UNIVERSIDADE PAULISTA

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CARLOS EDUARDO DE OLIVEIRA

MARCUS VINICIUS NUNES

PAULO SERGIO BRESSAN JUNIOR

RAFAEL ANTONIO LOPES ALVES

SIDNEY DE SOUZA JUNIOR

**PROJETO INTEGRADO MULTIDISCIPLINAR**

PROJETO E DESENVOLVIMENTO DE UM SISTEMA PARA GERENCIAMENTO DE UMA LOJA DE ARTIGOS ESPORTIVOS

RIBEIRÃO PRETO

2018

CARLOS EDUARDO DE OLIVEIRA

MARCUS VINICIUS NUNES

PAULO SERGIO BRESSAN JUNIOR

RAFAEL ANTONIO LOPES ALVES

SIDNEY DE SOUZA JUNIOR

**PROJETO INTEGRADO MULTIDISCIPLINAR**

Projeto e desenvolvimento de um sistema para gerenciamento de uma loja de artigos esportivos.

Trabalho de Integração Multidisciplinar para a conclusão do terceiro semestre do curso de Análise e Desenvolvimento de Sistemas apresentado à Universidade Paulista.

RIBEIRÃO PRETO

2018

CARLOS EDUARDO DE OLIVEIRA

MARCUS VINICIUS NUNES

PAULO SERGIO BRESSAN JUNIOR

RAFAEL ANTONIO LOPES ALVES

SIDNEY DE SOUZA JUNIOR

**PROJETO INTEGRADO MULTIDISCIPLINAR**

Projeto e desenvolvimento de um sistema para gerenciamento de uma loja de artigos esportivos.

Trabalho de Integração Multidisciplinar para a conclusão do terceiro semestre do curso de Análise e Desenvolvimento de Sistemas apresentado à Universidade Paulista.

Orientador:

Aprovado em: \_\_\_/\_\_\_/\_\_\_

BANCADA EXAMINADORA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_/\_\_\_/\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_/\_\_\_/\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_/\_\_\_/\_\_\_

Agradecimento

Agradecemos em primeiro lugar a Deus por ser à base das nossas conquistas. Aos professores, pela dedicação em suas orientações prestadas na elaboração deste trabalho, nos incentivando e colaborando no desenvolvimento de nossas ideias. Aos amigos e familiares pelo apoio e colaboração no trabalho, nos incentivando com dicas preciosas.

Dedicatória

Dedicamos este trabalho aos nossos familiares e amigos, que compreenderam os momentos de ausência necessários para o desenvolvimento deste trabalho. Muito obrigado a todos pela compreensão e paciência.

**RESUMO**

A empresa Intergalact Sports, uma loja de artigos esportivos quer informatizar seu negócio. Atualmente o controle de estoque e vendas é realizado através de anotações em cadernos e este não tem se mostrado eficiente. Para resolver esse problema a empresa procura uma fábrica de software para o desenvolvimento de um novo sistema para controlar suas atividades. O sistema deve contemplar um controle de cadastro de usuários e clientes, assim como registrar as vendas efetuadas e gerir o estoque dos produtos cadastrados. O sistema deve ser acessível para eventuais usuários com necessidades especiais e ser acessível através de dispositivos móveis, sem que haja perdas de desempenho e usabilidade.

**Palavras chave:** Esportivo, Software, ERP.

**ABSTRACT**

The company Intergalact Sports, a sporting goods store wants to computerize your business. Currently, inventory and sales control is carried out through notebooks and it has not been efficient. To solve this problem the company looks for a software factory to develop a new system to control its activities. The system should include a control of the registration of users and customers, as well as record the sales made and manage the inventory of the products registered. The system must be accessible to potential users with special needs and be accessible through mobile devices without loss of performance and usability.

**Keywords**: Sports, Software, ERP.

**LISTA DE ILUSTRAÇÕES**

**LISTA DE GRÁFICOS**

**LISTA DE QUADROS**

**LISTA DE ABREVIATURAS E SIGLAS**

**SUMÁRIO**

[1. INTRODUÇÃO 13](#_Toc515129494)

[1.1. Objetivo 15](#_Toc515129495)

[2. BASE CONCEITUAL 16](#_Toc515129496)

[2.1. Programação orientada a objetos. 16](#_Toc515129497)

[2.2. Interação humano-computador (IHC) 17](#_Toc515129498)

[2.3. Engenharia de Software 17](#_Toc515129499)

[2.4. Banco de Dados 18](#_Toc515129500)

[2.5. UML 19](#_Toc515129501)

[2.6. Trello 22](#_Toc515129502)

[2.7. Ferramenta de controle de versão (Git) 22](#_Toc515129503)

[3. COMPARAÇÃO DE SISTEMAS 23](#_Toc515129504)

[3.1. Sistema 1 23](#_Toc515129505)

[3.2. Sistema 2 23](#_Toc515129506)

[3.3. Sistema 3 23](#_Toc515129507)

[3.4. Funcionalidades 23](#_Toc515129508)

[3.4.1. Funcionalidade 1 23](#_Toc515129509)

[3.4.2. Funcionalidade 2 23](#_Toc515129510)

[3.4.3. Funcionalidade 3 23](#_Toc515129511)

[3.4.4. Funcionalidade 4 23](#_Toc515129512)

[3.4.5. Funcionalidade 5 23](#_Toc515129513)

[3.4.6. Comparativo funcionalidades 23](#_Toc515129514)

[4. DESENVOLVIMENTO 23](#_Toc515129515)

[4.1. Organização de tarefas (Trello) 24](#_Toc515129516)

[4.2. Ferramenta de controle de versão (Git) 25](#_Toc515129517)

[4.3. Linguagem de programação utilizada 25](#_Toc515129518)

[4.4. Interação humano-computador (IHC) 25](#_Toc515129519)

[4.4.1. Design do software 25](#_Toc515129520)

[4.4.2. Paleta de cores do sistema 25](#_Toc515129521)

[4.4.3. Avaliação realizada no Workshop 25](#_Toc515129522)

[4.5. Requisitos funcionais 25](#_Toc515129523)

[4.5.1. [RF01] Cadastro de categoria 25](#_Toc515129524)

[4.5.2. [RF02] Manter de usuário 26](#_Toc515129525)

[4.5.3. [RF03] Gerar relatório 26](#_Toc515129526)

[4.5.4. [RF04] Manter pedidos de produto 26](#_Toc515129527)

[4.5.5. [RF05] Manter cliente 26](#_Toc515129528)

[4.5.6. [RF06] Manter produto 27](#_Toc515129529)

[4.5.7. [RF07] Vender produto 27](#_Toc515129530)

[4.5.7. [RF07] Controle de caixa 27](#_Toc515129531)

[4.6. Requisitos não funcionais 28](#_Toc515129532)

[4.6.1. [RNF001] Usabilidade 28](#_Toc515129533)

[4.6.2. [RNF002] Manutenibilidade 28](#_Toc515129534)

[4.6.3. [RNF003] Confiabilidade 28](#_Toc515129535)

[4.6.4. [RNF004] Desempenho 28](#_Toc515129536)

[4.6.5. [RNF005] Portabilidade 28](#_Toc515129537)

[4.6.6. [RNF006] Segurança 28](#_Toc515129538)

[4.7. Diagrama de caso de uso 28](#_Toc515129539)

[4.8. Documentação caso de uso 29](#_Toc515129540)

[4.8.1. Manter produto. 29](#_Toc515129541)

[4.8.2. Manter categoria. 30](#_Toc515129542)

[4.8.3. Manter usuário. 31](#_Toc515129543)

[4.8.4. Manter pedidos de produtos. 31](#_Toc515129544)

[4.8.5. Gerar relatório. 32](#_Toc515129545)

[4.8.6. Manter Clientes 33](#_Toc515129546)

[4.8.7.Vender produto. 33](#_Toc515129547)

[4.8.8. Controlar caixa. 34](#_Toc515129548)

[4.9. Diagrama de classe. 35](#_Toc515129549)

[4.10. Diagrama Atividades 37](#_Toc515129550)

[4.10.1. Cadastrar produto 37](#_Toc515129551)

[4.10.2. Cadastro de cliente. 38](#_Toc515129552)

[4.10.3. Venda 38](#_Toc515129553)

[4.11. Software AFSport (telas do software e descrição breve) 39](#_Toc515129554)

[5. Conclusão 39](#_Toc515129555)

[5.1. Projetos futuros 39](#_Toc515129556)

[6. REFERÊNCIAS BIBLIOGRÁFICAS 40](#_Toc515129557)

1. INTRODUÇÃO

Segundo LAUDON (2007), a implantação de um sistema de informação em uma empresa pode significar uma vantagem competitiva através do controle de custos, traçando perfil dos clientes e fornecedores. Um sistema bem projetado se torna confiável e permite facilidade de acesso as informações, trazendo agilidade no levantamento de relatórios de vendas e estoque dos produtos.

Com base nas informações levantadas, uma loja de artigos esportivos procurou uma fábrica de software, para que fosse desenvolvido um software visando controlar suas atividades de venda e controle de estoque. O sistema conta também com um módulo de cadastro de clientes e filtros para geração de relatórios. O software possui uma interface intuitiva e de fácil acesso para eventuais usuários com necessidades especiais, além de ser possível seu acesso através de dispositivos móveis.

O sistema teve seu início com a realização do levantamento de requisitos funcionais e não funcionais de acordo com o modelo de negócio da empresa solicitante. Foram criados os diagramas de caso de uso com o intuito de facilitar o entendimento do que seria implementado no sistema, dando uma visão geral para a aprovação do projeto por parte da solicitante. Após a aceitação foram gerados os diagramas de classe, sequência e o Modelo Entidade Relacionamento do banco de dados para nortear o desenvolvimento do sistema.

O sistema teve como base em seu desenvolvimento uma linguagem de programação orientada a objetos (POO) chamada C# (lê-se C-Sharp), onde os objetos do mundo real são modelados para o mundo computacional. O C# é uma linguagem poderosa e simples que herdou muitos recursos do C++ e Microsoft Visual Basic, tornando-a uma linguagem mais limpa e lógica (SHARP, 2014).

Em todo sistema onde se deseja persistência de dados, é necessário o uso de um banco de dados. Na década de 1970 surgiram os bancos de dados relacionais que utilizam a linguagem SQL (*Structured Query Language*) como padrão. Bancos de dados relacionais possuem sua estrutura baseada em registros organizados e relacionados em tabelas, onde as linhas são os registros e as colunas descrevem as características dos registros (FERRARI, 2007).

Para auxiliar o desenvolvimento do sistema pelos desenvolvedores, foi utilizado o sistema de controle de versão chamado Git. Este sistema de versão permite controlar de forma segura as mudanças no código e desfazê-las caso seja necessário sem a necessidade de cópias manuais (PORTAL DEVMEDIA, 2018).

Com a implantação do sistema, espera-se maior agilidade no atendimento dos clientes e controle do fluxo de caixa, assim como estoque de produtos. Estas melhorias podem ser decisivas na fidelização de clientes que buscam realizar suas compras de maneira rápida.

1.1. Objetivo

Este trabalho visa o desenvolvimento de um software de computador, onde o mesmo fará todo o controle de vendas, pedidos de compra, estoque de produtos, cadastro de clientes e geração de relatórios pertinentes a toda movimentação realizada pelo sistema. Assim como o controle de acesso ao sistema por usuários cadastrados, com o intuito de proteger as informações nele armazenadas e manter a rastreabilidades das operações realizadas através do usuário que as realizou.

2. BASE CONCEITUAL

Nesse tópico será abordado uma breve explicação sobre as ferramentas e metodologia que serão utilizadas no desenvolvimento do projeto AFSport. Cada ferramenta utilizada é essencial para o desenvolvimento do projeto e a metodologia abordada tem o objetivo de organizar a equipe e o desenvolvimento do projeto.

2.1. Programação orientada a objetos.

A programação orientada a objetos ao contrário do que a maioria dos profissionais pensa, ela surgiu em 1967 junto com a linguagem Simula. Algum tempo depois nos anos 70 a XEROX criou o SmallTalk, considerada a primeira linguagem puramente orientada a objetos (PORTAL WIKILIVROS, 2018).

Segundo JOYANES AGUILAR (2011), na programação orientada a objetos há alguns conceitos, que juntos moldam suas características e a torna tão atrativa. Entre eles podemos citar:

• Abstração:

É como um objeto do mundo real é moldado para o mundo computacional, onde são levantadas suas características relevantes para cada tipo de aplicação.

• Classe:

Uma classe descreve a estrutura interna dos objetos que pertencem a ela, é o objeto representado pela abstração, nela estão definidos os atributos e as funcionalidades de cada objeto.

• Objeto:

É a instancia de uma classe. Um tipo abstrato de dados, onde seus atributos e métodos são definidos na classe a qual ele pertence.

• Encapsulamento:

É utilizado para impedir o acesso direto a estrutura de dados interna de uma classe pelo usuário dela. É a combinação dos dados e os métodos que podem ser executados, garantindo assim o acesso aos dados somente por meio dos próprios métodos do objeto.

• Herança:

O conceito de herança pode ser relacionado com a genética humana, sendo assim, um objeto pode transmitir suas propriedades para outros objetos, permitindo reutilizar objetos já definidos anteriormente reduzindo código redundante em programas. Ter suporte a herança, é um pré-requisito para que uma linguagem seja orientada a objetos.

• Polimorfismo:

É um conceito de grande importância na programação orientada a objetos. Objetos diferentes podem responder de maneira diferente quando ligados através da herança, como por exemplo, classes derivadas de superclasses podem ter métodos parecidos, porém, o mesmo terá comportamento diferente para cada classe derivada.

2.2. Interação humano-computador (IHC)

A interação humano computador trata da forma que as pessoas utilizam uma interface, sendo ela um display de um micro-ondas ou uma tela de software em um computador. Ela se preocupa desde o design, avaliação e a implementação de interfaces onde há interação com o ser humano, buscando a construção de uma interface intuitiva e amigável ao usuário e também promovendo facilidade na aprendizagem e maior produtividade (BARBOSA, 2011).

2.3. Engenharia de Software

O termo surgiu pela primeira vez em 1968 após a crise do software, período em que o desenvolvimento era problemático e defeituoso, resultando em um produto de difícil manutenção em que não seguia padrões.

Segundo Friedrich Ludwig Bauer: “Engenharia de Software é a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e trabalhe em maquinas reais”.

Engenharia de Software é a área da computação específica para a concepção de projeto de software, documentação, desenvolvimento, manutenção e qualidade utilizando de práticas gerenciais afim de definir e cumprir objetivos compreendidos e alinhados com toda a equipe envolvida.

Contextualização dos principais aspectos:

Analise de Requisitos. Através da análise de requisito é o momento onde se efetua o conhecimento do problema para desenvolver o software (JALOTE, 2005).

Design do Software. Pelo design do software é o momento que o engenheiro de software realiza o planejamento da solução do problema que foi levantado no documento de requisitos (JALOTE, 2005).

Codificação. A codificação é o momento que pega o problema resolvido no design do software e o transformara em uma linguagem de programação (JALOTE, 2005).

Teste. O teste é uma maneira de medir o controle da qualidade do software durante o desenvolvimento de software (JALOTE, 2005).

2.4. Banco de Dados

De acordo com MEDEIROS (2007 apud DATE, 2000) “um banco de dados é uma coleção de dados persistentes utilizadas pelos sistemas de aplicação de uma empresa”. Um arquivo de banco de dados é composto por tabelas que através de seus atributos gravam registros que descrevem entidades do mundo real, cujas informações podem ser manipuladas e recuperadas mais tarde por algum sistema de informação.

MEDEIROS (2007), afirma que “é necessário que um BD tenha uma representação eficiente que possibilite acesso a informações corretas, em tempo hábil”. Para um BD ser eficiente, alguns princípios listados abaixo devem ser observados:

• Redundância: Não deve haver duplicidade das informações dentro de um BD, como por exemplo, tabelas distintas contendo campos com informações idênticas.

• Inconsistência: Deve-se ter cuidado quanto ao inserir um dado em um campo respeitando seu tamanho e tipo por exemplo. A redundância pode gerar inconsistência, pois uma mesma informação gravada em tabelas diferentes pode ter a informação atualizada em somente um local.

• Integração: Estabelecer procedimentos para níveis de acesso para os diferentes setores da empresa, visto que atualizações podem comprometer a integridade das informações do BD, evitando que as atualizações realizadas por um setor não comprometam o outro.

Para garantir esses princípios utiliza-se um sistema gerenciador de banco de dados (SGBD), que é um software responsável por administrar o banco de dados e garantir a segurança e a integridade dos dados. São exemplos de SGBD, Oracle, MySQL, Microsoft SQL Server e PostgreSQL. Esses SGBD têm em comum, a linguagem de consulta estruturada SQL.

2.5. UML

Com o crescimento na utilização das linguagens orientadas a objetos e a complexidade dos novos projetos, surgiu à necessidade de se criar metodologias que possibilitassem projetar e planejar a construção dos sistemas. Foram criadas metodologias, porém nenhuma atendia totalmente a necessidade dos construtores (FOWLER, KENDAL, 2000).

Surge então em meados dos anos 90 a *Unified Modeling Language* (UML), criada por Grady Booch, Ivar Jacobson e James Rumbaugh, três estudiosos e criadores de métodos orientados a objetos. A UML não é uma metodologia, mais sim uma linguagem de modelagem, pois ela não define quais procedimentos precisam ser seguidos (FURLAN, 1998).

Ainda segundo FOWLER, KENDAL (2000), a UML baseia-se em uma notação principalmente gráfica, utilizada para expressar projetos.

A UML é constituída de vários diagramas, que expressam informações sobre alguns elementos de um modelo. Cada diagrama ilustra o que ocorre em cada parte de um sistema, dando uma visão diferente ao programador (FURLAN, 1998).

Fazem parte da UML os seguintes diagramas: Caso de uso, classes, atividade, interação, estado e implementação.

• Diagrama de caso de uso:

Segundo FOWLER, KENDAL (2000), o caso de uso representa como é a interação do usuário com o sistema, e apesar de ser amplamente utilizado, não é fundamental para a realização de projetos. Ele tem como objetivo descrever quais são os requisitos funcionais para um sistema, tendo como elementos básicos o ator, caso de uso, interação e sistema.

Figura 01 – Elementos básicos do diagrama de caso de uso



Fonte: o autor.

O ator interage com o sistema, podendo ser um usuário, outro sistema, dentre outros. O caso de uso descreve a ação do ator no sistema, já a interação trata do relacionamento entre atores e os casos de uso (FOWLER, KENDAL, 2000).

• Diagrama de classes:

Presente em quase todas as metodologias de análise orientadas a objetos, ele especifica as classes e os relacionamentos entre elas. Representada por um retângulo, uma classe pode ser dividida em até três partes onde na primeira parte está o nome da classe, na segunda parte seus atributos com seus respectivos tipos e visibilidade, e por ultimo seus métodos. Uma classe possui quatro tipos básicos de relacionamentos (FURLAN, 1998). São eles:

1. Agregação: indica que uma classe pertence à outra;
2. Associação: ligação entre classes;
3. Generalização/especificação: Relacionamento de herança entre classes;
4. Dependência: indica que uma classe depende de outra.

Figura 02 – Relacionamentos entre classes



Fonte: o autor.

• Diagrama de atividade:

Segundo SOMMERVILLE (2011), o diagrama de atividades descreve as atividades de um processo ou processamento de dados. Iniciado por um circulo preenchido, possui setas indicando o fluxo do processo onde as atividades são representadas por retângulos de cantos arredondados. Pode haver processos paralelos que são iniciados por uma barra sólida, que também servem para indicar o final de processos paralelos, onde todas as atividades paralelas devem ter sido completadas. Para indicar o final de uma atividade, utiliza-se um circulo preenchido dentro de outro circulo.

Figura 03 – Exemplo de diagrama de atividades.



Fonte: SOMMERVILLE (2011).

2.6. Trello

O Trello é um site que pode ser acessado utilizando os navegadores Google Chrome, Mozilla Firefox, Safari e Internet Explorer, além de também possuir aplicativos na Google Play e iTunes. É uma ferramenta de gerenciamento de projetos ou tarefas em forma de listas, ele é muito versátil e pode ser moldado de acordo com as necessidades de cada usuário. É possível criar quadros e definir quem pode ter acesso a eles, ver as atualizações e colaborar com os mesmos. Assim toda a equipe pode acompanhar de perto tudo o que se passa com o projeto.

2.7. Ferramenta de controle de versão (Git)

“É um sistema de controle de versão, de código aberto, usado no gerenciamento de código-fonte”, (WIKILIVROS, 2018). Criado por Linus Torvald, possui arquitetura descentralizada, todo repositório do Git possui um histórico e seguidor de revisões. O usuário faz uma cópia da pasta do repositório central, e depois pode publicar as mudanças para os usuários desse servidor (WIKILIVROS, 2018). O acesso pode ser realizado usando linhas de comando ou por alguns softwares disponíveis para conexão com o GIT, como por exemplo o GitHub.

3. COMPARAÇÃO DE SISTEMAS

3.1. Sistema 1

3.2. Sistema 2

3.3. Sistema 3

3.4. Funcionalidades

3.4.1. Funcionalidade 1

3.4.2. Funcionalidade 2

3.4.3. Funcionalidade 3

3.4.4. Funcionalidade 4

3.4.5. Funcionalidade 5

3.4.6. Comparativo funcionalidades

4. DESENVOLVIMENTO

Neste capítulo são abordados os requisitos funcionais e não funcionais, diagrama de caso de uso, diagrama de atividades, e diagrama de classes. Nesse capítulo também constam as telas do software, junto com a especificação das funcionalidades e as tecnologias e metodologias usadas na criação do software AFSport.

4.1. Organização de tarefas (Trello)

No planejamento deste projeto foi definido quatro listas para a organização das tarefas que são: a fazer, em andamento, concluído e base. Dentro dessas listas são adicionados os “cards”, assim chamados as tarefas, que descreve tudo o que precisa ser feito, o prazo para a entrega e quem está responsável por essa atividade.

Na primeira lista, “a fazer”, ficam todas as tarefas planejadas para o projeto e que não foram iniciadas.

Na segunda lista, “em andamento”, ficam as tarefas que estão sendo executadas pelos membros da equipe.

Na terceira lista, “concluído”, ficam todas as tarefas finalizadas e certificadas pelos membros da equipe.

Na quarta lista, “base”, foi utilizada para armazenar informações uteis que poderiam ajudar no desenvolvimento do projeto.

Figura 04 – Exemplo de diagrama de atividades do Trello.



Fonte: O autor

4.2. Ferramenta de controle de versão (Git)

4.3. Linguagem de programação utilizada

O sistema foi desenvolvido utilizando a linguagem de programação orientada a objetos C-Sharp (C#), utilizando o framework Visual Studio 2017.

4.4. Interação humano-computador (IHC)

O sistema AFSport foi desenvolvido com o intuito de maximizar sua utilização de forma simples e intuitiva, tendo como base os seguintes itens:

4.4.1. Design do software

O Software utiliza o Design minimalista para compor suas telas, botões e ícones. Tudo isso para poder entregar uma interface simples e amigável, tendo a experiência de uso como o seu aspecto central e fazendo da usabilidade um ponto forte.

4.4.2. Paleta de cores do sistema

A cor azul presente em todas as telas da interface tem como intuito, transmitir confiança e maturidade, além de proporcionar um ambiente acolhedor e confortável na experiência do usuário.

4.4.3. Avaliação realizada no Workshop

4.5. Requisitos funcionais

Neste capítulo estão listados os requisitos funcionais do sistema AFSport.

4.5.1. [RF01] Cadastro de categoria

O sistema permite cadastrar, alterar e excluir/inativar uma categoria.

Quadro 01: [RF01] Manter de categoria

|  |  |
| --- | --- |
| Parâmetros de Entrada | Nome, Descrição, ativa ou inativa. |
| Descrição | Dados levantados para cadastro de uma categoria |
| Restrições | Valida os dados e cadastra, altera ou exclui uma categoria. |
| Dados de Saída | No caso do sistema encontrar algum erro, uma mensagem será exibida. |

Fonte: O autor.

4.5.2. [RF02] Manter de usuário

O sistema permite cadastrar, alterar e excluir/inativar um usuário.

Quadro 02: [RF02] Manter usuário

|  |  |
| --- | --- |
| Parâmetros de Entrada | Login e senha |
| Descrição | Dados levantados para cadastro de usuário. |
| Restrições | Valida os dados e cadastra, altera ou exclui um usuário. |
| Dados de Saída | No caso do sistema encontrar algum erro, uma mensagem será exibida. |

Fonte: O autor.

4.5.3. [RF03] Gerar relatório

O sistema permite gerar relatórios a partir das informações inseridas.

Quadro 03: [RF03] Gerar relatório

|  |  |
| --- | --- |
| Parâmetros de Entrada | Data inicial e data final. |
| Descrição | Dados levantados para criação de relatórios. |
| Restrições | Valida os dados e busca as informações no banco de dados. |
| Dados de Saída | Mostra um relatório com as informações solicitadas |

Fonte: O autor.

4.5.4. [RF04] Manter pedidos de produto

O sistema permite cadastrar, alterar e excluir um pedido.

Quadro 04: [RF04] Manter de pedidos

|  |  |
| --- | --- |
| Parâmetros de Entrada | Produto e quantidade. |
| Descrição | Dados levantados para cadastro de operações. |
| Restrições | Valida os dados e cadastra, altera ou exclui um pedido. |
| Dados de Saída | No caso do sistema encontrar algum erro, uma mensagem será exibida. |

Fonte: O autor.

4.5.5. [RF05] Manter cliente

O sistema permite cadastrar, alterar e excluir/inativar um cliente.

Quadro 05: [RF05] Manter cliente

|  |  |
| --- | --- |
| Parâmetros de Entrada | Nome, e-mail, rua, número, bairro, cidade e estado. |
| Descrição | Dados levantados para cadastro de cliente. |
| Restrições | Valida os dados e cadastra, altera ou exclui um cliente. |
| Dados de Saída | No caso do sistema encontrar algum erro, uma mensagem será exibida. |

Fonte: O autor.

4.5.6. [RF06] Manter produto

O sistema permite cadastrar, alterar e excluir/inativar um produto.

Quadro 06: [RF06] Manter produto

|  |  |
| --- | --- |
| Parâmetros de Entrada | Nome, descrição, valor de compra, valor de venda e categoria. |
| Descrição | Dados levantados para cadastro de produto. |
| Restrições | Valida os dados e cadastra, altera ou exclui um produto. |
| Dados de Saída | No caso do sistema encontrar algum erro, uma mensagem será exibida. |

Fonte: O autor.

4.5.7. [RF07] Vender produto

O sistema permite cadastrar, alterar e excluir uma venda.

Quadro 07: [RF07] Vender produto

|  |  |
| --- | --- |
| Parâmetros de Entrada | Selecionar cliente, selecionar produtos e escolher forma de pagamento. |
| Descrição | Dados levantados para realizar uma venda. |
| Restrições | Valida os dados e cadastra, altera ou exclui uma venda. |
| Dados de Saída | No caso do sistema encontrar algum erro, uma mensagem será exibida. |

Fonte: O autor.

4.5.7. [RF07] Controle de caixa

O sistema permite gerenciar o fluxo de caixa.

Quadro 07: [RF07] Controle de caixa

|  |  |
| --- | --- |
| Parâmetros de Entrada | Selecionar a data do caixa, escolher uma opção do caixa para processar e exibir as informações solicitadas. |
| Descrição | Dados levantados para realizar o controle de fluxo do caixa. |
| Restrições | Valida os dados e exibe as informações solicitadas. |
| Dados de Saída | No caso do sistema encontrar algum erro, uma mensagem será exibida. |

Fonte: O autor.

4.6. Requisitos não funcionais

Neste capítulo estão listados os requisitos não funcionais do sistema AFSport.

4.6.1. [RNF001] Usabilidade

O sistema foi desenvolvido de forma que seja de fácil aprendizagem, com um esforço mínimo para aprender e focado na facilidade de utilização com uma interface agradável para a realização das tarefas. Ou seja, em pouco tempo e com esforço mínimo o usuário já irá alcançar um bom desempenho no uso do sistema.

4.6.2. [RNF002] Manutenibilidade

O sistema AFSport foi desenvolvido utilizando a linguagem de programação C-Sharp, que por ser uma linguagem orientada a objetos, permite a modularização do sistema. Sendo assim, erros encontrados em sua utilização, podem ser facilmente solucionados. Caso também seja necessária alguma alteração para inclusão de funcionalidades, por exemplo, a organização do código fonte permite fazê-la com facilidade sem que grandes alterações sejam necessárias.

4.6.3. [RNF003] Confiabilidade

O software AFSport foi testado para garantir que todos os requisitos implementados, fossem cumpridos sem ocorrência de erros e mantendo o desempenho esperado diante de cada operação.

4.6.4. [RNF004] Desempenho

4.6.5. [RNF005] Portabilidade

4.6.6. [RNF006] Segurança

4.7. Diagrama de caso de uso

A figura XX, representa o diagrama de caso de uso do sistema AFSport.



Fonte: O autor

4.8. Documentação caso de uso

Nesse tópico abordamos uma descrição dos casos de uso, onde são identificados as ações e os atores que interagem junto ao sistema para concluir o requisito.

4.8.1. Manter produto.

O quadro XX, representa as etapas e o relacionamento ator/sistema para manter um produto.

|  |  |  |
| --- | --- | --- |
| Nome do caso de uso | Manter Produto | |
| Caso de uso Geral |  | |
| Ator Principal | Funcionário | |
| Ator(res) Secundário(s) |  | |
| Resumo | Este caso de uso descreve as etapas percorridas pelo funcionário para cadastrar, alterar e excluir um produto. | |
| Pré condições |  | |
| Pós condições | Produto cadastrado | |
| Fluxo Principal | | |
| Ações do Ator | | Ações do Sistema |
| 1. Funcionário clica no botão do formulário de cadastro de produto | |  |
|  | | 1. Apresenta as opções cadastrar, alterar e excluir |
| 1. Funcionário preenche os dados | |  |
| 1. Funcionário clica em enviar formulário | |  |
|  | | 1. Verifica se os dados estão corretos |
|  | | 1. Sistema salva os dados no banco de dados |
|  | | 1. Mostra a tela de sucesso/efetivação |

Fonte: O autor.

4.8.2. Manter categoria.

O quadro XX, representa as etapas e o relacionamento ator/sistema para cadastrar uma categoria.

|  |  |  |
| --- | --- | --- |
| Nome do caso de uso | Manter Categoria | |
| Caso de uso Geral |  | |
| Ator Principal | Funcionário | |
| Ator(res) secundário(s) |  | |
| Resumo | Este caso de uso descreve as etapas percorridas pelo funcionário paracadastrar, alterar e excluir uma categoria | |
| Pré condições |  | |
| Pós condições | Categoria cadastrada | |
| Fluxo Principal | | |
| Ações do Ator | | Ações do Sistema |
| 1. Funcionário clica no botão do formulário de cadastro de categoria | |  |
|  | | 1. Apresenta a tela de cadastro |
| 1. Funcionário preenche os dados | |  |
| 1. Funcionário clica em enviar formulário | |  |
|  | | 1. Sistema salva os dados no banco de dados |
|  | | 1. Mostra a tela de sucesso/efetivação |

Fonte: O autor.

4.8.3. Manter usuário.

O quadro XX, representa as etapas e o relacionamento ator/sistema para cadastrar um usuário.

|  |  |  |
| --- | --- | --- |
| Nome do caso de uso | Manter Usuários | |
| Caso de uso Geral |  | |
| Ator Principal | Funcionário | |
| Ator(res) Secundário(s) |  | |
| Resumo | Este caso de uso descreve as etapas percorridas pelo funcionário para cadastrar, alterar e excluir um usuário | |
| Pré condições |  | |
| Pós condições | Usuário cadastrado | |
| Fluxo Principal | | |
| Ações do Ator | | Ações do Sistema |
| 1. Funcionário clica no botão do formulário de cadastro de usuário | |  |
|  | | 1. Apresenta a tela de cadastro |
| 1. Funcionário preenche os dados | |  |
|  | | 1. Verifica se os dados estão corretos |
| 1. Funcionário clica em enviar formulário | |  |
|  | | 1. Sistema salva os dados no banco de dados |
|  | | 1. Mostra a tela de sucesso/efetivação |

Fonte: O autor.

4.8.4. Manter pedidos de produtos.

O quadro XX, representa as etapas e o relacionamento ator/sistema para cadastrar um pedido.

|  |  |  |
| --- | --- | --- |
| Nome do caso de uso | Manter Pedidos de Produtos | |
| Caso de uso Geral |  | |
| Ator Principal | Funcionário | |
| Ator(res) Secundário(s) |  | |
| Resumo | Este caso de uso descreve as etapas percorridas pelo funcionário para cadastrar, alterar e excluir os pedidos de um produto | |
| Pré condições |  | |
| Pós condições | Pedido de produtos cadastrados | |
| Fluxo Principal | | |
| Ações do Ator | | Ações do Sistema |
| 1. Funcionário clica no botão do formulário de cadastro de pedido | |  |
|  | | 1. Apresenta a tela de cadastro |
| 1. Funcionário preenche os dados | |  |
|  | | 1. Verifica se os dados estão corretos |
| 1. Funcionário clica em enviar formulário | |  |
|  | | 1. Sistema salva os dados no banco de dados |
|  | | 1. Mostra a tela de sucesso/efetivação |

Fonte: O autor.

4.8.5. Gerar relatório.

O quadro XX, representa as etapas e o relacionamento ator/sistema para gerar um relatório.

|  |  |  |
| --- | --- | --- |
| Nome do caso de uso | Gerar relatórios | |
| Caso de uso Geral |  | |
| Ator Principal | Funcionário | |
| Ator(res) Secundário(s) |  | |
| Resumo | Este caso de uso descreve as etapas percorridas pelo funcionário para gerar os relatórios da loja. | |
| Pré condições |  | |
| Pós condições | Relatório gerado | |
| Fluxo Principal | | |
| Ações do Ator | | Ações do Sistema |
| 1. Funcionário clica no botão de gerar relatórios | |  |
|  | | 1. Sistema informa qual tipo de relatório pode ser gerado |
| 1. Funcionário específica o tipo de relatório que deve ser gerado | |  |
|  | | 1. Sistema gera o relatório |
|  | | 1. Sistema exibe o relatório. |

Fonte: O autor.

4.8.6. Manter Clientes

O quadro XX, representa as etapas e o relacionamento ator/sistema para cadastrar um cliente.

|  |  |  |
| --- | --- | --- |
| Nome do caso de uso | Manter Clientes | |
| Caso de uso Geral |  | |
| Ator Principal | Funcionário | |
| Ator(res) Secundário(s) |  | |
| Resumo | Este caso de uso descreve as etapas percorridas pelo funcionário para cadastrar, alterar e excluir um cliente | |
| Pré condições |  | |
| Pós condições | Cliente cadastrado | |
| Fluxo Principal | | |
| Ações do Ator | | Ações do Sistema |
| 1. Funcionário clica no botão do formulário de cadastro de clientes | |  |
|  | | 1. Apresenta a tela de cadastro |
| 1. Funcionário preenche os dados | |  |
|  | | 1. Verifica se os dados estão corretos |
| 1. Funcionário clica em enviar formulário | |  |
|  | | 1. Sistema salva os dados no banco de dados |
|  | | 1. Mostra a tela de sucesso/efetivação |

Fonte: O autor.

4.8.7.Vender produto.

O quadro XX, representa as etapas e o relacionamento ator/sistema para vender produto.

|  |  |  |
| --- | --- | --- |
| Nome do caso de uso | Vender produto | |
| Caso de uso Geral |  | |
| Ator Principal | Funcionário | |
| Ator(res) Secundário(s) | Cliente | |
| Resumo | Este caso de uso descreve as etapas percorridas pelo funcionário para gerenciar a venda de um produto. | |
| Pré condições | Para efetuar a venda de um produto o mesmo deverá estar cadastrado e ter a quantidade necessária mantida sobre o estoque. | |
| Pós condições | Produto será vendido | |
| Fluxo Principal | | |
| Ações do Ator | | Ações do Sistema |
| 1. Funcionário inicia o sistema | |  |
|  | | 1. Apresenta a tela inicial |
| 1. Funcionário lança os produtos e quantidade | |  |
|  | | 1. Verifica a disponiblidade no estoque |
| 1. Funcionário verifica a forma de pagamento | |  |
| 1. Cliente escolhe a forma de pagamento | |  |
| 1. Funcionário finaliza venda | |  |
|  | | 1. Sistema salva as alterações no banco de dados |
|  | | 1. Mostra a tela de sucesso/efetivação |
| Restrições/validações | | 1. Para efetuar a venda de um produto é necessário ter quantidade disponivel. |
|  |
| Fluxo de Exceção - Quantidade indisponivel | | |
| Ações do Ator | | Ações do Sistema |
|  | | 1.Informa ao funcionário que o produto não possui quantidade disponivel |
| 1. Funcionário cancela a venda | |  |

Fonte: O autor.

4.8.8. Controlar caixa.

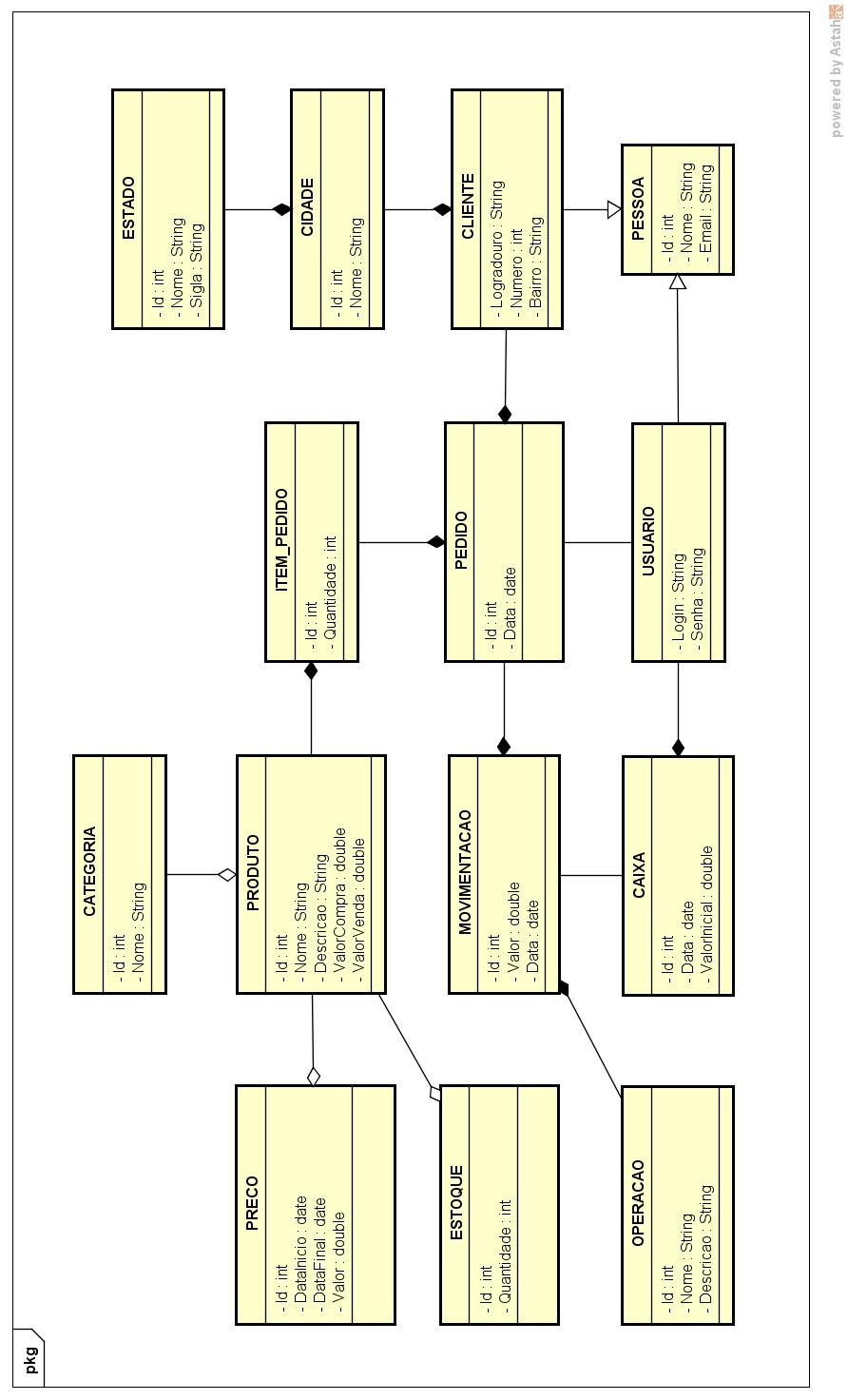
O quadro XX, representa as etapas e o relacionamento ator/sistema para controle de caixa.

|  |  |  |
| --- | --- | --- |
| Nome do caso de uso | Controlar caixa | |
| Caso de uso Geral |  | |
| Ator Principal | Funcionário | |
| Ator(res) Secundário(s) |  | |
| Resumo | Este caso de uso descreve as etapas percorridas pelo funcionário para controlar caixa. | |
| Pré condições |  | |
| Pós condições | O fluxo de caixa será controlado | |
| Fluxo Principal | | |
| Ações do Ator | | Ações do Sistema |
| 1. Funcionário clica no botão de relatórios | |  |
|  | | 1. Apresenta as opções de relatórios |
| 1. Funcionário escolhe a opção relatório de vendas | |  |
|  | | 1. Informa os dados das vendas |
| 1. Funcionário atualiza fluxo de caixa | |  |
|  | | 1. Sistema salva os dados no banco de dados |
|  | | 1. Mostra a tela de sucesso/efetivação |

Fonte: O autor.

4.9. Diagrama de classe.

A figura XX representa o diagrama de classes do sistema AFSport.



Fonte: O autor.

4.10. Diagrama Atividades

Abaixo estão listados os diagramas de atividades para cadastro de produto, cadastro de cliente e venda.

4.10.1. Cadastrar produto

A figura XX representa o diagrama com os passos até a conclusão do cadastro de um produto.

Fonte: O autor

4.10.2. Cadastro de cliente.

A figura XX representa o diagrama com os passos até a conclusão do cadastro de um cliente.



Fonte: O autor

4.10.3. Venda

A figura XX representa o diagrama com os passos até a conclusão da venda de um produto.



Fonte: O autor

4.11. Software AFSport (telas do software e descrição breve)

5. Conclusão

5.1. Projetos futuros

6. REFERÊNCIAS BIBLIOGRÁFICAS

LAUDON, Kenneth C., 1994**- Sistemas de Informações Gerenciais -** Kenneth C. Laudon e Jane P. Laudon; tradução Thelma Guimarães; revisãotécnica Belmiro N. João. – 7. Ed. – São Paulo: Pearson Prentice Hall, 2007.

SHARP, JOHN., Microsoft Visual C# 2013: **passo a passo [recurso eletrônico]**; tradução: João Eduardo Nóbrega Tortello; revisão técnica: Daniel Antonio Callegari. – Porto Alegre: Bookman, 2014.

PORTAL DEVMEDIA. **Git e Github.** Disponível em: <https://www.devmedia.com.br/guia/git-e-github/37585 >Acesso em 28 abril de 2018.

PORTAL WIKILIVROS. **Programação orientada a objetos.** Disponível em: <https://pt.wikibooks.org/wiki/Programação\_Orientada\_a\_Objetos/Introdução> Acesso em 29 de abril de 2018.

JOYANES AGUILAR, LUIS. **Fundamentos de programação: algoritmos, estruturas de dados e objetos** / Luis Joyanes Aguilar; tradução: Paulo Heraldo Costa do Valle; revisão técnica: Flávio Soares Corrêa da Silva. – 3. Ed. – Dados eletrônicos. – Porto Alegre: AMGH, 2011.

DATE, C. J. Introdução a sistemas de bancos de dados. 7. ed. Rio de

Janeiro: Campus, 2000.

Barbosa, Simone D. J. (Simone Diniz Junqueira). **Interação humano-computador** [recurso eletrônico] / Simone Diniz Junqueira Barbosa, Bruno Santana da Silva. · Rio de Janeiro: Elsevier, 2011.

Medeiros, Luciano Frontino de. **Banco de dados: princípios e prática** / Luciano

Frontino de Medeiros – Curitiba: Ibpex, 2007.

FOWLER, Martin; KENDAL, Scott. **UML Essencial : um breve guia para a linguagem padrão de modelagem de objetos**. Trad. Vera Pezerico e Christian Thomas Price. 2. ed. Porto Alegre: Bookman, 2000.

SOMMERVILLE, IAN. **Engenharia de Software** / Ian Sommerville; tradução Ivan Bosnic e Kalinka G. de O. Gonçalves; revisão técnica Kechi Hirama. — 9. ed. — São Paulo: Pearson Prentice Hall, 2011.

FURLAN, José Davi. **Modelagem de Objetos através da UML – the Unified Modeling, Language**. São Paulo: Makron Books, 1998.

CASTELLI, Ian. **Trello: como esta ferramenta pode ajudar você a organizar a sua vida.** TECMUNDO. Disponível em: <https://www.tecmundo.com.br/organizacao/75128-trello-ferramenta-ajudar-voce-organizar-vida.htm> Acesso em 12/05/2018.

**Trello.** Disponível em: < https://trello.com/about> Acesso em 12/05/2018.

FERRARI, Fabrício Augusto. **Crie banco de dados em MySQL** / Fabrício Augusto Ferrari. – São Paulo: Digerati Books, 2007.

PORTAL WIKILIVROS. **Git.** Disponível em: <https:// https://pt.wikibooks.org/wiki/Git> Acesso em 29 de abril de 2018.