

**Documentação de um**

**Produto de Software**

**Dog Walker**

**Eric Batista de Santana**

**Gabriel Ramalho Berti**

**Gabriel Maireno de Lima**

**Manoel Victor Castor Simões**

**Rafael Francisco Fortes Lacerda**

**2020**

**ÍNDICE DETALHADO**

**1.** **Introdução** 3

1.1. Tema 3

1.2. Objetivos a serem alcançados 3

1.3. Escopo principal 3

**2.** **Definição do Modelo de Processo** 4

**3.** **Requisitos do Sistema de Software** 4

3.1. Requisitos Funcionais 4

3.2. Requisitos Não-Funcionais 5

3.3 REGRAS DE negócio 5

**4.** **Projeto** 6

4.1. Arquitetura Lógica 6

**5.** **Protótipo de Interface** 8

5.1 Protótipo de Interface (Cliente) 8

5.2 Protótipo de Interface (Andador) 10

**6.** **Critérios de Qualidade de Software** 12

**7. Testes** 15

7.1. Plano de Testes 15

7.2. Roteiro de Testes 20

1. **Introdução**
   1. **Tema**

O tema do projeto é a criação de um aplicativo que visa facilitar os passeios com cachorros.

* 1. **Objetivos a serem alcançados**

O principal objetivo a ser alcançado com o desenvolvimento deste software, é facilitar a vida de pessoas que, por conta de fatores exteriores, acabam não tendo um tempo livre para passear com seu cachorro. O software busca atingir todos os públicos, desde recém-adultos a pessoas mais velhas. Nós buscamos a praticidade no uso, por esse motivo, a plataforma de desenvolvimento será voltada para o mobile, tendo em vista que a maioria da população possui um dispositivo celular e o mesmo está sempre presente. O aplicativo contará com a API do Google Maps, visando à facilitação do uso, tendo em vista que é uma API de fácil entendimento e por sua vez, acaba garantindo uma alta acessibilidade.

* 1. **Escopo principal**

O projeto visa garantir um aplicativo funcional e de fácil manuseio para quem busca uma ajuda quando o assunto é passear com seu cachorro. O aplicativo contém uma interface simples e de fácil entendimento, tendo em vista que um dos principais objetivos é garantir a acessibilidade para todos os públicos.

1. **Definição do Modelo de Processo**

No projeto do aplicativo Dog Walker, usamos alguns modelos de processo diferentes. Na parte inicial do projeto, foi utilizado o modelo incremental, onde produzimos pequenas partes da documentação do aplicativo e fomos entregando e refinando essas partes a cada integração. A utilização do protótipo foi de extrema importância nessa parte, pois foi usado como base para validar os requisitos, para a produção dos casos de uso, e, possibilitou uma visão mais real e interativa do aplicativo. No restante do projeto, por causa da aproximação do prazo de entrega final, optamos por utilizar uma metodologia ágil, onde integramos o método organizacional Kanban ao projeto, para que pudéssemos ser mais eficientes e finalizar dentro do prazo estabelecido.

1. **Requisitos do Sistema de Software**
   1. **Requisitos Funcionais**

|  |  |  |
| --- | --- | --- |
| Identificador | Descrição | Prioridade |
| RF01 | O sistema deve prover uma interface para cadastrar cachorros. | Alta |
| RF02 | O sistema deve possuir uma tela de Perfil do usuário. | Alta |
| RF03 | O sistema deve permitir alterações nas informações pessoais. | Alta |
| RF04 | O sistema deve possuir um mapa que mostrará em tempo real dos Andadores na região. | Alta |
| RF05 | O sistema deve possuir uma tela com o histórico de passeios dos cachorros. | Média |
| RF06 | O sistema deve definir uma rota de ida e volta como trajeto. | Alta |
| RF07 | O sistema deve possuir um *wallet* virtual. | Alta |
| RF08 | O sistema deve possuir avaliação do Andador. | Alta |
| RF09 | O sistema deve ter um chat para que o cliente se comunique com o Andador. | Alta |
| RF10 | O sistema deve incluir uma opção de denunciar o Andador. | Alta |
| RF11 | O sistema deve incluir opções de pagamento com cartão de débito e crédito. | Alta |
| RF12 | O sistema deve calcular o valor do passeio de acordo com o trajeto. | Alta |

* 1. **Requisitos Não-Funcionais**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identificador | Descrição | Categoria | Escopo | Prioridade |
| RNF01 | O sistema deve atender a lei de proteção de uso de dados. | Segurança de dados | Sistema | Alta |
| RNF02 | O sistema só pode ser acessado com um usuário previamente cadastrado. | Segurança de acesso | Sistema | Alta |
| RNF03 | O sistema deve ter uma fácil usabilidade. | Facilidade de Operação | Funcionalidade | Alta |
| RNF04 | O sistema deve possuir boa segurança. | Segurança de dados | Sistema | Alta |
| RNF05 | O sistema deve ser programado em Java. | Desenvolvimento | Sistema | Média |
| RNF06 | O sistema deve possuir Acessibilidade para pessoas deficientes. | Acessibilidade | Funcionalidade | Alta |
| RNF07 | O sistema deve utilizar a API do Google Maps para a geolocalização. | Desenvolvimento | Funcionalidade | Alta |

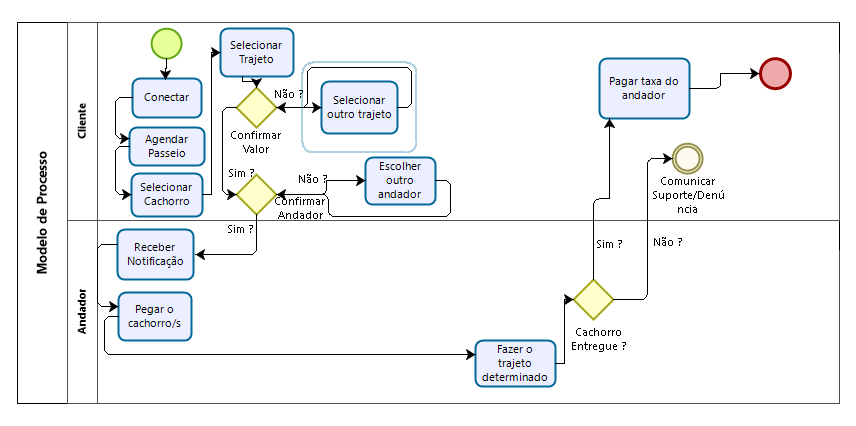
* 1. **Regras de Negócio**

|  |  |  |
| --- | --- | --- |
| Identificador | Descrição | Prioridade |
| RN01 | O limite de cachorros por passeio deverá ser menor ou igual a 5. | Alta |
| RN02 | O sistema deverá criar um trajeto de ida e volta com base nos endereços preenchidos nos campos “Casa” e “Destino” selecionados pelo usuário. | Alta |
| RN03 | O valor de cada passeio será calculado com base no porte do(s) cachorro(s) e na distância total do trajeto selecionado pelo usuário. | Alta |
| RN04 | Os cartões cadastrados deverão ser verificados pelo sistema toda vez que forem selecionados como forma de pagamento. | Alta |
| RN05 | O pagamento será realizado após o término do passeio. | Alta |
| RN06 | O pagamento será realizado caso o cliente solicite o cancelamento do passeio, uma vez que o andador já iniciou o processo. | Alta |
| RN07 | Caso o andador esteja a caminho do ponto de partida para buscar o(s) cachorro(s), e o cliente solicite o cancelamento do passeio, uma taxa será cobrada do cliente. | Alta |
| RN08 | Caso o cliente queira denunciar o andador, ele será direcionado a um site no qual preencherá um formulário que será enviado à equipe de suporte do aplicativo, assim sendo possível tomar as medidas necessárias para solucionar o problema. | Alta |

1. **Projeto**
   1. **Arquitetura Lógica**

Para a criação da Arquitetura Lógica, utilizamos dois diagramas, o Diagrama de Atividade e o Diagrama de Classes.

**Diagrama de Atividade**:



O Diagrama de Atividade é um pequeno gráfico de fluxo com o objetivo de representar o fluxo de atividade entre os diversos atores presentes no software.

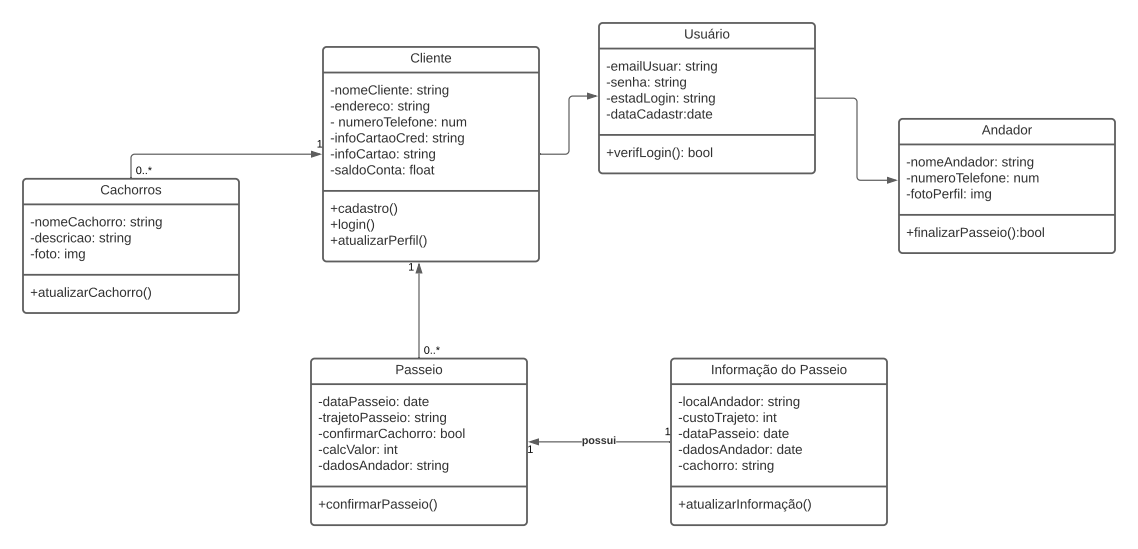
No Diagrama de Atividade apresentado acima, é possível ver o trajeto que o cliente irá realizar, os momentos em que haverá uma escolha, e o seu ponto final.

Após efetuar seu cadastro, o cliente se conectará ao aplicativo e já poderá agendar seu passeio. Após selecionar o cachorro que irá passear, selecionar o destino e confirmar o valor, basta o cliente aguardar a chegada do Walker.

A partir deste momento, outro indivíduo estará presente na ação, o passeador de cachorros, ou como o apelidamos, “Walker”. O Walker receberá uma chamada em seu aplicativo e se dirigirá para o local de busca dos cachorros. Ele terá o dever de realizar o trajeto determinado pelo cliente e entregá-los ao ponto de início. Quando o Walker chegar ao seu destino final, um botão ficará disponível para realizar o término do passeio.

Nesse momento, o cliente pegará de volta seu cachorro, e, por meio do aplicativo, avaliará o trabalho efetuado pelo Walker, podendo também denunciá-lo por alguma ação errônea cometida.

**Diagrama de Classes:**



No Diagrama de Classes é possível observar a representação da estrutura e as entradas necessárias para a formação de uma classe. É possível ver também, como as diferentes classes se relacionam entre si, estabelecendo multiplicidade entre elas. A multiplicidade nada mais é do que a relação que as classes estabelecem (0.1 ou 1..1) , como por exemplo a relação entre o Cachorro e o Cliente, onde o Cliente poderá adicionar um número ilimitado de Cachorros. Entretanto, o Cliente apenas poderá adicionar um limite de cinco cachorros por passeio.

**Interface gráfica do usuário, Aplicativo, Teams

Descrição gerada automaticamente**

1. 1. **Protótipo de Interface (Cliente)**

**Descrição Textual da Imagem:**

Na imagem acima, temos uma amostra do protótipo do aplicativo Dog Walker (Cliente), mostrando a sua principal funcionalidade. O protótipo está relacionando as telas do app com números e setas para melhor entendimento do processo.

1. Ao tocar na opção “Agendar passeio”, aparecerá uma janela com a lista de cachorros cadastrados pelo cliente no app, podendo então selecionar quais cães o andador levará para passear.
2. Após confirmar a seleção do(s) cachorro(s), o cliente será direcionado para próxima tela onde irá escolher o trajeto do passeio.
3. Quando o trajeto for selecionado, o botão “Calcular” irá aparecer, e ao tocá-lo, uma janela com o preço do passeio e as formas de pagamento aparecerá.
4. Após selecionar a forma de pagamento e tocar em “Confirmar”, o aplicativo começará a busca por um andador próximo do local.
5. Assim que o andador for encontrado, uma janela de confirmação com as informações dele irá aparecer, e assim, o cliente terá a opção de confirmar ou cancelar o passeio.
6. Ao tocar em “Confirmar”, o cuidador irá se dirigir até a casa do cliente.
7. Após o andador chegar ao local e pegar o cachorro, o passeio será iniciado.
8. Quando o andador voltar do passeio com o cachorro, o passeio será finalizado. O valor do passeio será cobrado e uma tela de avaliação do andador aparecerá para o cliente.

**Segue link para o protótipo Dog Walker (Cliente) completo:**

[**https://xd.adobe.com/view/ec86afd0-a564-49a0-949a-66e478ebf261-b9d5/**](https://xd.adobe.com/view/ec86afd0-a564-49a0-949a-66e478ebf261-b9d5/)

Interface gráfica do usuário, Diagrama, Aplicativo

Descrição gerada automaticamente

* 1. **Protótipo de Interface (Andador)**

**Descrição Textual da Imagem:**

Na imagem acima, temos uma amostra do protótipo do aplicativo Dog Walker (Cliente), mostrando a sua principal funcionalidade. O protótipo está relacionando as telas do app com números e setas para melhor entendimento do processo.

1. Quando o andador fizer *login* no aplicativo, ele terá um botão na tela principal do aplicativo que permitirá que ele alterne entre disponível ou indisponível para passeios.
2. Assim que um passeio for solicitado, uma tela de confirmação aparecerá para o andador com as informações do cliente.
3. Ao tocar em “Confirmar”, o app solicitará que o andador se dirija até o ponto de

partida.

1. Quando o cuidador chegar ao ponto de partida, o botão “Iniciar” irá aparecer.
2. Ao iniciar o passeio, o andador seguirá o trajeto selecionado pelo cliente.
3. Quando o andador finalizar o trajeto e voltar para o ponto de partida, o botão “Finalizar” irá aparecer, e ao tocá-lo, o passeio será finalizado e o andador receberá o valor do passeio em sua *wallet*.

**Segue link para o protótipo Dog Walker (Andador) completo:**

[**https://xd.adobe.com/view/9aa13ac1-57fb-466d-8481-3961331cd67b-cff2/**](https://xd.adobe.com/view/9aa13ac1-57fb-466d-8481-3961331cd67b-cff2/)

1. **Critérios de Qualidade de Software**

Para a definição da Qualidade de Software utilizamos quatro critérios, funcionalidade, confiabilidade, usabilidade e manutenibilidade.

* **Funcionalidade:** A Funcionalidade define a maneira como as necessidades do software estão sendo atendidas, dentre elas está:
  + Adequação (Propõe-se a fazer o que é apropriado?);
  + Acurácia (Gera resultados corretos ou conforme acordados?);
  + Interoperabilidade (É capaz de interagir com os sistemas especificados?);
  + Segurança de acesso (Evita o acesso não autorizado, acidental ou deliberado a programas e dados?);
  + Conformidade (Está de acordo com normas e convenções previstas em leis e descrições similares?).
* **Confiabilidade:** A Confiabilidade define com que frequência os erros ocorrem e de que maneira o software lida com eles.
  + Maturidade (Com que frequência apresenta falhas?);
  + Recuperabilidade (É capaz de recuperar dados após uma falha?).
* **Usabilidade:** A Usabilidade define se o software é simples de entender e de utilizar.
  + Inteligibilidade (É fácil entender os conceitos utilizados?);
  + Apreensibilidade (É fácil aprender a usar?);
  + Operacionalidade (É fácil de operar e controlar a operação?).
* **Manutenibilidade:** A Manutenibilidade define como o software encara os *bugs* encontrados e de que maneira lidamos com ele.
  + Analisabilidade (É fácil encontrar uma falha quando ocorre?);
  + Modificabilidade (É fácil modificar e remover defeitos?);
  + Estabilidade (Há grandes riscos de bugs quando se faz alterações?);
  + Testabilidade (É fácil testar quando se faz alterações?).

Diante dos critérios definidos, é possível analisar que o software cumpre todos os requisitos. O aplicativo conta com uma ótima funcionalidade, existindo um foco no tema apresentado e contando com uma regra de negócio que impede qualquer tipo de vazamento de informação.

A confiabilidade é um ponto crucial no desenvolvimento de um software. Por esse motivo, contamos com um aplicativo, de certa forma simples, e API’s que facilitam o uso e geram menos problema, diminuindo assim o aparecimento de falhas.

O ponto principal que buscamos neste software é a usabilidade. O software não apresenta telas complexas, muito pelo contrário, nós deixamos o ambiente da interface o mais *clean* possível, para que assim, o aplicativo seja de fácil compreendimento.

A manutenibilidade também não fica de fora. Como foi dito anteriormente, o software conta com API’s que facilitam o uso e geram menos problemas, sendo assim, as raras falhas que ocorrem podem ser revertidas em instantes. Outro ponto que é de extrema importância na manutenibilidade é a rastreabilidade de requisitos, que facilitam a correção de falhas que futuramente ocorreriam.

Matriz de Rastreabilidade (UC x RF)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | RF01 | RF02 | RF03 | RF04 | RF05 | RF06 | RF07 | RF08 | RF09 | RF10 | RF11 | RF12 |
| UC01 | X |  |  |  |  |  |  |  |  |  |  |  |
| UC02 |  |  |  | X |  | X |  |  |  |  |  |  |
| UC03 |  |  |  |  |  |  |  |  |  |  |  | X |
| UC04 |  |  |  |  |  |  | X |  |  |  | X |  |
| UC05 |  |  |  |  |  |  |  | X | X |  |  |  |
| UC06 |  |  |  |  |  |  |  |  | X |  |  |  |
| UC07 |  |  |  |  |  |  |  |  |  | X |  |  |
| UC08 |  |  |  |  |  |  |  | X |  |  |  |  |

Matriz de Rastreabilidade (UC x RNF)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | RNF01 | RNF02 | RNF03 | RNF04 | RNF05 | RNF06 | RNF07 |
| UC01 |  | X | X |  | X | X |  |
| UC02 | X | X | X | X | X | X | X |
| UC03 | X | X | X | X | X | X |  |
| UC04 | X | X | X | X | X | X |  |
| UC05 |  | X | X |  | X | X |  |
| UC06 | X | X | X | X | X | X |  |
| UC07 | X | X | X | X | X | X |  |
| UC08 | X | X | X | X | X | X |  |

Matriz de Rastreabilidade (UC x RN)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | RN01 | RN02 | RN03 | RN04 | RN05 | RN06 | RN07 | RN08 |
| UC01 | X |  |  |  |  |  |  |  |
| UC02 |  | X |  |  |  |  |  |  |
| UC03 |  |  | X |  |  |  |  |  |
| UC04 |  |  |  | X | X | X | X |  |
| UC05 |  |  |  |  |  |  |  |  |
| UC06 |  |  |  |  |  |  |  |  |
| UC07 |  |  |  |  |  |  |  | X |
| UC08 |  |  |  |  |  |  |  |  |

**7. Testes**

**7.1. Plano de Testes**

**Teste de Ciclo de Negócios**

|  |  |
| --- | --- |
| Objetivo da Técnica: | Garantir que o sistema funcione apropriadamente durante um ciclo de atividades relativas ao negócio, e que ao final deste ciclo, garantir que todos os resultados esperados serão obtidos. |
| Técnica: | O teste simulará vários ciclos de negócios executando o seguinte:   * Os testes destinados a inspecionar o funcionamento do objetivo do teste serão modificados ou melhorados para aumentar o número de vezes que cada função é executada, a fim de simular vários usuários diferentes ao longo de um período de tempo especificado. * Todas as funções que mudam com as datas ou o tempo serão executadas usando datas ou períodos de tempo válidos e inválidos. * Todas as funções que ocorrerem segundo uma programação periódica serão executadas ou iniciadas no momento adequado. * O teste incluirá o uso de casos válidos e inválidos para verificar se:   + Os resultados esperados ocorrerão quando forem usados dados válidos.   + As mensagens de erro ou de aviso apropriadas serão exibidas quando forem usados dados inválidos.   + Cada regra de negócio será aplicada de forma adequada. |
| Ferramentas Necessárias: | A técnica exige as seguintes ferramentas:   * Ferramenta de Automação de Scripts de Teste * Restaurador e reprodutor de imagem da configuração básica * Ferramentas de backup e de recuperação * Ferramentas de geração de dados |
| Critérios de Êxito: | A técnica suporta o teste de todos os ciclos de negócios essenciais. |
| Considerações Especiais: | * Os eventos e as datas do sistema poderão exigir atividades de suporte especiais. * É necessário um modelo de negócios para identificar requisitos e procedimentos de testes adequados. |

**Teste de Segurança e de Controle de Acesso**

|  |  |
| --- | --- |
| Objetivo da Técnica: | Segurança no nível do aplicativo: Os atores terão acesso somente às funções e aos dados que foram permitidas pelo seu nível de acesso predefinido.  Segurança no nível do aplicativo: Os atores com acesso ao sistema e ao aplicativo serão os únicos a ter permissão para acessar determinadas funções. |
| Técnica: | Testes serão criados para cada tipo de usuário e serão realizados todos os tipos de ações para verificar se as permissões concedidas estão sendo seguidas.  Modificar o tipo de usuário e executar novamente os testes para os mesmos usuários. Em cada caso, verificar se as funções / dados adicionais estão corretamente disponíveis ou se têm seu acesso negado. |
| Ferramentas Necessárias: | Ferramenta de Automação de Scripts de Teste.  Ferramentas de investigação e contra a violação da segurança  por *hackers*.  Ferramentas de Administração da Segurança do Sistema  Operacional. |
| Critérios de Êxito: | A técnica suporta, para cada tipo de ator conhecido, o teste das funções apropriadas ou dos dados afetados pelas configurações de segurança. |
| Considerações Especiais: | O acesso ao sistema deve ser revisado / discutido com o administrador da rede ou do sistema apropriado. Talvez esse teste não seja necessário, pois pode ser uma função de administração da rede ou do sistema. |

**Plano de Teste para o Protótipo de Arquitetura**

**Versão 1.0**

**Teste de função**

|  |  |
| --- | --- |
| Objetivo da técnica: | Experimentar a funcionalidade do objetivo do teste, incluindo a  navegação, a entrada, o processamento e a recuperação de dados a fim de observar e registrar o comportamento-alvo. |
| Técnica: | * Os resultados esperados ocorrerão quando forem usados dados válidos. * As mensagens de erro ou de aviso apropriadas serão exibidas quando forem usados dados inválidos. * Cada regra de negócio será aplicada de forma adequada. |
| Ferramentas necessárias: | * Ferramenta de Automação de Scripts de Teste. * Restaurador e reprodutor de imagem da configuração básica. * Ferramentas de backup e de recuperação. * Ferramentas de monitoramento de instalação (registro, disco rígido, CPU, memória, etc...). * Ferramentas de geração de dados. |
| Critérios de Êxito: | A técnica suporta o teste de:   * Todos os principais cenários de caso de uso. * Todos os principais recursos. |

**Teste da Interface do Usuário**

|  |  |
| --- | --- |
| Objetivo da Técnica: | Experimentar o seguinte para observar e registrar a conformidade com padrões e o comportamento-alvo:   * A navegação pelo objetivo do teste para verificar se reflete os requisitos e as funções de negócios, incluindo a navegação janela a janela e campo a campo, e o uso de métodos de acesso. * Os objetos e as características das janelas poderão ser experimentados como, por exemplo, menus, tamanho, posição, estado e foco. |
| Técnica: | Crie ou modifique testes para cada janela a fim de verificar a navegação adequada e os estados de objeto apropriados para cada janela e objeto do aplicativo. |
| Ferramentas Necessárias: | A técnica necessita da Ferramenta de Automação de Scripts de Teste. |
| Critérios de Êxito: | A técnica suporta o teste de cada tela ou janela principal que será muito usada pelo usuário final. |
| Considerações Especiais: | Nem todas as propriedades referentes a objetos personalizados e de terceiros poderão ser acessadas. |

**7.2. Roteiro de Testes**

Os testes do aplicativo passarão por três etapas. A primeira etapa será um teste de caixa branca realizado pelos próprios desenvolvedores do aplicativo. Este teste consiste basicamente em desenvolvedores de uma parte específica do aplicativo trocar de lugar com outro desenvolvedor, e assim realizar o teste. Desta forma, os desenvolvedores não se atêm às próprias linhas de código, cessando assim o problema de avaliar seu próprio código.

A segunda etapa será realizada por uma equipe contratada de *testers* profissionais com o objetivo de realizar tanto testes de caixa branca, quanto testes de caixa preta, e outros que haja uma possível necessidade. O fato de essa equipe não participar diretamente do desenvolvimento do app aumenta o êxito em encontrar possíveis falhas, averiguar se o app está funcional e verificar se o software atende as necessidades do cliente.

A terceira e última etapa será realizada por um usuário final, de preferência que não tenha tido uma participação no desenvolvimento do app. O teste que realizado será o teste de caixa preta, onde será avaliada a interface do app e suas funcionalidades, verificar se as mesmas atendem as necessidades do cliente e verificar se há possíveis falhas no app.