



**UNIVERSIDADE FEDERAL DO CEARÁ**  
**Campus de Quixadá**

# **Relatório de Medição de Qualidade**

*Felipe Raulino Lemos*  
*Ítalo Rodrigues Nascimento*  
*Mateus Gonçalves Loiola*

**Responsável:** Equipe CliniVet

# ÍNDICE

GLOSSÁRIO	3
HISTÓRICO DE REVISÕES	3
1. INTRODUÇÃO	4
<b>1.1. Descrição do Produto</b>	4
<b>1.2. Objetivos da Avaliação</b>	4
2. MÉTODO	4
<b>2.1. Participantes (caso necessite)</b>	4
<b>2.2. Contexto de Uso</b>	4
<b>2.3. Procedimentos da Avaliação</b>	4
<b>2.4. Medidas de Software Coletadas</b>	4
2.4.1. Eficácia	4
2.4.2. Eficiência	5
2.4.2. Manutenibilidade	5
3. RESULTADOS	5
4. REFERÊNCIAS	5

## GLOSSÁRIO

<b>Siglas</b>	<b>Definição</b>
UFC	Universidade Federal do Ceará
LOC	Lines of Code / Linhas de Código
Tarefa 1	Realizar cadastro no aplicativo
Tarefa 2	Realizar login no aplicativo
Tarefa 3	Realizar agendamento de serviço
Tarefa 4	Visualizar os agendamentos de consultas
Tarefa 5	Administrador verificar os agendamentos finalizados
Tarefa 6	Administrador cadastrar os serviços disponibilizados pela clínica veterinária
Tarefa 7	Administrador alterar status de um serviço agendado

## HISTÓRICO DE REVISÕES

<b>Data</b>	<b>Versão</b>	<b>Descrição</b>	<b>Responsável</b>
03/12/2022	1.0	Criação do documento	CliniVet
05/12/2022	1.1	Atualização do documento	CliniVet
06/12/2022	1.2	Atualização do documento	CliniVet
07/12/2022	2.0	Atualização para apresentação	CliniVet

# 1. INTRODUÇÃO

Esse documento destina-se a todas as partes interessadas no aplicativo mobile CliniVet. O documento apresenta uma visão objetiva, através de dados coletados empiricamente, sobre métricas que avaliam aspectos específicos do aplicativo. O público pode ser referenciado desde desenvolvedores até consumidores administrativos da plataforma. De qualquer forma, o documento fica disponibilizado para todos que o tenham interesse.

## 1.1. Descrição do Produto

O produto a ser avaliado, a partir das métricas propostas por esse documento, é o aplicativo CliniVet. O app tem como propósito, entre outras funcionalidades, facilitar o processo de agendamento de serviços comumente atrelados a uma clínica veterinária. Dessa forma, espera-se obter vantagens nas duas pontas do processo (Cliente - Administrador), de acordo com os objetivos específicos dessas duas partes.

## 1.2. Objetivos da Avaliação

A avaliação tem como objetivo analisar e validar atributos de qualidade da aplicação em questão. Isso será feito a partir das métricas selecionadas, com elas podemos inferir interpretações sobre aquilo que se foi proposto e comparar verificando se os resultados são compatíveis com os resultados. As métricas escolhidas foram:

1. Eficácia
  - a. Completude da tarefa
  - b. Frequência de erros
2. Eficiência de desempenho
  - a. Tempo de resposta por tarefas
  - b. Tempo de resposta instantânea
3. Manutenibilidade
  - a. Quantidade de linhas por métodos de uma classe
  - b. Quantidade de linhas por classe

Analisar	Aplicativo CliniVet
Para o propósito de	Avaliar e certificar os atributos de qualidade desejáveis a partir das métricas selecionadas para a coleta de dados.
Com respeito a	Foco na análise e correta execução das tarefas propostas pelo aplicativo.
Do ponto de vista	Pessoas que possuem interesse no campo de atuação do aplicativo. Donos de clínicas veterinárias, donos de pets...
No contexto de	Dispositivos móveis com sistema operacional Android

## **2. MÉTODO**

### **2.1. Participantes (caso necessite)**

Inicialmente os testes passarão por 3 usuários reais que dispõem de certa experiência no uso de aplicativos em dispositivos móveis com o sistema operacional Android. Os participantes estão em uma faixa etária de 18 a 24 anos. No grupo de participantes incluíram apenas os usuários que possuem um pet para dar mais veracidade nas informações prestadas. Nessa perspectiva, as experiências de uso de sistemas mobile variam, o que pode representar um aspecto positivo para a avaliação. Dessa forma, como os usuários escolhidos possuem familiaridades diferentes no uso de aplicativos, a capacidade de encontrar possíveis bugs/erros é maior, desse modo os testes com os participantes ajudaram o time de desenvolvimento melhorar aplicação.

### **2.2. Contexto de Uso**

- Tarefas:
  - o Realizar cadastro no aplicativo
  - o Realizar login no aplicativo
  - o Realizar agendamento de serviço
  - o Visualizar os agendamentos de consultas
  - o Administrador verificar os agendamentos finalizados.
  - o Administrador cadastrar os serviços disponibilizados pela clínica veterinária
  - o Administrador alterar status de um serviço agendado
- Ambiente:
  - o Ambiente real 1, Smartphone: Moto G5s, Android 8.1, 2GB RAM
  - o Ambiente real 2, Smartphone: Motorola One Macro, Android 10, 4GB RAM

### **2.3. Procedimentos da Avaliação**

A execução das tarefas selecionadas inicialmente passará pela equipe CliniVet, portanto, atributos de eficácia e eficiência de desempenho serão testados e os dados serão colhidos a partir dessa abordagem. Posteriormente, serão realizados testes com possíveis usuários da aplicação.

Na fase de avaliação com os participantes será solicitado o preenchimento de um formulário para entendermos o nível de aptidão dos usuários participantes com tecnologias de modo geral, com o uso de aplicativos e com o uso de aplicativos com o nosso propósito específico. Por ética, sempre devemos informar aos integrantes que participaram dos testes que estamos analisando atributos de qualidade do aplicativo e não a capacidade individual dos mesmos.

## 2.4. Medidas de Software Coletadas

### 2.4.1. Eficácia

Nome	Descrição	Função de Medição	Método
Completeness da tarefa	Qual a proporção das tarefas são concluídas corretamente?	<b><math>X = A/B</math></b> A = Número de tarefas concluídas B = Número total de tarefas que o usuário tentou executar	Teste com usuários
Frequência de erros	Qual é a frequência de erros cometidos pelo usuário em relação a um valor-alvo?	<b><math>X = A/B</math></b> A = número de erros cometidos pelos usuário B = número de tarefas (ou pode ser o tempo)	Teste com usuários

### 2.4.2. Eficiência de desempenho

Nome	Descrição	Função de Medição	Método
Tempo da tarefa	Quanto tempo leva para o usuário completar uma tarefa?	<b><math>X = \text{tempo da tarefa}</math></b>  Note: Você pode comparar com o tempo de um usuário especialista	Teste com usuários

### 2.4.3. Manutenibilidade

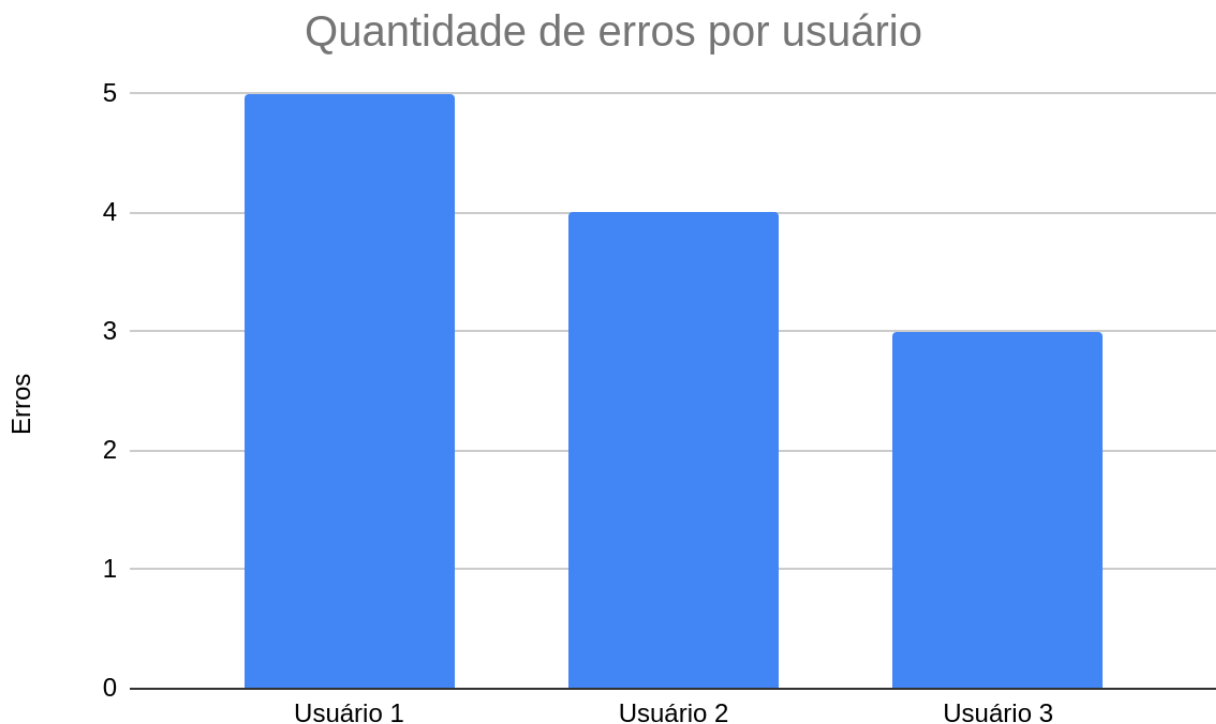
Nome	Descrição	Função de Medição	Método
Quantidade de linhas por métodos de uma classe	Qual a quantidade de linhas em média dos métodos de uma classe ?	<b><math>X = A/B</math></b> A = Linhas de código B = Quantidade de métodos  Nota: As linhas de código desconsideram comentários e linhas de espaço	Avalia a complexidade e de métodos de uma classe

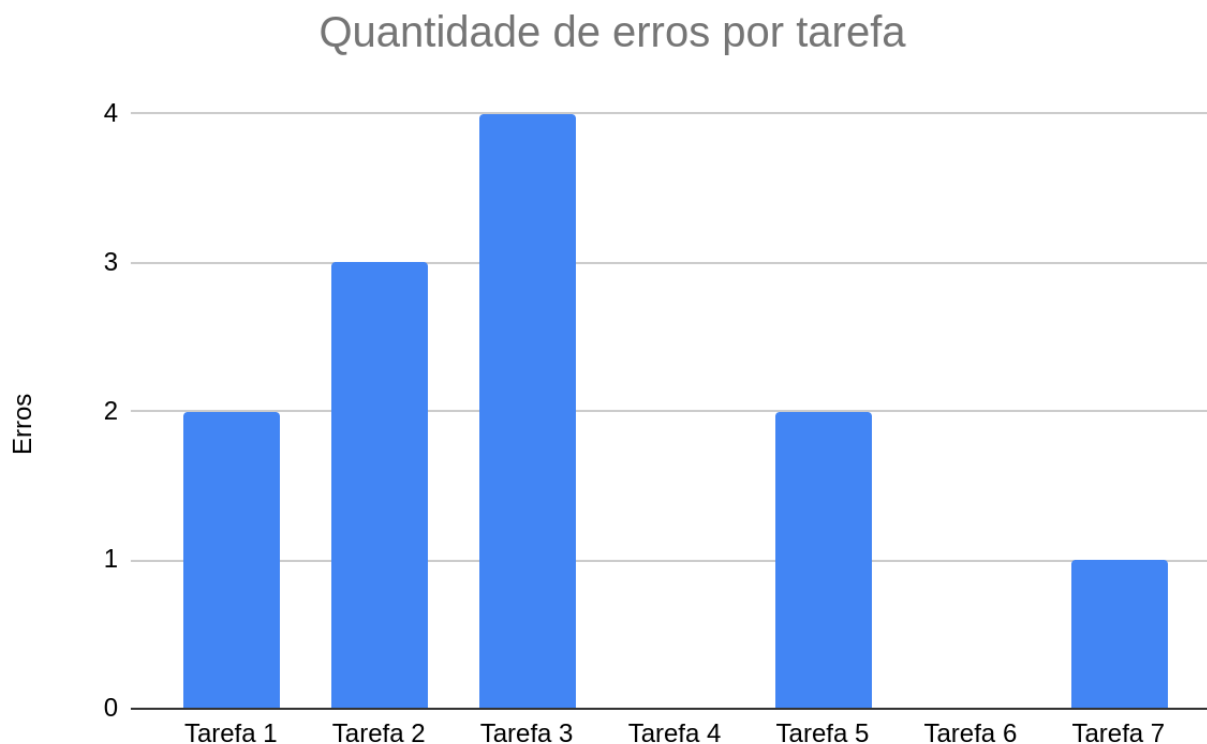
Nome	Descrição	Função de Medição	Método
Quantidade de linhas por classe	Qual a quantidade de linhas em uma determinada classe ?	<b><math>X = \text{Quantidade de linhas da classe}</math></b>	Avalia a possível complexidade e facilidade de interpretação de uma classe

### 3. RESULTADOS

#### 3.1. Métricas de Eficácia

Usuário	Compleitude da tarefa	Frequência de erros
#1	A = 7 tarefas concluídas B = 7 tarefas passadas	A = 5 erros B = 7 tarefas
	X = 100% das tarefas concluídas	X = 0,86 erros por tarefa
#2	A = 7 tarefas concluídas B = 7 tarefas passadas	A = 4 erros B = 7 tarefas
	X = 100% das tarefas concluídas	X = 0,58 erros por tarefa
#3	A = 7 tarefas concluídas B = 7 tarefas passadas	A = 3 erros B = 7 tarefas
	X = 100% das tarefas concluídas	X = 0,43 erros por tarefa





Os dados permitem a interpretação de que as tarefas 3 e 2 tiveram os maiores números de erros dos usuários ao tentarem executá-la com assertividade. Um dos motivos da quantidade de erros na tarefa 3 é o fato da necessidade de se ter um pet para realizar o agendamento de um serviço.

## 3.2. Métricas de Eficiência de Desempenho

### 3.2.1. Resultados por tarefa

#### 1. Realizar cadastro no aplicativo

Usuário	Tempo da tarefa
#1	X = 55,37 segundos
#2	X = 42,38 segundos
#3	X = 34,59 segundos





## 2. Realizar login no aplicativo

Usuário	Tempo da tarefa
#1	X = 40,86 segundos
#2	X = 12,16 segundos
#3	X = 12,26 segundos



### 3. Realizar agendamento de serviço

Usuário	Tempo da tarefa
#1	X = 75,29 segundos
#2	X= 70,12 segundos
#3	X= 91,45 segundos

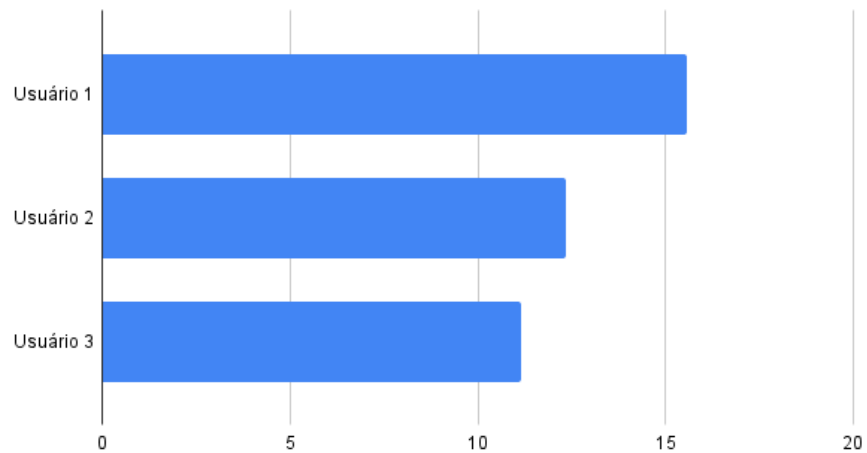


### 4. Cliente visualizar os agendamentos de consultas.

Usuário	Tempo da tarefa
#1	X = 15,57 segundos
#2	X= 12,34 segundos

**#3** X= 11,17 segundos

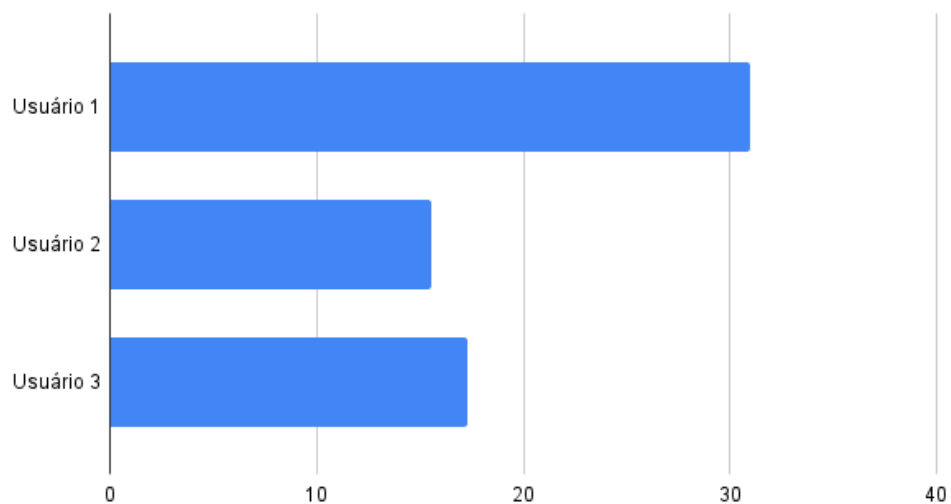
**Tempo de realização da tarefa em segundos**



**5. Administrador verificar os agendamentos finalizados.**

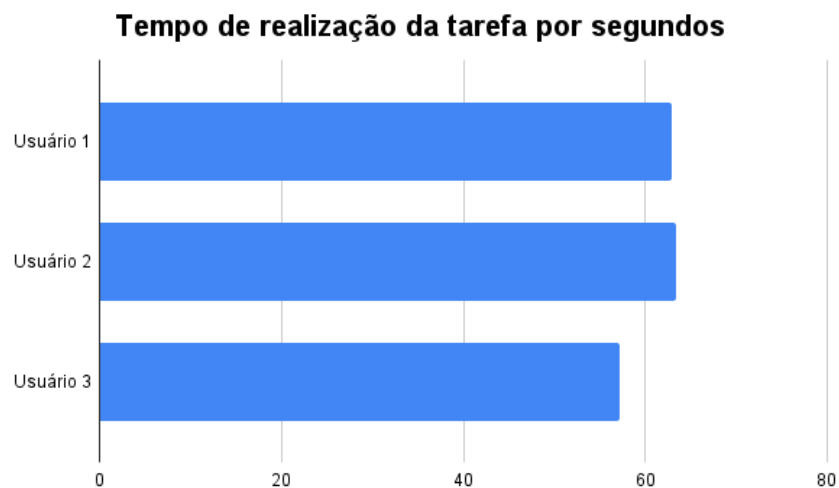
Usuário	Tempo da tarefa
<b>#1</b>	X = 30,96 segundos
<b>#2</b>	X= 15,52 segundos
<b>#3</b>	X= 17,28 segundos

**Tempo de realização da tabela em segundos**



**6. Administrador cadastrar os serviços disponibilizados pela clínica veterinária**

Usuário	Tempo da tarefa
#1	X = 62,84 segundos
#2	X = 63,32 segundos
#3	X = 57,12 segundos



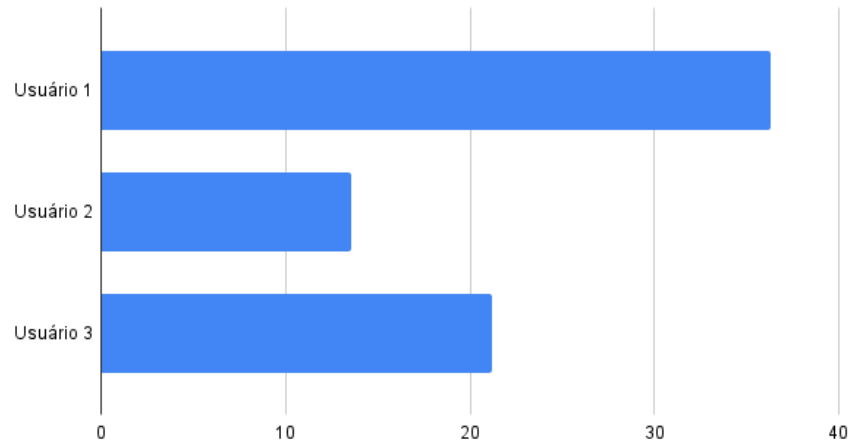
**7. Administrador alterar status de um serviço agendado**

Usuário	Tempo da tarefa
#1	X = 36,31 segundos
#2	X = 13,54 segundos

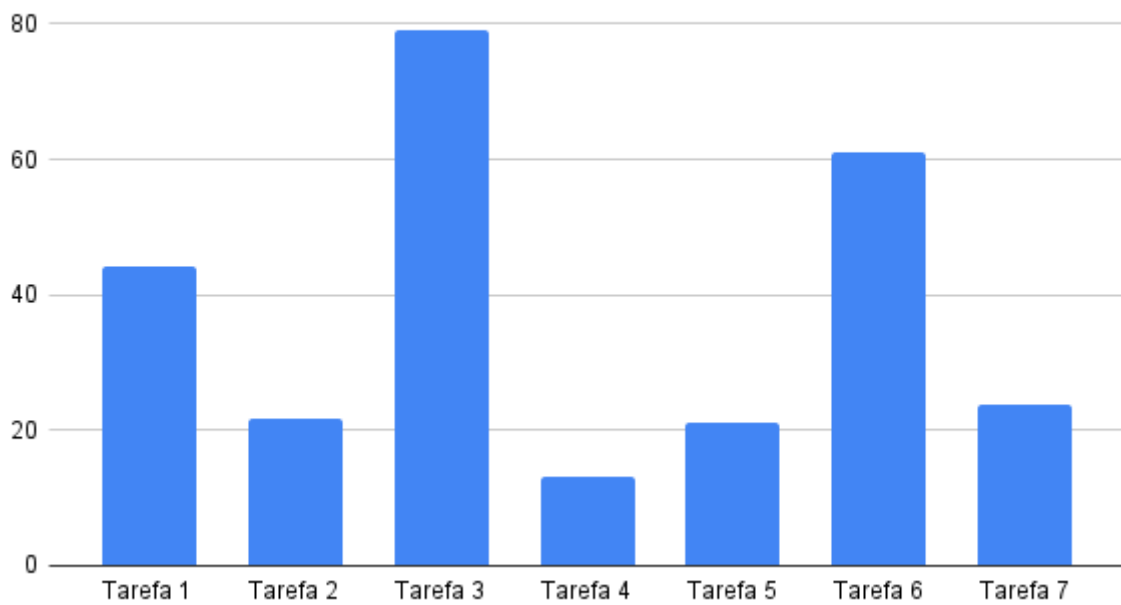
**#3**

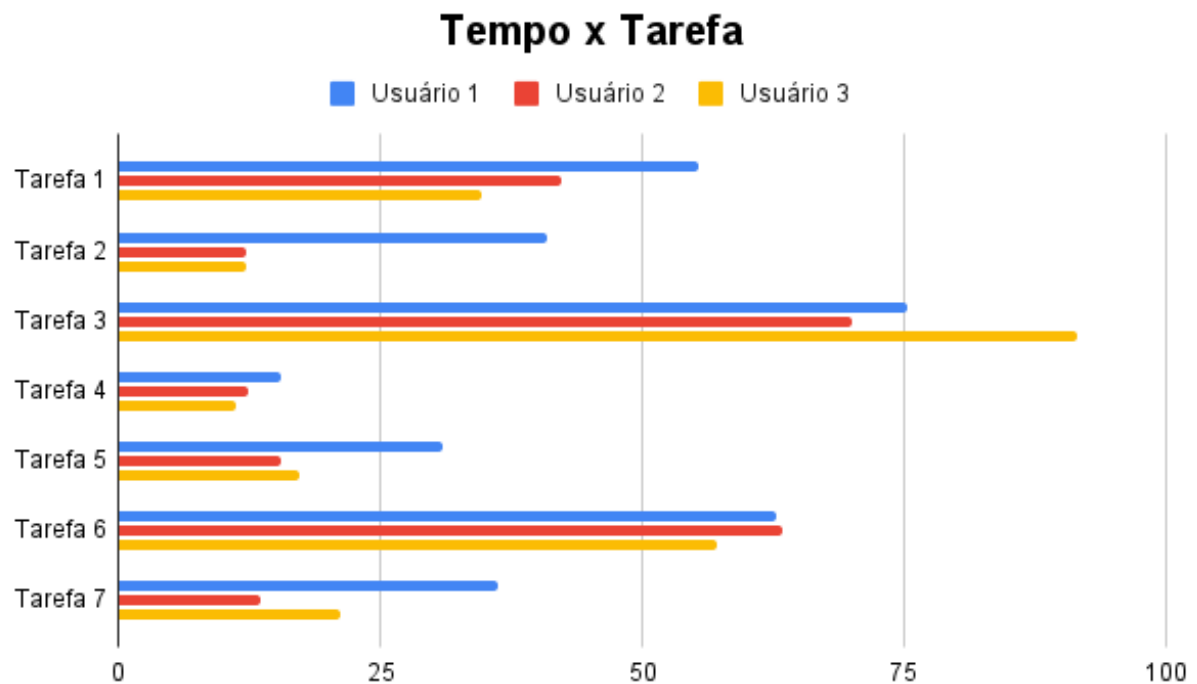
X = 21,18 segundos

**Tempo de realização da tarefa por segundos**



**Média de tempo dos usuários por tarefa**





De acordo com os resultados obtidos, traçando-se uma relação entre tempo e tarefas sendo realizadas, chegamos na conclusão que a tarefa 3, realizar agendamento de serviço, foi a tarefa com um maior tempo de conclusão. Atribuímos esses resultados ao fato do fluxo de execução do aplicativo exigir que o usuário já tenha um pet para realizar o agendamento, possivelmente esse fator não tenha ficado claro aos usuários na execução da tarefa. Os resultados podem servir como pontos a serem melhorados, com o objetivo de tornar o aplicativo o mais claro e fácil de ser utilizado.

### 3.3. Métricas Internas de Manutenibilidade

Para a coleta das métricas internas dos atributos de qualidade foram utilizadas duas plataformas, são elas, [Codefactor](#) e [Codacy](#). Ambas as plataformas têm como objetivo através de uma análise do código, encontrar erros, más práticas, problemas em geral, que prejudiquem aspectos como capacidade e facilidade de interpretação de trechos de código. Em ambas as plataformas a análise é feita de forma simples, basta colar o endereço do repositório na página principal de ambos os sites. O objetivo ao utilizar ambas as plataformas foi a de ter uma análise mais segura sobre os dados obtidos em ambas as verificações.

#### 3.3.1. Sistema de avaliação e resultados

Ambas as plataformas possuem um sistema de avaliação em notas que assegura a qualidade do código do repositório fornecido. Tanto na Codefactor como na Codacy as notas são feitas em um intervalo de A até F. Na Codacy a nota representa uma média dos atributos de qualidade avaliados, são eles, issues, complexidade, duplicação, e cobertura. Em ambas as plataformas, após os testes serem realizados, obteve-se nota A, uma nota que representa poucos problemas de escrita de código segundo os critérios e avaliações da plataforma.

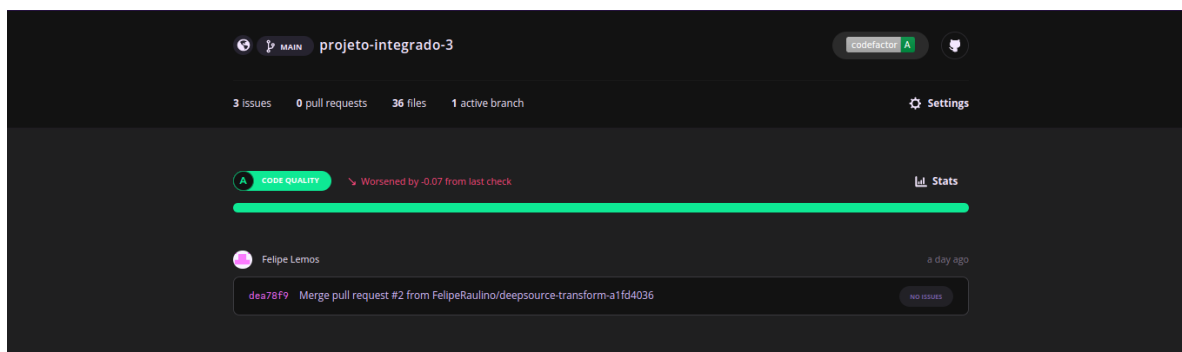


Imagem 1: Avaliação geral na Codefactor

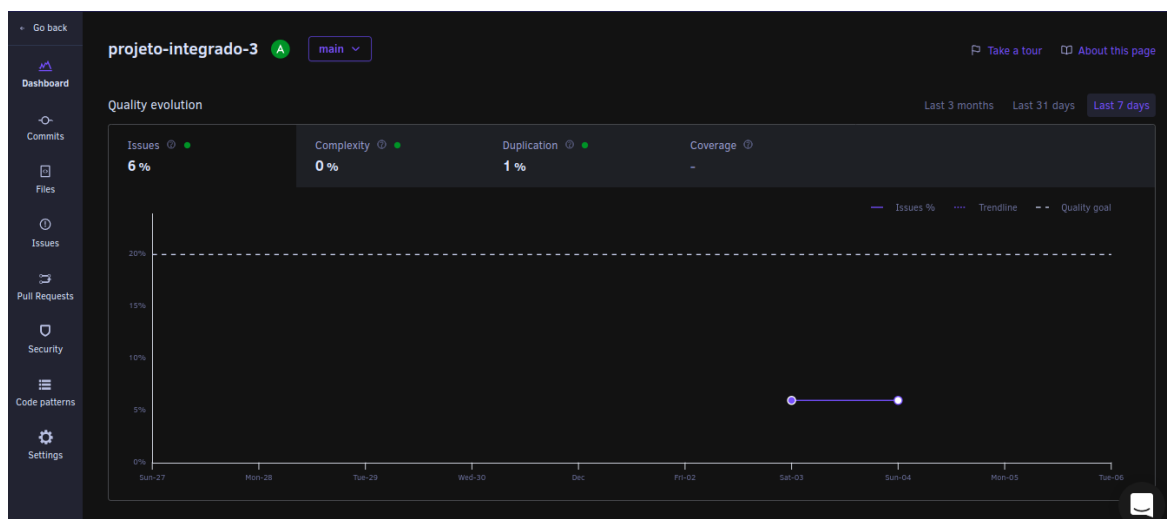


Imagem 2: Avaliação geral na Codacy

### 3.3.2. Quantidade de linhas por métodos de uma classe

Em [artigo](#) produzido por Jim Bird, é discutida a análise de qualidade de código a partir do tamanho de componentes que compõem um projeto, sejam eles os pacotes, as classes, os métodos. No artigo referênciam-se citações como a de Bob Martin, em Clean Code:

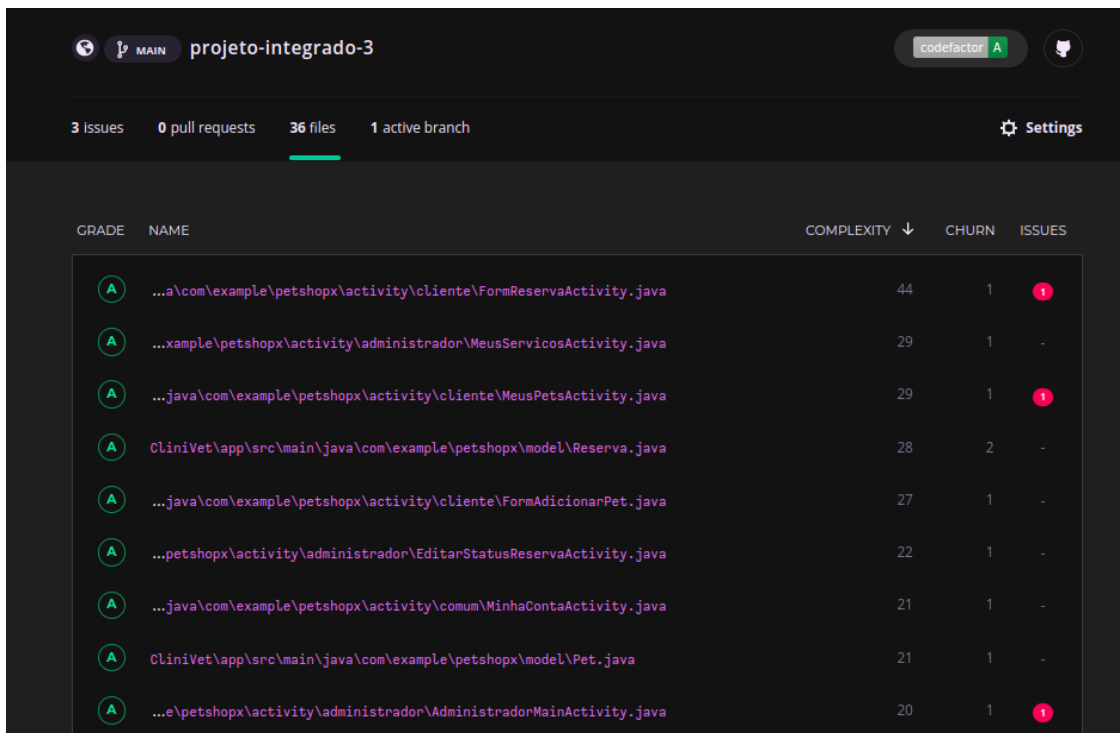
*“A primeira regra para funções é que elas devem ser pequenas. A segunda regra é que elas devem ser menores que isso. Funções não podem ter um tamanho de mais de 100 linhas. As Funções devem, no máximo, terem 20 linhas de tamanho.”*

No artigo, é citado o [Refactoring in Large Software Projects](#), de Martin Lippert e Stephen Roock, nele define-se a regra das/dos 30. No contexto da nossa métrica, recomenda-se que métodos não devem ter uma média de linhas de código maior que 30.

O Codefactor fornece o cálculo da média de linhas por método de uma classe. Portanto, para colher os resultados dessa avaliação utilizaremos o Codefactor. O Codefactor permite listarmos as classes do projeto de acordo com a complexidade, a complexidade de uma classe é medida da seguinte forma:

**contagem do número de caminhos independentes que a classe pode executar até seu fim / número de funções ou métodos.**

Essas foram as classes mais complexas da nossa aplicação:



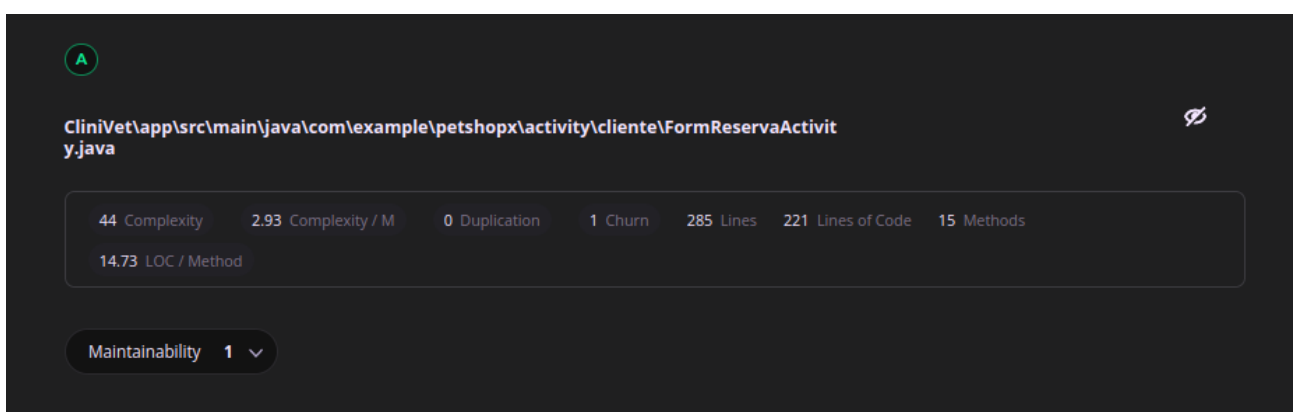
GRADE	NAME	COMPLEXITY ↓	CHURN	ISSUES
A	...a\com\example\petshopx\activity\cliente\FormReservaActivity.java	44	1	1
A	...xample\petshopx\activity\administrador\MeusServicosActivity.java	29	1	-
A	...java\com\example\petshopx\activity\cliente\MeusPetsActivity.java	29	1	1
A	CliniVet\app\src\main\java\com\example\petshopx\model\Reserva.java	28	2	-
A	...java\com\example\petshopx\activity\cliente\FormAdicionarPet.java	27	1	-
A	...petshopx\activity\administrador\EditarStatusReservaActivity.java	22	1	-
A	...java\com\example\petshopx\activity\comum\MinhaContaActivity.java	21	1	-
A	CliniVet\app\src\main\java\com\example\petshopx\model\Pet.java	21	1	-
A	...e\petshopx\activity\administrador\AdministradorMainActivity.java	20	1	1

Imagem 3: Listagem das classes mais complexas da aplicação feita pela Codefactor

Dessa forma, a equipe achou prudente colher os resultados das 5 classes mais complexas para realizar a avaliação. Esses foram os resultados:

**Classe:** FormReservaActivity.java

**Resultado:** 14.73 LOC/Method



CliniVet\app\src\main\java\com\example\petshopx\activity\cliente\FormReservaActivit y.java						
44 Complexity	2.93 Complexity / M	0 Duplication	1 Churn	285 Lines	221 Lines of Code	15 Methods
14.73 LOC / Method						
Maintainability 1						

Imagem 4: Resultados da classe FormReservaActivity.java

**Classe:** MeusServicosActivity.java

**Resultado:** 9.53 LOC/Method



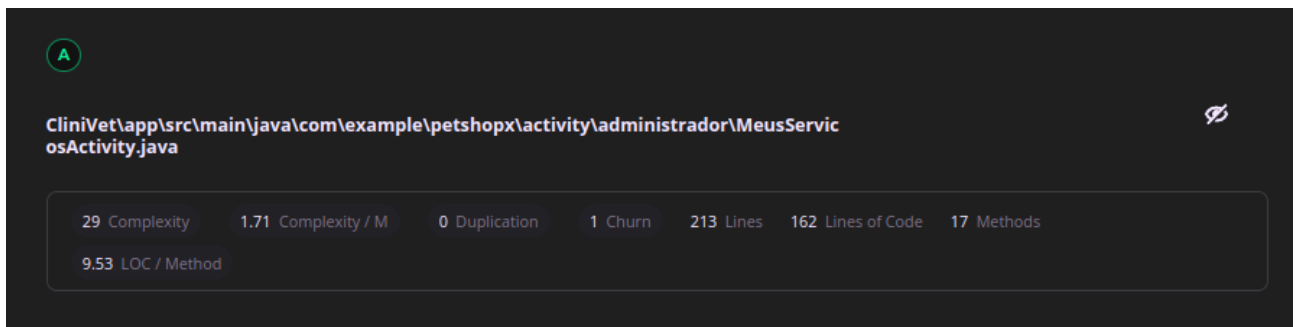


Imagem 5: Resultados da classe MeusServicosActivity.java

**Classe:** MeusPetsActivity.java  
**Resultado:** 9.76 LOC/Method

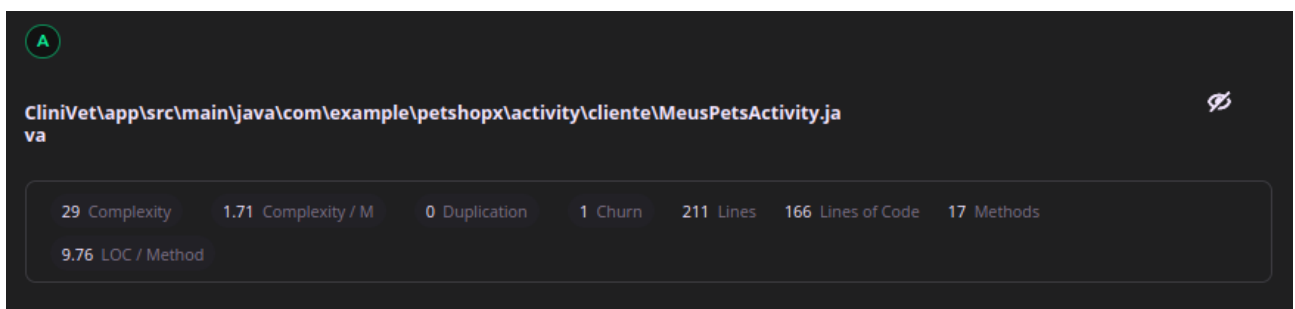


Imagem 6: Resultados da classe MeusPetsActivity.java

**Classe:** Reserva.java  
**Resultado:** 4.64 LOC/Method

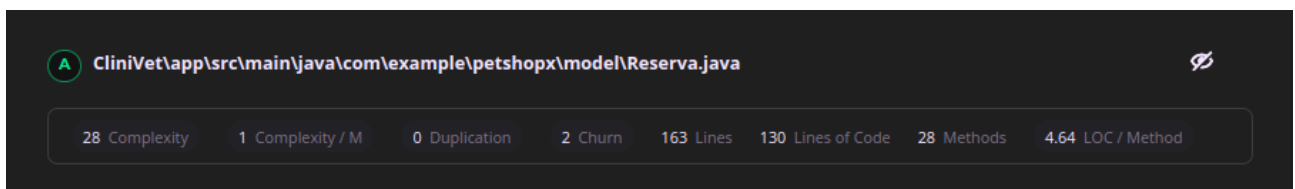


Imagem 7: Resultados da classe Reserva.java

**Classe:** FormAdicionarPet.java  
**Resultado:** 16.5 LOC/Method

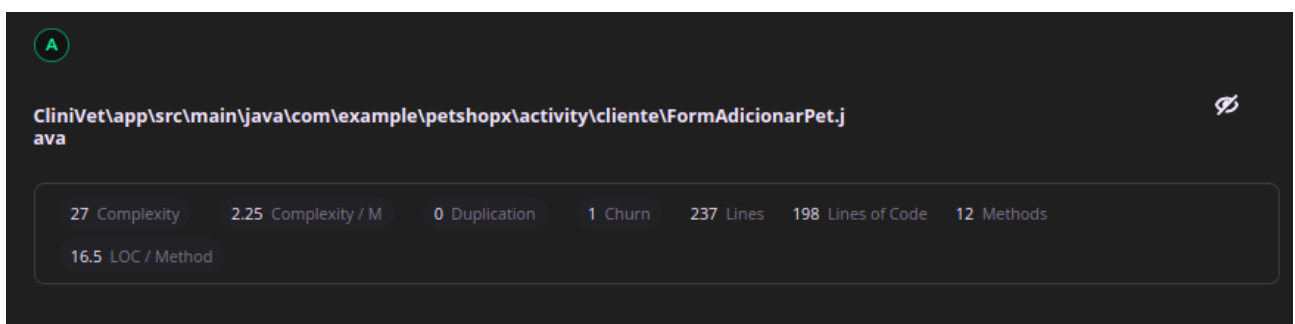


Imagem 8: Resultados da classe FormAdicionarPet.java

Logo, podemos inferir que os resultados foram satisfatórios para as cinco classes mais complexas da aplicação, segundo o Codefactor.

### 3.3.3. Quantidade de linhas por classe

Tendo como base a mesma abordagem utilizada na métrica anterior, a equipe achou prudente avaliar as cinco classes mais complexas. Porém, agora, analisa-se a quantidade de linhas da classe. Vale ressaltar que o Codefactor oferece a possibilidade de retirarmos a quantidade de linhas de código, ou seja, linhas de espaços ou de comentários são descartadas. No livro Clean Code, destaca-se que uma classe, assim como funções ou métodos, devem ser “*Menor do que uma classe considerada pequena*”. O ganho seria a facilidade de leitura e compreensão de um bloco de código. Os resultados foram:

**Classe:** FormReservaActivity.java

**Resultado:** 221 linhas de código

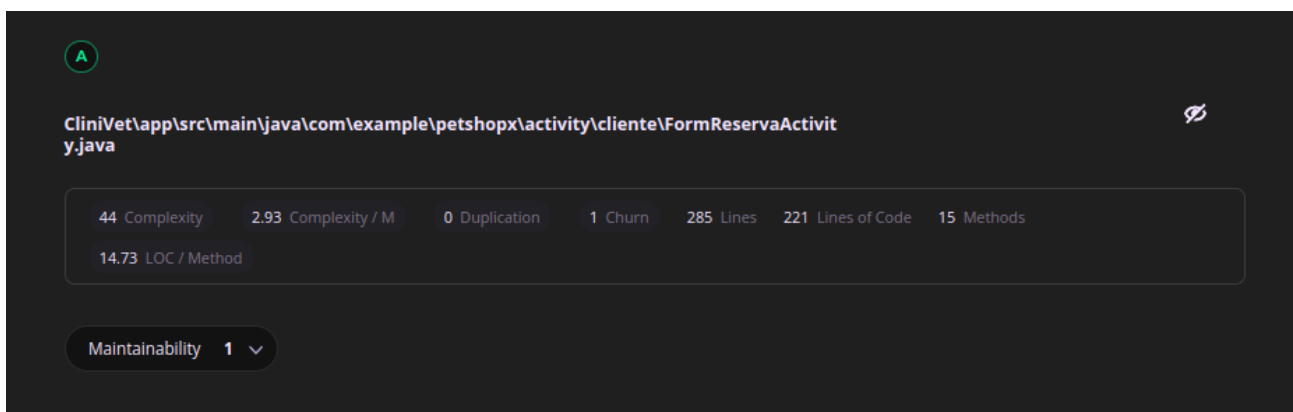


Imagem 4: Resultados da classe FormReservaActivity.java

**Classe:** MeusServicosActivity.java

**Resultado:** 162 linhas de código

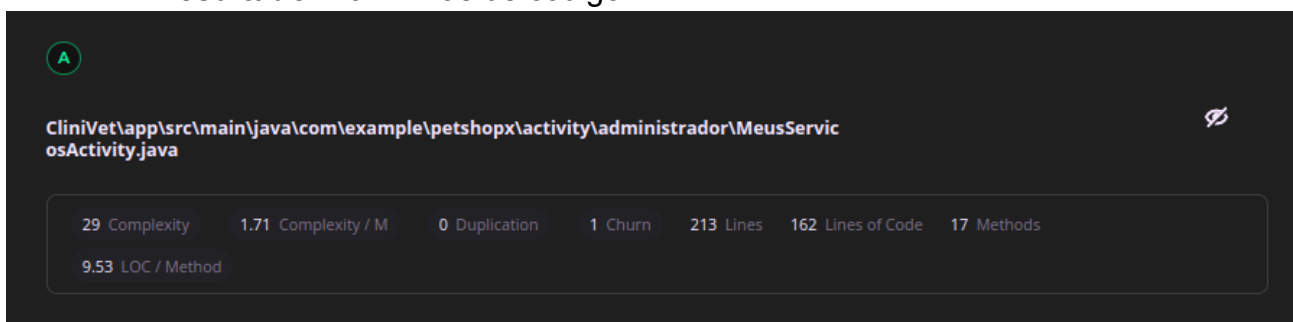


Imagem 5: Resultados da classe MeusServicosActivity.java

**Classe:** MeusPetsActivity.java

**Resultado:** 166 linhas de código

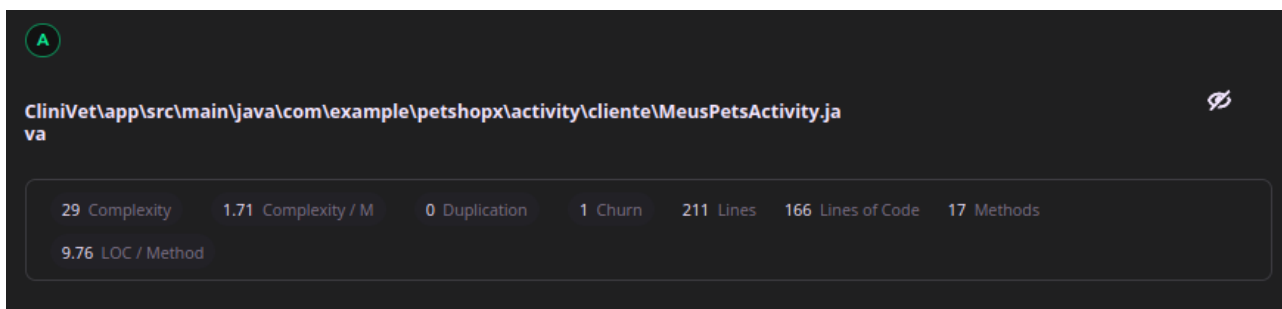


Imagem 6: Resultados da classe MeusPetsActivity.java

**Classe:** Reserva.java

**Resultado:** 130 linhas de código

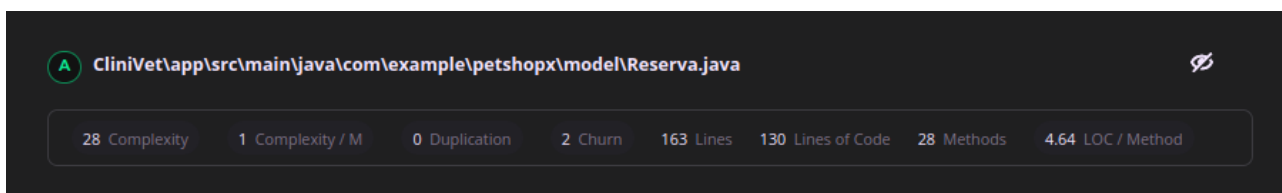


Imagem 7: Resultados da classe Reserva.java

**Classe:** FormAdicionarPet.java

**Resultado:** 198 linhas de código

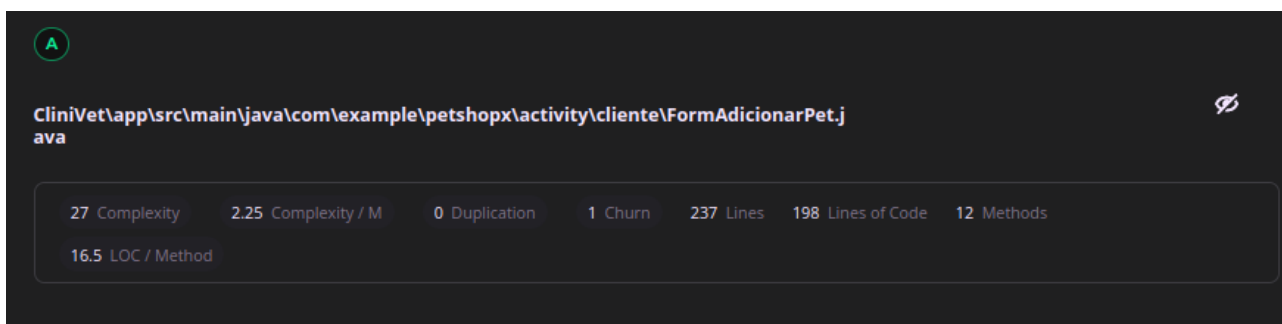


Imagem 8: Resultados da classe FormAdicionarPet.java

Tendo em vista os resultados obtidos e aquilo que se é pregado por metodologias ágeis e a discussão do artigo, as cinco classes mais complexas da aplicação, segundo o Codefactor, estão dentro de uma adequação e por isso possuem uma boa avaliação.

## 4. REFERÊNCIAS

ISO/IEC 25000. Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE. v. 2005, 2005.

ISO/IEC 9126. Software Engineering – Product Quality – Part 1. 2001

[Artigo para embasamento das métricas internas](#)