**ANHANGUERA EDUCACIONAL**

**sorocaba**

Análise e desenvolvimento de sistemas

**Adiel Ferreira Fontebaixa, Jeferson Santos Júnior, Júlio Cesar Correa da Costa**

**INFO TECH – ANALISE DE UM BANCO DE DADOS E CRIAÇÃO DE UM SISTEMA DE VENDAS**

**Sorocaba/SP**

**2020**

**Adiel Ferreira Fontebaixa, Jeferson Santos Júnior, Júlio César Correa da Costa**

**INFO TECH – ANALISE DE UM BANCO DE DADOS E CRIAÇÃO DE UM SISTEMA DE VENDAS**

Trabalho de conclusão de Curso apresentado ao curso de Análise e desenvolvimento Sistemas da Anhanguera Educacional como requisito parcial à obtenção do título de Graduação.

Orientador: Edson Martin Feitosa

**Sorocaba/SP**

**2020**

**Adiel Ferreira Fontebaixa, Jeferson Santos Júnior, Júlio Cesar Correa da Costa**

**INFO TECH – ANALISE DE UM BANCO DE DADOS E CRIAÇÃO DE UM SISTEMA DE VENDAS**

Trabalho de Conclusão de Curso apresentado ao curso de Análise e Desenvolvimento de Sistemas da Anhanguera como requisito parcial à obtenção do título de Graduação.

Aprovado em:<<Data de aprovação>>

BANCA EXAMINADORA:

Prof.(a) <<Titulação e nome completo do orientador>>

Faculdade Anhanguera

Prof.(a) Nome Completo do(a) Examinador(a)

Instituição a que ele(a) pertence

Prof.(a) Dr.(a) Nome Completo do(a) Examinador(a)

Instituição a que ele(a) pertence

Dedico...

**AGRADECIMENTOS**

Agradeço...

“Epígrafe”

**RESUMO**

Nesta monografia tratamos sobre diversos assuntos relacionados a banco de dados e sua arquitetura. Nosso projeto se trata de uma pesquisa aplicada onde traremos um banco de dados real utilizando tudo que aprendemos no semestre, aqui falaremos sobre o objetivo do nosso problema, tipos de banco de dados, como será utilizado o banco de dados, Modelo Entidade Relacionamento (MER) que será usado para planejar nossas tabelas e ajudará no script do nosso banco.

**Palavras-chave:** Banco. SQL. Aplicativo. Sistema. Site.

**ABSTRACT**

In this monograph we deal with several issues related to database and its architecture. Our project is an applied research where we will bring a real database using everything we learned in the semester, here we will talk about the purpose of our problem, database types, how the database will be used, Relationship Entity Model (MER) that will be used to plan our tables and will help in our bank script.

**Keywords:** Bank. SQL. Application. System. Site.

**LISTA DE ILUSTRAÇÕES**

[Figura 1 - Diagrama de Classes 27](#_Toc82283997)

[Figura 2 - Diagrama de Atividade 32](#_Toc82283998)

[Figura 3 - Modelo Entidade Relacionamento (MER) 33](#_Toc82283999)

[Figura 4 - Caso de uso de alto nível 34](#_Toc82284000)

[Figura 5 - Caso de uso de alto nível 35](#_Toc82284001)

[Figura 6 - Caso de uso de alto nível 35](#_Toc82284002)

[Figura 7- Diagrama de sequencia 37](#_Toc82284003)

[Figura 8 - Diagrama de sequencia 38](#_Toc82284004)

[Figura 9 - Identidade visual........................................................................................39](#_Identidade_visual_1)

**LISTA DE SIGLAS E ABREVIATURAS**

SQL (Structured Query Language)

ACID (Atomicidade, Consistência, Isolamento, Durabilidade)

XML (eXtensible Markup Language)

SGBD (Sistema Gerencial de Banco de Dados)

MER (Modelo Entidade-Relacionamento)

GNU/GPL (Licença Pública Geral – General Public License)

HTML (Hyper Text Markup Language)

CSS (Cascading Style Sheet)

JS (JavaScript)

**SUMÁRIO**

[Análise e desenvolvimento de sistemas 1](#_Toc82204248)

[1 INTRODUÇÃO 12](#_Toc82204249)

[2 Embasamento teórico 13](#_Toc82204250)

[2.1 Descrição do Problema 13](#_Toc82204251)

[2.2 Stackholders 13](#_Toc82204252)

[2.3 Projeto de Pesquisa 13](#_Toc82204253)

[2.4 Levantamento de Requisitos 14](#_Toc82204254)

[2.5 O que é SQL? 14](#_Toc82204255)

[2.6 Arquiteturas de Banco de Dados 14](#_Toc82204256)

[2.7 CASOS DE USO DE ALTO NIVEL 17](#_Toc82204257)

[2.8 MODELO ENTIDADE RELACIONAMENTO 17](#_Toc82204258)

[2.9 Classe DAO 18](#_Toc82204259)

[2.10 Diagrama de Classes 18](#_Toc82204260)

[2.11 Diagrama de Atividades. 18](#_Toc82204261)

[2.12 Diagrama de Sequência o que é. 21](#_Toc82204262)

[2.13 Código Fonte de Classes 21](#_Toc82204263)

[2.14 Identidade visual..............................................................................................21](#_Identidade_visual)

[2.15 Tecnologias Front end.....................................................................................21](#_Tecnologias_Front_end)

[2.16 Tecnologias Back end......................................................................................23](#_Tecnologia_back-end)

[3 Metodologia 22](#_Toc82204264)

[**3.1** **Classe DAO** 22](#_Toc82204265)

[3.2 Diagrama de Classes 23](#_Toc82204266)

[3.3 Código Fonte Diagrama de Classes 24](#_Toc82204267)

[3.4 Diagrama de Atividade 28](#_Toc82204268)

[3.5 Projeto Científico 28](#_Toc82204269)

[3.6 MODELO ENTIDADE RELACIONAMENTO 28](#_Toc82204270)

[3.7 Script banco de dados 29](#_Toc82204271)

[3.8 Casos de uso de alto nível 30](#_Toc82204272)

[3.9 Casos de uso expandido 31](#_Toc82204273)

[3.10 Diagrama de Sequência 32](#_Toc82204274)

[3.11 Tecnologia front end........................................................................................39](#_Tecnologia_Front_end)

[3.12 Identidade visual..............................................................................................40](#_Identidade_visual_1)

[4 Considerações Finais 35](#_Toc82204275)

[Referências 36](#_Referências)

# INTRODUÇÃO

Trabalho teórico dedicado a produção de um banco de dados para uma empresa de produtos eletrônico, neste trabalho abordaremos sobre a criação do banco de dados e traremos também sobre as arquiteturas de bancos de dados, as diferenças de cada uma delas para termos as qualidades e diferença de cada uma e decidirmos qual seria a melhor opção pra utilizarmos para desenvolver o banco de dados.

# Embasamento teórico

## Descrição do Problema

Vamos criar um banco de dados e um aplicativo para uma empresa de produtos eletrônicos, Mauricio dono da empresa, está se queixando que seu banco atual não está lhe servindo bem, ele precisa que o banco dele seja mais completo e eficaz, que faça controle das entregas da empresa, do estoque da empresa, o tipo do produto, a data em que o produto foi enviado para a entrega, o local de envio, os clientes que fizeram o pedido, Ele se queixa do atual banco de dados dele e nos procurou para criarmos um banco de dados melhor e mais eficaz da maneira que ele gostaria de ter.

Também vamos criar um aplicativo novo para que ele faça o gerenciamento de toda sua empresa.

## Stackholders

O gerente de logística da empresa reclamou para o dono Maurício, que o banco de dados não está funcional, a área de expedição logística não está conseguindo organizar as entregas, a área do almoxarifado não está conseguindo organizar o estoque, a área de recebimento também está muito confusa, e também a área de planejamento e controle da produção não está conseguindo organizar os clientes.

## Projeto de Pesquisa

Para conseguir resolver os problemas da empresa, vamos utilizar o SGBD (Sistema Gerencial de Banco de Dados) porque ele se comporta melhor para empresas, a arquitetura Relacional que é a que está atualmente não funciona muito bem para uma empresa que está em constante crescimento e por isso ela acaba gerando gargalos e criando todos esses problemas.

## Levantamento de Requisitos

Para este projeto vamos precisar de um modelo relacional, uma linguagem de programação e de um Sistema Gerenciador de Banco de Dados que possua os fatores ACID (atomicidade, consistência, isolamento, durabilidade).

## O que é SQL?

“SQL significa “Structured Query Language”, ou “Linguagem de Consulta Estruturada”, em português. Resumidamente, é uma linguagem de programação para lidar com banco de dados relacional (baseado em tabelas). Foi criado para que vários desenvolvedores pudessem acessar e modificar dados de uma empresa simultaneamente, de maneira descomplicada e unificada”. Segundo (Bacca, 2020)

## Arquiteturas de Banco de Dados

**Banco de Dados usando sistemas hierárquicos e de rede:**

Os modelos hierárquicos possuíam semelhante à de uma árvore, muito rígida. Caso fosse necessária a adição de uma nova informação, o banco dados em sua totalidade precisariam ser reorganizado ou redefinido. Os bancos de dados em rede assemelhavam-se com uma teia, tecida por uma aranha ou uma rede conectada de informações, porém existia uma restrição de um determinado número de relacionamentos que poderiam ser realizados entre os registros.

**Banco de dados relacional:**

Os bancos de dados relacionais começaram a surgir comercialmente a partir de 1980. Navathe e Ramez (2005) afirmam que os bancos de dados relacionais foram originalmente projetados para separar a forma de armazenamento, diferenciando o projeto físico do projeto conceitual do banco de dados. Ofereceram uma flexibilidade maior no desenvolvimento dos sistemas para os mesmos clientes da fase anterior, tornando-se rapidamente uma tendência para todos os desenvolvedores daquela época até os dias atuais e com a possibilidade de ser utilizado em uma infinidade de aplicações nos softwares, por exemplo, para servidores em grandes sistemas, pequenas empresas e em muitos sites na internet.

**Banco de dados relacional/objetos:**

Conforme pode ser observado em Date (2003), os bancos de dados relacionais/objetos são uma evolução do banco de dados relacional. Com o aparecimento das linguagens orientadas a objetos, a evolução dos bancos de dados foi naturalmente surgindo. Inicialmente, os bancos de dados orientados a objetos surgiram como uma concorrência ao relacional, mas na prática é usado com a possibilidade de utilização de recursos da orientação a objetos como herança, encapsulamento e tipos de dados abstratos, de acordo com Navathe e Ramez (2005).

**Banco de dados para web com XML:**

A internet possibilitou que muitos sistemas ficassem on-line, com sistemas de banco de dados distintos trocando informações entre si. O XML (eXtensible Markup Language ou linguagem de marcação que permite a integração de dados) possui um formato que, independentemente da plataforma que o sistema utiliza, permite a comunicação entre sistemas de banco de dados diferentes. Navathe e Ramez (2005) afirmam que a linguagem XML combina os conceitos de modelos empregados nos sistemas de documentos com os conceitos de modelos de banco de dados.

**Banco de dados para nuvem:**

A computação em nuvem está revolucionando a forma de armazenamento, de processamento dos dados e principalmente no quesito de infraestrutura, com pagamentos relacionados ao uso dos recursos disponibilizados. Os Sistema Gerencial de Banco de Dados (SGBD) estão migrando para os serviços em nuvens, diminuindo os custos com equipamentos e softwares, principalmente os e-commerce.

**Oracle é** um SGBD proprietário e sua licença precisa ser adquirida, portanto, não é freeware. É utilizado em médias e grandes empresas e foi projetado para sistemas que requerem alto desempenho e segurança. As versões atuais possuem recursos para computação em nuvem, big data (grandes volumes de informação), multiplataformas e também muitas ferramentas de administração e de desenvolvimento de aplicações que servem como interface, possibilitando mais facilidades no acesso ao banco de dados.

O SQL **Server** pertence à empresa Microsoft e possui versões gratuitas e pagas, sendo que as pagas são de valores bem inferiores ao seu principal concorrente (visto anteriormente, o Oracle). É utilizado em diversos segmentos de empresas que precisam de um SGBD estável e seguro (e claro, não tão caro!). Um dos problemas deste SGBD era sua plataforma que funcionava somente com o sistema operacional Windows. As novas versões permitem que o SGBD funcione no LINUX e em Container Docker (tecnologia que oferece um conjunto de ferramentas empacotadas e isoladas, como se estivem em um contêiner). Atualmente, o SQL Server, é um dos SGBDs mais utilizados no mercado, porém está perdendo espaço para outros que têm código aberto.

O MySQL é Open Source ou código aberto e possui licenças GNU/GPL (Licença Pública Geral – General Public License) permitindo que qualquer usuário edite o seu código fonte de forma que atenda aos requisitos de uma determinada aplicação que está sendo implementada. Atualmente, pertence ao Oracle, cujo objetivo, com a disponibilização freeware, é a fomentação do uso desta tecnologia. Sua capacidade de processamento de transações é muito grande e pode ser utilizado por grandes empresas. Um fator a ser considerado é o quesito segurança. Por ser código aberto, o cuidado com falhas de segurança deve ser redobrado. O MySQL possui uma versão bem mais robusta, porém paga, com diversos recursos avançados, como auditoria, firewalls, monitoramento e backups avançados.

**Postgree** é um SGBD muito utilizado por rodar em várias plataformas de desenvolvimento como código aberto (Open Source) e de desenvolvimento livre. É usado em sistemas mais robustos, em que a base de dados é muito grande, em empresas

(Werlich, pp. 17,18,19)

## CASOS DE USO DE ALTO NIVEL

Segundo (Vieira, 2020)

O diagrama de Casos de Uso auxilia no levantamento dos requisitos funcionais do sistema, descrevendo um conjunto de funcionalidades do sistema e suas interações com elementos externos e entre si.

Ou seja, o diagrama de Casos de Uso ajuda nos projetos servindo como uma tabela que demonstra tudo que será necessário para que o projeto atinja todos os objetivos com seu usuário.

## MODELO ENTIDADE RELACIONAMENTO

Segundo (Rodrigues, 2020)

O Modelo Entidade Relacionamento (também chamado Modelo ER, ou simplesmente MER), como o nome sugere, é um modelo conceitual utilizado na Engenharia de Software para descrever os objetos (entidades) envolvidos em um domínio de negócios, com suas características (atributos) e como elas se relacionam entre si (relacionamentos).

Em geral, este modelo representa de forma abstrata a estrutura que possuirá o banco de dados da aplicação. Obviamente, o banco de dados poderá conter várias outras entidades, tais como chaves e tabelas intermediárias, que podem só fazer sentido no contexto de bases de dados relacionais.

O princípio para criar um MER, é ter um projeto descrito e nessa descrição de projeto, questionar, e assim começar a criar o modelo conceitual tirando características dessa descrição através de perguntas como: “São vários ou só um?”, “É um objeto ou não?”, entre outras perguntas possíveis que agregariam ao modelo. Após realizar essas perguntas, você obterá as entidades e os atributos nas respostas delas, sendo uma entidade um objeto e algo único, e um atributo algo abstrato e diverso.

## Classe DAO

Em resumo, DAO é um padrão de projetos onde um objeto:

* provê uma interface que abstrai o acesso a dados;
* lê e grava a partir da origem de dados (banco de dados, arquivo, memória, etc.);
* encapsula o acesso aos dados, de forma que as demais classes não precisam saber sobre isso. (Quesado, 2018)

## Diagrama de Classes

Segundo (Microsoft, 2021) o que você pode fazer com diagramas de classe.

Projetar, Visualizar e refatorar classes e outros tipos em seu código com Designer de classe no Visual Studio. Use diagramas de classe para criar e editar classes em seu projeto C#, Visual Basic ou C++. Você também pode usar diagramas de classe para entender melhor a estrutura do projeto ou reorganizar o código.

* **Criar**: edite o código do projeto editando o diagrama de classe. Adicione novos elementos e exclua os que não deseja mais. As alterações serão refletidas no código.
* **Visualizar**: entenda a estrutura do projeto exibindo as classes do projeto em um diagrama. Personalize o diagrama para que você possa se concentrar nos detalhes do projeto mais importantes para você. Salve o diagrama para usar posteriormente na demonstração ou documentação.
* **Refatorar**: substitua métodos, renomeie identificadores, refatore parâmetros e implemente interfaces e classes abstratas.

## Diagrama de Atividades.

Segundo (Inc., 2020)

Um diagrama de atividade é essencialmente um fluxograma que mostra as atividades executadas por um sistema.

Antes de criar um diagrama de atividade, é necessário primeiro entender sua composição. Alguns dos componentes mais comuns de um diagrama de atividade incluem:

* **Ações:** uma etapa da atividade em que o usuário ou software realiza uma determinada tarefa. No Lucidchart, ações são simbolizadas por retângulos de cantos arredondados.
* **Nó de decisão:** um ramo condicional no fluxo representado por um diamante. Inclui uma única entrada e duas ou mais saídas.
* **Fluxos de controle:** outro nome dado aos conectores que mostram o fluxo entre as etapas no diagrama.
* **Nó inicial:** simboliza o início da atividade. É representado por um círculo preto.
* **Nó final:** representa a etapa final da atividade. É representado por um círculo preto delineado.

|  |  |  |
| --- | --- | --- |
| símbolo de início | Símbolo de início | Representa o começo de um processo ou fluxo de trabalho em um diagrama de atividade. Ele pode ser usado por si só ou com um símbolo de nota que explica o ponto de partida. |
| símbolo de atividade | Símbolo de atividade | Indica as atividades que compõem um processo modelado. Estes símbolos, que incluem descrições breves dentro da forma, são os principais componentes de um diagrama de atividade. |
| Símbolo de conector | Símbolo de conector | Mostra o fluxo de direção, ou fluxo de controle, da atividade. Uma seta de entrada inicia um passo de uma atividade. Uma vez concluído o passo, o fluxo continua com a seta de saída. |
| símbolo de junta | Símbolo de junta / Barra de sincronização | Combina duas atividades simultâneas e as reintroduz em um fluxo onde apenas uma atividade ocorre por vez. Representado por uma linha espessa vertical ou horizontal. |
| símbolo de garfo | Símbolo de garfo | Divide um único fluxo de atividade em duas atividades simultâneas. É simbolizado com várias linhas com setas de uma junção. |
| Símbolo de decisão | Símbolo de decisão | Representa uma decisão e sempre tem pelo menos dois caminhos ramificados e com texto de condição, permitindo aos usuários visualizarem opções. Este símbolo representa a ramificação ou fusão de diferentes fluxos, com o símbolo atuando como um quadro ou contêiner. |
| Símbolo de nota | Símbolo de nota | Permite aos criadores ou colaboradores do diagrama comunicar mensagens adicionais que não se encaixam dentro do próprio diagrama. Deixe observações para uma maior clareza e especificação. |
| símbolo de enviar sinal | Símbolo de enviar sinal | Indica que um sinal está sendo enviado a uma atividade recebedora. |
| símbolo de receber sinal | Símbolo de receber sinal | Demonstra a aceitação de um evento. Após o evento ser recebido, o fluxo que vem desta ação é concluído. |
| símbolo de história rasa pseudoestado | Símbolo de história rasa pseudoestado | Representa uma transição que invoca o último estado ativo. |
| símbolo de opção em loop | Símbolo de opção em loop | Permite ao criador modelar uma sequência repetitiva dentro do símbolo de opção em loop. |
| símbolo de final de fluxo | Símbolo de final de fluxo | Representa o final de um fluxo de processos específicos. Este símbolo não deve representar o fim de todos os fluxos em uma atividade. Nesse caso, use o símbolo de término. O símbolo final do fluxo deve ser colocado no final de um processo em um fluxo único de atividade. |
| texto de condição | Texto de condição | É colocado ao lado de um marcador de decisão para avisar em qual condição um fluxo de atividade deve se separar nesse sentido. |
| símbolo de término | Símbolo de término | Marca o estado final de uma atividade e representa a conclusão de todos os fluxos de um processo. |

## **Diagrama de Sequência o que é**.

Segundo (Ventura, 2018)

Diagrama de Sequência é um diagrama usado em UML (*Unified Modeling Language*), representando a sequência de processos (mais especificamente, de mensagens passadas entre objetos) num programa de computador. Demonstra de forma ilustrativa a sequência de um processo de forma simples e lógica. Com um diagrama de sequência fica fácil de forma ilustrativa e lógica você perceber como funciona um processo de mensagens passadas entre objetos.

O diagrama de sequência dá ênfase a ordenação temporal em que as mensagens são trocadas entre os objetos de um sistema. Entende-se por mensagens os serviços solicitados de um objeto a outro, e as respostas desenvolvidas para as solicitações.

## Código Fonte de Classes

O código fonte é encarregado de armazenar o agrupamento de instruções necessárias para o funcionamento do software neste caso este é um código fonte de uma classe de um diagrama de Classe.

## Identidade visual

A identidade visual é algo muito importante a se analisar na criação de um sistema, sites ou em outros tipos de produtos, pois ela é quem diferencia e destaca produtos dentre o meio de diversos produtos.

Segundo (Ramos, 2019) A identidade visual é uma construção de vários elementos gráficos e visuais responsáveis por criar uma atmosfera a respeito de quem é a empresa, quais são seus valores e até mesmo como ela vê o mundo e a sociedade.

Alguns dos principais elementos que compõem o desenvolvimento de uma identidade visual são a logomarca, tipografia, paleta de cores, materiais de divulgação (flyers, cartazes e outdoors, por exemplo) e, mais recentemente, elementos das redes sociais como imagens dos posts, avatar e a capa.

## Tecnologias Front end

Existem diversas tecnologias que te ajudam no desenvolvimento front end, aqui citaremos algumas delas:

### **Backbone.js**

O backbone.js fornece uma estrutura para aplicativos web provindo modelos com vinculação de key-value e custom events, ele também conta com uma coleção de APIs que contém inúmeras funções.

Exemplos de sistemas utilizando essa estrutura são o Airbnb, Coursera e o Trello.

(Tableless, 2017)

### **React**

O React é uma biblioteca nova do Facebook que permite uma abordagem para a criação de dynamic UIs, manutenção de estado e modularização de grandes e complexos aplicativos de JavaScript.

Exemplos de sistemas utilizando essa biblioteca são o Tumblr, Instagram e Uber.

(Tableless, 2017)

### **Bootstrap**

O Bootstrap é um framework HTML, CSS e JS popular pela capacidade de auxiliar no desenvolvimento responsivo de projetos mobile first na web.

Exemplos de sistemas utilizando o Bootstrap são o Twitter, Udacity e DataMiner.

(Tableless, 2017)

### **AngularJS**

O HTML é ótimo para declarar documentos estáticos, mas ele deixa a desejar quando usado para declarar dynamic views em aplicações web. O AngularJS permite expandir o vocabulário HTML para sua aplicação resultando em um ambiente expressivo, legível e rápido para desenvolver.

Exemplos de sistemas utilizando AngularJS são o Wix, The Guardian, SmartAnalytics.

(Tableless, 2017)

### **jQuery**

O jQuery trata-se de uma biblioteca JS rápida, pequena e rica em recursos. Ela torna as coisas como travessia e manipulação de documentos HTML, manuseio de eventos, animação e Ajax muito mais simples com uma API fácil de usar que funciona em uma infinidade de navegadores.

Exemplos de sistemas utilizando jQuery são o Snipcart, WorkFlowy e Playo.

(Tableless, 2017)

## Tecnologia back-end

O que é back-end?

Essa forma de desenvolvimento se relaciona com o que está por trás das aplicações desenvolvidas na programação. Ou seja, tudo que dá estrutura e apoio às ações do usuário da máquina é chamado de back-end.

Quando acessamos um site, por exemplo, por trás de toda sua apresentação amigável esteticamente, há uma comunicação das informações trocadas entre banco de dados e navegador. Portanto, por trás da interface gráfica do realizador, o back-end está sempre agindo.

Um servidor é o responsável por fornecer serviços aos computadores. Os bancos de dados, por sua vez, são como uma enorme fonte de informações, e é por meio deles que as aplicações se tornam dinâmicas.

Ao fazer uma pesquisa, o resultado é trazido de acordo com a busca no banco de dados. Podemos dizer então que esse centro de informações é acessado de maneira indireta, a partir de uma aplicação externa.

O desenvolvimento da linguagem back-end passa por diversas etapas, desde o planejamento até a construção, a implantação e os cuidados com a manutenção da estrutura que suporta a aplicação.

Como o back-end é considerado o centro nervoso de qualquer site, suas tarefas podem até parecer menos palpável, uma vez que não se relacionam diretamente com a parte de design e jamais são vistas pelo usuário externo.

Portanto, é indicado conhecer linguagens de programação como Java, TypeScript, Node.js, PHP, Python, etc. Além disso, possíveis problemas devem ser antecipados e evitados antes mesmo que aconteçam. (TOTVS, 2020)

### **Exemplos de tecnologias back-end**

### Java - Java é uma linguagem de programação e plataforma computacional lançada pela primeira vez pela Sun Microsystems em 1995. Existem muitas aplicações e sites que não funcionarão, a menos que você tenha o Java instalado, e mais desses são criados todos os dias. O Java é rápido, seguro e confiável. De laptops a datacenters, consoles de games a supercomputadores científicos, telefones celulares à Internet, o Java está em todos os lugares! (java, 2021)

### **TypeScript** - TypeScript é um superset para JavaScript, ou um conjunto adicional de instruções, keywords e estruturas, criado pela Microsoft.

Ou seja, você ainda estará programando JavaScript, mas com “super poderes”.

E que super poderes são esses?

O primeiro, mais direto e mais simples é a tipagem estática, muito comum em linguagens como Java, C e derivados, mas algo que passa longe do JavaScript geralmente.

Mas TypeScript vai muito além disso, adicionando várias outras validações, oferecendo suporte a Generics, interfaces e muito mais, além de estender enormemente as possibilidades do uso de [classes em JS](https://www.luiztools.com.br/post/como-criar-classes-em-javascript-es6-e-node-js/).

Note que todos estes benefícios são em tempo de codificação, pois em produção, você terá o bom e velho JS sendo transpilado a partir do TypeScript novamente. (Duarte, Luiz Tools, 2020)

### **Node.js** - Node.js não é uma linguagem de programação. Você programa utilizando a linguagem JavaScript, a mesma usada há décadas no client-side das aplicações web. Javascript é uma linguagem de scripting interpretada, embora seu uso com Node.js guarde semelhanças com linguagens compiladas, uma vez que máquina virtual V8 (veja mais adiante) faz etapas de pré-compilação e otimização antes do código entrar em operação. (Duarte, Luiz Tools, 2020)

### **PHP** - O PHP (um acrônimo recursivo para PHP: Hypertext Preprocessor) é uma linguagem de script open source de uso geral, muito utilizada, e especialmente adequada para o desenvolvimento web e que pode ser embutida dentro do HTML. (PHP, 2020)

# Metodologia

## Classe DAO

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ProjetoIntegrado

{

    class ConexaoFuncionario

    {

        SqlConnection conn;

        public ConexaoFuncionario()

        {

            conn = new SqlConnection(@"Data Source=DESKTOP-4CDHGQH\SQLEXPRESS;Initial Catalog=Projeto\_Integrado;User ID=sa;Password=123456");

        }

        public List<Funcionario> Listar()

        {

            List<Funcionario> lstfunc = new List<Funcionario>();

            SqlCommand listar = new SqlCommand("select \* from Funcionario", conn);

            listar.CommandType = System.Data.CommandType.Text;

            conn.Open();

            SqlDataReader dr = listar.ExecuteReader();

            while (dr.Read())

            {

                Funcionario fu = new Funcionario();

                fu.cod\_funcionario = Convert.ToInt32(dr["cod\_funcionario"]);

                fu.Nome = dr["nome"].ToString();

                fu.CPF = dr["cpf"].ToString();

                fu.Setor = dr["setor"].ToString();

                lstfunc.Add(fu);

            }

            conn.Close();

            return lstfunc;

        }

        public void InserirFuncionario(string Nome, string Senha, string CPF,string Setor)

        {

            SqlCommand cmd = new SqlCommand("insert into Funcionario values (@nome,@cpf,@setor,@senha)", conn);

            cmd.CommandType = System.Data.CommandType.Text;

            cmd.Parameters.Add(new SqlParameter("@nome", System.Data.SqlDbType.VarChar).Value = Nome);

            cmd.Parameters.Add(new SqlParameter("@cpf", System.Data.SqlDbType.VarChar).Value = CPF);

            cmd.Parameters.Add(new SqlParameter("@setor", System.Data.SqlDbType.VarChar).Value = Setor);

            cmd.Parameters.Add(new SqlParameter("@senha", System.Data.SqlDbType.VarChar).Value = Senha);

            conn.Open();

            cmd.ExecuteNonQuery();

            conn.Close();

        }

    }

}

## Diagrama de Classes

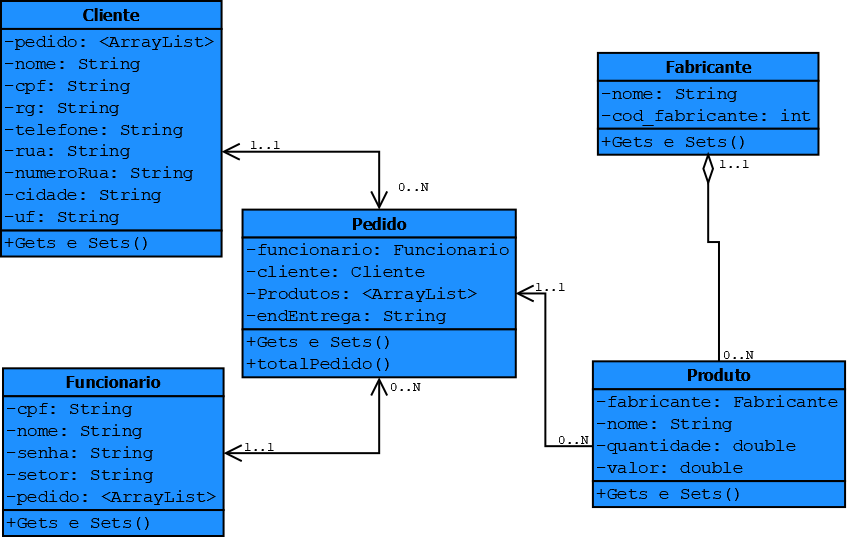


Figura 1 - Diagrama de Classes

## Código Fonte Diagrama de Classes

**Classe Pedido**

package ProjetoIntegrado;

import java.util.ArrayList;

public class Pedido {

private Funcionario funcionario;

private Cliente cliente;

private String endEntrega;

private ArrayList <Produto> produtos = new ArrayList<>();

public **Pedido**(){}

public **Pedido**(Funcionario funcionario, Cliente cliente, String endEntrega) {

this.funcionario = funcionario;

this.cliente = cliente;

this.endEntrega = endEntrega;

}

public Funcionario **getFuncionario**() {

return funcionario;

}

public void **setFuncionario**(Funcionario funcionario) {

this.funcionario = funcionario;

}

public Cliente **getCliente**() {

return cliente;

}

public void **setCliente**(Cliente cliente) {

this.cliente = cliente;

}

public String **getEndEntrega**() {

return endEntrega;

}

public void **setEndEntrega**(String endEntrega) {

this.endEntrega = endEntrega;

}

public ArrayList<Produto> **getProdutos**() {

return produtos;

}

public void **setProdutos**(ArrayList<Produto> produtos) {

this.produtos = produtos;

}

public void setProdutos(Produto produto){

this.produtos.add(produto);

}

public double **totalPedid**o(){

double valorTotal = 0;

for (Produto produto : produtos) {

valorTotal += produto.getQuantidade() \* produto.getValor();

}

return valorTotal;

}

}

**Classe Produto**

package ProjetoIntegrado;

public class Produto {

private Fabricante fabricante;

private String nome;

private double quantidade;

private double valor;

public **Produto**(){}

public **Produto**(Fabricante fabricante, String nome, double quantidade, double valor) {

this.fabricante = fabricante;

this.nome = nome;

this.quantidade = quantidade;

this.valor = valor;

}

public Fabricante **getFabricante**() {

return fabricante;

}

public void **setFabricante**(Fabricante fabricante) {

this.fabricante = fabricante;

}

public String **getNome**() {

return nome;

}

public void **setNome**(String nome) {

this.nome = nome;

}

public double **getQuantidade**() {

return quantidade;

}

public void **setQuantidade**(double quantidade) {

this.quantidade = quantidade;

}

public double **getValor**() {

return valor;

}

public void **setValor**(double valor) {

this.valor = valor;

}

}

**Classe Fabricante**

package ProjetoIntegrado;

import java.util.ArrayList;

public class Fabricante {

private String nome;

private int codFabricante;

//private ArrayList<Produto> produtos = new ArrayList<>();

public **Fabricante**(String nome, int codFabricante) {

this.nome = nome;

this.codFabricante = codFabricante;

}

public **Fabricante**(){}

public String **getNome**() {

return nome;

}

public void **setNome**(String nome) {

this.nome = nome;

}

public int **getCodFabricante**() {

return codFabricante;

}

public void **setCodFabricante**(int codFabricante) {

this.codFabricante = codFabricante;

}

// public ArrayList<Produto> **getProdutos**() {;

// return produtos;

// }

//

// public void **setProdutos**(ArrayList<Produto> produtos) {

// this.produtos = produtos;

// }

// public void **setProdutos**(Produto produto) {

// this.produtos.add(produto);

// }

}

**Classe Funcionario**

package ProjetoIntegrado;

import java.util.ArrayList;

public class Funcionario {

private String nome;

private String cpf;

private String senha;

private String setor;

private ArrayList<Pedido> pedidos = new ArrayList<>();

public **Funcionario**(){}

public **Funcionario**(String nome, String cpf, String senha, String setor) {

this.nome = nome;

this.cpf = cpf;

this.senha = senha;

this.setor = setor;

}

public String **getNome**() {

return nome;

}

public void **setNome**(String nome) {

this.nome = nome;

}

public String **getCpf**() {

return cpf;

}

public void **setCpf(**String cpf) {

this.cpf = cpf;

}

public String **getSenha**() {

return senha;

}

public void **setSenha**(String senha) {

this.senha = senha;

}

public String **getSetor**() {

return setor;

}

public void **setSetor**(String setor) {

this.setor = setor;

}

public ArrayList<Pedido> **getPedidos**() {

return pedidos;

}

public void **setPedidos**(ArrayList<Pedido> pedidos) {

this.pedidos = pedidos;

}

public void **setPedidos**(Pedido pedido){

this.pedidos.add(pedido);

}

}

**Classe Cliente**

package ProjetoIntegrado;

import java.util.ArrayList;

public class Cliente {

private String nome;

private String cpf;

private String rg;

private String telefone;

private String rua;

private String numeroRua;

private String cidade;

private String uf;

private ArrayList<Pedido> pedidos = new ArrayList<>();

public **Cliente**(){}

public **Cliente**(String nome, String cpf, String rg, String telefone, String rua, String numeroRua, String cidade, String uf) {

this.nome = nome;

this.cpf = cpf;

this.rg = rg;

this.telefone = telefone;

this.rua = rua;

this.numeroRua = numeroRua;

this.cidade = cidade;

this.uf = uf;

}

public String **getNome**() {

return nome;

}

public void **setNome**(String nome) {

this.nome = nome;

}

public String **getCpf**() {

return cpf;

}

public void **setCpf**(String cpf) {

this.cpf = cpf;

}

public String **getRg**() {

return rg;

}

public void **setRg**(String rg) {

this.rg = rg;

}

public String **getTelefone**() {

return telefone;

}

public void **setTelefone**(String telefone) {

this.telefone = telefone;

}

public String **getRua**() {

return rua;

}

public void **setRua**(String rua) {

this.rua = rua;

}

public String **getNumeroRua**() {

return numeroRua;

}

public void **setNumeroRua**(String numeroRua) {

this.numeroRua = numeroRua;

}

public String **getCidade**() {

return cidade;

}

public void **setCidade**(String cidade) {

this.cidade = cidade;

}

public String **getUf**() {

return uf;

}

public void **setUf**(String uf) {

this.uf = uf;

}

public ArrayList<Pedido> **getPedidos**() {

return pedidos;

}

public void **setPedidos**(ArrayList<Pedido> pedidos) {

this.pedidos = pedidos;

}

public void **setPedidos**(Pedido pedido){

this.pedidos.add(pedido);

}

public double valortotalPedidos(){

double valortotal = 0;

for (Pedido pedido : pedidos) {

valortotal += pedido.totalPedido();

}

return valortotal;

}

}

## Diagrama de Atividade

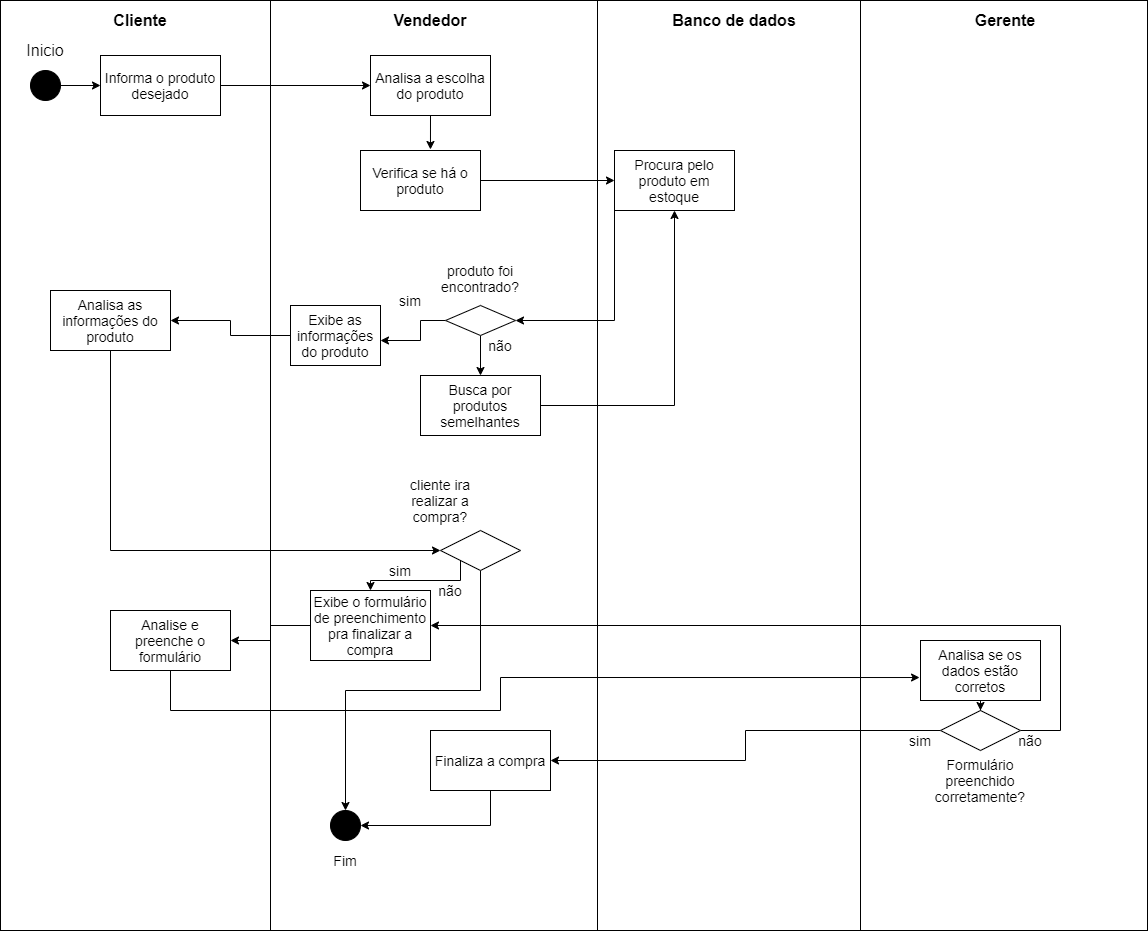


Figura 2 - Diagrama de Atividade

## Projeto Científico

O trabalho trata-se de uma pesquisa aplicada de abordagem descritiva com utilização de métodos e procedimentos em SQL e modelos ER, cujo objetivo geral é criar um banco de dados para um aplicativo

## MODELO ENTIDADE RELACIONAMENTO

Para a Criação do Banco de dados precisamos de um Modelo de Entidade Relacional (MER):



Figura 3 - Modelo Entidade Relacionamento (MER)

## Script banco de dados

E precisamos criar um script de banco de dados:

create database Projeto\_Integrado

use Projeto\_Integrado

create table Cliente

(

cod\_cliente int identity(1,1) primary key,

rg varchar(10) not null,

nome varchar(50) not null,

cpf varchar (18) not null,

rua varchar (150) not null,

numero\_rua varchar (20) not null,

cidade varchar (50) not null,

uf char (2),

telefone varchar (15),

)

create table Funcionario

(

cod\_funcionario int identity (1,1) primary key,

nome varchar(50) not null,

cpf varchar(18) not null,

setor varchar(15) not null,

senha varchar (11) not null,

)

create table Pedido

(

cod\_pedido int identity (1,1) primary key,

datasaida datetime not null,

endereco\_entrega varchar (150) not null,

produto varchar (50) not null,

quantidade varchar (20) not null,

destinatario varchar (50) not null,

cod\_cliente int references Cliente(cod\_cliente) not null,

cod\_funcionario int references Funcionario (cod\_funcionario) not null,

)

create table Produto

(

cod\_produto int identity (1,1) primary key,

nome varchar (50) not null,

quantidade varchar (20) not null,

fabricante varchar (50) not null,

valor numeric not null,

)

create table Produto\_pedido

(

cod\_produto int references Produto (cod\_produto) not null,

cod\_pedido int references Pedido (cod\_pedido) not null,

quantidade float not null,

valor float not null,

)

## Casos de uso de alto nível

Com a arquitetura Relacional o Gerente Logístico realizará todo o processo de expedição de mercadorias, não terá mais lentidão quando utilizar o banco de dados.



Figura 4 - Caso de uso de alto nível

A equipe de almoxarifado terá mais organização e eficácia dando mais rapidez e ganho de tempo no controle de estoque, recebimento de mercadorias e anotações.

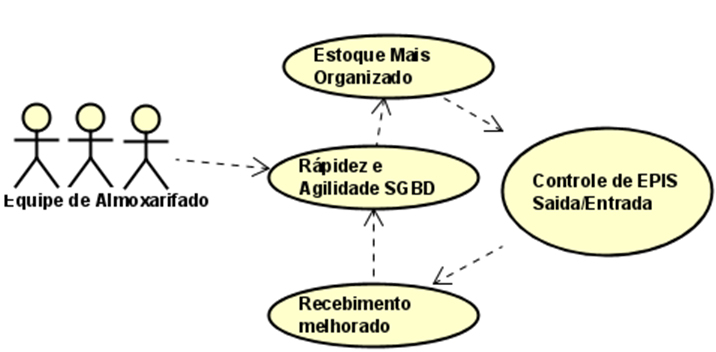


Figura 5 - Caso de uso de alto nível

A equipe de processo de produção também terá mais facilidade ao encontrar projetos, clientes e separar os produtos para qual cliente. Sem se quer alguma lentidão, tendo nossa arquitetura SGBD implantada na sua empresa.

Figura 6 - Caso de uso de alto nível

## Casos de uso expandido

Entregas

1. [IN] O profissional de expedição entra no sistema;

2. [OUT] o sistema informa os produtos disponíveis no estoque;

3. [IN] O profissional de expedição escolhe o produto;

4. [OUT] O sistema informa a quantidade disponível do produto e um campo para endereço e hora de saída do produto;

5. [IN] O profissional de expedição escolhe a quantidade e o endereço para entrega do produto;

6. [OUT] O sistema cadastra e processa a informação gerada pelo profissional de expedição;

Estoque

1. [IN] O almoxarifado entra no sistema de estoque;

2. [OUT] O sistema mostra as categorias do estoque, produtos recém chegado, recém saído, disponíveis para venda, produtos em falta;

3. [IN] O almoxarifado escolhe a categoria;

4. [OUT] o sistema mostra os produtos da categoria

5. [IN] O almoxarifado, indica para o sistema o que está faltando no estoque;

6. [OUT] O sistema grava as informações

Receber produtos

1. [IN] O profissional de recebimento, recebe o produto e vai registrar no sistema;

2. [OUT] O sistema mostra um campo de quantidade e nome do produto;

3. [IN] O profissional coloca o nome e a quantidade do produto;

4. [OUT] O sistema grava a informação em estoque;

## Diagrama de Sequência

Nesta imagem explicando na prática como funciona o diagrama de sequência no sistema.

O funcionário entra no sistema, escolhe o produto e o sistema guarda as informações



Figura 7- Diagrama de sequencia

Como podemos ver na imagem o funcionário da Expedição está escolhendo um produto, o sistema verifica a quantidade disponível em estoque e cadastra a informação gerada pelo funcionário.

O processo da expedição é finalizado com o produto despachado para a entrega, após a sequência de processos como mostra no diagrama.

O processo de sequência do Estoque não muda, o profissional entra no sistema do estoque. O sistema informa os produtos que recém chegaram ao estoque, os que recém saíram do estoque.

O profissional indica ao sistema qual o produto que está faltando no estoque, o sistema grava em seu banco de dados as informações.

O profissional do recebimento recebe os produtos e faz o registro no sistema, o sistema mostra em um campo a quantidade e tipo do produto a ser recebido. O profissional coloca o nome e a quantidade do produto, o sistema guarda as informações.



Figura 8 - Diagrama de sequencia

## Tecnologia Front end

Em nosso projeto optamos por utilizar o AngularJS como nossa tecnologia front end pois nos familiarizamos mais com essa tecnologia e achamos que ela seria a melhor opção pra usarmos no projeto.

## Identidade visual

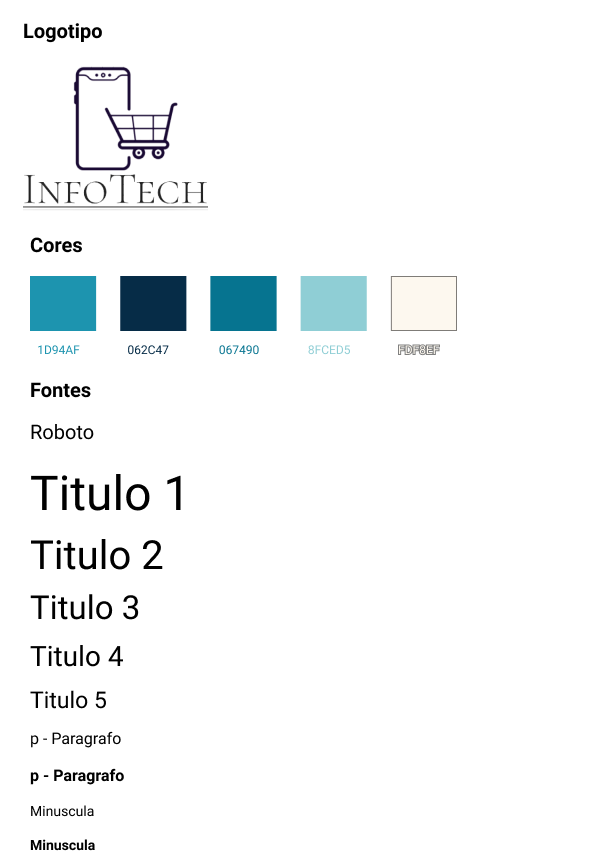


Figura 9 – Identidade visual

# Considerações Finais

Nesta pesquisa com o que aprendemos no segundo semestre de Análise e Desenvolvimento de Sistemas conseguimos criar, um caso de uso expandido, telas de prototipagem e um script de banco de dados, com isso conseguimos criar um banco de dados para um futuro aplicativo.

# Referências

Bacca, C. C. (2020).

Duarte, L. (14 de Outubro de 2020). *Luiz Tools*. Fonte: https://www.luiztools.com.br/post/como-usar-typescript-com-node-js/?utm\_source=google&utm\_medium=cpc&utm\_campaign=12231680300&utm\_content=115851466463&utm\_term=como%20usar%20typescript&gclid=Cj0KCQjwtMCKBhDAARIsAG-2Eu9f3S7TttDu9xn\_s20\_9kD28Tyl3MVqZ-HMyRMT

Duarte, L. (28 de Março de 2020). *Luiz Tools*. Fonte: https://www.luiztools.com.br/post/o-que-e-nodejs-e-outras-5-duvidas-fundamentais/?utm\_source=google&utm\_medium=cpc&utm\_campaign=902557428&utm\_content=127000463008&utm\_term=para%20que%20serve%20node%20js&gclid=Cj0KCQjwtMCKBhDAARIsAG-2Eu8w98AwRUlWHODmNosL2r

Inc., L. S. (2020). *LucidChart*. Fonte: Lucid Software Inc.: https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-atividades-uml

java. (26 de Setembro de 2021). *java*. Fonte: https://www.java.com/pt-BR/download/help/whatis\_java.html

Microsoft. (2021). *docs.microsoft*. Fonte: microsoft: https://docs.microsoft.com/pt-br/visualstudio/ide/class-designer/designing-and-viewing-classes-and-types?view=vs-2019

PHP. (26 de Setembro de 2020). *PHP*. Fonte: https://www.php.net/manual/pt\_BR/intro-whatis.php

Quesado, J. (8 de Março de 2018). *stackOverflow*. Fonte: stackOverflow: https://pt.stackoverflow.com/questions/113840/como-funciona-o-padr%C3%A3o-dao

Ramos, A. J. (18 de Setembro de 2019). *O que é identidade visual e quais as diferenças para o branding.* Fonte: Rockcontent: https://rockcontent.com/br/blog/identidade-visual/

Rodrigues, J. (2020).

Tableless. (14 de Novembro de 2017). *QUAIS SÃO AS PRINCIPAIS TECNOLOGIAS DE FRONT-END UTILIZADAS PELAS STARTUPS.* Fonte: Tableless: https://tableless.com.br/quais-sao-as-principais-tecnologias-front-end-utilizadas-pelas-startups/

TOTVS. (18 de Maio de 2020). *TOTVS*. Fonte: TOTVS: https://www.totvs.com/blog/developers/back-end/

Ventura, P. (4 de Outubro de 2018).

Vieira, R. (2020).

Werlich, C. (s.d.). Modelagem de Dados.