



FACULDADE METROPOLITANA DA GRANDE FORTALEZA
SISTEMA DE INFORMAÇÃO E ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

WILSON COSTA DE CASTRO (1-2021221251)
LUIS RICARDO ALVES SANTOS (1-2022121904)
CARLOS ENRIQUE DA COSTA MONTEIRO (1-2021120422)

**AVALIAÇÃO PRÁTICA SUPERVISIONADA - SISTEMA DE VENDA DE
PRODUTOS**

FORTALEZA
2022

WILSON COSTA DE CASTRO (1-2021221251)
LUIS RICARDO ALVES SANTOS (1-2022121904)
CARLOS ENRIQUE DA COSTA MONTEIRO(1-2021120422)

AVALIAÇÃO PRÁTICA SUPERVISIONADA - SISTEMA DE VENDA DE PRODUTOS

Avaliação prática supervisionada sobre conceitos de banco de dados, do curso de Sistema de Informação e Análise e Desenvolvimento de Sistemas da Faculdade Metropolitana da Grande Fortaleza – FAMETRO – como requisito parcial para aprovação na disciplina.

Orientador prof.º Eduardo Julião Máximo.

FORTALEZA

2022

LISTA DE ILUSTRAÇÕES

Figura 1 –	Script de criação da tabela de usuários	12
Figura 2 –	Script de inserção de três registros na tabela de usuários	12
Figura 3 –	Script de listagem de todos usuários e atributos na tabela de usuários	13
Figura 4 –	Tabela resposta do script apresentado na FIGURA 3	13
Figura 5 –	Script de alteração do campo “isAtivo” dos usuários que sejam da unidade federativa “RJ”	13
Figura 6 –	Listagem das colunas respectivas ao nome e unidade federativa dos usuários ativos	14
Figura 7 –	Script de criação da tabela de clientes	15
Figura 8 –	Script de inserção de um registro na tabela de clientes	15
Figura 9 –	Script de listagem de todos as colunas de clientes trazendo informações de usuário	16
Figura 10 –	Resposta do Script da FIGURA 9	16
Figura 11 –	Script de criação da tabela de funcionários	17
Figura 12 –	Script de inserção de funcionários	18
Figura 13 –	Script de listagem de funcionários	18
Figura 14 –	Resposta do script de listagem anterior	18
Figura 15 –	Comando de criação da tabela de categoria	20
Figura 16 –	Inserção de algumas categorias na tabela de categorias	20
Figura 17 –	Comando de criação da tabela de produtos	21
Figura 18 –	Script de inserção produtos na tabela de produtos	23
Figura 19 –	Script de criação da tabela de estoque	23
Figura 20 –	Script de inserção de dois registros na tabela de estoque	24
Figura 21 –	Script de listagem de todos os produtos e seus campos com as informações do seu estoque	24
Figura 22 –	Script de criação da tabela de pedidos	25
Figura 23 –	Ilustração do relacionamento entre funcionários e estoque	26
Figura 25 –	Demonstração do relacionamento de cliente e produtos	27

SUMÁRIO

1	INTRODUÇÃO	05
1.1	Tema	06
1.2	Objetivos	06
2	TRATAMENTO DA INFORMAÇÃO	06
2.1	SOBRE AS TECNOLOGIAS ESCOLHIDAS	07
2.1.1	Linguagem SQL	07
2.1.2	Ferramenta MySQL	07
2.2	MODELO DE NEGÓCIOS ADOTADO	08
2.2.1	Base do modelo de negócio escolhido	08
2.2.2	Regras de negócio gerais	09
2.3	ENTIDADES ENVOLVIDAS	09
2.3.1	Usuários	09
2.3.1.1	Definição	10
2.3.1.2	Atributos:	10
2.3.1.3	Manuseio e representação de responsabilidade em SQL	12
2.3.2	Clientes	14
2.3.2.1	Definição	14
2.3.2.2	Atributos	14
2.3.2.3	Manuseio e representação de responsabilidade em SQL	14
2.3.3	Funcionários	16
2.3.3.1	Definição	16
2.3.3.2	Atributos	17
2.3.3.3	Manuseio e representação de responsabilidade em SQL	17
2.3.4	Produto	18
2.3.4.1	Definição	18
2.3.4.2	Atributos	19
2.3.4.3	Relacionamento Categoria do Produto	19
2.3.4.4	Manuseio e representação de responsabilidade em SQL	20
2.3.5	Estoque	21
2.3.5.1	Definição	21
2.3.5.2	Atributos	22
2.3.5.3	Manuseio e representação de responsabilidade em SQL	22

2.3.6 Pedido	24
2.3.6.1 Definição	24
2.3.6.2 Atributos	25
2.3.6.3 Manuseio e representação de responsabilidade em SQL	25
2.4 RELACIONAMENTOS ENTIDADES	25
2.4.1 Estoque: envolvendo Produtos e Funcionários	25
2.4.1.1 Definição	25
2.4.1.2 Ilustração	26
2.4.2 Pedidos: envolvendo Clientes e Produtos	27
2.4.2.1 Definição	27
2.4.1.2 Ilustração	27
3 METODOLOGIA	28
4 RESULTADOS E DISCUSSÃO	29
5 CONSIDERAÇÕES FINAIS	30
REFERÊNCIAS	31
ANEXOS	32

1 INTRODUÇÃO

Tal documento tem por finalidade transpor todo o processo de desenvolvimento de uma base de dados que porta todos os principais conceitos do ensino de banco de dados. Visto que, a fim de aprovação na disciplina de Fundamentos de Banco de Dados, foi-se dado o objetivo de construir uma base de dados que abordasse modelos de negócios reais cotidianos. Como principais premissas, tinha-se a escolha de um tema para compor o trabalho e a escolha de uma linguagem de programação para a criação do banco de dados em nível de tecnologia. Além disso, foi requisitada a criação deste documento visando o registro descritivo de todas as fases de desenvolvimento e especificações e explicações das regras de negócio e composição do banco de dados.

Em um primeiro momento, foi feita a composição do time, por volta do dia treze de outubro do ano de criação deste trabalho; e em seguida houve a discussão da equipe sobre qual linguagem seria empregada para programar o banco de dados. A equipe no fim decidiu optar pelo SQL e como ferramenta o MySQL, posto a proximidade de todos com ela e também sua facilidade de uso. Depois da seleção da linguagem veio a seleção do tema, seguindo as mesmas premissas da escolha da linguagem de programação, foi optado pelo desenvolvimento de um modelo baseado na administração de mercadorias e clientes, o qual, ainda que traga uma grande simplicidade e facilitação no desenvolvimento, possibilita que todos os conceitos básicos estudados sobre banco de dados possam ser utilizados ao máximo. Na mesma semana dessas seleções, aconteceu a divisão de responsabilidades do time, ficando a caráter de alguns tanto a parte de desenvolvimento de módulos: Cliente, Mercadoria, Estoque, quanto o das partes que compõem este documento que seguem o modelo de desenvolvimento de trabalhos científicos da ABNT (Associação Brasileira de Normas de Técnicas); enquanto, para alguns ficou unicamente a responsabilidade de explicitação da descrição da regra e entidades do modelo de negócios escolhido.

Por fim, vale-se ressaltar que os autores deste trabalho se utilizaram de tabelas, ilustrações e conceituações para composição do seu projeto. Da mesma forma, foi feita a gravação da apresentação do trabalho uma semana depois da escolha do tema. A Equipe fez o trabalho de forma incremental, assim a cada parte

se foi criando outra e em caso de programação houve o seu registro e explicação neste documento.

1.1 Tema

Trata-se da projeção da base de dados de um ponto de venda (**PDV**), que consiste em um negócio que em sua base engloba o cadastro de mercadoria, deleção de itens, atualização das informações de mercadoria e demonstração dos itens que estão salvos no banco de dados. Também é de caráter de um PDV disponibilizar a administração de clientes, funcionários e estoque. Para todos, há a necessidade das funcionalidades básicas vistas no padrão **CRUD**, acrônimo para *Create, Read, Update and Read*, — padrão que define o manuseio de dados por meio das operações de criação, deleção, atualização e leitura. Além disso, como todo ponto de venda, este projeto tem como fim a exposição de informações baseadas em seleções dinâmicas, assim se utilizando dos conceitos de Banco de Dados e das melhores técnicas de tratamento e organização de banco de dados. Por fim, cabe apontar que o plano esquema do banco em questão também tem por fim caracterizar partes de entidades, como suas classificações, e especificações destas.

1.2 Objetivos

Colocar em prática todos os conceitos aprendidos em sala de aula ao decorrer do semestre, enquanto, utilizando-se do aprendizado empírico que também foi propiciado pelos ensinamentos. De tal forma, faz-se imprescindível citar, também, o estudo: da linguagem de programação escolhida para exercer o desenvolvimento em nível de tecnologia do esquema do banco de dados; o estudo dos métodos de rascunho, planejamento e conclusão do plano de projeto do banco de dados; e as metodologias de documentação e registro de atividades, fundamentada no modelo de trabalhos acadêmicos da Faculdade Metropolitana da Grande Fortaleza.

2 TRATAMENTO DA INFORMAÇÃO

2.1 SOBRE AS TECNOLOGIAS ESCOLHIDAS

A maioria dos bancos de dados utilizam o padrão de linguagem SQL, que pode ser utilizado em diversas ferramentas como **SQL server**, **PostgreSQL**, **MySQL** dentre outros. A linguagem SQL servirá para a consulta e manipulação de dados solicitados, desse jeito o cliente não ficará preso à plataforma **Oracle**. Tais ferramentas servem para tratar da ferramenta da interface para a conexão e uso da linguagem SQL, que é o **MySQL**.

2.1.1 Linguagem SQL

A maioria das organizações guardam suas informações em grandes bancos de dados, sejam **Oracle**, **SQLserver**, **PostgreSQL** ou qualquer outro banco de dados, e para consumir tais informações, utilizamos a linguagem SQL que é a principal linguagem de consulta e manipulação de dados. No início dos anos 70, o trabalho produtivo do colega de pesquisa da IBM **E.F. Codd** levou a tal desenvolvimento de um produto de modelo de dados relacional que foi chamado de **SEQUEL** ou Linguagem de Consulta em Inglês Estruturado (**Structured English Query Language**), porém ultimamente o **SEQUEL** se transformou em SQL que é um mnemônico que desdobrado quer dizer **Structured Query Language** ou **Linguagem de Consulta Estruturada**.

2.1.2 Ferramenta MySQL

O MySQL nada mais é do que um sistema gerenciamento de banco de dados que baseado em cliente server, ou seja, deve-se ter um MySQL em um servidor e outro na máquina (computador/notebook) para a execução de comandos, solicitar e mandar informações ao banco de dados. O MySQL utiliza de um protocolo de comunicação que é uma linguagem muito popular no mercado para banco de dados denominado **Standard Query Language** ou SQL que foi criado em 1890 na Suécia.

O MySQL pode ser usado em diversas plataformas, sendo Windows, Mac ou Linux as mais populares e é compatível com várias linguagens de programação, como **Java**, **Php**, **C++**, **Visual Basic**, **Python** e **Ruby**. Ele é conhecido pelo seu

excelente desempenho e estabilidade, pois exige muito pouco de recursos de hardware da máquina onde está instalada a versão servidor. Além disso, o MySQL é um software livre, ou seja, gratuito e segue o padrão de licença **GPL** e possui suporte a conexões transacionais que significa o cliente solicitar várias transações SQL e decide se vai enviar ou fechar tais instruções realizadas no servidor ou **Rollback** (reverter) em todas as transações entre outros recursos.

2.2 MODELO DE NEGÓCIOS ADOTADO

2.2.1 Base do modelo de negócio escolhido

O modelo de negócios escolhido foi um CRUD de mercadorias, isto é, um sistema que propõe todas as operações de administração de produtos: registro, atualizações, amostragem e exclusão. Além disso, no modelo adotado também é previsto o controle de usabilidade. Assim, também irá empacotar em suas capacidades o gerenciamento de usuários, neste caso, são definidos por clientes, aqueles que farão compra de produtos, e funcionários, aqueles que ficarão dispostos por parte do estabelecimento que adotar o uso do sistema.

Mais esclarecidamente, o modelo adotado segue o ramo dos CRUD 's voltados a pedidos e compras diretos, cabendo neste, por exemplo, restaurantes ou lanchonetes, os quais trabalham com registro da compra de um produto por quantidade e também o registro da ação com suas especificações.

Quanto às entidades, resumidamente são constituídas nos objetos que causam e sofrem ações no fluxo de atendimento: produtos, pedidos, clientes e funcionários. Clientes e funcionários foram tratados a forma de se abstrair e otimizar a usabilidade, assim são considerados ambos 'usuários' para o sistema, o que quer dizer que um funcionário, posto que é um usuários, também pode ser um cliente. Quanto ao produto, este recebe todas as características que são possíveis ver em um cardápio: seu valor unitário e definições. Os pedidos, apesar de estranho considerá-los como entidades, são pois se definem como objetos criados a partir da interação de outros dois tipos de entidades, os produtos e clientes. Além disso, visando seguir as regras de normalização, foi se escolhido a separação de algumas características de produto, as quais são suas informações de estoque e também categoria, que no sistema irão constituir outras duas entidades.

2.2.2 Regras de negócio gerais

Basicamente, as regras são tem três pólos básicos fundamentados pelas entidades centrais, as quais são Usuários, Pedidos e Produtos. Os Usuários são as pessoas que utilizam o sistema, podendo ser cliente e funcionário. Um Funcionário tem a responsabilidade de criar e atualizar o Estoque de algum Produto, assim um estoque só pode ser atualizado por pelo menos um Funcionário. O Estoque de um produto correspondendo às suas informações de logística, deve pertencer a um produto, havendo um Estoque para por produto. Um Produto tem entre suas características, além da dependência indireta de um funcionário para registrá-lo em Estoque, sua categoria, a qual corresponde ao tipo que pode ser de vários Produtos. Ademais, é de possibilidade um produto estar em um Pedido, o qual necessidade que exista um cliente que o solicitou, assim o Pedido depende de um Cliente e da existência de pelo menos um Produto. Um cliente pode fazer um pedido, deste que consiga pagar pelo valor deste em sua determinada quantidade e que ela esteja em estoque. Como dito o cliente é um usuário, e, assim como funcionário, para atuar ele precisa estar ativo. Mais uma vez mostrando que não se pode existir cliente ou funcionário que não seja um Usuário.

2.3 ENTIDADES ENVOLVIDAS

2.3.1 Usuários

As entidades são os elementos envolvidos no negócio e cada entidade terá associado uma série de atributos. O cliente é uma entidade e ele terá uma série de atributos como ID do cliente, nome do cliente, CPF e dentre outros. Por tanto, a entidade é como um substantivo que representa de forma clara alguma função dentro de um negócio.

Cada banco de dados inclui um guest. As permissões concedidas ao usuário guest são herdadas pelo usuário que têm acesso ao banco de dados, mas que não têm uma conta de usuário no banco de dados. O usuário guest não pode ser

removido, mas pode ser desabilitado pela revogação da permissão CONNECT. Essa permissão pode ser revogada executando REVOKE CONNECT FROM GUEST.

As entidades podem ter 3 tipos de relacionamento, que são.

Relacionamento **1** para **1**.

Relacionamento **1** para **n**.

Relacionamento **n** para **n**.

2.3.1.1 Definição

Entidade que irá representar a generalização de todos aqueles envolvidos no uso do fluxo do modelo de negócios: Clientes e Funcionários. Ela irá fornecer as suas especificações e os campos necessários para todas as operações. Sua importância é para evitar a replicação de definições e também tornar mais seguro e abrangente o registro de utilizadores do sistema, os usuários finais. Cabe dizer que suas especificações não são uma existência em si, mas sim uma qualificação de um usuário; o que irá definir as responsabilidades e capacidades do usuário seguindo as regras de negócio. Além disso, os registros de clientes não são deletados, mas sim inativados; e também são identificados principalmente pelos seus códigos, garantindo mais segurança e acurácia nas buscas. Cabe ressaltar que não um registro de usuário não pode ser deletado enquanto houver outros registros de suas especificações dependendo de informações suas, ou seja, nenhuma tupla da tabela de usuários será excluída enquanto nas tabelas de Clientes ou Funcionários tiver uma linha que faça referência ao usuário que se deseja excluir.

2.3.1.2 Atributos:

1. **Código:** será representado pela coluna “codigo” que armazenará somente números inteiros não negativos, sendo classificado como uma chave primária, logo, não aceita valores nulos, e também é incrementado com um baseado na quantidade de outros registros já

armazenados. É sempre único por registro e ajuda para identificação atômica de registros.

2. **Nome:** será representado pela coluna “nome”, a qual poderá ser alterada livremente e só aceitará armazenar valores que sejam um cadeia de caracteres tendo limite por duzentos e cinquenta e cinco caracteres, não aceitando valores nulos ou que tenham mais do que a quantidade de caracteres do seu limite. Este atributo corresponde ao nome do usuário, será usado para listagem, seleções ou amostragens dinâmicas de um ou muitos dados.
3. **Telefone:** corresponde ao número de telefone do usuário, podendo ser alterada e armazenar o contato de diversas formas pois a coluna atrelada a este, a coluna “telefone”, aceita uma cadeia de caracteres que não cinquenta caracteres; não armazena valores nulos mas aceita armazenar tantos os número quanto números com máscara, exemplos: “(85) 9555-5555” e “8595555555”. Será usada na em consultas, sejam criteriosas ou não, desde que envolvam a necessidade de demonstrar este dado.
4. **Cidade, Bairro e Rua:** sua coluna respectivamente correspondentes são: “cidade”, “bairro” e “rua” as quais armazenam valores que sejam uma cadeia de caracteres com tamanho máximo de cem caracteres, podendo ser nula. Significa a localização dos usuários dividida em níveis, visando facilidade nas buscas e normalização da tabela. Atributo alterável.
5. **UF:** significa a unidade federativa em que o usuário se localiza, representada pela coluna “uf”, podendo armazenar valores que contenham até cinco caracteres, podendo ser nulo e mudada.
6. **Atividade do usuário:** Seguindo o modelo de negócios e a regra de não se poder deletar usuários, fez-se então um atributo que representará o status do usuários, variando entre 1 — usuário ativo— ou 0 — usuários inativo —, logo, só aceitará armazenar valores inteiro não nulos e tem valor padrão como 1, em casos de omissão do valor.

2.3.1.3 Manuseio e representação de responsabilidade em SQL

Criação da tabela de usuários vai seguir todos os parâmetros passados definidos nos requisitos na regra, como demonstrado na FIGURA 1:

FIGURA 1 - SCRIPT DE CRIAÇÃO DA TABELA DE USUÁRIOS

```
> CREATE TABLE usuario (  
    codigo INT PRIMARY KEY AUTO_INCREMENT,  
    nome VARCHAR(255) NOT NULL,  
    telefone VARCHAR(50) NOT NULL,  
    cidade VARCHAR(100),  
    bairro VARCHAR(100),  
    rua VARCHAR(100),  
    numero_casa INT,  
    uf VARCHAR(5),  
    isAtivo INT NOT NULL DEFAULT 1  
);
```

FONTE: autoria própria dos autores do documento, 2022.

As inserções deverão seguir as restrições definidas na tabela, que correspondem às propostas pelas regras de negócio, e deverão ter o mesmo padrão do apresentado na FIGURA 2 a seguir:

FIGURA 2 - SCRIPT DE INSERÇÃO DE TRÊS REGISTROS NA TABELA USUÁRIO

```
INSERT INTO  
    usuario(nome, telefone, cidade, bairro, rua, numero_casa, uf, isAtivo)  
VALUES  
    ("Wilson Costa", "(85) 98888-8888", "Fortaleza", "Alvaro Legal", "Frederico Legal", 82, "CE", 1),  
    ("Wilson Castro", "(85) 2222-2222", "Fortaleza", "Carleio", "Omar Barroso", 23, "CE", 1),  
    ("Carioca", "93848-8448", "Copacabana", "Bairro de copacabana", "Rua de copacabana", 43, "RJ", 1)  
;
```

FONTE: autoria da equipe do trabalho, 2022.

A seguir a FIGURA 3 demonstra o comando de exibição da lista de todas as colunas e registros da tabela de usuários, o qual teve como resultado a tabela apresentada na FIGURA 4:

FIGURA 3 - SCRIPT DE LISTAGEM DE TODOS USUÁRIOS E COLUNAS

```
SELECT
    codigo, nome, telefone, cidade, bairro, rua, numero_casa, uf, isAtivo
FROM usuario
;
```

FONTE: autoria dos escritores do trabalho, 2022.

FIGURA 4 - TABELA RESPOSTA DO SCRIPT DA FIGURA 3.

codigo	nome	telefone	cidade	bairro	rua	numero_casa	uf	isAtivo
1	Wilson Costa	(85) 98888-8888	Fortaleza	Alvaro Legal	Frederico Legal	82	CE	1
2	Wilson Castro	(85) 2222-2222	Fortaleza	Carleio	Omar Barroso	23	CE	1
3	Carioca	93848-8448	Copacabana	Bairro de copacabana	Rua de copacabana	43	RJ	1

FONTE: autoria dos autores do projeto, 2022.

A seguir a FIGURA 5 demonstra um script que desativa todos os usuários que sejam da unidade federativa “RJ” — Rio de Janeiro —, e a FIGURA 6 a listagem somente das colunas “nome” e “uf” dos usuários ativos, isto é, com valor “1” no campo “isAtivo”, da tabela de usuários:

FIGURA 5 - SCRIPT DE ALTERAÇÃO DO CAMPO “isAtivo” DOS USUÁRIOS QUE SEJAM DA UNIDADE FEDERATIVA “RJ”

```
UPDATE usuario
SET
    isAtivo = 0
WHERE
    uf = "RJ";
```

FONTE: autoria dos escritores do trabalho, 2022.

FIGURA 6 - LISTAGEM DAS COLUNAS RESPECTIVAS AO NOME E UNIDADE FEDERATIVA DOS USUÁRIOS ATIVOS

nome	uf
Wilson Costa	CE
Wilson Castro	CE

FONTE: autoria dos autores do trabalho, 2022.

Como foi possível ver na figura 4 só existia um usuário que continha o UF como Rio de Janeiro, e este não aparece na listagem da figura 6 pois teve seu valor de status de ativação alterado na figura 5.

2.3.2 Clientes

2.3.2.1 Definição

A entidade de clientes, é responsável por armazenar informações dos Usuários, pertencentes ao grupo de clientes, compartilhando uma ligação direta com a Entidade de usuários, visto que todo cliente se trata de um usuário, onde o mesmo, para se diferenciar dos demais, possui atributos diferentes de outros grupos, como o de funcionários a ser citado posteriormente. Por cada cliente fazer parte da Entidade de usuários, a tabela de clientes faz o controle do crédito do cliente, possuindo somente dois atributos: *Código* e *limite crédito*.

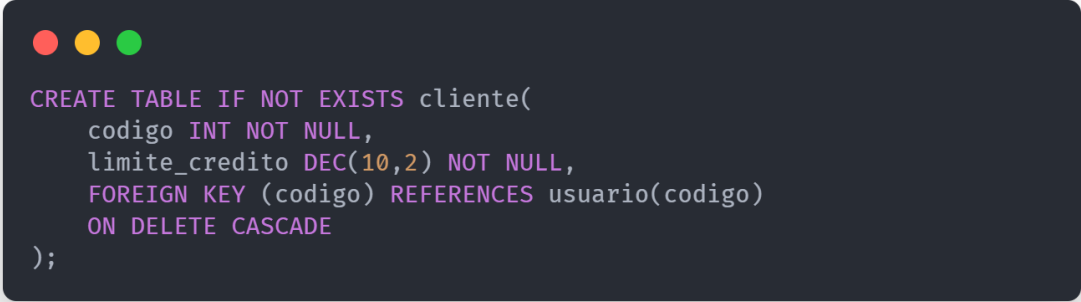
2.3.2.2 Atributos

1. **código:** Representa o código de usuário, sendo este, uma chave estrangeira da entidade usuários, onde o mesmo será responsável por informar o código do cliente que possui crédito no sistema PDV, onde possui as mesmas características do campo o qual se refere (Não nulo, somente valores negativos e inteiros, "auto_increment", etc...)
2. **limite crédito:** Armazena o limite de crédito do cliente, ou seja, é o valor total correspondente ao que poderá ser usado pelo cliente em compras dentro do sistema, sendo este um campo do tipo decimal, podendo ser nulo (Haja vista o cliente esteja devendo), tendo tamanho máximo de 9 casas para inteiros e 2 casas para decimais.

2.3.2.3 Manuseio e representação de responsabilidade em SQL

Script para a criação da tabela de clientes, feita com base nos atributos e regras acima:

FIGURA 7 - SCRIPT DE CRIAÇÃO DA TABELA DE CLIENTES

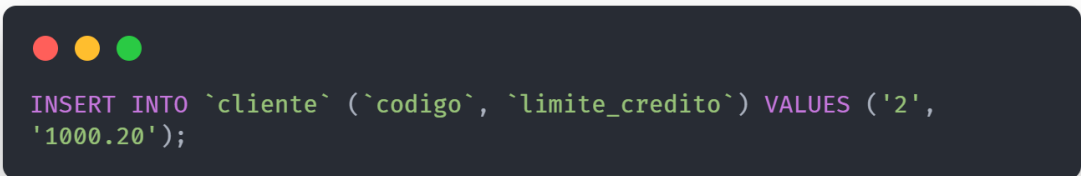


```
CREATE TABLE IF NOT EXISTS cliente(  
    codigo INT NOT NULL,  
    limite_credito DEC(10,2) NOT NULL,  
    FOREIGN KEY (codigo) REFERENCES usuario(codigo)  
    ON DELETE CASCADE  
);
```

FONTE: autoria dos autores do trabalho, 2022.

Posteriormente é realizada a “população” da tabela, preenchendo os campos necessários para a inclusão de um novo registro, como na imagem abaixo:

FIGURA 8 - SCRIPT DE INSERÇÃO DE UM REGISTRO DE CLIENTES



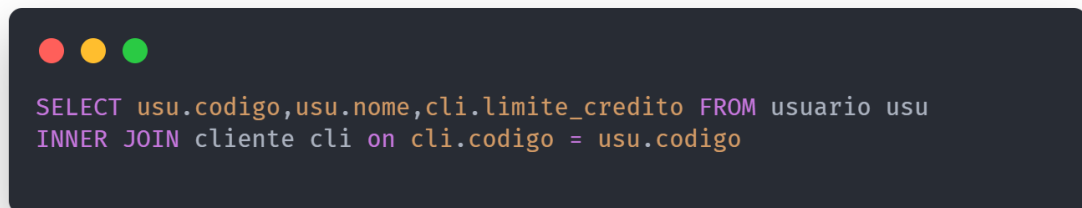
```
INSERT INTO `cliente` (`codigo`, `limite_credito`) VALUES ('2',  
'1000.20');
```

FONTE: autoria dos autores do trabalho, 2022.

no atributo “*codigo*” deve ser informada uma chave cadastrada na entidade de usuários, caso contrario, será retornado um erro, informando sua inexistência na usuarios

Após a inserção dos dados, é realizada uma consulta, visando relacionar as entidades(com a chave “codigo”) de usuário ou cliente, retornando somente, clientes que possuam cadastro em AMBAS as entidade, para isto, foi utilizado o comando de consulta SELECT e um comando de junção JOIN, como representado na figura abaixo:

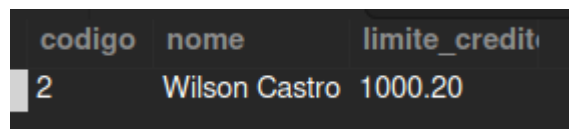
FIGURA 9 - SCRIPT DE LISTAGEM DE TODAS AS COLUNAS DE CLIENTES TRAZENDO INFORMAÇÕES DE USUÁRIO

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains a SQL script for a database query.

```
SELECT usu.codigo,usu.nome,cli.limite_credito FROM usuario usu
INNER JOIN cliente cli on cli.codigo = usu.codigo
```

FONTE: autoria dos autores do trabalho, 2022.

FIGURA 10 - RESPOSTA DO SCRIPT DA FIGURA 9

A table with a dark background and light-colored text, displaying the results of the SQL query. It has three columns: 'codigo', 'nome', and 'limite_credito'. The first row shows the value '2' under 'codigo', 'Wilson Castro' under 'nome', and '1000.20' under 'limite_credito'.

codigo	nome	limite_credito
2	Wilson Castro	1000.20

FONTE: autoria dos autores do projeto, 2022.

2.3.3 Funcionários

2.3.3.1 Definição

Uma especialização da tabela de usuários, que fica com a responsabilidade de gerência dos produtos e também de pedidos. Isso porque sua função pode determinar qual a sua responsabilidade. Intencionalmente não foi criada uma tabela à parte para haver a possibilidade de ser mais abrangente quanto a função do funcionário. Ele precisa da existência de um usuário base que irá armazenar suas informações, e quanto a sua especificação ele irá armazenar a função. Como supramencionado, por depender do registro de usuário, caso este venha a ser deletado, seu registro como funcionário também irá ser.

2.3.3.2 Atributos

1. **código:** Representa o código de usuário, sendo este, uma chave estrangeira da entidade usuários, onde o mesmo será responsável por informar o código do usuário que possui crédito no sistema PDV, onde possui as mesmas características do campo o qual se refere (Não nulo, somente valores negativos e inteiros, "auto_increment", etc...)
2. **função:** Representa os deveres do funcionário no fluxo do sistema, podendo variar mas sempre seguindo o eixo de controle e gerência de produtos e estoque, e também o de pedidos. Terá tabela respectiva 'funcao', podendo esta armazenar valores textuais com tamanho máximo de duzentos e cinquenta e cinco caracteres, não podendo ficar nulo em nenhum momento.

2.3.3.3 Manuseio e representação de responsabilidade em SQL

Primeiramente, a manipulação começa com a criação da tabela de funcionários com a cláusula de caso não exista seja criada, assim como demonstrado a seguir:

FIGURA 11 - SCRIPT DE CRIAÇÃO DA TABELA DE FUNCIONÁRIOS

```
CREATE TABLE IF NOT EXISTS funcionario (  
    codigo INT NOT NULL,  
    funcao varchar(255) NOT NULL,  
    FOREIGN KEY (codigo)  
        REFERENCES usuario (codigo)  
        ON DELETE CASCADE  
);
```

FONTE: autoria dos autores do projeto, 2022.

Após a criação da tabela, apresenta-se a inserção de registros de funcionários, com distintas funções:

FIGURA 12 - SCRIPT DE INSERÇÃO DE FUNCIONÁRIOS

```
INSERT INTO funcionario(codigo, funcao) VALUES (1, "Gerente de nível atendimento");
INSERT INTO funcionario(codigo, funcao) VALUES (2, "Gerente de nível produtos");
INSERT INTO funcionario(codigo, funcao) VALUES (3, "Garçom");
```

FONTE: autoria dos autores do projeto, 2022.

Abaixo, tem-se a listagem dos funcionários inseridos e suas informações de usuário:

FIGURA 13 - SCRIPT DE LISTAGEM DE FUNCIONÁRIOS

```
SELECT
    usu.codigo, usu.nome, usu.telefone, usu.cidade, usu.bairro, func.funcao
FROM usuario usu
INNER JOIN funcionario func on func.codigo = func.codigo;
```

FONTE: autoria dos autores do projeto, 2022.

FIGURA 14 - RESPOSTA DO SCRIPT DE LISTAGEM ANTERIOR

codigo	nome	telefone	cidade	bairro	funcao
1	Wilson Costa	(85) 98888-8888	Fortaleza	Alvaro Legal	Gerente de nível atendimento
2	Wilson Castro	(85) 2222-2222	Fortaleza	Carleio	Gerente de nível atendimento
3	Carioca	93848-8448	Copacabana	Bairro de copacabana	Gerente de nível atendimento

FONTE: autoria dos autores do projeto, 2022.

2.3.4 Produto

2.3.4.1 Definição

Referente a mercadoria que será vendida por quantidade. Abstraída para os campos de nome, preço e o código da categoria, cujo é uma referência à tabela de categorias. Um produto será único, assim como indica o seu código. Um produto terá um estoque, o produto não depende de estoque, o que significa que não há produto em estoque. Quanto a categoria, trata-se obrigatória a criação de uma entidade separada, a qual é a tabela de Categoria, que irá armazenar apenas seu código inteiro não nulo e não negativo, controlado de forma incremental e única,

além da sua descrição, o valor textual respectivo a definição daquela categoria. Vale ressaltar que a descrição da categoria é obrigatória e foi criada na intenção de deixar a tabela de produtos mais normalizada possível aproveitando, assim, o espaço.

2.3.4.2 Atributos

1. **Código:** identificador único inteiro não nulo e não negativo, controlado incrementalmente. Usado para isolar individualmente um produto dos outros, será usado nas outras tabelas de relacionamentos para definir mais precisamente qual produto é referido, assim, será usado como referência a tupla do respectivo produto;
2. **Código da Categoria:** identificador da categoria do produto, valor inteiro e não nulo, será a referência a categoria do produto registrada na tabela de categorias. Visto que é uma referência, põe-se como dependência da tabela de produtos à de categoria, desta forma, pode-se dizer que um produto depende de uma categoria existente no momento de sua criação. Em caso de deleção da categoria, este recebe a restrição 'ON DELETE CASCADE' que diz que se o registro da referência for excluído todas as tuplas que dependem dele também serão excluídas, sua coluna respectiva na tabela é "cod_categoria";
3. **Preço:** referente ao valor unitário do produto, sendo um valor decimal com até duas casas decimais e não aceitando valores nulos. Sua respectiva coluna é 'preco'. No fluxo o preço será usado para definir o valor final da compra do produto pela quantidade escolhida;
4. **Nome:** a descrição do produto, que tem a coluna de mesmo nome, a qual aceita armazenar valores de cadeia de caracteres não nulos com tamanho máximo de duzentos e cinquenta e cinco caracteres, vai caracterizar o produto que vai ser vendido ao cliente, assim este não pode ser nulo, como já citado.

2.3.4.3 Relacionamento Categoria do Produto

O relacionamento de categorias e produtos é definido como opcional um para muitos, em que há somente a dependência obrigatória de produtos ter uma

categoria. Neste sentido, uma categoria pode não estar associada a um produto, no entanto, é obrigatório que um produto esteja associado a uma categoria.

FIGURA 15 - COMANDO DE CRIAÇÃO DA TABELA DE CATEGORIAS

```
CREATE TABLE IF NOT EXISTS categoria (  
    codigo INT PRIMARY KEY auto_increment NOT NULL,  
    descricao varchar(255) NOT NULL  
);
```

FONTE: autoria dos escritores do trabalho, 2022.

FIGURA 16 - INSERÇÃO DE ALGUMAS CATEGORIAS NA TABELA DE CATEGORIA

```
insert into categoria(descricao) values("Bebida");  
insert into categoria(descricao) values("Pequeno Almoço");  
insert into categoria(descricao) values("Refeição");  
insert into categoria(descricao) values("Fast Food");  
insert into categoria(descricao) values("Doce");  
insert into categoria(descricao) values("Salgado");
```

FONTE: autoria dos escritores do trabalho, 2022.

2.3.4.4 Manuseio e representação de responsabilidade em SQL

A seguir se tem a criação da tabela de produtos, identificável nesta a referência a tabela de categorias, e, em seguida, também há uma figura de inserção de registros na tabela de produtos, a qual apresenta a criação do vínculo entre categoria e produtos:

FIGURA 17 - COMANDO DE CRIAÇÃO DA TABELA DE PRODUTOS

```
CREATE TABLE IF NOT EXISTS produto (  
    codigo INT PRIMARY KEY AUTO_INCREMENT NOT NULL,  
    cod_categoria INT NOT NULL,  
    preco DEC(10 , 2 ) NOT NULL,  
    nome VARCHAR(255) NOT NULL,  
    FOREIGN KEY (cod_categoria)  
        REFERENCES categoria (codigo)  
        ON DELETE CASCADE  
);
```

FONTE : criadores deste documento, 2022.

FIGURA 18 - SCRIPT DE INSERÇÃO DE PRODUTOS NA TABELA DE PRODUTOS

```
insert into produto(cod_categoria, preco, nome) values (1, 5.10, "Suco de Goiaba");  
insert into produto(cod_categoria, preco, nome) values (6, 3.00, "Pastel de Frango");
```

FONTE : criadores deste documento, 2022.

2.3.5 Estoque

2.3.5.1 Definição

Entidade fraca que representa o armazenamento de produtos, o estoque de produtos. Irá contar com a referência ao produto e sua quantidade em armazenamento. Segundo o fluxo adotado, um estoque só poderá ser alterado e criado por um usuário do tipo funcionário, assim, um estoque também depende de um funcionário. Também armazena a informação da data em que houve a última alteração na tupla. Por ser uma entidade fraca, caso funcionário ou produto que o estoque dependa seja excluído, a linha que faz referência a um destes também será excluída. Esta entidade não tem identificador próprio em um único campo, mas sim uma chave de identificação composta por código do produto e também pelo usuário que alterou a linha por último.

2.3.5.2 Atributos

1. **Código do Produto:** é a referência do produto do estoque ao registro cadastrado, e, dito isso, é notório que assim como o código do produto e por ser uma parte da chave primária da tabela de estoque, armazena valores inteiros não nulos e não negativos. É a dependência de produtos que a tabela de estoque tem, assim é um atributo derivado, um chave do tipo estrangeira;
2. **Usuário Alteração:** é um atributo usado para auditoria, registro de informações de logística. Sua coluna 'usuario_alteracao' aceita apenas valores inteiro não nulos e não negativos, sendo a referência a algum funcionário da tabela de funcionários. A partir da criação do produtos este valor já deve ser preenchido e é passível de alterações caso hajam mudanças na tupla respectiva. Assim como código do produto é uma parte da chave primária e uma dependência para outra tabela, o que é definido como um chave estrangeira;
3. **Data de Alteração:** Assim como o campo de **usuário alteração** é um campo de auditoria, mas neste caso será armazenado valores que representam as datas e horas que houve mudanças no registro de um produto. A coluna 'data_alteracao' armazena valores não nulos dentro do tipo "*timestamp*", que mais do que valores de data comum, ele também considera os fusos horários em questão;
4. **Quantidade:** Sua coluna respectiva tem o mesmo nome e armazena valores inteiros não nulos e não negativos. Este atributo define o quanto de produtos de determinado tipo existem armazenados no depósito;

2.3.5.3 Manuseio e representação de responsabilidade em SQL

Primeiramente, começa-se com o script de criação da tabela e nele já é perceptível a dependência nas chaves estrangeiras "**FOREIGN KEY**":

FIGURA 19 - SCRIPT DE CRIAÇÃO DA TABELA DE ESTOQUE

```
CREATE TABLE IF NOT EXISTS estoque (  
    codigo_produto INT NOT NULL,  
    usuario_alteracao INT NOT NULL,  
    data_alteracao TIMESTAMP NOT NULL,  
    quantidade INT NOT NULL,  
    PRIMARY KEY (codigo_produto , usuario_alteracao),  
    FOREIGN KEY (codigo_produto)  
        REFERENCES produto (codigo)  
        ON DELETE CASCADE,  
    FOREIGN KEY (usuario_alteracao)  
        REFERENCES funcionario (codigo)  
        ON DELETE CASCADE  
);
```

FONTE: autoria dos escritores deste documento, 2022.

E na imagem seguinte se tem a inserção de dois registros de estoque na tabela de estoque, o que pode se considerar como o abastecimento de um produto por um funcionário:

FIGURA 20 - SCRIPT DE INSERÇÃO DE DOIS REGISTROS NA TABELA DE ESTOQUE

```
INSERT INTO estoque(codigo_produto, usuario_alteracao, data_alteracao, quantidade) values  
(1, 2, current_timestamp(), 10),  
(2, 2, current_timestamp(), 2)  
;
```

FONTE: autoria dos escritores deste documento, 2022.

O trecho “current_timestamp()” é a chamada da função que traz o valor em timestamp da ponto no tempo de acordo com a localização e o dados de onde está o banco de dados, o ambiente onde ele se encontra.

FIGURA 21 - SCRIPT DE LISTAGEM DE TODOS OS PRODUTOS E SEUS CAMPOS COM AS INFORMAÇÕES DO SEU ESTOQUE

```
SELECT
    prod.*,
    est.quantidade,
    est.usuario_alteracao,
    est.data_alteracao
FROM
    produto prod
    JOIN
    estoque est ON est.codigo_produto = prod.codigo;
```

FONTE: autoria própria dos autores, 2022.

FIGURA 22 - RESULTADO DO SCRIPT APRESENTADO NA FIGURA 21

codigo	cod_categoria	preco	nome	quantidade	usuario_alteracao	data_alteracao
1	1	5.10	Suco de Goiaba	10	2	2022-10-30 08:29:51
2	6	3.00	Pastel de Frango	2	2	2022-10-30 08:29:51

FONTE: autoria própria dos autores, 2022.

2.3.6 Pedido

2.3.6.1 Definição

A entidade de pedidos, é responsável por representar as demandas dos usuários, o que eles compraram em determinada quantidade. Assim, a entidade de pedidos é uma entidade fraca, que depende da existência de um cliente e de pelo menos um produto. Sua composição prática também depende dos relacionamentos das entidades citadas com a própria. Além disto, a visto de auditoria, ela contém seu identificador e a data de sua criação. Portanto, ela tem suas tabelas filhas mas que tem dependência unilateral. Seus relacionamentos são descritos pela linha Pedidos-Clientes, relacionamento de uma entidade com muitas de outras, no caso, muitas entidades de clientes podendo ter vários pedidos, mas um pedido apenas tendo um cliente. Do mesmo modo, há o relacionamento entre Pedidos-Produtos,

onde muitos produtos poderão estar em diversos pedidos, e um pedido pode ter vários produtos, assim classificado como muitos para muitos.

2.3.6.2 Atributos

1. **Número:** consiste na identificação de um registro, e é um número inteiro não nulo e não negativo que sempre é incrementado em um com a quantidade de valores que existem cadastrados;
2. **Data de Elaboração:** corresponde a data em que foi criado o pedido, em timestamp, não aceitando valores nulos e deve ser inserido no momento de criação do registro.

2.3.6.3 Manuseio e representação de responsabilidade em SQL

Em primeiro lugar, exemplifica-se a criação da tabela na FIGURA 22:

FIGURA 23 - SCRIPT DE CRIAÇÃO DA TABELA PEDIDOS

```
CREATE TABLE IF NOT EXISTS pedido (  
    numero INT PRIMARY KEY NOT NULL,  
    data_elaboracao TIMESTAMP NOT NULL  
);
```

FONTE: autoria dos criadores do projeto, 2022.

2.4 RELACIONAMENTOS ENTIDADES

2.4.1 Estoque: envolvendo Produtos e Funcionários

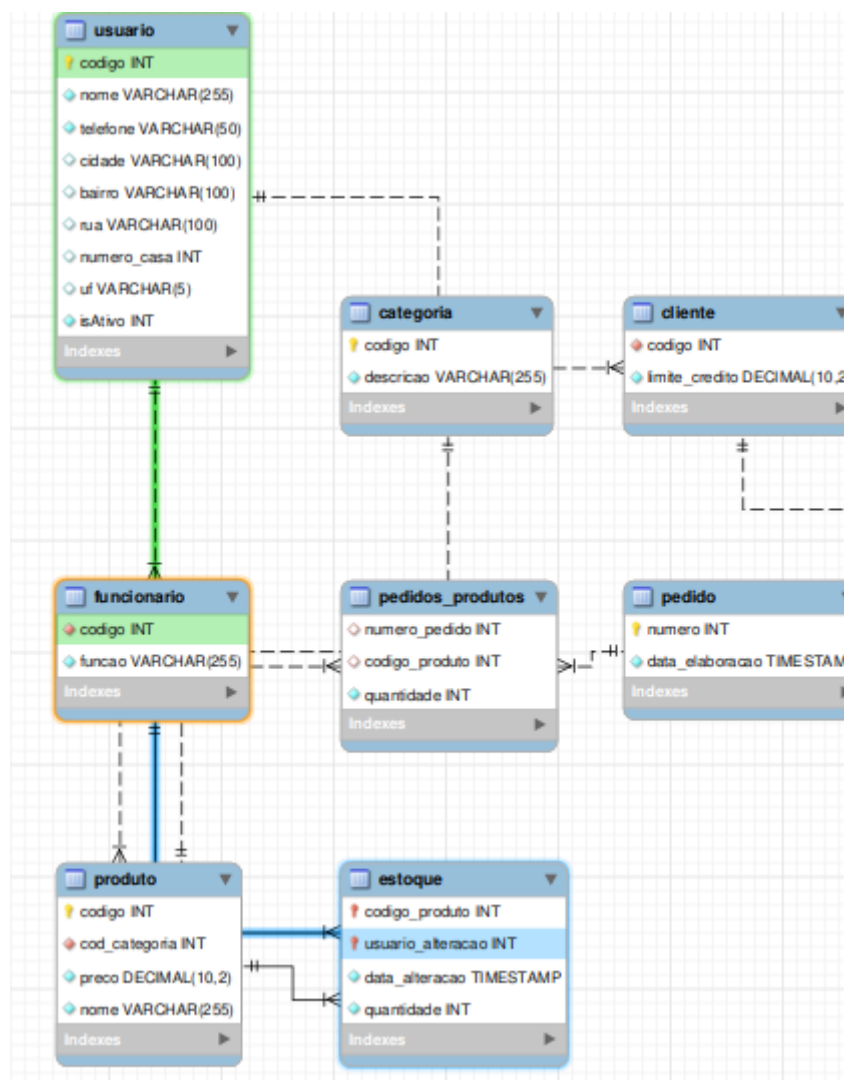
2.4.1.1 Definição

Como já mencionado, um produto indiretamente depende da existência de um funcionário, isto porque para existir em estoque ele precisa de um funcionário. Logo, pode-se dizer que o relacionamento entre funcionários e estoque é um para

muitos, onde um estoque pode ter somente um funcionário, enquanto um funcionário pode cadastrar vários estoques.

2.4.1.2 Ilustração

FIGURA 24 - ILUSTRAÇÃO DO RELACIONAMENTO ENTRE FUNCIONÁRIOS E ESTOQUE



FONTE: autoria dos escritores do trabalho, 2022.

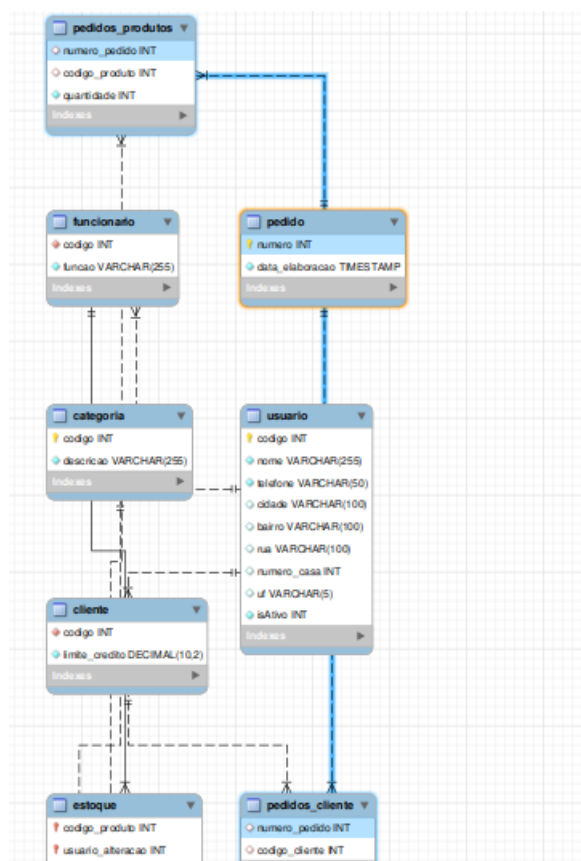
2.4.2 Pedidos: envolvendo Clientes e Produtos

2.4.2.1 Definição

Como também já mencionado, um pedido é uma entidade fraca que depende das entidades de clientes e produtos. O pedido irá representar a ação do cliente de adquirir determinado produto em determinada quantidade. Assim, enquanto a pedidos, um usuário pode fazer vários pedidos mas um pedido é feito apenas por um único usuários, este em sua generalização do tipo cliente. Então, um produto pode estar em diversas quantidades em diversos produtos, desde que esteja dentro do cabível em seu estoque, do mesmo modo, um pedido pode ter vários produtos.

2.4.1.2 Ilustração

FIGURA 25 - DEMONSTRAÇÃO DO RELACIONAMENTO DE CLIENTES E PRODUTOS



FONTE: autoria dos escritores do trabalho, 2022.

3 METODOLOGIA

Como ponto de partida, após o envio da requisição de construção deste trabalho, o grupo se reuniu e realizou uma troca de ideias e opiniões, momento onde todos opinaram sobre como prosseguir: quais ferramentas utilizar, qual modelo de negócio usar e como seria feita a divisão das tarefas de cada um, no fim ficou resolvido que ao decorrer da mesma semana seriam feitas discussões para cada tema em dias distintos.

Assim, para a escolha e definição do Sistema de Gerenciamento de Banco de Dados (SGBD), foi utilizado o modelo de pesquisa bibliográfica e documental, o que ocorreu durante os dias quinze e dezesseis de outubro do ano de criação deste trabalho. Após isso, veio a escolha de qual modelo de negócios iriam abordar, no dia dezessete, dia a qual houve a entrega dos membros da equipe ao supervisor deste trabalho, ficando como tema a criação de ponto de venda de produtos básicos de consumo.

Além disso, houve também entrevistas aos veteranos dos autores visando um norteamento de como suceder com a criação do documento escrito que deveria conter toda a descrição do que foi feito em todos os níveis e ser entregue; este que foi é fundamentado nos modelos de trabalhos científicos da Unifametro e seguindo as normas do padrão ABNT e conter anexos respectivo aos scripts de criação e outras formas de manuseios da base de dados.

Durante todo o processo do trabalho a equipe utilizou a ferramenta *Workbench CE* — software que interfacializa, integrando todo SQL, o manuseio do banco de dados trazendo ainda diversos utilitários — para a programação do banco de dados, catálogo das imagens e usos em consultas e testes na execução do plano do banco de dados. Outrossim, do mesmo modo foi empregada a ferramenta *Draw.io* — ferramenta que possibilita o desenho e planejamento de rascunhos e planos concretos, incluindo relacionamentos e outras especificações, de entidades e suas interações dentro do contexto do negócio estabelecido pelo modelo a ser adotado.

Ao final, a equipe reuniu todo o construído e seguido as recomendações apresentou ao supervisor este trabalho em uma gravação contendo todos os membros, e explicação das partes do trabalho e do banco de dados desenvolvido.

4 RESULTADOS E DISCUSSÃO

A criação de uma base de dados envolve diversas fases, diversas formas de se fazer cada uma delas e com diversos indivíduos com distintas responsabilidades. Uma realidade que exige capacitação e experiência para se superar. Faz-se perceptível agora tal fato com a experimentação supervisionada de um caso adaptado ao estudo mas que era pautado em realidades de negócios presentes no mercado, o que aconteceu ao decorrer da pesquisa, organização das tecnologias e ambiente de informações e o que cada pessoa teve que fazer ao decorrer deste projeto.

Não só isso, mas também o fato de haver a necessidade de constante evolução do material já existente, seja esta para abarcar novos aspectos ou para interagir com outras novas partes. Para Além, é urgente que se cite um dos principais aspectos da construção de projetos: gradação no desenvolvimento. Como no plano de um banco, parte-se da parte inicial de rascunho e projeção para colocar em primeiro momento aquilo que representa a base do modelo de negócios. Feito isso, cabe lapidar o que se foi posto para tornar mais consistente e próximo do que será feito a nível de tecnologia. O passo seguinte deve ser voltado para delimitações de requisitos para aperfeiçoar o feito para garantir o comprimento das regras de negócio. Após toda essa validação e escolha de tecnologias mais cabíveis, é feito a implementação na programação e depois os testes para validação do resultado.

Concluindo, cabe apenas às considerações de registros do que foi feito e também o registro de manual de uso. Portanto, o fluxo supramencionado, corrobora reiterando a ideia anteriormente mencionada trazida pelo aprendizado: A composição de uma base de dados que segue um modelo de negócio com regras e necessidades reais é um processo gradativo, evolutivo e cooperativo.

5 CONSIDERAÇÕES FINAIS

Ao final do trabalho é notório a urgência e importância no estudo dos conceitos de banco de dados para aqueles que desejam seguir pela área da tecnologia da informação. No seguimento das partes do trabalho a equipe pode perceber o quão também é importante a existência de atividades práticas que demonstrem diretamente como é o processo do desenvolvimento em uma realidade ou demanda próxima a encontrada no mercado de trabalho dos segmentos da tecnologia da informação.

Ademais, também foi proveitoso aos autores do projeto o trabalho de criar se criar este documento, não só pelo exercício da pesquisa e documentação, mas concomitantemente por causa do processo de se esmiuçar as características que fundamentam um banco de dados. O que semelhante acontecimento no mercado de trabalho.

Conquanto, fez-se óbvio o peso da constituição desse documento para a carreira dos envolvidos e a importância de outrem semelhantes a esse, da mesma forma, o estudo da disciplina de Fundamentos de Banco de Dados.

REFERÊNCIAS

CRUD. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2019.
Disponível em: <<https://pt.wikipedia.org/w/index.php?title=CRUD&oldid=55190154>>.
Acesso em: 29 out. 2022.

MYSQL. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2021.
Disponível em: <<https://pt.wikipedia.org/w/index.php?title=MySQL&oldid=60786002>>.
Acesso em: 29 out. 2022.

SQL. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2022.
Disponível em: <<https://pt.wikipedia.org/w/index.php?title=SQL&oldid=64645917>>.
Acesso em: 29 out. 2022.

ANEXO - Scripts de cria e manuseio da Base de Dados

SCRIPT DE CRIAÇÃO E USO BANCO DE DADOS:

1. create database pdv_fdb_database;
2. use pdv_fdb_database;

SCRIPTS DE USUÁRIOS:

- 1) Criação da tabela de usuários:

```
CREATE TABLE usuario (  
    codigo INT PRIMARY KEY AUTO_INCREMENT,  
    nome VARCHAR(255) NOT NULL,  
    telefone VARCHAR(50) NOT NULL,  
    cidade VARCHAR(100),  
    bairro VARCHAR(100),  
    rua VARCHAR(100),  
    numero_casa INT,  
    uf VARCHAR(5),  
    isAtivo INT NOT NULL DEFAULT 1  
);
```

- 2) Inserts na tabela de usuários:

INSERT INTO

usuario(nome, telefone, cidade, bairro, rua, numero_casa, uf, isAtivo)

VALUES

("Wilson Costa", "(85) 98888-8888", "Fortaleza", "Alvaro Legal", "Frederico Legal", 82, "CE", 1),

("Wilson Castro", "(85) 2222-2222", "Fortaleza", "Carleio", "Omar Barroso", 23, "CE", 1),

("Carioca", "93848-8448", "Copacabana", "Bairro de copacabana", "Rua de copacabana", 43, "RJ", 1)

;

3) Select de todas as colunas e todos os usuários:

```
SELECT
    codigo, nome, telefone, cidade, bairro, rua, numero_casa, uf, isAtivo
FROM usuario
;
```

4) Alteração no status de ativação de usuários de UF igual a "RJ":

```
UPDATE usuario
SET
    isAtivo = 0
WHERE
    uf = "RJ";
```

5) Select das colunas do nome e uf dos usuários ativos:

```
SELECT
    nome, uf
FROM
    usuario
WHERE
    isAtivo = 1;
```

SCRIPTS DE CLIENTES:

1) Criação da tabela de clientes:

```
CREATE TABLE IF NOT EXISTS cliente (
    codigo INT NOT NULL,
    limite_credito DEC(10 , 2 ) NOT NULL,
    FOREIGN KEY (codigo)
        REFERENCES usuario (codigo)
        ON DELETE CASCADE
);
```

2) Inserts na tabela de clientes:

```
INSERT INTO cliente(codigo, limite_credito) VALUES (2, 1000.20);
```

3) Select do código de usuário e nome junto com as informações de cliente :

```
SELECT
    usu.codigo, usu.nome, cli.limite_credito
FROM usuario usu
INNER JOIN cliente cli on cli.codigo = usu.codigo;
```

SCRIPTS DE FUNCIONÁRIOS:

1) Criação da tabela funcionários:

```
CREATE TABLE IF NOT EXISTS funcionario (
    codigo INT NOT NULL,
    funcao varchar(255) NOT NULL,
    FOREIGN KEY (codigo)
        REFERENCES usuario (codigo)
        ON DELETE CASCADE
);
```

2) Scripts de inserção de funcionários:

```
INSERT INTO funcionario(codigo, funcao) VALUES (1, "Gerente de nível
atendimento");
INSERT INTO funcionario(codigo, funcao) VALUES (2, "Gerente de nível produtos");
INSERT INTO funcionario(codigo, funcao) VALUES (3, "Garçom");
```

3) Script de listagem de funcionários:

SELECT

usu.codigo, usu.nome, usu.telefone, usu.cidade, usu.bairro, func.funcao

FROM usuario usu

INNER JOIN funcionario func on func.codigo = func.codigo;

SCRIPTS DE PRODUTOS:

1) Criação da tabela de categoria:

CREATE TABLE IF NOT EXISTS categoria (

codigo INT PRIMARY KEY auto_increment NOT NULL,

descricao varchar(255) NOT NULL

);

2) Criação da tabela produtos:

CREATE TABLE IF NOT EXISTS produto (

codigo INT PRIMARY KEY AUTO_INCREMENT NOT NULL,

cod_categoria INT NOT NULL,

preco DEC(10 , 2) NOT NULL,

nome VARCHAR(255) NOT NULL,

FOREIGN KEY (cod_categoria)

REFERENCES categoria (codigo)

ON DELETE CASCADE

);

3) Inserção de categorias de produtos:

insert into categoria(descricao) values("Bebida");

insert into categoria(descricao) values("Pequeno Almoço");

insert into categoria(descricao) values("Refeição");

insert into categoria(descricao) values("Fast Food");

insert into categoria(descricao) values("Doce");

```
insert into categoria(descricao) values("Salgado");
```

4) Inserção de produtos:

```
insert into produto(cod_categoria, preco, nome) values (1, 5.10, "Suco de Goiaba");  
insert into produto(cod_categoria, preco, nome) values (6, 3.00, "Pastel de Frango");
```

5) Listagem de produtos e categorias respectivas:

```
SELECT produto.*, categoria.descricao FROM produto join categoria on  
produto.cod_categoria = categoria.codigo;
```

SCRIPTS DE ESTOQUE:

1) Script de criação da tabela de estoque:

```
CREATE TABLE IF NOT EXISTS estoque (  
    codigo_produto INT NOT NULL,  
    usuario_alteracao INT NOT NULL,  
    data_alteracao TIMESTAMP NOT NULL,  
    quantidade INT NOT NULL,  
    PRIMARY KEY (codigo_produto , usuario_alteracao),  
    FOREIGN KEY (codigo_produto)  
        REFERENCES produto (codigo)  
        ON DELETE CASCADE,  
    FOREIGN KEY (usuario_alteracao)  
        REFERENCES funcionario (codigo)  
        ON DELETE CASCADE  
);
```

2) Script de inserção de estoque de produtos:

```
INSERT INTO estoque(codigo_produto, usuario_alteracao, data_alteracao,
quantidade) values
(1, 1, current_timestamp(), 10),
(2, 1, current_timestamp(), 2)
;
```

SCRIPTS DE PEDIDOS:

1) Script de criação da tabela de pedidos:

```
CREATE TABLE IF NOT EXISTS pedido (
    numero INT PRIMARY KEY NOT NULL,
    data_elaboracao TIMESTAMP NOT NULL
);
```

2) Script de criação da tabela de relacionamento de pedidos e clientes:

```
CREATE TABLE IF NOT EXISTS pedidos_cliente (
    numero_pedido INT,
    codigo_cliente INT,
    FOREIGN KEY (numero_pedido)
        REFERENCES pedido (numero)
        ON DELETE SET NULL,
    FOREIGN KEY (codigo_cliente)
        REFERENCES cliente (codigo)
        ON DELETE SET NULL
);
```

3) Script de criação da tabela de relacionamento de pedidos e produtos:

```
CREATE TABLE IF NOT EXISTS pedidos_produtos (  
    numero_pedido INT,  
    codigo_produto INT,  
    quantidade INT NOT NULL,  
    FOREIGN KEY (numero_pedido)  
        REFERENCES pedido (numero)  
        ON DELETE SET NULL,  
    FOREIGN KEY (codigo_produto)  
        REFERENCES produto (codigo)  
);
```