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

Dedicatória

**RESUMO**

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**

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 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](#_Toc513899949)

[2. BASE CONCEITUAL 15](#_Toc513899950)

[2.1. Programação orientada a objetos. 15](#_Toc513899951)

[2.2. Interação humano-computador (IHC) 16](#_Toc513899952)

[2.3. Engenharia de Software 16](#_Toc513899953)

[2.4. Banco de Dados 17](#_Toc513899954)

[2.5. UML 18](#_Toc513899955)

[3. COMPARAÇÃO DE SISTEMAS 21](#_Toc513899956)

[3.1. Sistema 1 21](#_Toc513899957)

[3.2. Sistema 2 21](#_Toc513899958)

[3.3. Sistema 3 21](#_Toc513899959)

[3.4. Funcionalidades 21](#_Toc513899960)

[3.4.1. Funcionalidade 1 21](#_Toc513899961)

[3.4.2. Funcionalidade 2 21](#_Toc513899962)

[3.4.3. Funcionalidade 3 21](#_Toc513899963)

[3.4.4. Funcionalidade 4 21](#_Toc513899964)

[3.4.5. Funcionalidade 5 21](#_Toc513899965)

[3.4.6. Comparativo funcionalidades 21](#_Toc513899966)

[4. DESENVOLVIMENTO 21](#_Toc513899967)

[4.1. Organização de tarefas (Trello) 21](#_Toc513899968)

[4.2. Ferramenta de controle de versão (Git) 21](#_Toc513899969)

[4.3. Linguagem de programação utilizada 21](#_Toc513899970)

[4.4. Banco de dados utilizado 21](#_Toc513899971)

[4.5. IHC 21](#_Toc513899972)

[4.6. Requisitos funcionais 21](#_Toc513899973)

[4.6.1. Requisito 1 21](#_Toc513899974)

[4.6.2. Requisito 2 21](#_Toc513899975)

[4.6.3. Requisito 3 21](#_Toc513899976)

[4.6.4. Requisito 4 21](#_Toc513899977)

[4.6.5. Requisito 5 22](#_Toc513899978)

[4.6.6. ......... 22](#_Toc513899979)

[4.7. Requisitos não funcionais 22](#_Toc513899980)

[4.7.1. Requisito 1 22](#_Toc513899981)

[4.7.2. Requisito 2 22](#_Toc513899982)

[4.7.3. Requisito 3 22](#_Toc513899983)

[4.7.4. Requisito 4 22](#_Toc513899984)

[4.7.5. Requisito 5 22](#_Toc513899985)

[4.7.6. .......... 22](#_Toc513899986)

[4.8. Diagrama de caso de uso 22](#_Toc513899987)

[4.9. Documentação caso de uso 22](#_Toc513899988)

[4.9.1. Caso de uso 1 22](#_Toc513899989)

[4.9.2. Caso de uso 2 22](#_Toc513899990)

[4.9.3. Caso de uso 3 22](#_Toc513899991)

[4.9.4. .......... 22](#_Toc513899992)

[4.10. Diagrama de classe (colar diagrama de classe) 22](#_Toc513899993)

[4.11. Diagrama Atividades 22](#_Toc513899994)

[4.11.1. Atividade 1 22](#_Toc513899995)

[4.11.2. Atividade 2 22](#_Toc513899996)

[4.11.3. Atividade 3 22](#_Toc513899997)

[4.12. Software AFSport (telas do software e descrição breve) 22](#_Toc513899998)

[4.13. Interação com usuário 22](#_Toc513899999)

[4.13.1 Avaliação realizada no Workshop 22](#_Toc513900000)

[5. Conclusão 22](#_Toc513900001)

[5.1. Projetos futuros 22](#_Toc513900002)

[6. REFERÊNCIAS BIBLIOGRÁFICAS 23](#_Toc513900003)

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). O banco de dados escolhido foi o xFalta decidir o bancox.

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.

2. BASE CONCEITUAL

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, 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.



Fonte: o autor.

Figura 01 – Elementos básicos do diagrama de caso de uso

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.



Fonte: o autor.

Figura 02 – Relacionamentos entre classes

• 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.



Fonte: SOMMERVILLE (2011).

Figura 03 – Exemplo de diagrama de atividades.

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

4.1. Organização de tarefas (Trello)

4.2. Ferramenta de controle de versão (Git)

4.3. Linguagem de programação utilizada

4.4. Banco de dados utilizado

4.5. IHC

4.6. Requisitos funcionais

4.6.1. Requisito 1

4.6.2. Requisito 2

4.6.3. Requisito 3

4.6.4. Requisito 4

4.6.5. Requisito 5

4.6.6. .........

4.7. Requisitos não funcionais

4.7.1. Requisito 1

4.7.2. Requisito 2

4.7.3. Requisito 3

4.7.4. Requisito 4

4.7.5. Requisito 5

4.7.6. ..........

4.8. Diagrama de caso de uso

4.9. Documentação caso de uso

4.9.1. Caso de uso 1

4.9.2. Caso de uso 2

4.9.3. Caso de uso 3

4.9.4. ..........

4.10. Diagrama de classe (colar diagrama de classe)

4.11. Diagrama Atividades

4.11.1. Atividade 1

4.11.2. Atividade 2

4.11.3. Atividade 3

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

4.13. Interação com usuário

4.13.1 Avaliação realizada no Workshop

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.