**HELP DELIVERY**

**Solicitações e Entregas**

**Cleiton Eduardo Ramos**

**Ruan Felipe Sasse**

**Professor**

**Jocemar José Freire**

Centro Universitário Leonardo da Vinci – UNIASSELVI

### Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

### (TSI0066) – Seminários em Análise e Desenvolvimento de Sistemas

04/04/2020

Resumo

Será entregue ao Termino do Artigo

Introdução

Na atualidade, o tempo disponível das pessoas está cada vez mais escasso, e com a tecnologia podemos buscar formas para poupar o tempo das pessoas. Baseado neste contexto, voltamos nossa visão para a mobilidade urbana, onde cada ano aumenta o número de veículos em circulação, gerando problemas de mobilidade urbana.

Verificamos que existe uma demanda nos transporte de produtos de pequeno e médio porte, Quando necessário levar uma pequena encomenda até determinado local as pessoas muitas precisam se deslocarem, em um trajeto de ida e volta, verificamos o desperdício de tempo, recursos e aumento na circulação de veículos gerados por esta demanda de entrega dessas mercadorias.

Como uma solução para este tipo de situação, iremos desenvolver um software que irá aproximar as pessoas comuns que poderão solicitar uma corrida para levar uma mercadoria (Solicitantes) e as pessoas que se disponibilizam a realizar a coleta e entrega por um preço justo (Entregadores), assim fomentando uma fonte de renda extra para as pessoas gerando oportunidades, e também agilizando a comunicação entre as partes interessadas.

Primeiramente iremos compreender as necessidades que um sistema deste precisa e com este intuito será realizado a análise dos requisitos, em consequência desta análise, e para uma melhor compreensão do software que será desenvolvido, criaremos diagramas de UML. Os dados e informações que serão gerados ficaram armazenados em um banco de dados e por este motivo realizaremos a modelagem do banco de dados do sistema,

Desenvolveremos a aplicação web, que possibilitara que pessoas contratem o transporte de algum produto mercadoria até um endereço especifico, e pessoas que estejam dispostas a uma renda extra poderão se disponibilizar a realizar estas coletas e entregas.

Desenvolviment**o**

O primeiro passo no processo de análise de sistemas é a identificação de necessidades (PRESSMAN, 1995 apud Djone Kochanski, 2013, p. 41). De acordo com Pfleeger (2004, p. 111), “um requisito é uma característica do sistema ou a descrição de algo que o sistema é capaz de realizar, para atingir os seus objetivos”. Estes requisitos podem ser classificados conforme citação abaixo.

Em relação às categorias de requisitos, podemos classificá-los como funcionais ou não funcionais. Os requisitos funcionais são as declarações de funções que o software deverá oferecer, como entradas específicas deverão ser tratadas e qual o comportamento esperado em determinadas situações. Já os requisitos não funcionais tipicamente representam restrições sobre os serviços e/ ou funções oferecidas pelo software. (SOMMERVILLE, 2003 apud Djone Kochanski, 2013, p. 47).

Através da análise de requisitos pode ser desenvolvido a modelagem do software que é uma forma de documentar e o que será desenvolvido (PRESSMAN, 2011, p. 123) afirma:

Modelagem abrange tanto análise quanto projeto, descrevendo representações do software que se tornam progressivamente mais detalhadas. O objetivo dos modelos é solidificar a compreensão do trabalho a ser feito e providenciar orientação técnica aos implementadores do software. Os princípios de modelagem servem como infraestrutura para os métodos e para a notação utilizada para criar representações do software.

A modelagem pode ser representada com os diagramas da UML, segundo o autor Larman (2007, p.39) UML é: “[...] uma linguagem visual para especificar, construir e documentar os artefatos dos sistemas [OMG03a]. A palavra visual na definição é um ponto chave – a UML é a notação diagramática padrão, de fato, para desenhar ou apresentar figuras (com algum texto) relacionadas a software – principalmente software orientação a objeto, existe vários tipos de diagramas da UML cada um com seu propósito de representar de forma detalhada aspectos do software.

O diagrama de caso de uso mostra as principais funcionalidades do sistema, ou seja, o que o sistema faz, de um ponto de vista do usuário. Segundo Larman (2007, p.116): “Um diagrama de caso de uso é uma excelente imagem do contexto do sistema; ele é um bom diagrama de contexto, ou seja, mostra a fronteira de um sistema, o que está fora dele e como o sistema é usado. Serve como uma ferramenta de comunicação que resume o comportamento do sistema e seus autores[...]”. Alguns diagramas podem se complementar para elencar vários pontos de vista. “Um diagrama de atividades UML complementa o caso de uso através de uma representação gráfica do fluxo de interação em um cenário específico. Similar ao fluxograma, um diagrama de atividades usa retângulos com cantos arredondados para representar determinada função do sistema, setas para representar o fluxo através do sistema [...]” (PRESSMAN, 2011, p. 161).

Diagrama de classes mostra um conjunto de classes e a relação que há entre elas, o diagrama de classes possui entidades, relacionamentos e exemplo contendo as anotações. Para o autor Larman (2007, p.266): “A UML inclui diagramas de classe para ilustrar classes, interfaces e suas associações. Eles são usados para modelagem estática de objeto. Já introduzimos e usamos esse diagrama UML ao fazermos modelagem de domínio, aplicando diagramas de classe em uma perspectiva conceitual”. Através deste diagrama fica evidentes as classes e entidades estas que de uma forma geral poderão ser transformadas e tabelas para armazenamento de dados.

Segundo o autor PRESSMAN se entre os requisitos do software for necessário a criação ou a extensão ou interfacear um banco de dados, ou se as estruturas tiverem que ser construídas manipuladas, pode se optar pela criação de um modelo de dados como parte dos requisitos.

Um analista ou engenheiro de software define todos os objetos de dados processados no sistema, os relacionamentos entre os objetos de dados e outras informações pertinentes aos relacionamentos. O *diagrama entidade-relacionamento* (*entity-relationship* *diagram*) trata das questões e representa todos os objetos de dados introduzidos, armazenados, transformados e produzidos em uma aplicação.(Pressman, 2011, p. 163)

Sobre a forma como estes dados se relacionam afirma Ferrari: “O modelo relacional para gerência de bancos de dados (SGBD) é um modelo de dados baseado em lógica de predicados e na teoria de conjuntos.” (Ferrari, 2007, p7). Sendo assim o modelo de banco de dados relacional utiliza como linguagem padrão o SQL do inglês *Structured Query Language*, é apenas vagamente remanescente do modelo matemático (Ferrari, 2007, p7).E para o gerenciamento organizações destes dados o responsável é o SGBD,

Um Sistema Gerenciador de Bancos de Dados (SGBD) é constituído por um conjunto de dados associados a um conjunto de programas para acesso a esses dados. O conjunto de dados, comumente chamado banco de dados, contém informações sobre uma empresa em particular. O principal objetivo de um SGBD é proporcionar um ambiente tanto conveniente quanto eficiente para a recuperação e armazenamento das informações do banco de dados.(Silberschatz, 2004, p1.)

Sobre a forma como estes dados se relacionam afirma Ferrari: “O modelo relacional para gerência de bancos de dados (SGBD) é um modelo de dados baseado em lógica de predicados e na teoria de conjuntos.” (Ferrari, 2007, p7). Sendo assim o modelo de banco de dados relacional utiliza como linguagem padrão o SQL do inglês *Structured Query Language*, é apenas vagamente remanescente do modelo matemático (Ferrari, 2007, p7). “SQL é essencialmente uma linguagem não procedural ou não algorítmica, ou seja, a sintaxe da linguagem específica que resultados se deseja obter e não uma ‘sequência de instruções.” GUIMARÃES (2003, p.100/101).

Como a linguagem SQL não é algo simples, os usuários comuns precisam de algo que facilite a manipulação das informações no banco de dados, é necessário um programa que possibilite realização destes comandos no banco de dados, afim de mostrar de uma forma visual para o usuário os resultados desta manipulação (Casati, 2016, p 14). Baseado neste pressuposto:

“Sistemas web é comum se deparar com páginas que possuem formulários. Estes formulários fazem parte da interação entre o cliente e o servidor, fazendo assim com que os usuários possam enviar informações que são processadas pelo servidor. Estes formulários estão do lado cliente, enquanto o processamento das informações advindas destes é feito no lado servidor” (Casati, 2016, p 14).

Uma das linguagem de programação que possibilita construção de página web é o Java Script sobre o Java Script (HAVERBEKE, 2018) cita.

O JavaScript foi introduzido em 1995, como uma forma de adicionar programas a páginas da web no navegador Netscape. A linguagem foi adaptada pela maioria dos navegadores gráficos da web. Ele fez a atual geração de aplicações web possível - clientes de email baseado no navegador, mapas e redes sociais - e também é usado em sites mais tradicionais para fornecer várias formas de interatividade e inteligência. (HAVERBEKE, 2018).

O JavaScript também possui vários módulos e bibliotecas que auxiliam com diversas funcionalidades o desenvolvimento de um sistema web, sobre o uso da biblioteca NPM do JavaScript (HAVERBEKE, 2018) cita:

Existem muito mais coisas no NPM além de npm install. Ele pode ler arquivos package,json, que contém informações codificadas em JSON sobre o programa ou biblioteca, como por exemplo outras bibliotecas que depende. Rodar npm install em um diretório que contém um arquivo como esse vai instalar automaticamente todas as dependencias, assim como as dependencias das dependencias. A ferramenta npm também é usada para publicar bibliotecas para o repositório NPM online de pacotes para que as pessoas possam encontrar, transferir e usá-los (HAVERBEKE, 2018).

Sobre a biblioteca Node.js (HAVERBEKE, 2018) afirma: “[...]Node.js, um programa que permite que você aplique suas habilidades de JavaScript fora do navegador. Com isso, você pode construir desde uma ferramenta de linha de comando até servidores HTTP dinâmicos”.

Metodologia

Para realizar a análise de requisitos do software, será esclarecido quais são os objetivos da construção deste softwares o que se espera de resultado do mesmo e como seria sua usabilidade, pode ser dado o início do processo de levantamento de requisitos com a identificação de interessados e suas sugestões. “No início, devemos criar uma lista das pessoas que irão contribuir com sugestões à medida que os requisitos são obtidos” (PRESSMAN, 2011, p 131). A lista inicial crescerá à medida que os interessados forem contatados, pois para cada um deles será feita a pergunta: “Com quem mais você acha que eu deva falar?” (PRESSMAN, 2011, p 131).

Com o levantamento dos interessados então vários pontos de vistas serão elencados para esclarecer as funcionalidades e requisitos e que o software deve a presentar conforme cita. “Pelo fato de existirem muitos interessados diferentes, os requisitos do sistema serão explorados Sob vários pontos de vista” (PRESSMAN, 2011, p 131).

Para uma melhor definição destes requisitos será feito uma modelagem do software utilizando os diagramas da UML, “os arquitetos de software criam diagramas UML para ajudar os desenvolvedores de software a construir o software” (PRESSMAN, 2011, p 727).

O primeiro diagrama de UML abordado será o de caso de uso conforme proposto por PRESSMAN, 2011, p 732), pois representam bem os aspectos de uso do software e também como os usuários irão interagir com o sistema definindo os passos necessários para atingir um objetivo especifico. Método para o desenvolvimento deste caso de uso os usuários serão representados por atores, estes serão conectados aos casos de uso que são representados esferas ovais estes ficam dentro de um retângulo, e os atores se encontram fora do retângulo em alguns casos de uso podem estar relacionados uns com os outros nestes casos é usada a palavra include para representá-lo desenvolvido com a ferramenta diagrams.net.

Conforme o autor PRESSMAN o diagrama de classes consiste em uma representação gráfica que auxilia o entendimento dos atributos operações, relações e associações com outras classes, e também fornece uma visão estrutural de um sistema. As classes são representadas por caixas que possuem 3 subdivisões a primeira contém o nome da classe a segunda à lista de atributos e a terceira contém as operações ou comportamentos, os atributos pode ter um nome, um tipo e um nível de visibilidade, os diagramas de classe também podem exibir relações entre classes, associação, dependência, agregação a ferramenta utilizada diagrams.net.

Outro diagrama abordado e o diagrama de atividades também proposto por PRESSMAN, que informa que de certa forma é muito parecido com um fluxograma com algumas características que os distinguem, e mostra o comportamento dinâmico de um sistema ou parte de um sistema através do fluxo de controle entre ações que o sistema executa. Segundo PRESSMAN este diagrama possui o nó de ação representado por um retângulo arredondado que corresponde a uma tarefa executada, as setas representam o fluxo de controle o início representada por um ponto preto solido e o final por um ponto preto envolvido por um círculo preto, também possui fork que serve para separar duas ou mais atividades e o join que serve para juntar atividades concorrentes, nó de decisão representado por um triangulo, e faixas que representam os participantes, ferramenta utilizada diagrams.net.

Metodologia sobre entidade e relacionamento,

**RESULTADOS E DISCUSSÃO**

Para realizar o levantamento dos requisitos primeiramente buscamos conversar com várias pessoas para ouvir sobre opiniões diferentes sobre o tema, professore, colegas, colegas de profissão. Também votamos nossa visão para o mercado e procuramos nos aplicativos da google play aplicativos similares para extrair ideias avaliar se algum destes já realizam a mesma funcionalidade. E como resultado desta analise chegamos aos seguintes requisitos funcionais:

**RF1** - Realizar cadastro de usuários. **Descrição:** Como uma forma de segurança o usuário deverá se cadastrar, para poder usufruir das funcionalidades do software. Para o primeiro acesso será necessário informar um e-mail válido, também com suas informações pessoais.

**RF2** - Cadastro de solicitante e entregadores. **Descrição:** Após o usuário já estar registrado no sistema ele deverá escolher se será um entregador ou um solicitante, porém o acesso será permitido após cadastramento de todas as informações pessoais solicitadas especificamente para cada um deles.

**RF3 -** Cliente poderá cadastrar produtos/encomendas. **Descrição:** Para poder solicitar uma corrida uma condição será informar o que será transportado neste cadastro deverá conter informações sobre o produto como peso dimensões e também fotos. Dessa forma o ficará mais transparente possível para o entregador ter ciência do produto a ser transportado.

**RF4 -** Entregador poderá cadastrar seus meios de transporte. **Descrição:** Será cadastrado no sistema informações sobre o veículo o meio de transporte utilizado, informações do documento e fotos, uma forma para esclarecer para o solicitante qual veículo ira vir buscar suas encomendas

**RF5** - Solicitante poderá contratar uma corrida. **Descrição:** Para contratar uma corrida o solicitante passará pelas consistências do seu perfil. Então deverá escolher a carga a ser enviada já previamente cadastrado, deverá informar qual endereço de entrega horário para recebimento data e hora limite para entrega e forma de pagamento.

**RF6 -** Sistema irá gerar o valor da corrida. **Descrição:** Baseado no tamanho da encomenda e da distância de entrega o software irá gerar o valor da corrida, evitando negociação entre os usuários.

**RF7** - Entregador escolherá solicitações de corrida de seu interesse. **Descrição:** Entregador terá acesso a visualizar as corridas solicitadas que ainda estão em aberto, terá opção de visualizar as informações do produto as informações do solicitante como avaliações feitas por outros usuários, o endereço da entrega, data e horários do recebimento como o valor e forma de pagamento.

**RF8** - Solicitante pode recusar entregador pretendente. **Descrição:** O entregador terá a possibilidade de visualizar o entregador que aceitou sua corrida primeiro, ele poderá então avaliar as informações cadastrais do entregador como informações pessoais veículo que será usado para o transporte para então recusar ou aprovar a corrida gerando um pedido.

**RF9** - Informar término da corrida e avaliação. **Descrição:** Após entrega do produto, tanto usuário quanto entregador deverão informar que a corrida foi concluída, obrigatoriamente será necessário avaliar a entrega, tanto o solicitante quando o entregador ambos serão avaliados.

**RF10** - Retirada do produto. **Descrição:** Quando o entregador for realizar a retirada do produto do local de partida ele deverá verificar se o produto está conforme combinado na descrição da corrida se estiver conforme o que foi contratado pelo aplicativo, ele valida que retirou o produto e que está sob sua posse. Caso contrário ele poderá cancelar a corrida e o solicitante será analisado tendo seu perfil bloqueado para novas solicitações.

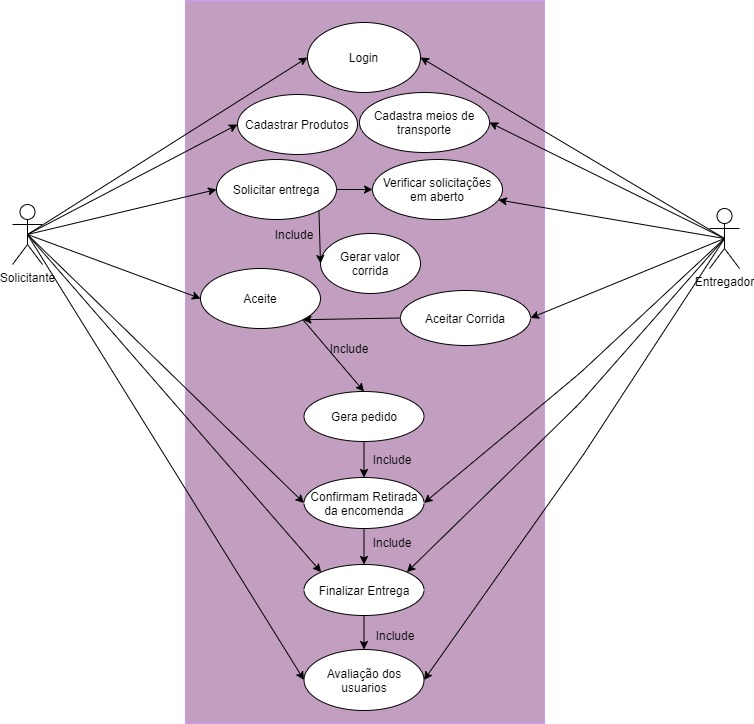
Como requisito não funcional foram levantados as seguintes premissas:

**RNF01 -** Infraestrutura - Será necessário acesso à internet para utilização do software.

**RNF02 -** Funcionamento - Deverá possuir navegador compatível com o software.

A figura abaixo representa um dos diagramas da UML o caso de uso conforme metodologia proposta neste trabalho

FIGURA 1 – DIAGRAMA DE CASOS DE USO



FONTE: Elaborado pelos autores © 2005-2020 diagrams.net. All rights reserved.

Referente a figura 1 apresentamos a descrição do caso de uso nele representado.

Atores envolvidos: **1. Solicitante 2. Entregador**

Pré-condições: 1. Ambos deve estar com os perfis completos conforme respectiva função que irá exercer no sistema

Pós-condições:

1. Solicitada uma corrida.

2. Entregador aceita a corrida.

3. Entregador entrega a mercadoria.

4. Avaliação dos Usuários.

Cenário principal:

1. Solicitante cadastrar um item para enviar.

2. Solicitante requisita uma corrida para determinado endereço para determinado produto cadastrado.

3. Sistema calcula o valor da corrida automaticamente para o solicitante saber quanto irá custar.

4. Entregador cadastrar os veículos que o mesmo dispõe para realizar o transporte.

5. Verifica as solicitações de corrida em aberto e escolhe a de seu interesse.

6. Entregador aceita a corrida.

7. Solicitante Avalia se aceita o entregador pretendente.

8. Conforme aceite do solicitante pedido é gerado.

9. Quando Entregador retira o produto ele deve confirmar que o produto está conforme foi especificado no pedido. Solicitante e entregador confirma no sistema que produto foi levado. Gerando a confirmação que a encomenda está sob posse do entregador.

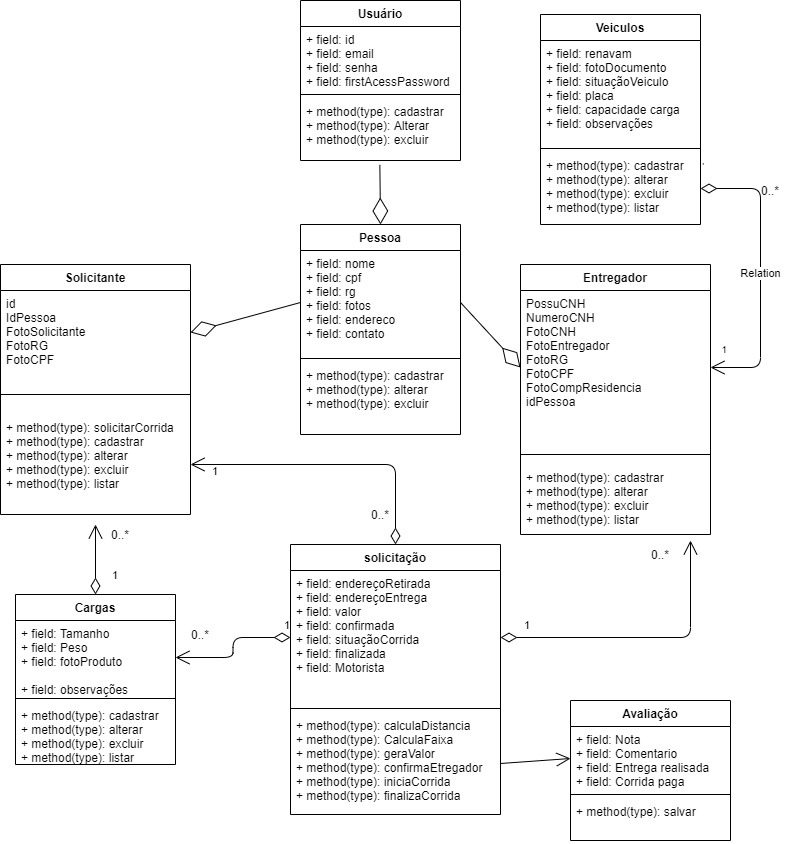
10. Ao realizar a entrega do produto ambos finalizam a corrida no sistema.

11. Ao termino da corrida o sistema exige a avaliação por parte de ambos os atores.

O próximo diagrama desenvolvido é o diagrama de classe que irá presentar as classes do sistema seus métodos e atributos e os relacionamentos entre elas. Segue uma listagem contendo uma breve descrição das classes, atributos e métodos deste diagrama:

* Usuário – Representa a conta de acesso ao sistema seus atributos são id que representa o código do usuário internamente para o sistema, email, usuário e senha de acesso e primeiro acesso utilizado para informar se é a primeira vez que o usuário está acessando o sistema. Métodos desta classe cadastrar para criar um novo acesso, alterar para realizar alterações nos dados de acesso e exclusão.
* Pessoa – seus atributos representam os dados pessoas necessários para serem cadastrados para o usuário usufruir completamente do sistema, seus métodos são para adicionar alterar ou excluir.
* Solicitante – Que é uma agregação da classe pessoa representa o usuário que pode realizar contratações de entregas seus atributos são documentos específicos para essa funcionalidade no sistema e seus métodos server para manipular as solicitações de corridas e lista-las.
* Cargas – São os produtos que o solicitante pode cadastrar para solicitar entregas para os mesmos, atributos desta classe representam o produto físico para dentro do sistema.
* Entregador – Classe que representa a pessoa responsável por retirar a carga e entrega-la em seu destino seus atributos representam todas as documentações necessária para realizar essa atividade, que poderão ser manipulados pelos métodos da classe.
* Veículos – Esta classe representa os meios de transporte que o entregador utilisa ele os atributos representam os documentos do veículos e seus características.
* Solicitação – Representa a requisição de uma corrida onde o solicitante informa quais produto serão transportados endereço de retirada e destino o status da corrida o entregador foi escolhido para realizar a corrida, esta classe possui o método para calcular a distância gerar uma faixa com valor para corrida, confirmação do entregador confirmação da entrega e finalização da corrida.
* Avaliação – Após termino da corrida ambos solicitante e entregador realizaram a avaliação um do outro para servir como parâmetro de confiabilidade do usuário perante a utilização do sistema.

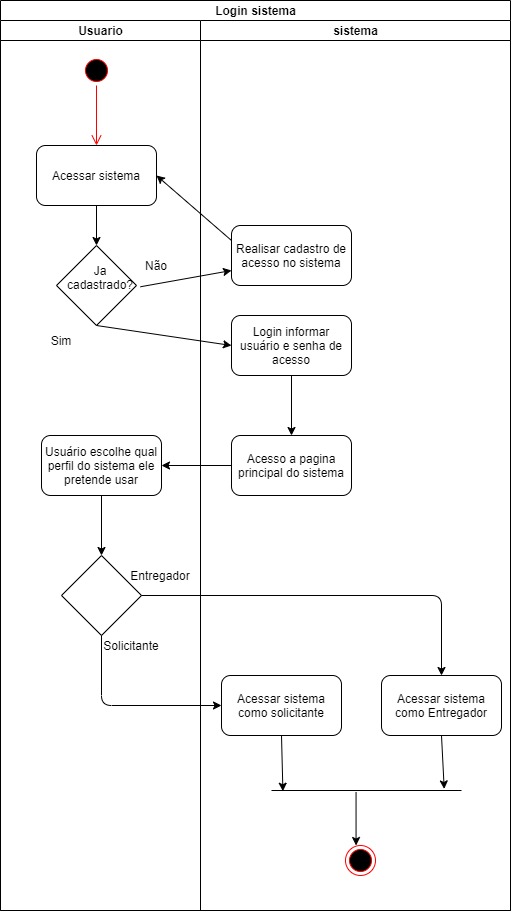
FIGURA 2 – DIAGRAMA DE CLASSES



FONTE: Elaborado pelos autores © 2005-2020 diagrams.net. All rights reserved.

Diagrama de atividades desenvolvido conforme metodologia proposta será demostrado em partes devido ao seu tamanho, desta forma o entendimento da representação gráfica ficara mais clara e detalhada.

A figura 3 representa o acesso ao sistema e a escolha de qual perfil o usuário irá utilizar, conforme as funcionalidades especificas de cada perfil.

FIGURA 3 – DIAGRAMA DE ATIVIDADESFONTE: Elaborado pelos autores © 2005-2020 diagrams.net. All rights reserved.

A figura 4 representa a abertura de uma nova corrida pelo solicitante. Após ingressar o sistema como um solicitante, ele poderá requisitar uma nova corrida, será necessário informar a carga a ser transportada cadastrando um novo produto, ou escolhendo algum produto que já tenha sido previamente cadastrado, informar também o local de retirada e o local de entrega com essa informação o sistema ira informar o valor da corrida, se o solicitante estiver de acordo ele confirma e a corrida ficaram com status em aberto para que entregadores se ofereçam para realizá-la.

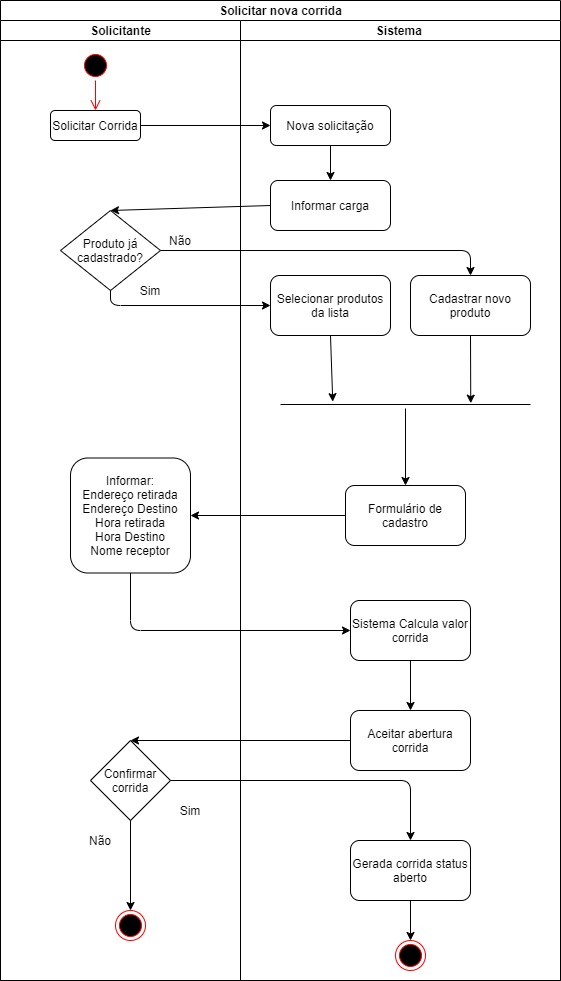
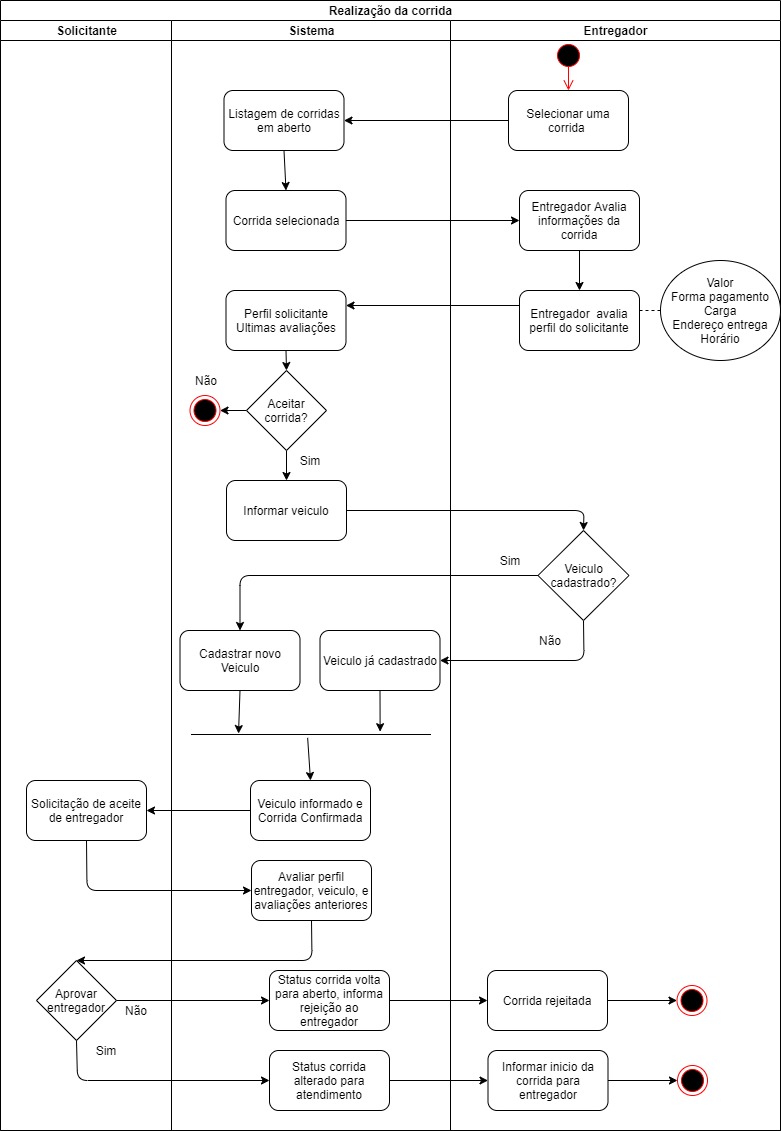
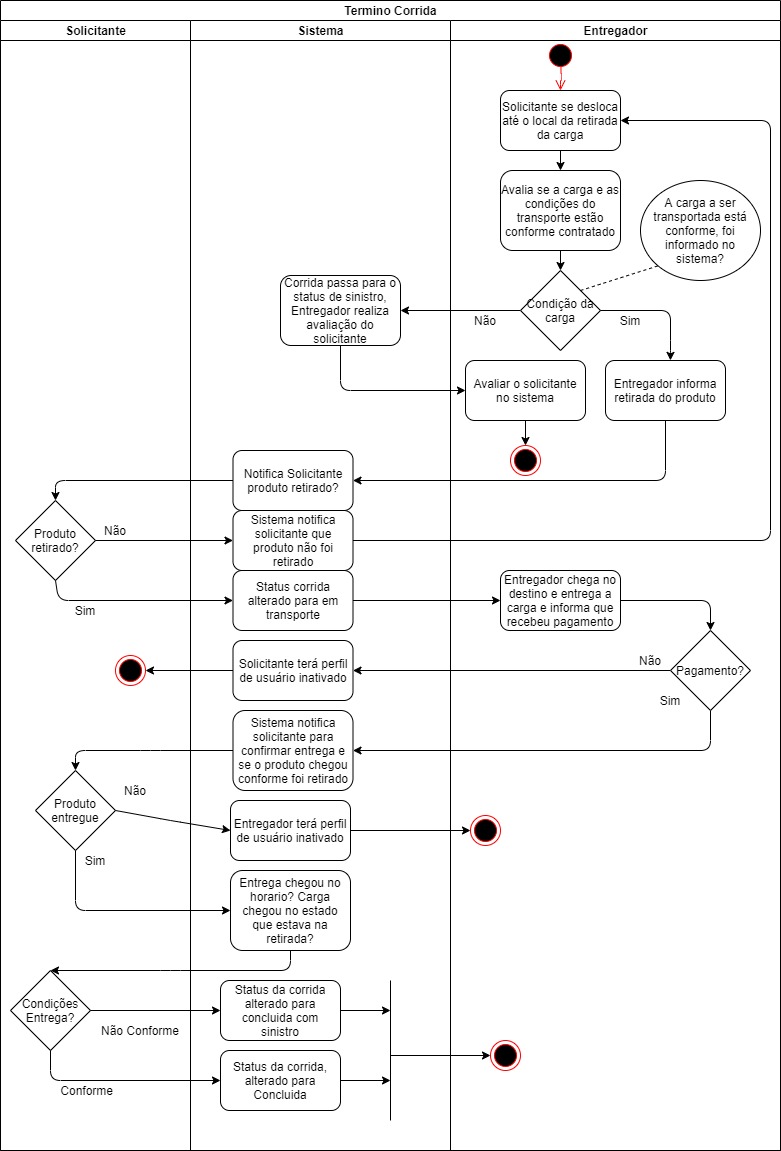
FIGURA 4 – DIAGRAMA DE ATIVIDADESFONTE: Elaborado pelos autores © 2005-2020 diagrams.net. All rights reserved.

FIGURA 5 – DIAGRAMA DE ATIVIDAD****

FONTE: Elaborado pelos autores © 2005-2020 diagrams.net. All rights reserved

.

A figura 5 acima representada pelo diagrama de atividades demostra a início da corrida, o entregador verifica no sistema quais são as corridas com status em aberto, ele verifica as informações que foram preenchidas previamente pelo solicitante e o perfil incluindo avaliações de usuário anteriores, se for de seu interesse realizá-la ele informa com qual veículo pretende realizar o transporte, o sistema neste momento encaminhara uma aprovação para solicitante se ele aceita ou não o entregador pretendente, caso aceite a corrida passa para o status de atendimento.

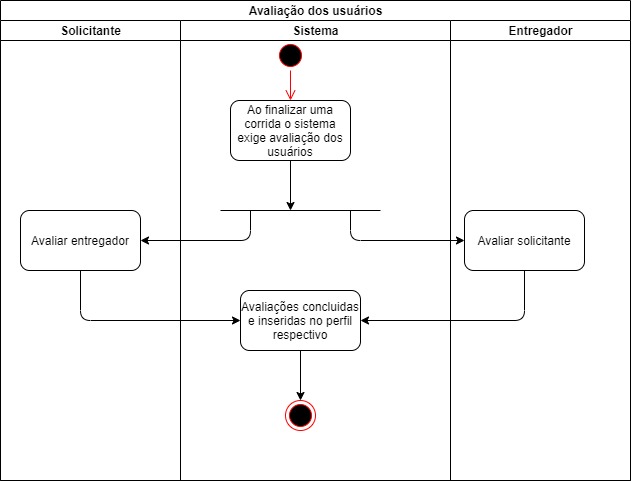
FIGURA 6 – DIAGRAMA DE ATIVIDADES FONTE: Elaborado pelos autores © 2005-2020 diagrams.net. All rights reserved

O diagrama de atividade representado na figura 6, demostra o fluxo necessário para se finalizar uma corrida, primeiramente a corrida já deve estar em atendimento pelo entregador então o mesmo se desloca até o local para retirar o produto a ser transportado que iremos tratar como “carga”, se estiver conforme as imagens a e as condições informadas no sistema, o intregador apontara a carga como retirada caso contrário a corrida será cancelada e seus status será alterado para “sinistro” informando houve uma má conduta por parte do solicitante.

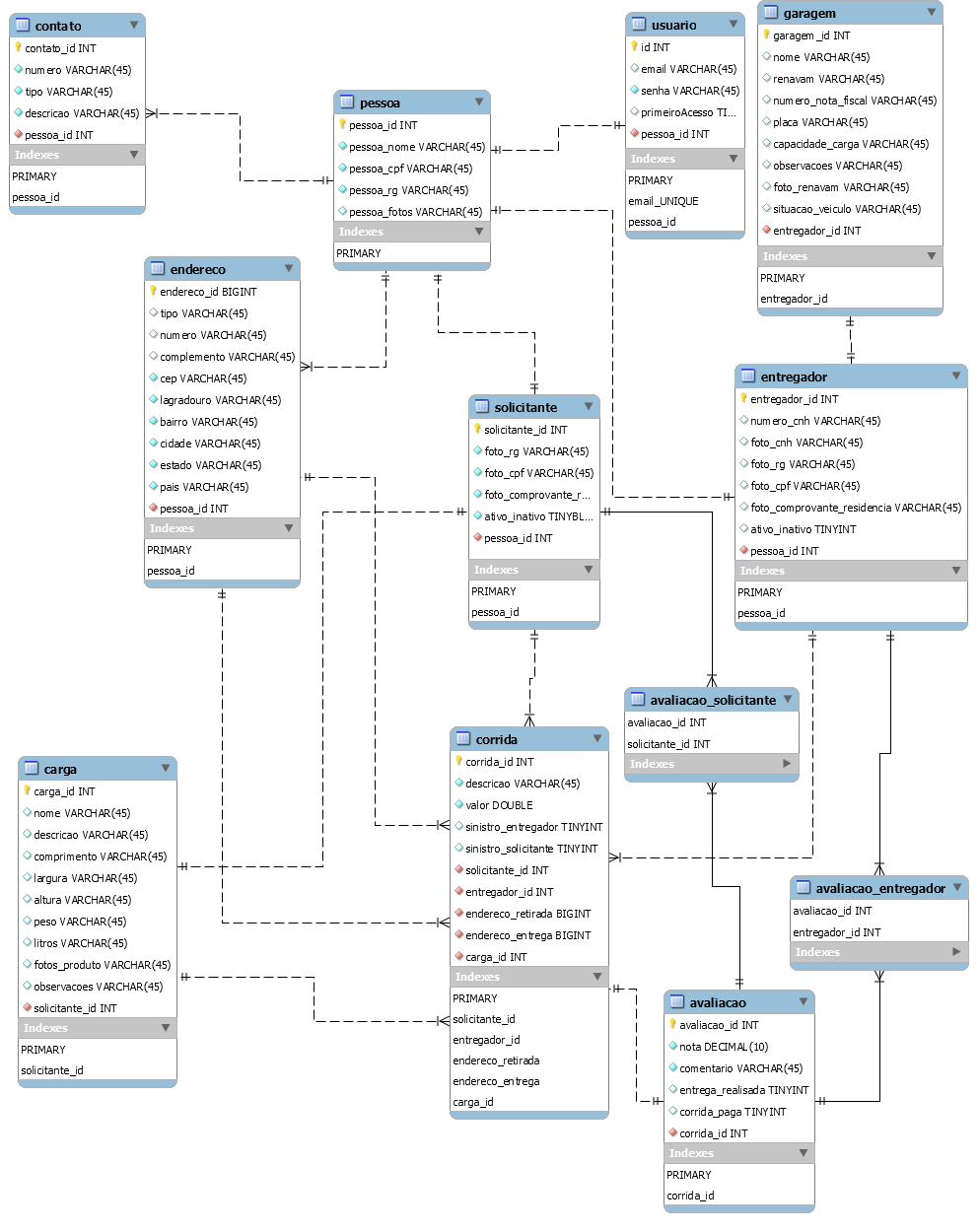
Ao chegar no local de destino com a mercadoria a entregador deverá informar que realizou a entrega e recebeu o pagamento, o não pagamento da corrida implica na inativação do usuário para que o mesmo não possa mais solicitar corridas. Fica a cargo então do solicitante informar se a entrega foi realizada, caso o produto não for entregue o entregador terá seu perfil inativado. Se a carga foi entregue o solicitante deverá informar o estado que se encontra a carga e se não houve atrasos, esses casos geram uma entrega com sinistro que ficara atrelada ao histórico do entregador, após esta etapa a corrida ficara com status Concluída ou concluída com restrição.

No diagrama informado na figura 7 após a conclusão da corrida será necessário o entregador avaliar o solicitante e o solicitante avaliar o entregador, está avaliação ficara atrelada ao perfil do usuário e servira como referência para outros usuários quando forem avaliar seus perfis.

FIGURA 7 – DIAGRAMA DE ATIVIDADES

FONTE: Elaborado pelos autores © 2005-2020 diagrams.net. All rights reserved

Com a conclusão destes diagramas da UML, é possível ter uma visão das classes de seus relacionamentos, a interação que será realizada pelos usuários e como o software devera se comportar, faltando representar a forma que os dados serão armazenados no sistema.

FIGURA 8 – MODELO DE ENTIDADE RELACIONAMENTOFONTE: Elaborado pelos autores, MySQL Workbench © 2020, Oracle Corporation and/or its affiliates

**Referências**

PRESSMAN, Roger S. Engenharia de software. Rio de Janeiro: Pearson Makron Books, 1995.

PFLEEGER, Shari Lawrence. Engenharia de software: teoria e prática. 2 ed. SãoPaulo: Prentice Hall, 2004.

Pressman, Roger S.Engenharia de software [recurso eletrônico] : uma abordagem profi ssional / Roger S. Pressman ; tradução Ariovaldo Griesi ; revisão técnica Reginaldo Arakaki, Julio Arakaki, Renato Manzan de Andrade. – 7. ed. – Dados eletrônicos. – Porto Alegre : AMGH, 2011.

Kochanski, Djone Engenharia de software / Djone Kochanski. Indaial : Uniasselvi, 2013.

HAVERBEKE, Marijn. **Eloquent JavaScript**. 2ª ed. San Francisco: No Starch Press, 2014. Disponível em: <eloquentjavascript.net>. Acesso em: 20 abr. 2018.

DELAMARO, Marcio. Introdução ao Teste de Software. Rio de Janeiro: Elsevier, 2007.

SOARES, Wallace. PHP 5: conceitos, programação e integração com banco de dados. São Paulo: Editora Érica Ltda, 2004.

Ferrari, Fabrício Augusto. Crie banco de dados em MySQL / Fabrício Augusto Ferrari. – São Paulo: Digerati Books, 2007. 128 p. ISBN 978-85-60480-25-8

Casati, João Paulo Programação cliente em sistemas WEB / João Paulo Casati. Rio de Janeiro: SESES, 2016. 120 p: il. isbn: 978-85-5548-333-2

SILVA, Maurício Samy. Construindo sites com CSS e (X)HTLM. São Paulo: Novatec Editora Ltda, 2008.

DELAMARO, Márcio. Introdução ao teste de software. Rio de Janeiro: Elsevier Editora Ltda, 2007.

AREZI, Fábio. Casos de uso. Disponível em: <https://arezi.wordpress.com/2010/10/20/casos-de-uso-diferencas-entre-include-extend-e-generalizacao/>. Acesso em: 02/10/2015.

GUEDES, Marylene. SQL vs NoSQL, qual usar?. Disponível em: <https://www.treinaweb.com.br/blog/sql-vs-nosql-qual-usar//>. Acesso em: 10/03/2020.

INFOQ. Analisando gráficos de burndown.Disponível em: <http://www.infoq.com/br/news/2010/02/deciphering-burndown-charts>. Acesso em: 16/09/2015.

LINHA DE CÓDIGO. Diagramas UML.Disponível em: <http://www.linhadecodigo.com.br/artigo/2151/primeiras-imagens-do-visual-studio-team-system-2010.aspx>. Acesso em: 12/09/2015.

JAVA FREE. Diagramas de classes e objetos.Disponível em: <http://javafree.uol.com.br/artigo/876366/url>. Acesso em: 12/09/2015.