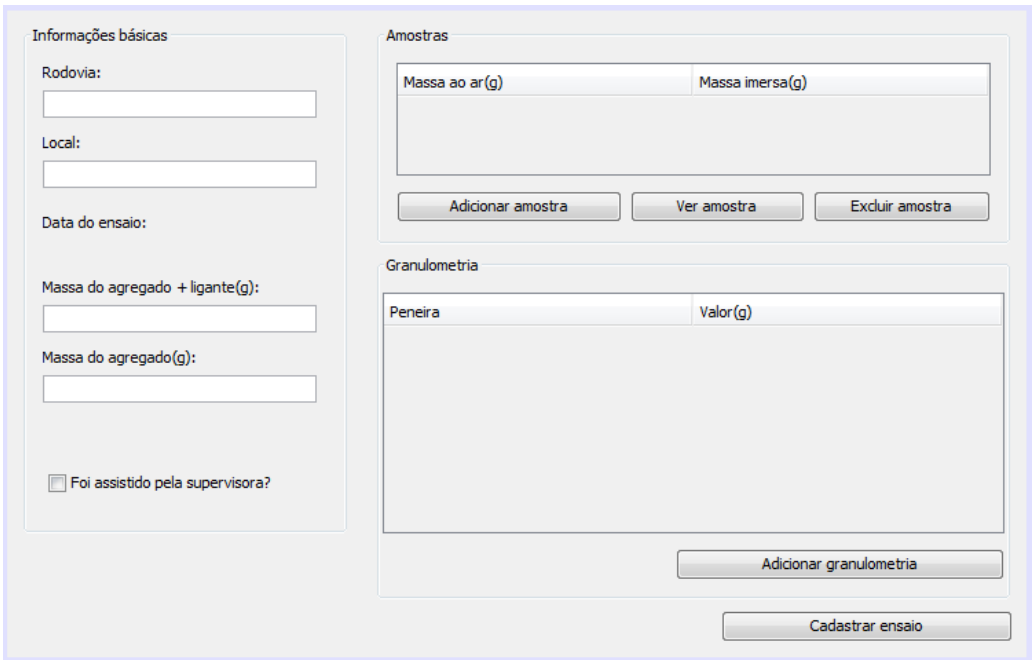
```
                    .addComponent(jPanel3,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    252,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(txtCadastrar)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>

    private void btnAddAmostraActionPerformed(java.awt.event.ActionEvent evt) {
        if(tblAmostras.getRowCount() < 3){
        new frmNovaAmostra(this.ensaio).setVisible(true);
}else{
            showErrorMessage("VocÃª sÃ³ pode cadastrar no mÃ¡ximo trÃªs amostras");
}// TODO add your handling code here:
    }

    private void btnAddGranulometriaActionPerformed(java.awt.event.ActionEvent evt)
{
        if(ensaio.getGranulometria() == null){
            new frmNovaGranulometria(ensaio).setVisible(true);
        }else{
new frmNovaGranulometria(ensaio.getGranulometria(), ensaio).setVisible(true);
}
    }

    private void txtCadastrarActionPerformed(java.awt.event.ActionEvent evt) {
if(isCamposPreenchidos()){
            montarEnsaio();
            if(getController().cadastrarEnsaio(ensaio)){
                showSuccessMessage("Ensaio cadastrado com sucesso!");
dispose();
            }else{
                showErrorMessage();
            }
        }else{
            showCamposVaziosMessage();
        }
    }
```

```
    private void formWindowGainedFocus(java.awt.event.WindowEvent evt) {
preencherAmostras(ensaio.getAmostras());
        if(ensaio.getGranulometria() != null){
            btnAddGranulometria.setText("Editar granulometria");
            preencherGranulometria(ensaio.getGranulometria());
            wrapperGranulometria.setMapFuncion(o ->{
                Peneira p = (Peneira) o;
                Object[] linha = {p.toString(), ensaio.getGranulometria().getPeneira(p) +
p.getUnidade()};
return linha;
            });
            wrapperGranulometria.setItems(Peneira.values());
            wrapperGranulometria.setOnDoubleClickListener((item)->new
frmNovaGranulometria(ensaio.getGranulometria(), ensaio).setVisible(true));

        }
    }

    private void btnVerAmostraActionPerformed(java.awt.event.ActionEvent evt) {
        if(wrapperAmostras.hasItemSelect()){
            new                    frmNovaAmostra(wrapperAmostras.getSelectedItem(),
this.ensaio).setVisible(true);
}else{
            showErrorMessage("Selecione uma amostra para visualizar suas
informaÃ§Ãµes");
}// TODO add your handling code here:
    }

    private void txtPesoActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void btnExcluirAmostraActionPerformed(java.awt.event.ActionEvent evt) {
if(wrapperAmostras.hasItemSelect()){
            ensaio.getAmostras().remove(wrapperAmostras.getSelectedItem());
            preencherAmostras(ensaio.getAmostras());
        }else{
            showErrorMessage("Selecione uma amostra para excluir.");
}// TODO add your handling code here:        // TODO add your handling code here:
}

    public void preencherAmostras(List<Amostra> amostras){
        wrapperAmostras.setItems(amostras);
    }
```

```
    private void montarEnsaio(){
        this.ensaio.setAssistidoPelaSupervisora(checkAssistido.isSelected());
        this.ensaio.setDataEnsaio(txtDataEnsaio.getDate());
        this.ensaio.setLugar(txtLocal.getText());
this.ensaio.setPeso(Double.parseDouble(txtPeso.getText().replace(",", ".")));

this.ensaio.setPesoAgregado(Double.parseDouble(txtPesoAgregado.getText().replac
e(",", ".")));
this.ensaio.setRodovia(txtRodovia.getText());
    }
    public void preencherGranulometria(Granulometria granulometria){

}
    public boolean isCamposPreenchidos(){
        return true;
    }

    // Variables declaration - do not modify
    private javax.swing.JButton btnAddAmostra;
    private javax.swing.JButton btnAddGranulometria;
private javax.swing.JButton btnExcluirAmostra;
    private javax.swing.JButton btnVerAmostra;
    private javax.swing.JCheckBox checkAssistido;
    private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JProgressBar jProgressBar1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JScrollPane jScrollPane4;
    private javax.swing.JTable jTable1;
    private org.jdesktop.swingx.painter.RectanglePainter rectanglePainter1;
    private javax.swing.JTable tblAmostras;
    private javax.swing.JTable tblGranulometria;
    private javax.swing.JButton txtCadastrar;
    private org.jdesktop.swingx.JXDatePicker txtDataEnsaio;
    private javax.swing.JTextField txtLocal;
    private javax.swing.JTextField txtPeso;
    private javax.swing.JTextField txtPesoAgregado;
```

```
    private javax.swing.JTextField txtRodovia;
// End of variables declaration
}
```

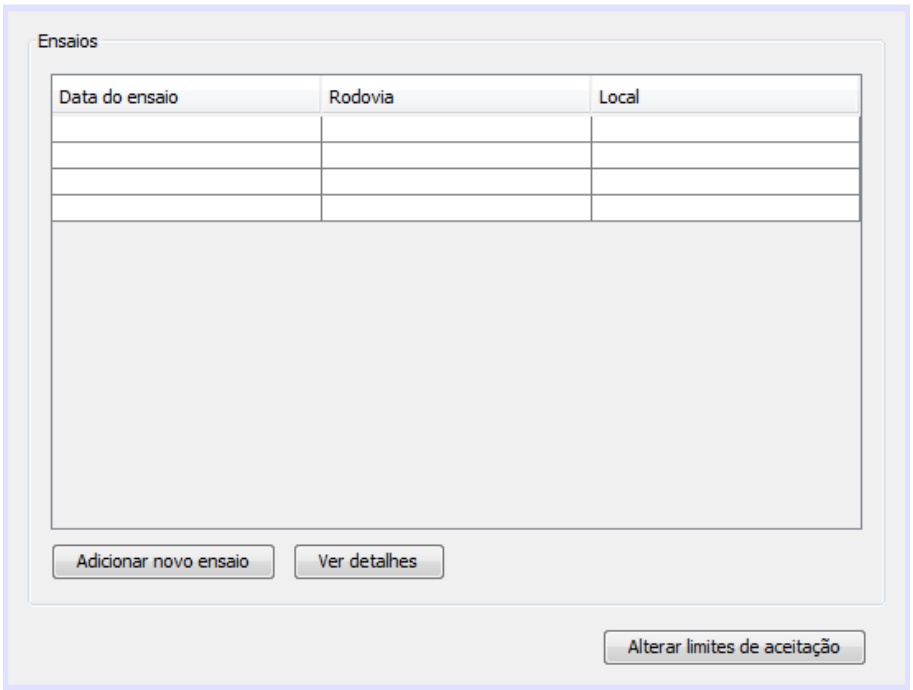
Figura 36 - Tela criada para adicionar um novo ensaio



FONTE: Autoria própria (2017).

- **Dados Principais**

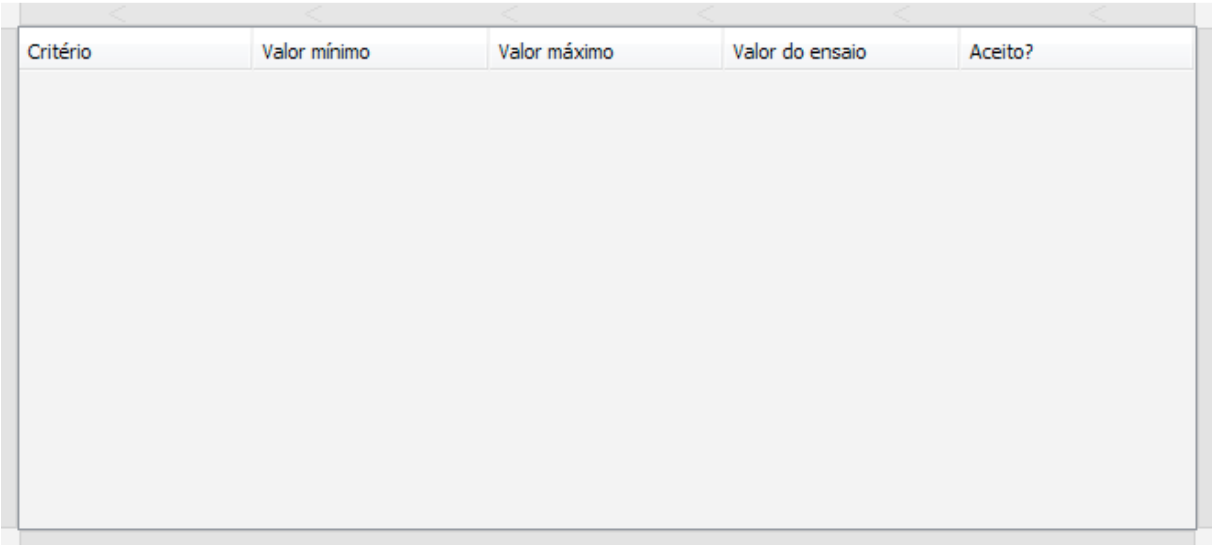
Figura 37 - Tela criada para mostrar os principais dados do ensaio



FONTE: Autoria própria (2017).

- **Tabela de Aceitação**

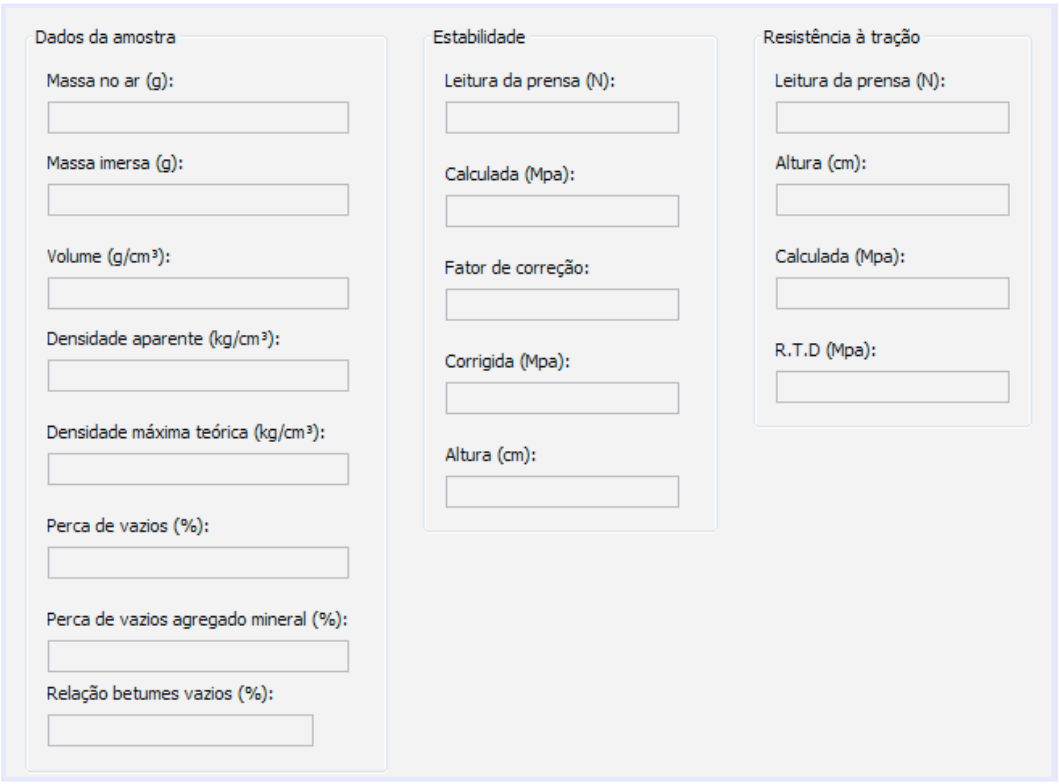
Figura 38 - Tela criada para mostrar a tabela de aceitação do ensaio



FONTE: Autoria própria (2017).

- **Ver Amostra**

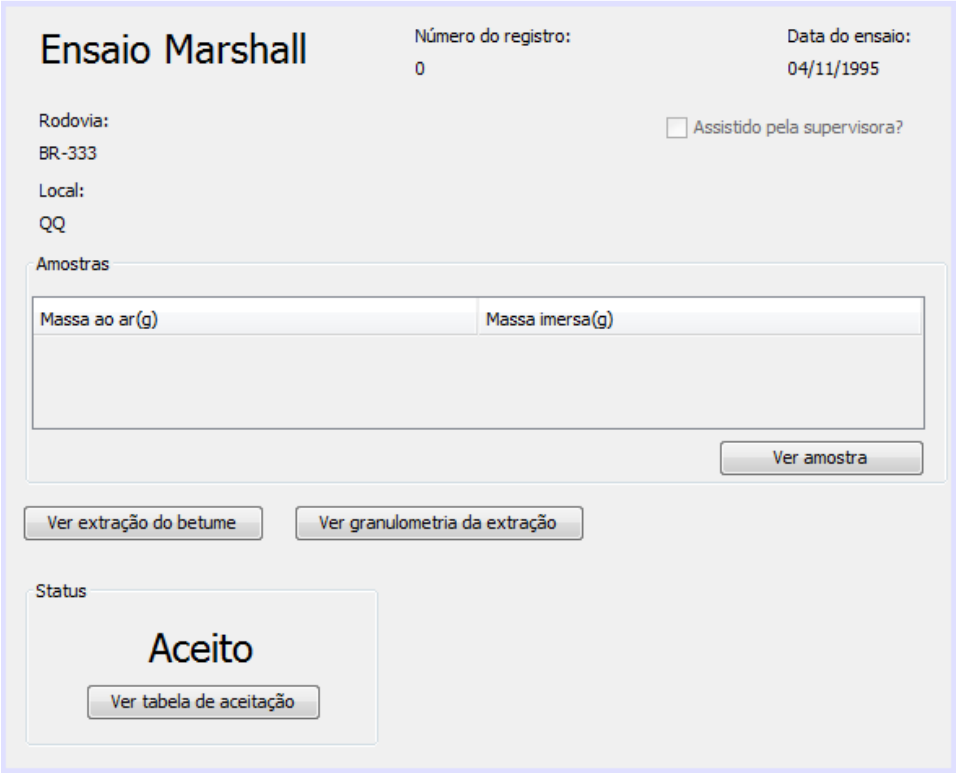
Figura 39 - Tela criada para mostrar os dados das amostras do ensaio



FONTE: Autoria própria (2017).

- **Ver Ensaio**

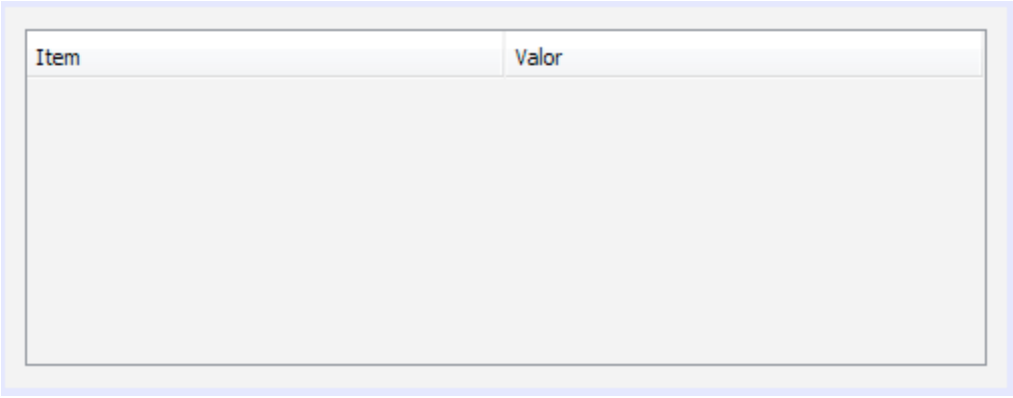
Figura 40 - Tela criada para mostrar o resumo do ensaio



FONTE: Autoria própria (2017).

- **Ver Extração**

Figura 41 - Tela criada para mostrar a extração do ensaio



FONTE: Autoria própria (2017).

- **Ver Granulometria da Extração**

Figura 42 - Tela criada para mostrar a granulometria extração do ensaio

FONTE: Autoria própria (2017).

- **Init**

```
package view;

import javax.swing.JOptionPane;
import model.Projeto;
import org.hibernate.cfg.Configuration;
import org.hibernate.tool.hbm2ddl.SchemaExport;

public class Init {
    public static void main(String args[]){
try{
        Projeto projeto = Projeto.getInstance();
new frmPrincipal().setVisible(true);

        if(!projeto.hasValoresPadroes()){
JOptionPane.showMessageDialog(null, "Iremos iniciar a configuraÃ§Ã£o dos valores de aceitaÃ§Ã£o.");
new frmConfigInicial().setVisible(true);
```

```
        }
        }catch(Exception e){
e.printStackTrace();
            JOptionPane.showMessageDialog(null, "PossÃ-vel erro no acesso ao banco de dados. Verifique sua conexÃ£o com o mySQL!");

        }
    }
}
```

- **JTableWrapper**

```
package view;

import java.awt.Point;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.function.Function;
import java.util.stream.Collectors;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import model.Ensaio;

public class JTableWrapper<T> {
    private JTable table;
    private Function function;
    private List<T> items = new ArrayList<>();
    public JTableWrapper(JTable table) {
        this.table = table;
    }
    public void setMapFuncion(Function f){
        this.function = f;
        setItems(items); // to refresh table
    }
    public void setOnDoubleClickListener(OnJTableDoubleClickListener<T> listener){
        table.addMouseListener(new MouseAdapter() {
            public void mousePressed(MouseEvent me) {
                T itemClicked = getSelectedItem();
                if (me.getClickCount() == 2 && itemClicked != null) {
                    listener.run(itemClicked);
                }
```

```
            }
        });
    }

    public void setItems(List<T> items){
        DefaultTableModel tableModel = (DefaultTableModel) table.getModel();
        tableModel.setRowCount(0);
        List<Object[]> lista = (List<Object[]>)
items.stream().map(this.function).collect(Collectors.toList());
        for(Object[] line: lista){
            tableModel.addRow(line);
        }
        table.setModel(tableModel);
        tableModel.fireTableDataChanged();
        this.items = items;
    }
    public void setItems(T[] items){
        setItems(Arrays.asList(items));
    }
    public T getSelectedItem(){
        int row = table.getSelectedRow();
        if(row > -1){
            return items.get(row);
        }else{
            return null;
        }
    }

    public Boolean hasItemSelect(){
        return table.getSelectedRowCount() == 1;
    }
    public Integer getRowCount(){
return table.getRowCount();
    }
}
```

- **On JTable Double Click Listener**

```
package view;

@FunctionalInterface
interface OnJTableDoubleClickListener<T> {
    public void run(T item);
}
```

- **View**

```
package view;
import controller.Controller;
import javax.swing.JOptionPane;

public class View<T extends Controller> extends javax.swing.JFrame {
    private T controller;
    View(){
        super();
        setResizable(false);

    }
    public T getController() {
        return controller;
    }

    public void setController(T controller) {
        this.controller = controller;
    }

    public void showErrorMessage(){
        JOptionPane.showMessageDialog(null,      getController().getErro(),      "Erro",
JOptionPane.ERROR_MESSAGE);
    }
    public void showErrorMessage(String message){
        JOptionPane.showMessageDialog(null,                message,                "Erro",
JOptionPane.ERROR_MESSAGE);
    }

    public void showSuccessMessage(String mensagem){
        JOptionPane.showMessageDialog(null, mensagem);
    }

    public void showCamposVaziosMessage(){
String mensagem = "Os campos nÃ£o estÃ£o preenchidos corretamente.";
JOptionPane.showMessageDialog(null, mensagem);
    }
}
```

Portanto, vale ressaltar que o “Controller” do modelo MVC utilizadoage como intercessor entre os códigos do “Model” e do “View”, promovendo a interação necessária entre eles.

ANEXO A – FAIXA DE TRABALHO DA MISTURA ASFÁLTICA

Quadro 3 - Faixa de trabalho da mistura asfáltica: Análise granulométrica

Peneira	Massa Mínima passante nas peneiras	Massa Máximapassante nas peneiras
1/2"	100	100
1"	100	100
3/4"	100	100
1/2"	83	97
3/8"	74,5	88,5
Nº4	50,6	60,6
Nº10	29,1	39,1
Nº40	13,2	23,2
Nº80	8,3	14,3
Nº200	4,5	8,5

Fonte: Autoria própria (2017).

Quadro 4 - Faixa de trabalho da mistura asfáltica: Demais parâmetros

Parâmetros Marshall	Mínimo	Máximo
Teor de Asfalto (%)	4,8	5,4
Estabilidade (kgf)	500	-
Deformação (%)	-	-
Resistência à Tração (MPa)	0,65	-
Densidade Aparente (kg/cm³)	-	-
Densidade Real (kg/cm³)	-	-
% Vazios	3	5
% Vazios cheios de Betume	-	-
% V.A.M.	15	-
% R.B.V	75	82

Fonte: Autoria própria (2017).

ANEXO B – EXEMPLO DE ENSAIO MARSHALL DENTRO DOS LIMITES DA FAIXA DE TRABALHO DA MISTURA ASFÁLTICA

Quadro 5 – Primeiro exemplo de ensaio Marshall

INFORMAÇÕES BÁSICAS	
Rodovia	BR-487
Local	ACOST LD 1 CAM KM 238+910
Data do Ensaio	31/10/2017
Massa do Agregado + Ligante (g)	863,4
Massa do Agregado (g)	818,5
Assistido pela Supervisora?	Sim
AMOSTRAS	
Massa do corpo de prova ao ar (g)	1199,4 1199,3 1200,0
Massa do corpo de prova imerso (g)	738,4 737,7 737,4
Deformação do corpo de prova (%)	1,0 1,1 0,9
Leitura da Prensa para Estabilidade (N)	547 539 550
Altura do corpo de prova para Estabilidade (cm)	6,35 6,39 6,35
Leitura da Prensa para Resistência à Tração (N)	547 539 550
Altura do corpo de prova para Resistência à Tração (cm)	5,76 5,76 5,77
GRANULOMETRIA - MASSA RETIDA POR PENEIRA (g)	
1" 1/2	0
1"	0
3/4"	0
1/2"	53,6
3/8"	126
Nº 4	342,9
Nº 10	568,8
Nº 40	640,8
Nº 80	734,1
Nº 200	757,9
FUNDO	818,5

Fonte: Autoria própria (2017).

ANEXO C – EXEMPLO DE ENSAIO MARSHALL FORA DOS LIMITES DA FAIXA DE TRABALHO DA MISTURA ASFÁLTICA

Quadro 6 - Segundo exemplo de ensaio Marshall

INFORMAÇÕES BÁSICAS	
Rodovia	BR-487
Local	3º FAIXA LE 1 CAM KM 241+050
Data do Ensaio	22/11/2017
Massa do Agregado + Ligante (g)	768,7
Massa do Agregado (g)	718,4
Assistido pela Supervisora?	Sim
AMOSTRAS	
Massa do corpo de prova ao ar (g)	1199,1 1199,5 1199,2
Massa do corpo de prova imerso (g)	737,5 738,7 738,1
Deformação do corpo de prova (%)	1,0 0,9 0,9
Leitura da Prensa para Estabilidade (N)	539 559 559
Altura do corpo de prova para Estabilidade (cm)	6,39 6,35 6,35
Leitura da Prensa para Resistência à Tração (N)	539 559 559
Altura do corpo de prova para Resistência à Tração (cm)	5,76 5,82 5,8
GRANULOMETRIA - MASSA RETIDA POR PENEIRA (g)	
1" 1/2	0
1"	0
3/4"	0
1/2"	100,9
3/8"	175,9
Nº 4	307,4
Nº 10	484,8
Nº 40	581,2
Nº 80	650,6
Nº 200	684,2
FUNDO	730,3

Fonte: Autoria própria (2017).