

INSTITUTO SUPERIOR POLITÉCNICO DE KANGONJO - ISKA

ÁREA DE FORMAÇÃO DE ENGENHARIA

CURSO DE ENHENHARIA INFORMÁTICA

DOCUMENTAÇÃO

SISTEMA DE GERENCIAMENTO DE RESTAURANTE FLOR DO SEQUELE

**Nome**: Félix Eugénio Mavila & Domingos Silva

**Nº** 210678

**Sala:** 10

Maio de 2023

INSTITUTO SUPERIOR POLITÉCNICO DE KANGONJO

(I.P.I.L)

**Tema:** Desktop

**Subtema**: Sistema de Gerenciamneto de Restaurante Flor do Sequele

Projeto Tecnológico apresentado como requisito parcial para prova de aptidão profissional, Área de Formação Informática, Curso Engenharia Informática, Instituto Superior Politécnico de Kangonjo.

**Orientador**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Gabriel Graça**

**Dedicatória**

Eu Félix Eugénio Mavila, dedico este trabalho a Deus pelo dom da vida, a minha família pelo total apoio nessa caminhada vitoriosa. E para todos os professores do ISKA.

“Somente através da ajuda da inteligência infinita de Deus que este trabalho foi concluído de forma satisfatória dedicamos a Deus”.

Agradecimentos

Primeiramente gostaria de agradecer a Deus, agradecemos também ao nosso orientador Senhor Professor **Gabriel Graça** por aceitar conduzir o nosso Projeto Tecnológico.

Eu, Félix Eugénio Mavila gostaria de Agradecer a todos que no decorrer desses anos de formação no ISKA contribuíram direta ou indiretamente para a minha formação, agradeço especialmente a todos os meus diretores de turma enquanto aluno do ISKA.

Aos meus Pais que sempre estiveram ao meu lado, apoiando-me ao longo de toda a minha trajetória.

Sumário

Este trabalho tem como objetivo o desenvolvimento de um Sistema *DESKTOP*, destinado à Gerenciamento de Restaurante Flor do Sequele , As empresas no sector de vendas no seu dia-a-dia actuam nas suas actividades vendendo seus produtos em seus Estabalecimentos causando uma gama de problemas, o volume de informação gerado aumenta exponencialmente, sendo que uma parcela significativa das informações encontra-se em formato textual. Entretanto, face ao grande volume de informação tem se notado que há atrasos por parte dessas lojas em exercer as suas actividades de forma segura , rápida e eficiente, e que por sinal afeta também o vendedor na obtenção e visualização de seus Produtos. Por esta razão eu Félix Eugénio Mavila, Para dar uma solução a esse Problema resolvi desenvolver um sistema que visa à melhoria no acto de Gerenciamneto do Establecimento (Restaurante), gerando menos frustrações os vendedores. Por fim, todos os objetivos estabelecidos inicialmente foram cumpridos, visto que o sistema foi desenvolvido e possui todas as funcionalidades essenciais, sendo possível que seja implementado em sua integridade.

Palavras-Chave: Gerenciamento de Restaurante Flor do Sequele , Sistema *Desktop*, Desenvolvimento *Desktop*.

**Objetivo Geral**

Desenvolver um Sistema Desktop para Gerenciamneto de Restaurante Flor do Sequele.

**Objetivos específicos**

* Analisar e Levantar os requisitos.
* Pesquisar Ferramentas para Desenvolvimento Web.
* Permitir ao Administrador Cadastrar Fucionários.
* Permitir ao Funcionário Fazer Vendas.
* Permitir ao Funcionário Cadastrar Produtos.
* Permitir ao Administrador Visualizar os Fucionários.

Índice

[1 Introdução 1](#_Toc136845983)

[2 Base de Dados 3](#_Toc136845984)

[Definição 3](#_Toc136845985)

[Classificação das bases de dados 3](#_Toc136845986)

[2.1.1 Quanto ao Modelo de Dados 3](#_Toc136845987)

[2.1.2 Quanto ao Número de Usuários 5](#_Toc136845988)

[2.1.3 Quanto á Localização 6](#_Toc136845989)

[Sistemas Gerenciadores de Banco de Dados SGBD 9](#_Toc136845990)

[O Modelo Relacional 9](#_Toc136845991)

[2.1.4 Noção de Registro 10](#_Toc136845992)

[2.1.5 Noção de Campo 11](#_Toc136845993)

[2.1.6 Classificação dos Campos de uma Tabela 11](#_Toc136845994)

[2.1.7 Tipos de Relacionamentos 12](#_Toc136845995)

[Normalização 14](#_Toc136845996)

[2.1.8 Formas Normais 14](#_Toc136845997)

[Structured Query Language (SQL) 16](#_Toc136845998)

[2.1.9 Subconjuntos do SQL 16](#_Toc136845999)

[3 Electron.js 20](#_Toc136846000)

[Desktop Application (Aplicação Desktop) 20](#_Toc136846001)

[O que é o Eelectron 20](#_Toc136846002)

[Blocos de construção do Electron 22](#_Toc136846003)

[Módulos do Electron 24](#_Toc136846004)

[4 Introdução a paginas web 25](#_Toc136846005)

[Origem da Internet 25](#_Toc136846006)

[A Web ou a WWW 26](#_Toc136846007)

[Provedor de Serviço de Internet (ISP) 26](#_Toc136846008)

[Como funciona a web 27](#_Toc136846009)

[4.1.1 Protocolo HTTP 27](#_Toc136846010)

[5 Desenvolvimento Web 30](#_Toc136846011)

[As linguagens Web 30](#_Toc136846012)

[5.1.1 Linguagens de marcação 30](#_Toc136846013)

[5.1.2 Linguagens do tipo client side script (script do lado do cliente) 30](#_Toc136846014)

[5.1.3 Linguagens do tipo server side script (script do lado do servidor) 31](#_Toc136846015)

[Instituto Superior Politécnico de Kangonjo – Área de Formação de Informática 33](#_Toc136846016)

[6 Análise do Sistema 35](#_Toc136846017)

[Requisitos Não Funcionais 35](#_Toc136846018)

[Disponibilidade 35](#_Toc136846019)

[Taxa de Ocorrencias de Falhas 35](#_Toc136846020)

[Probabilidade de Indisponibilidade 35](#_Toc136846021)

[Tempo medio para Falhas 35](#_Toc136846022)

[Requisitos de Resposta 35](#_Toc136846023)

[Requisitos de Processamento 35](#_Toc136846024)

[Requisitos de Temporização 36](#_Toc136846025)

[Requisitos de Espaço 36](#_Toc136846026)

[Requisitos de Portabilidade 36](#_Toc136846027)

[Requisitos de Segurança 36](#_Toc136846028)

[Requisitos de Usabilidade 36](#_Toc136846029)

[Requisitos Funcionais 37](#_Toc136846030)

[Regra de Negócio 39](#_Toc136846031)

[Diagrama de contexto 40](#_Toc136846032)

[6.1.1 Dicionário de dados 41](#_Toc136846033)

[7 Cronograma de atividades 41](#_Toc136846034)

[8 Modelo Entidade Relacionamento 42](#_Toc136846035)

[Ferramentas usadas para construção da base de dados 43](#_Toc136846036)

[9 Ferramentas usadas para construção da aplicação 45](#_Toc136846037)

[10 Anexos 48](#_Toc136846038)

[11 Conclusão 51](#_Toc136846039)

[12 Referencias Bibliográficas 52](#_Toc136846040)

Lista de Figuras

Figura 1 - Tela do Atom. 20

Figura 2 - Blocos de Construção do electron. 22

Figura 3 - Bloco de construção do electron forma resumida. 22

Figura 4 - Módulos do processo principal. 24

Figura 5 - Módula do processo de Renderização. 25

Figura 6 - Origem da Internet. 26

Figura 7 - Exemplo de transação HTTP 27

Figura 8 - Modelo Cliente/Servidor. 28

Figura 9 - Exemplo de uma URL. 29

Figura 10 - Modelo Entidade Relacionamento “um”. 39

Figura 11- Modelo Entidade Relacionamento “dois”. 40

Figura 12 – Login - parte desktop 46

Figura 13 - Menu inicial - parte desktop 46

Figura 14 - Ver alunos - parte desktop 47

Lista de Tabelas

Tabela 1 - Cronograma de Atividades 38

Lista de abreviaturas

**AJAX** - Asynchronous JavaScript And XML.

**API** - Application Program Interface.

**CSS** – Cascading Style Sheet.

**C#** - C-Sharp

**HTML** - Hypertext Markup Language.

**JS** - JavaScript.

**MVC** - Model View Controller.

**NPM** - Node Package Management.

**PHP** - Hypertext Preprocessor.

**POO** - Programação Orientada á Objeto.

**Rest** - Respresentation State Transference.

**RT** - *Runtime*, sistema de tempo de execução.

**SQL** - Structured Query Language.

**VB** - Visual Basic.

**WAMP** - Windows, Apache, MySQL, PHP

**WINDOWS** - NT- família de sistemas operacionais produzidos pela Microsoft

**XML -** Extensible Markup Language.

# Introdução

O desenvolvimento crescente de tecnologias traz a necessidade de realizar tarefas de maneira cada vez mais simples e informatizada, ou seja, de modo que seja exigido o menor esforço possível por parte das pessoas.

As aplicações do tipo Desktop são exemplos dessa informatização, já que cada vez mais tarefas, desde o envio de mensagens até o pagamento de contas, podem ser realizadas de maneira online, utilizando apenas um computador com acesso à Internet. O que antigamente não era muito aceito pela população por ser considerado perigoso, hoje em dia se tornou frequente na vida da maioria das pessoas, que utilizam essas aplicações para tarefas comuns do dia-a-dia, como, por exemplo, a compra de produtos.

Nesse entanto, propus-me a desenvolver um sistema DESKTOP que visa à melhoria no acto de Gerenciamento de Restaurante.



# Base de Dados

## Definição

Um banco de dados é uma coleção de informações que é organizado para que possa ser facilmente acessado, gerenciado e atualizado.

“Segundo Korth, um banco de dados é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico”, ou seja, sempre que for possível agrupar informações que se relacionam e tratam de um mesmo assunto, posso dizer que tenho um banco de dados.

**Exemplo**

* Uma lista telefônica.
* Livro de endereços
* Um sistema de controle de RH de uma empresa.
* Lista dos Clientes de um Banco.

## Classificação das bases de dados

**As Bases de Dados classificam-se em:**

1. Quanto ao Modelo de Dados
2. Quanto ao Número de Usuários
3. Quanto á Localização

### Quanto ao Modelo de Dados

**Os Bancos de dados classificam-se em:**

1. Banco de Dados Hierárquico
2. Banco de Dados Em Rede
3. Banco de Dados Orientado á Objetos
4. Banco de Dados Relacionais

**Banco de Dados Hierárquico**

Uma base de dados hierárquica é um tipo de sistema de gerenciamento de banco de dados que conecta registos numa estrutura de dados em árvore através de ligações de tal modo que cada tipo de registo tenha apenas um possuidor.

**Banco de Dados Em Rede**

Sua organização é semelhante à dos Bancos de Dados hierárquicos, é o modelo de dados que eliminou o conceito de hierarquia, permitindo que um mesmo registro estivesse envolvido em várias associações, um registro filho pode ser ligado a mais de um registro pai, criando conexões bastante complexas e são bastante utilizados em sistemas para computadores de grande porte (mainframe).

**Banco de Dados Orientado á Objetos**

Um banco de dados orientado a objetos é um banco de dados em que cada informação é armazenada na forma de objetos, ou seja, utiliza a estrutura de dados denominada orientação a objetos, a qual permeia as linguagens mais modernas. Começou a ser comercialmente viável em 1980. O gerenciador do banco de dados para um orientado a objeto é referenciado por vários como ODBMS ou OODBMS

 Este modelo define um banco de dados como uma coleção de objetos, ou elementos de software reutilizáveis, com recursos e métodos associados. Existem vários tipos de bancos de dados orientados a objetos: Um banco de dados multimídia incorpora mídias, como imagens, que não poderiam ser armazenadas em um banco de dados relacional.

O modelo de banco de dados orientado a objetos é o modelo de banco de dados pós-relacional mais conhecido, pois incorpora tabelas, mas não se limita a tabelas. Esses modelos também são conhecidos como modelos de banco de dados híbridos

**Banco de Dados Relacional**

Um banco de dados relacional, inventado por EF Codd na IBM em 1970, é um banco de dados tabular no qual os dados são definidos para que possam ser reorganizados e acessados de várias maneiras diferentes.

Os bancos de dados relacionais são compostos por um conjunto de tabelas com dados que se enquadram em uma categoria predefinida. Cada tabela possui pelo menos uma categoria de dados em uma coluna, e cada linha possui uma determinada instância de dados para as categorias que são definidas nas colunas.

O modelo mais comum, o modelo relacional, classifica os dados em tabelas, também conhecidas como relações, cada uma das quais consiste em colunas e linhas. Cada coluna lista um atributo da entidade em questão, como preço, código postal ou data de nascimento. Juntos, os atributos em uma relação são chamados de domínio. Um determinado atributo ou combinação de atributos é escolhido como uma chave primária que pode ser referenciada em outras tabelas, quando é chamada de chave estrangeira.

 Cada linha, também chamada de tupla, inclui dados sobre uma instância específica da entidade em questão, como um determinado funcionário. O modelo também leva em conta os tipos de relacionamento entre essas tabelas, incluindo relacionamentos um para um, um para muitos e muitos para muitos.

Os bancos de dados relacionais são normalmente escritos em Structured Query Language (SQL).

### Quanto ao Número de Usuários

**Classificam se em:**

1. Banco de Dados MonoUsuário
2. Banco de Dados Multiusuário

**Banco de Dados MonoUsuário**

Permitem que apenas um usuário por vez acesse o banco de dados Antigos (1980-1990) e direcionados a uso pessoal: dBASE III, dBASE IV, FoxBase, FoxPro.

**Banco de Dados Multiusuário**

Suporta o acesso de vários usuários ao mesmo tempo A maioria dos bancos de dados atuais oferece suporte a multiusuários

### Quanto á Localização

1. Banco de Dados Centralizados
2. Banco de Dados Distribuídos
3. Banco de dados Heterogéneos

**Banco de Dados Centralizados**

Localizados em uma única máquina denominada Servidor de Banco de Dados Embora centralizados, podem oferecer suporte a acesso concorrente de vários usuários.

**Banco de Dados Distribuídos**

O sistema gerenciador e o banco de dados estão localizados em diferentes máquinas interligadas em redes (LANS ou WANS) Independentemente de serem centralizados ou distribuídos os SGBDs atualmente trabalham dentro da arquitetura cliente-servidor

**Banco de dados Heterogéneos**

Tendência que vem crescendo muito atualmente: distribuir na arquitetura de SGBDs vários bancos de dados de fornecedores diferentes

**Elementos que Intervêm num Banco de Dados**

1. Administradores de banco de dados
2. Projetistas de banco de dados
3. Analistas de sistema e programadores
4. Usuários finais.

**Administrador de Banco de Dados (DBA)**

**Administrador de banco de dados**, comumente chamado de DBA (sigla em inglês de Database administrator), é o profissional responsável por gerenciar, instalar, configurar, atualizar e monitorar um banco de dados ou sistemas de bancos de dados

Ou Administrador de Banco de Dados é o responsável pela autorização de acesso ao Banco de Dados e pela coordenação e monitoração de seu uso. É a pessoa que, numa equipe de desenvolvimento, centraliza tanto o controle dos dados quanto os programas de acesso a eles. É conhecido com a sigla em inglês: DBA (Database Administrator).

**Habilidades que um DBA deve possuir:**

* Conhecimentos da linguagem estruturada SQL.
* Conhecimentos em estrutura de banco de dados, mais precisamente em diagrama entidade-relacionamento.
* Entendimento básico de Arquitetura de computadores.
* Um bom entendimento do funcionamento dos sistemas operacionais.
* Domínio de SGBDs, tais como Microsoft SQL Server, PostgreSQL, MySQL, Oracle database, DB2.

**Funções de um DBA**

Um **Administrador de banco de dados** deve ser capaz de exercer as seguintes tarefas:

* Criação e testes de backup para garantir a recuperabilidade dos dados em caso de falha de hardware ou outros problemas severos.
* Instalar e atualizar as ferramentas do banco de dados.
* Alocar o espaço do sistema reservado ao banco e garantir um alocamento futuro no sistema
* Saber modificar a estrutura do banco de dados.
* Saber os comandos básicos e exclusivos de cada SGBD.
* Ter um controle de acesso, ou privilégios, aos dados como quem pode acessar e o que pode acessar e talvez quando possa acessar.
* Garantir o acesso ao banco de dados no maior tempo possível.
* Garantir o máximo de desempenho para as consultas ao banco de dados.
* Auxiliar a equipe de desenvolvimento e a equipe de testes a maximizar o uso e desempenho do banco de dados.

**Projetista de Banco de Dados (DB Designer)**

O profissional com este perfil realiza as atividades pertinentes ao projeto do banco de dados. É responsável pela definição de tabelas, índices, views, constraints, triggers, stored procedures, tablespaces ou parâmetros de armazenamento e outros elementos necessários para o armazenamento, a recuperação e a deleção de objetos persistentes.

O **Projetista de Banco de Dados** é responsável pela identificação dos dados que devem ser armazenados no Banco de Dados. Ele escolhe a estrutura mais adequada para representar e armazenar esses dados.

**Analista de Sistemas e programador**

Os **analistas** de sistemas determinam os requisitos dos usuários finais e desenvolvem especificações para transações que atendam estes requisitos. Os programadores de aplicações implementam estas especificações com os programas, testando, depurando, documentando e dando manutenção aos mesmos. São profissionais em computação que interagem com o sistema por meio de DMLs, envolvidas em programas escritos em diferentes linguagens hospedeiras. Um Analista de Sistemas elabora um sistema de acordo com as necessidades de uma determinada empresa.

**Programador** é um profissional que trabalha no desenvolvimento de aplicativos e softwares. Ele é responsável por escrever em linguagem de programação como C, Java, Objective-C, Python, Perl, dentre outras, através de instruções e rotinas que são descritas para a máquina. Ele utiliza essas linguagens para facilitar a escrita dos algoritmos, pois a linguagem de máquina, interpretada pelo computador, é difícil de ser compreendida. Essa profissão também pode ser chamada de programador de sistemas ou desenvolvedor de aplicativos.

**Usuários Finais**

Existem basicamente quatro categorias de usuários de Banco de Dados, que fazem operações mais básicas nos SGBD, tais como consultas, atualizações e geração de documentos:

**Usuários Casuais**: acessam o Banco de Dados casualmente, mas que podem necessitar de diferentes informações a cada acesso. Utilizam normalmente sofisticadas linguagens de consulta para especificar suas necessidades;

**Usuários Novatos ou Paramétricos**: utilizam visões do Banco de Dados, utilizando consultas preestabelecidas que já foram exaustivamente testadas. São também chamados de usuários navegantes, ou seja, usuários comuns que interagem com o sistema através de interfaces pré-definidas;

**Usuários Sofisticados**: são usuários que estão familiarizados com o SGBD e realizam consultas mais complexas;

**Usuários Especialistas**: usuários sofisticados que chegam a escrever aplicações especializadas.

## Sistemas Gerenciadores de Banco de Dados SGBD

Um **Sistema de Gerenciamento de Banco de Dados** (SGBD) do inglês Data Base Management System (DBMS) - é o conjunto de programas de computador (softwares) responsáveis pelo gerenciamento de um banco de dados. Seu principal objetivo é retirar da aplicação cliente a responsabilidade de gerenciar o acesso, a persistência, a manipulação e a organização dos dados. O SGBD disponibiliza uma interface para que seus clientes possam incluir alterar ou consultar dados previamente armazenados.

**Exemplos de SGBD’s**

* PostgreSQL
* MYSQL
* Microsoft Acess
* MariaDB
* Oracle
* SQL-Server
* MongoDB

## O Modelo Relacional

O modelo relacional é um modelo de dados representativo (ou de implementação), adequado a ser o modelo subjacente de um Sistema Gerenciador de Banco de Dados (SGBD), que se baseia no princípio de que todos os dados estão armazenados em tabelas (ou, matematicamente falando, relações). Toda sua definição é teórica e baseada na lógica de predicados e na teoria dos conjuntos.

Historicamente ele é o sucessor do modelo hierárquico e do modelo em rede. Estas arquiteturas antigas são até hoje utilizadas em alguns data centers com alto volume de dados, onde a migração é inviabilizada pelo custo que ela demandaria; existem ainda os novos modelos baseados em orientação ao objeto, que na maior parte das vezes são encontrados como kits em linguagem formal.

Um Banco de Dados relacional é um conjunto de Tabelas Lógicas relacionadas entre si. Assim, uma Tabela é a Estrutura onde estão armazenadas uma classe de Dados do banco de Dados.

Em definição simplificada, o modelo baseia-se em dois conceitos: conceito de entidade e relação.

**Uma entidade ou tabela:** é um conjunto de objetos da realidade modelada nas quais se deseja manter a informação no banco de dados.

Exemplo: Numa biblioteca, uma das entidades ou Tabelas, pode ser :

1. Livros
2. Categorias
3. Funcionário
4. Subcategoria

O aspecto mais importante do modelo Relacional é a Entidade, também chamada de

**Tabela.**

Uma Entidade ou Tabela é composta por linhas e colunas. As linhas são chamadas de Registos e as Colunas são chamadas de Campos.

A relação determina o modo como cada registro de cada tabela se associa a registros de outras tabelas.

### Noção de Registro

Um **Registro** é a estrutura onde está armazenado uma única entidade de uma classe

de dados. Ou seja, dentro de uma Tabela cada linha desta tabela representa um único Registro.

Ex: Um livro, uma categoria, um funcionário.

### Noção de Campo

Um **Campo** é a estrutura onde está armazenada uma única informação relacionada ás entidades de uma classe de dados.

Ex: Nome do Livro, Nome da Categoria, idade de um Funcionário.

### Classificação dos Campos de uma Tabela

Os campos de uma Tabela podem ser:

* Chave Primária
* Chave Candidata
* Chave Estrangeira

**Chave Primária (PRIMARY KEY)**

È Um atributo que permite identificar univocamente os registros de uma Tabela Este tipo de chave, refere-se aos conjuntos de um ou mais campos, cujos valores, considerando a combinação de valores de todos os campos da tupla (registro), nunca se repetem e que podem ser usadas como um índice para os demais campos da tabela do banco de dados. Em chaves primárias, não pode haver valores nulos nem repetição de tuplas.

Quando a chave é composta, ou seja, formada por mais de um campo, os valores podem se repetir, mas não a combinação desses valores

**Características de uma Chave Primária**

1. Não pode haver duas ocorrências de uma mesma entidade com o mesmo conteúdo na Chave Primária;
2. A chave primária não pode ser composta por atributo opcional, ou seja, atributo que aceite nulo;
3. Os atributos identificadores devem ser o conjunto mínimo que pode identificar cada instância de uma entidade.
4. Não deve conter informação volátil

**Chave Estrangeira (FOREIGN KEY)**

È uma Chave que é primária numa tabela, mas que aparece em uma outra tabela. Este tipo de chave é utilizado para criar os relacionamentos entre as tabelas.

Com essa chave estrangeira, podemos facilitar as consultas e fazer cruzamento de dados através destas referências, o que poderia gerar uma consulta que iria pegar o nome do produto e o nome da categoria que ele pertence para exibirmos em uma listagem dos dados.

Perceba que a chave estrangeira não será única dentro da tabela de produtos, já que podemos ter vários produtos de uma categoria. Já no caso da chave primária, sempre será e deverá ser única.

### Tipos de Relacionamentos

De acordo com a cardinalidade existem 3 tipos básicos de relacionamentos entre as entidades.

* Relacionamentos um para um
* Relacionamentos um para muitos
* Relacionamentos muitos para muitos

**Um para UM 1:1**

São relacionamentos em que uma ocorrência de uma entidade em A está associada no máximo a uma ocorrência em uma entidade B e uma ocorrência na entidade B está associada no máximo a uma ocorrência na entidade A.

**Exemplo**: Um esposo só pode casar com uma esposa. Uma esposa só pode casar com um esposo.

**Um para Muitos 1:N**

São relacionamentos em que uma ocorrência de uma entidade em A está associada há mais de uma ocorrência em uma entidade B e uma ocorrência na entidade B está associada no máximo a uma ocorrência na entidade A.

**Exemplo**: Um aluno poder frequentar vários cursos no centro de formação, vários curso podem ser frequentado por um aluno.

**Muitos para Muitos**

Em um relacionamento m:m, criamos uma terceira entidade, chamada entidade associativa que é usada para associar as entidades por meio de dois relacionamentos 1:m. De maneira geral, é razoavelmente fácil nomear essa terceira entidade. Nesse exemplo, essa terceira entidade, geralmente conhecida como entidade associativa.

**Exemplo**: Um professor pode lecionar varias disciplinas e varias disciplina podem ser lecionada por vários professor.

**Por que criamos uma terceira entidade?**

Quando temos um relacionamento m:m e precisamos manter informações sobre este relacionamento, criamos uma entidade associativa para armazenar informações sobre o relacionamento. Neste caso, armazenamos dados sobre as mercadorias vendidas. Não podemos armazenar estes dados em VENDAS, pois uma venda pode ter muitos itens e uma entidade só armazena ocorrências de valores simples. Da mesma maneira, não podemos armazenar esses dados em MERCADORIAS, porque um código de mercadoria pode aparecer em muitas vendas.

**Integridade**

**Existem três tipos de integridade:**

* Integridade de domínio: os valores de cada coluna devem ser atómicos e pertencentes ao domínio do atributo;
* Integridade de entidade: os valores dos atributos que correspondem à chave primária não podem ser nulos nem iguais a outros já existentes na tabela. Notar que, como os tuplas são diferenciados pela chave primária, se os valores dos atributos correspondentes à chave primária fossem nulos não seria possível distinguir entre tuplas.
* Integridade referencial: a chave forasteira numa tabela deve referenciar sempre uma chave primária existente numa outra tabela.

## Normalização

Normalização de banco de dados é um conjunto de regras que visa, principalmente, a organização de um projeto de banco de dados para reduzir a redundância de dados, aumentar a integridade de dados e o desempenho. Para normalizar o banco de dados, deve-se examinar as colunas (atributos) de uma entidade e as relações entre entidades (tabelas), com o objetivo de se evitar anomalias observadas na inclusão, exclusão e alteração de registros.

### Formas Normais

**Primeira Forma Normal**

'Uma tabela está na 1FN, se e somente se, todos os valores das colunas da tabela forem atómicos'

Assim, podemos dizer que os relacionamentos, como definidos acima, estão necessariamente na 1FN. Uma relação está na 1FN quando todos os atributos da relação estiverem baseados em um domínio simples, não contendo grupos ou valores repetidos.

**Passagem à 1FN**

* Encontre a chave primária da tabela;
* Fique ciente de quais são as colunas da tabela que apresentam dados repetidos

para que sejam removidas;

* Crie uma tabela para esses dados repetidos, com a chave primária da anterior;
* Por fim, estabeleça relação entre a nova tabela e a principal.

Outra forma de identificar se a tabela não está na 1FN é verificando se existe tabela aninhadas, ou seja, mais de um registro para uma chave primária.

**Segunda Forma Normal**

Uma relação está na 2FN se, e somente se, estiver na 1FN e cada atributo não-chave for dependente da chave primária inteira, isto é, cada atributo não-chave não poderá ser dependente de apenas parte da chave.

No caso de tabelas com chave primária composta, se um atributo depende apenas de uma parte da chave primária, então esse atributo deve ser colocado em outra tabela.

**Passagem à 2FN**

* Geração de novas tabelas com DFs (Dependências Funcionais) completas.

**Terceira Forma Normal**

Uma relação R está na 3FN se ela estiver na 2FN e cada atributo não-chave de R não possuir dependência transitiva, para cada chave candidata de R. Todos os atributos dessa tabela devem ser independentes uns dos outros, ao mesmo tempo que devem ser dependentes exclusivamente da chave primária da tabela.

**Passagem à 3FN**

* Para estar na 3FN precisa estar na 2FN;
* Geração de novas tabelas com DF diretas;
* Análise de dependências funcionais entre atributos não chave;
* Verificar a dependência exclusiva da chave primária;
* Entidades na 3FN também não podem conter atributos que sejam resultados
* de algum cálculo de outro atributo.

**Quarta Forma Normal**

Uma tabela está na 4FN, se e somente se, estiver na 3FN e não existirem dependências multivaloradas.

* Redundância para representar todas as informações;
* Representação não-uniforme (repete alguns elementos ou posições nulas).

## Structured Query Language (SQL)

Linguagem de Consulta Estruturada ou SQL, é a linguagem de pesquisa declarativa padrão para banco de dados relacional (base de dados relacional). Muitas das características originais do SQL foram inspiradas na álgebra relacional.

A linguagem é um grande padrão de banco de dados. Isto decorre da sua simplicidade e facilidade de uso. Ela se diferencia de outras linguagens de consulta a banco de dados no sentido em que uma consulta SQL especifica a forma do resultado e não o caminho para chegar a ele. Ela é uma linguagem declarativa em oposição a outras linguagens procedurais. Isto reduz o ciclo de aprendizado daqueles que se iniciam na linguagem.

### Subconjuntos do SQL

A linguagem SQL é dividida em subconjuntos de acordo com as operações que queremos efetuar sobre um banco de dados, tais como:

**DML - Linguagem de Manipulação de Dados**

O primeiro grupo é a DML (Data Manipulation Language - Linguagem de manipulação de dados). DML é um subconjunto da linguagem SQL que é utilizado para realizar inclusões, consultas, alterações e exclusões de dados presentes em registros. Estas tarefas podem ser executadas em vários registros de diversas tabelas ao mesmo tempo. Os comandos que realizam respectivamente as funções acima referidas são INSERT, SELECT, UPDATE e DELETE.

**DDL - Linguagem de Definição de Dado**s

O segundo grupo é a DDL (Data Definition Language - Linguagem de Definição de Dados). Uma DDL permite ao utilizador definir tabelas novas e elementos associados. A maioria dos bancos de dados de SQL comerciais tem extensões proprietárias no DDL.

Os comandos básicos da DDL são poucos:

**CREATE:** cria um objeto (uma Tabela, por exemplo) dentro da base de dados.

**DROP:** apaga um objeto do banco de dados.

Alguns sistemas de banco de dados usam o comando ALTER, que permite ao usuário alterar um objeto, por exemplo, adicionando uma coluna a uma tabela existente.

Outros comandos DDL:

* CREATE TABLE
* CREATE INDEX
* CREATE VIEW
* ALTER TABLE
* DROP VIEW

**DCL - Linguagem de Controle de Dados**

O terceiro grupo é o DCL (Data Control Language - Linguagem de Controle de Dados). DCL controla os aspectos de autorização de dados e licenças de usuários para controlar quem tem acesso para ver ou manipular dados dentro do banco de dados.

Duas palavras-chaves da DCL:

**GRANT -** autoriza ao usuário executar ou setar operações.

**REVOKE -** remove ou restringe a capacidade de um usuário de executar operações.

**DTL - Linguagem de Transação de Dados**

* BEGIN WORK (ou START TRANSACTION, dependendo do dialeto SQL)Pode ser usado para marcar o começo de uma transação de banco de dados que pode ser completada ou não.
* COMMIT finaliza uma transação dentro de um sistema de gerenciamento de banco de dados.
* ROLLBACK faz com que as mudanças nos dados existentes desde o último

**COMMIT** ou **ROLLBACK** sejam descartadas. COMMIT e ROLLBACK interagem com áreas de controle como transação e locação. Ambos terminam qualquer transação aberta e liberam qualquer cadeado ligado a dados. Na ausência de um BEGIN WORK ou uma declaração semelhante, a semântica de SQL é dependente da implementação.

**DQL - Linguagem de Consulta de Dados**

Embora tenha apenas um comando, a DQL é a parte da SQL mais utilizada. O comando SELECT permite ao usuário especificar uma consulta ("query") como uma descrição do resultado desejado. Esse comando é composto de várias cláusulas e opções, possibilitando elaborar consultas das mais simples às mais elaboradas.



# Electron.js

## Desktop Application (Aplicação Desktop)

Uma aplicação desktop é um é uma peça autônoma que é executada na área de trabalho ou no laptop. Essas aplicações de desktop são desenvolvidas em linguagens com VB, C#, Java etc. Aplicações desktop são famosas entre os usuários devido as seguintes benefícios:

* Aplicativos disponíveis off-line.
* Aplicativo Possui uma resposta mais rápida.
* Fornece uma experiência de usuário rica.

## O que é o Eelectron

Electron é um framework voltado para o desenvolvimento desktop utilizando as principais tecnologias web disponíveis, foi desenvolvida pelo Github, inicialmente o eléctron foi projetado para o Editor da Atom. A seguinte figura mostra a tela do Atom.

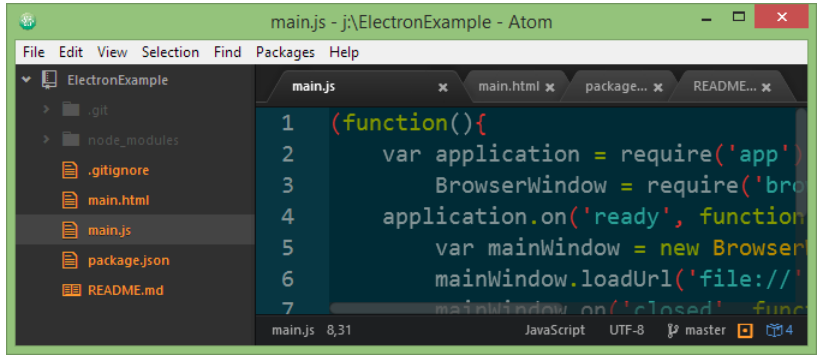


Figura 1 - Tela do Atom.

Da captura de tela anterior podemos observar que parece um aplicativo de desktop nativo, enquanto que no background ele usa a framework do electron com as tecnologias web como HTML, CSS e JavaScript. Na figura anterior podemos encontrar menus e barra de status semelhantes aos de uma janela nativa. Há um monte de outras aplicações desktop que foram construídas no Electron, as mais notáveis são:

* Stack cliente for Mac
* Visual Studio Code
* GitKraken git client
* Wordpress desktop app
* Nuclide IDE
* Wagon SQL Editor
* Nylas Email Client

**Criar uma aplicação com Electron significa:**

* Sistema de Código único, cross-platfrom.
* Usar padrões HTML 5
* Usar Bootstrap, Angular, React, jQuery e mais.
* Uma única versão de navegador como alvo.
* Não é necessário usar o tempo de download para arquivos locais.

## Blocos de construção do Electron

A seguinte figura mostra os três blocos de construção mais importantes do electron.

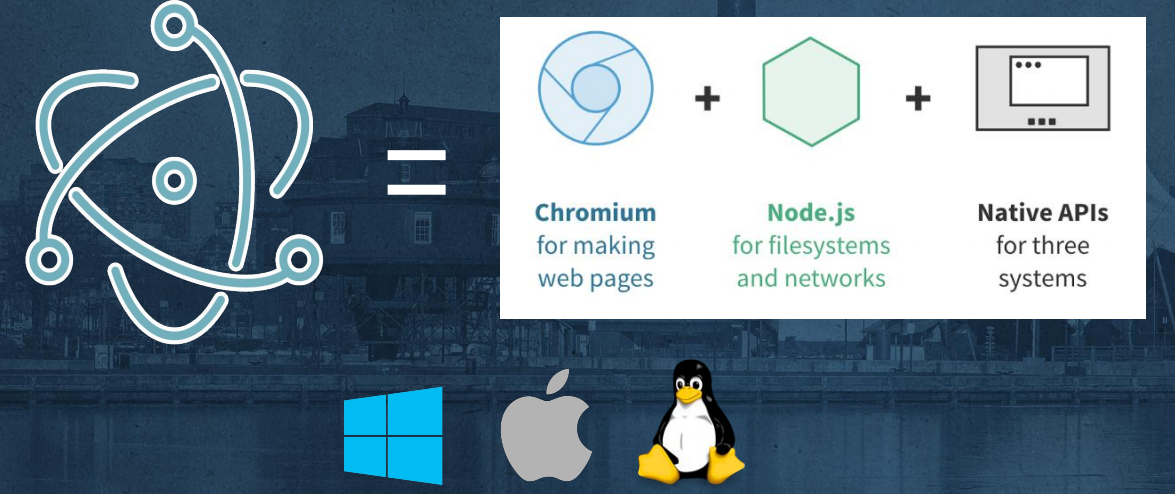


Figura 2 - Blocos de Construção do electron.

**Chormium:** é o navegador mais famoso projetado pela equipa do Google. É um projeto de código aberto, Chromiun se encarrega do processo de Renderização (pacotes web).

**Node.js :** trata-se de um software open-source, cross-platfrom, de um runtime de javaScript que executa código de javaScript a nível backend e frontend.

**Native APIs:** é leve interface de programação de aplicativo(API) usada pelo Windows NT e modo de usuário. No elecron é responsável por fazer a aplicação funcionar nos 3 sistemas operativos: Windows, MAC e Linux.

Mas podemos resumir em 2 grandes blocos de construção Chromiun + Node.js



Figura 3 - Bloco de construção do electron forma resumida.

**Como funciona o electron**

Uma aplicação desktop construída em electron é controlada por dois processos distintos, processo principal e processo de Renderização.

**Processo Principal (Main Process):** Processo responsável por criar as páginas web usando uma instância BrowserWindow ( navegador em forma de janela desktop) e recursos da URL. Em outras palavras, o processo principal é o processo do navegador.

**Processo de Renderização (Render Process):** é o processo responsável por apresentar as páginas web no BorwserWindow.

## Módulos do Electron

**Módulos do Processo Principal ( Main Process Modules)**

O Processo Principal controla a inicialização da aplicação electron e cria o BorwserWindow que exibirá o conteúdo. Para conseguir isso ele tem controlo sobre uma lista de módulos. O seguinte diagrama apresenta os principais módulos específicos do processo:

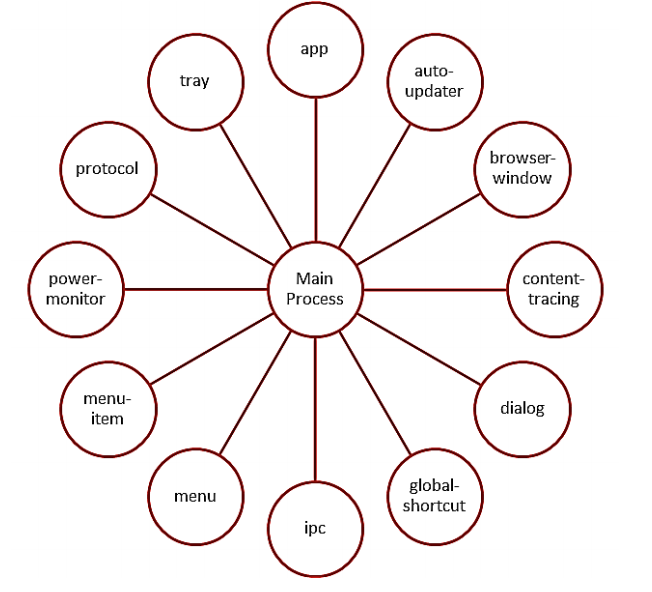


Figura 4 - Módulos do processo principal.

**Módulos do Processo de Renderização (Render Process Modules)**

O Processo de Renderização controla o aplicativo do electron para apresentar o conteúdo na janela do navegador. A Seguinte figura mostra os módulos específicos do processo:

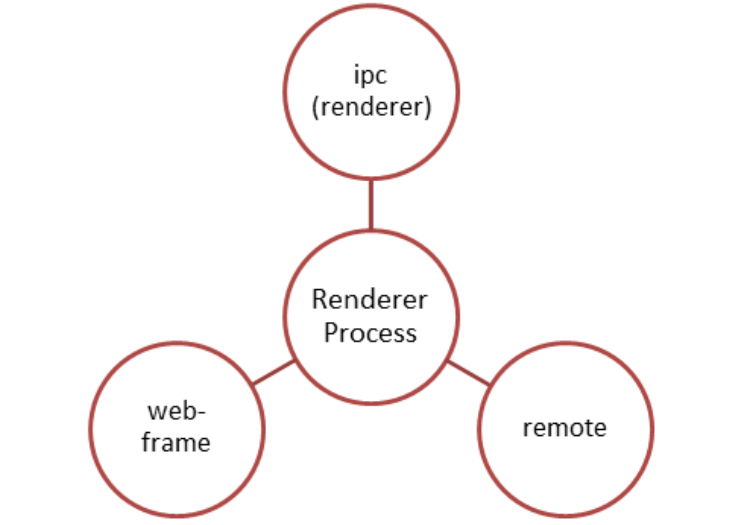


Figura 5 - Módula do processo de Renderização.

# Introdução a paginas web

## Origem da Internet

A internet é uma rede global, que consiste em milhares de redes independentes de  
computadores, de empresas privadas, entidades governamentais e instituições científicas e educativas. Surgiu por volta de 1970, designado por Arpanet, com o objetivo principal de partilhar informações entre grupo de instituições científicas e universidades desenvolvidas no projeto. Esta rede global, torna viável para os computadores enviar mensagens entre si. Uma das acções mais importantes realizadas na Web é pesquisar informações variadas para diferentes fins. Para auxiliar esta tarefa de procurar informações nas páginas Web o utilizador tem à sua disposição os chamados “motores de pesquisa”, “motores de busca” ou search engines. Por exemplo, a Google (www.google.com), a Yahoo (www.yahoo.com), entre outros motores de pesquisa.

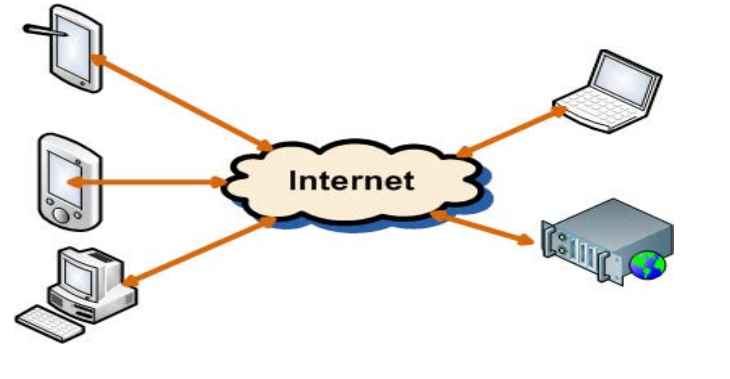


Figura 6 - Origem da Internet.

## A Web ou a WWW

A World Wide Web (WWW), foi criada por Tim Berners Lee em 1989, com o objetivo de, trocar as informações entre os pesquisadores. A WWW é o programa utilizado para obter informações solicitadas pelos utilizadores. Progressivamente, foi-se alimentando com a introdução de novas aplicações Web, não só para ambientes de pesquisas, como também para fins comerciais disponíveis na internet. Pode-se dizer que atualmente, a Web é o universo de informações sem fronteiras com várias caraterísticas. Exemplo, interface gráfica consistente, integração de um conjunto de tecnologias e documentos, etc.

## Provedor de Serviço de Internet (ISP)

Um ISP é simplesmente uma empresa que fornece o acesso à Internet. Cada computador que está conectado à Internet faz parte de uma rede, inclusive um em sua casa. Quando a conexão é estabelecida com o seu ISP, isto significa que a sua máquina se torna parte de uma rede maior. A Internet é uma rede de redes.

## Como funciona a web

### Protocolo HTTP

O protocolo HTTP permite o transporte de recursos na Web. Os clientes (browser) envia uma solicitação HTTP, para aceder os recursos do servidor, e o servidor Web responde às solicitações através da emissão de respostas HTTP. O HTTP é a sigla Protocolo de Transferência de Hypertexto (Hypertext Transfer Protocol), é um protocolo de comunicação utilizado para sistemas de informação de hipermídia, distribuídos e colaborativos. Esta atividade apresenta uma visão geral das transações do protocolo HTTP utilizando um navegador Web (Google Chrome/ Firefox).

Para desenvolver e depurar aplicações Web, é importante ter bom entendimento de como o protocolo HTTP funciona e os servidores Web.

**Exemplo de transações HTTP**

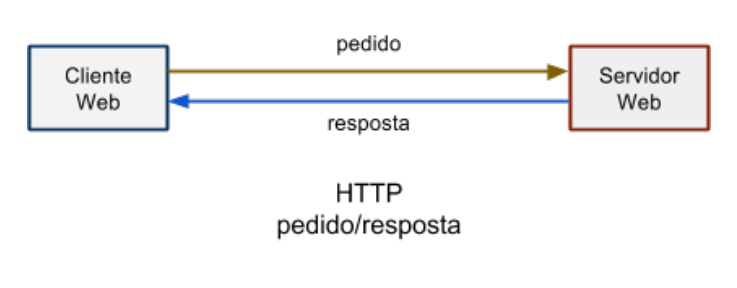


Figura 7 - Exemplo de transação HTTP

**O modelo de comunicação: Cliente/Servidor**

O serviço Web é baseado na arquitetura Cliente/Servidor, ou seja, o cliente solicita um serviço e o servidor, por sua vez, responde à solicitação. A figura seguinte mostra a  
comunicação entre o cliente e o servidor.



Figura 8 - Modelo Cliente/Servidor.

**Localizador Universal de Recursos (URL)**

Os recursos disponibilizados na Web têm um único endereço para o acesso a qualquer página Web. Os endereços utilizados são chamados de URLs (Localizador Universal de Recursos).

**Exemplo:**

http://www.avu.org/Service/content-development.html

**Este endereço identifica:**

* O protocolo de acesso ao recurso (HTTP);
* O servidor remoto a ser contactado www.avu.org;
* Caminho do diretório até ao recurso /Service/;
* O recurso ou documento Web obtido /content-development.htm.

URL (Uniform Resource Locator) são nomes padrão dos recursos de internet em geral (incluindo os recursos da web em particular). Eles indicam a localização destes recursos (o servidor que os contém) e como interagir com eles para o obter (protocolo padrão).

Estes são os endereços que são utilizados para localizar exclusivamente cada recurso disponível na Web. Veja o exemplo a seguir, solicitação de um recurso através de um navegador para o servidor remoto da AVU:

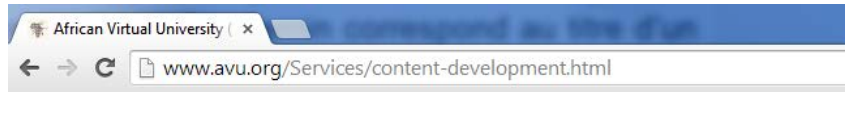


Figura 9 - Exemplo de uma URL.

**O que é um website**

Um website, também chamado apenas de site, é um conjunto de arquivos HTML conectado através de links de hipertexto, armazenado em um servidor web, ou seja, um computador permanentemente ligado à Internet.

.

# Desenvolvimento Web

## As linguagens Web

O termo aplicação Web é utilizado principalmente para descrever a utilização das linguagens para a criação de conteúdos web dinâmica. O ecossistema de desenvolvimento web é composto de linguagens de marcação e linguagens de programação.

As linguagens de marcação (Markup) são utilizadas para escrever páginas Web, para colocar os hiperlinks e definir as suas aparências (layout) nos navegadores.

### Linguagens de marcação

HTML (HyperText Markup Language): Atualmente, é muito difícil para um desenvolvedor web suceder sem ter o domínio da linguagem Web;  
CSS (Cascading Style Sheet): utilizada para definir a apresentação de documentos escritos em uma linguagem de marcação, como HTML ou XML. Seu principal benefício é prover a separação entre o formato e o conteúdo de um documento;

XML (Extensible Markup Language): esta linguagem é amplamente utilizada para troca de dados na internet. As linguagens do lado do servidor web permitem colocar o código interpretável em HTML e assim gerar o conteúdo da página web com o processamento no servidor.

### Linguagens do tipo client side script (script do lado do cliente)

JavaScript, frequentemente abreviado como JS, é uma linguagem de programação interpretada de alto nível, caracterizada também, como dinâmica, fracamente tipada, prototype-based e multi-paradigma. Juntamente com HTML e CSS, o JavaScript é uma das três principais tecnologias da World Wide Web. JavaScript permite páginas da Web interativas e, portanto, é uma parte essencial dos aplicativos da web. A grande maioria dos sites usa, e todos os principais navegadores têm um mecanismo JavaScript dedicado para executá-lo. É atualmente a principal linguagem para programação client-side em navegadores web. É também bastante utilizada do lado do servidor através de ambientes como o node.js.

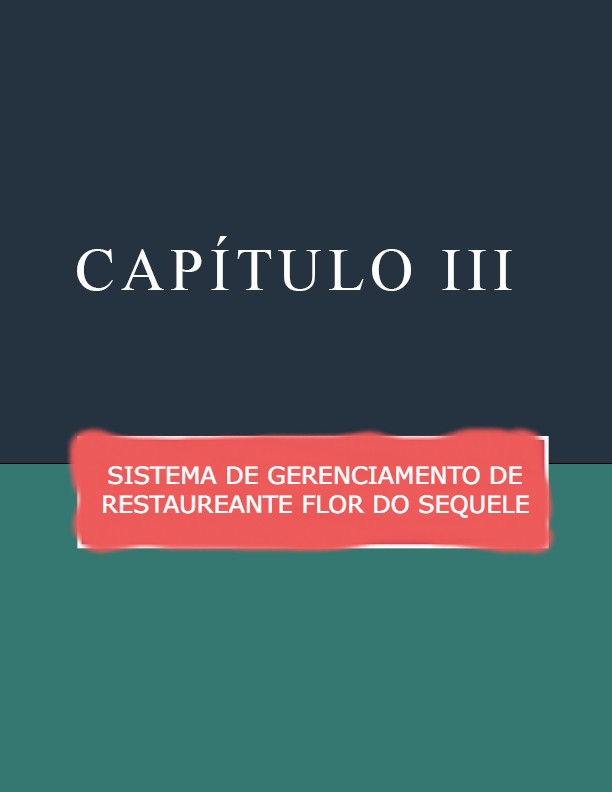
### Linguagens do tipo server side script (script do lado do servidor)

PHP: Esta linguagem cuja sintaxe é baseada na linguagem em C, é, certamente, a linguagem mais popular da Web. A evidência empírica deste sucesso é o grande número de CMS escrito em PHP. Exemplo: Joomla, Drupal, Wordpress, Magento,... Sua popularidade é devido à sua facilidade de aprendizagem para iniciantes.

JSP/Servlet: É uma linguagem muito popular no desenvolvimento de sistemas relativamente complexos;

ASP.NET: É uma linguagem da Microsoft. É uma linguagem que oferece uma série de  
componentes de interface (Widget). Facilita o desenvolvimento quando o ambiente de  
implementação é a Microsoft;

Python, Ruby e outros: Python e Ruby são linguagens que se encontram em crescimento através do uso dos seus respetivos frameworks: Django e Ruby on Rails.



# Instituto Superior Politécnico de Kangonjo – Área de Formação de Informática

# Análise do Sistema

A aplicação “Desktop ” terá como objetivo ajudar a organizar todas as tarefas do Restaurante , ou gerenciamento adequado de todos os seus recursos, sejam eles físicos, humanos ou financeiros.

## Requisitos Não Funcionais

1. O sistema não Deve Permitir a Exclusão de Produtos já vendidos
2. O Sistema deve impedir a duplicação de dados
3. O sistema não deve permitir Acesso sem autenticação prévia do Usuário
4. O sistema deve permitir a recuperação Prévia dos dados
5. O tempo de Resposta Prévia do Sistema não deve ultrapassar 30 Segundos

Requisitos de Confiabilidade

## Disponibilidade

RNF01 - O sistema deve ter uma disponibilidade 999 / 1000.

## Taxa de Ocorrencias de Falhas

RNF01 - O sistema deve ter uma Ocorrencia de Falhas de 2 / 1000.

## Probabilidade de Indisponibilidade

RNF01 - O sistema devera apresentar sucesso nas operacoes 999 / 1000.

## Tempo medio para Falhas

RNF01 – O sistema devera apresentar 500 horas de tempo medio sem falhas no cadastros.

Requisitos de Desempenho

## Requisitos de Resposta

RNF01 – O sistema deve apresentar a Interface de Login em 1 Segundo.

## Requisitos de Processamento

RNF01- O sistema deve realizar 10 transações por segundo.

## Requisitos de Temporização

RNF01 – O sistema deve Efectuar Leitura de dados 5 vezes por segundo

## Requisitos de Espaço

RNF01 – O sstema não deve consumir alem de 40 % da memória Principal.

## Requisitos de Portabilidade

RNF01 – O sistema deve executar eficientemente em Sistemas Windows.

## Requisitos de Segurança

RNF01 – O sistema deve exigir autenticação prévia dos seus usuários.

RNF02 – O sistema deve restringir Acesso de acordo com o tipo de Usuário.

RNF03 – O sistema deve Fazer backup de dados a cada 24 h Horas .

## Requisitos de Usabilidade

RNF01 – O sistema deve ter interface para Usuarios Iniciantes.

RNF03 – O sistema deve garantir mensagens que informem erros de utilização.

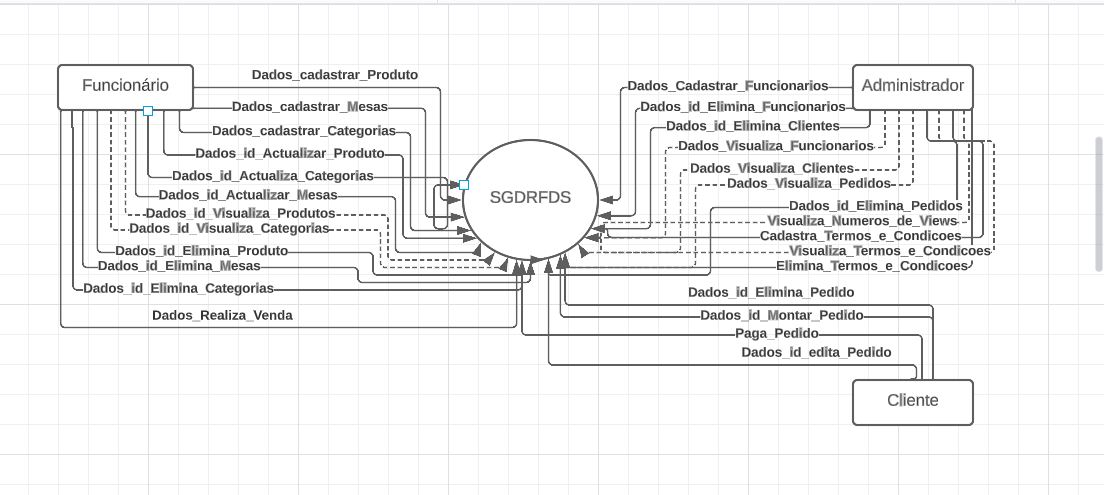
## 

## Requisitos Funcionais

1. Administrador é cadastrado no sistema [E, F]
   1. Sistema faz a validação dos dados inseridos [I, C]
   2. Administrador faz o login pela primeira vez [E, F]
   3. Sistema verifica os dados Inseridos para efetuar login [I, C]
2. Administrador gere Funcionários
   1. Administrador cadastra Funcionários [E, F]
   2. Administrador edita Funcionários [E, C]
   3. Administrador elimina Funcionários [E, C]
   4. Administrador visualiza Funcionários [E, C]
3. Funcionário gere Produtos
   1. Funcionário cadastra Produtos [E, F]
   2. Funcionário edita Produtos [E, C]
   3. Funcionário elimina Produtos [E, C]
   4. Funcionário visualiza Produtos [E, C]
4. Funcionário gere Categorias
   1. Funcionário cadastra categoria [E, F]
   2. Funcionário edita categoria [E, C]
   3. Funcionário elimina categoria [E, C]
   4. Funcionário visualiza categoria [E, C]
5. Funcionário gere Pedidos
   1. Funcionário elimina Pedidos [E, C]
   2. Funcionário visualiza Pedidos [E, C]
   3. Funcionário edita Pedidos [E, C]
6. Funcionário gere Mesas
   1. Funcionário cadastra Mesa [E, F]
   2. Funcionário edita Mesa [E, C]
   3. Funcionário elimina Mesa [E, C]
   4. Funcionário visualiza Mesa [E, C]
7. Funcionário gere Suas Informações
   1. Funcionário edita suas Informações [E, F]
   2. Funcionário elimina suas Informações [E, C]
   3. Funcionário Visualiza suas Informções [E, C]
8. Administrador gere Suas Informações
   1. Administrador edita suas Informações [E, F]
   2. Administrador elimina suas Informações [E, C]
   3. Administrador Visualiza suas Informções [E, C]

## Regra de Negócio

## Diagrama de contexto



### Dicionário de dados

**Usuário** = @num\_de\_identificaca­o + nome + senha + email  
**Administrador** = @ num\_de\_identificacao­ + DadosPessoais  
**DadosPessoais** = BI + sexo  
**clientes** = @num\_de\_identificaca­o + DadosPessoais

# Cronograma de atividades

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Atividades | Duração em meses | | | | | | | | | | | |
| Jan | Fev | Mar | Abr | Mai | Jun | Jul | Ago | Set | Out | Nov | Dez |
| Escolha do tema |  |  |  |  |  |  |  |  |  |  |  |  |
| Pesquisa |  |  |  |  |  |  |  |  |  |  |  |  |
| Análise do Sistema |  |  |  |  |  |  |  |  |  |  |  |  |
| Elaboração do  relatório |  |  |  |  |  |  |  |  |  |  |  |  |
| Revisão do relatório |  |  |  |  |  |  |  |  |  |  |  |  |
| Criação da aplicação |  |  |  |  |  |  |  |  |  |  |  |  |
| Revisão geral da aplicação |  |  |  |  |  |  |  |  |  |  |  |  |
| Conclusão definitiva da aplicação |  |  |  |  |  |  |  |  |  |  |  |  |

Tabela 1 - Cronograma de Atividades

**Banco de Dados**

Para armazenarmos os dados do Sistema usamos o SGBD MySQL. O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês Structured Query Language) como interface. É atualmente um dos sistemas de gerenciamento de bancos de dados mais populares da Oracle Corporation, com mais de 10 milhões de instalações pelo mundo.

# Modelo Entidade Relacionamento

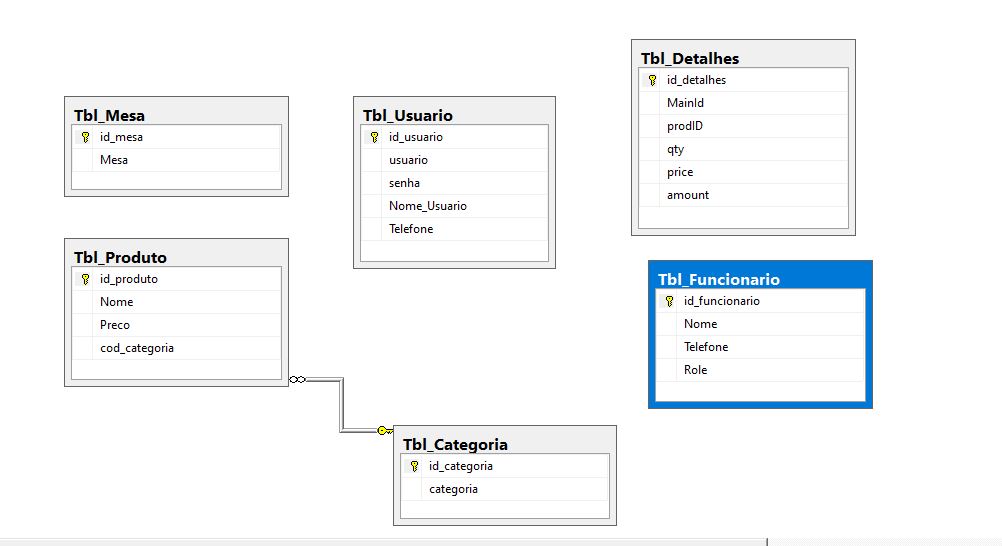


Figura 10 - Modelo Entidade Relacionamento “um”.

## Ferramentas usadas para construção da base de dados

**Wampserver**

Wampserver é um software que efetua a instalação automática de um conjunto de softwares no computador, de modo a facilitar a configuração de um software interpretador de scripts local e um banco de dados no sistema Windows. Conjunto desenvolvido pela PHP Team sob licença GNU General Public License. A partir da versão 2 ocorreu a mudança do nome, de WAMP5 para WampServer. Possui suporte a adição de add-ons (funções extras), disponíveis por download, como por exemplo, a função de ativar o uso de PHP 4 ao invés do PHP 5.Uma ferramenta que auxilia no desenvolvimento, permitindo aos programadores de websites testarem o trabalho em seus próprios computadores, sem necessitar acesso à Internet.

**MySql Workbench**

O MySQL Workbench é uma ferramenta de design de banco de dados visual que integra desenvolvimento, administração, design de banco de dados, criação e manutenção de SQL em um único ambiente de desenvolvimento integrado para o sistema de banco de dados MySQL.

**A Aplicação Web e Desktop**

A aplicação web foi construída utilizando as tecnologias webs mais populares, a aplicação desktop foi construída utilizando o electron.js, que utiliza tecnologias web para desenvolvimento de aplicações desktop.

**O Padrão MVC (Model-View-Controller)**

Para a nossa aplicação utilizamos o padrão MVC, que possibilita a divisão do projeto em camadas muito bem definidas. Cada uma delas, o Model, o Controller e a View, executa o que lhe é definido e nada mais do que isso. A utilização do padrão MVC trás como benefício isolar as regras de negócios da lógica de apresentação, a interface com o usuário. Isto possibilita a existência de várias interfaces com o usuário que podem ser modificadas sem que haja a necessidade da alteração das regras de negócios, proporcionando assim muito mais flexibilidade e oportunidades de reuso das classes. Uma das características de um padrão de projeto é poder aplicá-lo em sistemas distintos. O padrão MVC pode ser utilizado em vários tipos de projetos como, por exemplo, desktop, web e mobile. O modelo (model) consiste nos dados da aplicação, regras de negócios, lógica e funções.

**Camada de apresentação ou visualização** - Não se dedica em saber como o conhecimento foi retirado ou de onde ela foi obtida, apenas mostra a referência. Segundo Gamma et al (2006), ”A abordagem MVC separa a View e Model pela organização de um protocolo inserção/notificação (subscribe/notify). Uma View deve garantir que sua expressão reflita o estado do Model. Sempre que os dados do Model mudam, o Model altera as Views que dependem dele. Em resposta, cada View tem a oportunidade de modificarse”. Adiciona os elementos de exibição ao usuário HTML, ASP, XML, Applets. É a camada de interface com o usuário. É utilizada para receber a entrada de dados e apresentar visualmente o resultado.

**Camada de lógica da Aplicação** - É o coração da execução. Responsável por tudo que a aplicação vai fazer controle de um ou mais elementos de dados, responde a perguntas sobre a sua condição e responde a instruções para mudar de posição. O modelo sabe o que o aplicativo quer fazer e é a principal estrutura computacional da arquitetura, pois é ele quem modela o problema que está se tentando resolver. Modela os dados e o comportamento por atrás do processo de negócios se preocupa apenas com o armazenamento, manipulação e geração de dados. É um encapsulamento de dados e de comportamento independente da apresentação.

**Camada de Controle** - É responsável por interpretar as ações de entrada através do mouse e teclado realizados pelo usuário. O Controle (Controller) envia essas ações para o Modelo (Model) e para a janela de visualização (View) onde serão realizadas as alterações necessárias.

**POO**

Programação orientada a objetos (POO, ou OOP segundo as suas siglas em inglês) é um paradigma de programação baseado no conceito de "objetos", que podem conter dados na forma de campos, também conhecidos como atributos, e códigos, na forma de procedimentos, também conhecidos como métodos. Uma característica de objetos é que um procedimento de objeto pode acessar, e geralmente modificar, os campos de dados do objeto com o qual eles estão associados (objetos possuem uma noção de "this" (este) ou "self" (próprio)).

**REST API´**

# Ferramentas usadas para construção da aplicação

**Visual Studio Code**

O Visual Studio Code é um editor de código-fonte desenvolvido pela Microsoft para Windows, Linux e macOS. Ele inclui suporte para depuração, controle Git incorporado, realce de sintaxe, complementação inteligente de código, snippets e refatoração de código. Ele também é customizável, fazendo com que os usuários possam mudar o tema do editor, teclas de atalho e preferências. Ele é um software livre e de código aberto, apesar do download oficial estar sob uma licença proprietária. O Visual Studio Code é baseado no Electron, uma *framework* que é usada para desenvolver aplicativos Node.js para o desktop rodando no motor de layout Blink. Apesar de usar o Electron como *framework*, o software não usa o Atom e em seu lugar emprega o mesmo componente editor (codenomeado "Monaco") usado no Visual Studio Team Services (anteriormente chamado de Visual Studio Online)

**Google Chrome**

O Google Chrome é um navegador de internet, desenvolvido pela companhia Google com visual minimalista. Foi lançado pela primeira vez em setembro de 2008, para o Microsoft Windows, e mais tarde foi portado para Linux, Mac, iOS e Android. Compilado com base em componentes de código licenciado como o motor de renderização o WebKit. O Google Chrome também é o principal componente do Chrome OS onde funciona como uma plataforma para executar aplicativos da web. e sua estrutura de desenvolvimento de aplicações (framework).

**Tecnologias usadas para construção do front-end**

**Bootstrap**

Bootstrap é um framework web com código-fonte aberto para desenvolvimento de componentes de interface e front-end para sites e aplicações web usando HTML, CSS e JavaScript, baseado em modelos de design para a tipografia, melhorando a experiência do usuário em um site amigável e responsivo. O Bootstrap é um dos projetos mais bem avaliado no site GitHub, com mais de 111 600 estrelas e 51 500 forks. Em 29 de outubro de 2014, Mark Otto anunciou que o Bootstrap 4 estava em desenvolvimento. Em 6 de setembro de 2016, Mark suspendeu os trabalhos no Bootstrap 3, a fim de liberar mais tempo para trabalhar no Bootstrap 4. Mais de 4.000 mudanças foram feitas para o código base do Bootstrap 4 até agora.

**JQuery**

jQuery é uma biblioteca de funções JavaScript que interage com o HTML, desenvolvida para simplificar os scripts interpretados no navegador do cliente (client-side). Usada por cerca de 74.4% dos 10 mil sites mais visitados do mundo, jQuery é a mais popular das bibliotecas JavaScript.

Esta biblioteca é um projeto da Fundação JS (em inglês: JS Foundation), que por sua vez faz parte da Fundação Linux (em inglês: Linux Foundation) e seu desenvolvimento é realizado pelo time jQuery (em inglês: The jQuery Team).

* Principais funcionalidades do jQuery:
* Resolução da incompatibilidade entre os navegadores.
* Redução de código.
* Reutilização do código através de plugins.
* Utilização de uma vasta quantidade de plugins criados por outros

**Ajax**

Ajax é o uso metodológico de tecnologias como Javascript e XML, providas por navegadores, para tornar páginas Web mais interativas com o usuário, utilizando-se de solicitações assíncronas de informações. Foi inicialmente desenvolvida pelo estudioso Jessé James Garret e mais tarde por diversas associações.

**Segurança na Aplicação e no Banco de Dados**

Uma das maiores ameaça que as organizações enfrentam nos dias actuais são os softwares desprotegidos, uma vez que os usuários mal-intencionados voltam sua atenção para os aplicativos e softwares que formam a infra-estrutura de Tecnologia de Informação (T.I) das organizações, assim a melhor forma de protecção é criar um software seguro desde o início. Um sistema a ser desenvolvido deve se ter em conta alguns pressupostos de segurança de modo a evitar problemas tais como:

* Pessoas mal-intencionadas verem, modificarem os dados de outros usuários.
* Pessoas mal intencionadas submeterem sql injection nos formulario
* Utilizadores não autorizados, terem acesso a serviços remotos.

**Criptografia**

A criptografia é uma das melhores soluções para se armazenar ou transferir dados, suponha que alguma informação caia em mãos erradas, se ela estiver cifrada, ou seja, que tenha sido usado um algoritmo de criptografia, a pessoa que a obteve terá dificuldades em conseguir encontrar o significado real, pois a criptografia mascara a informação trocando os caracteres por outros, disfarçando o sentido das palavras.

# Anexos

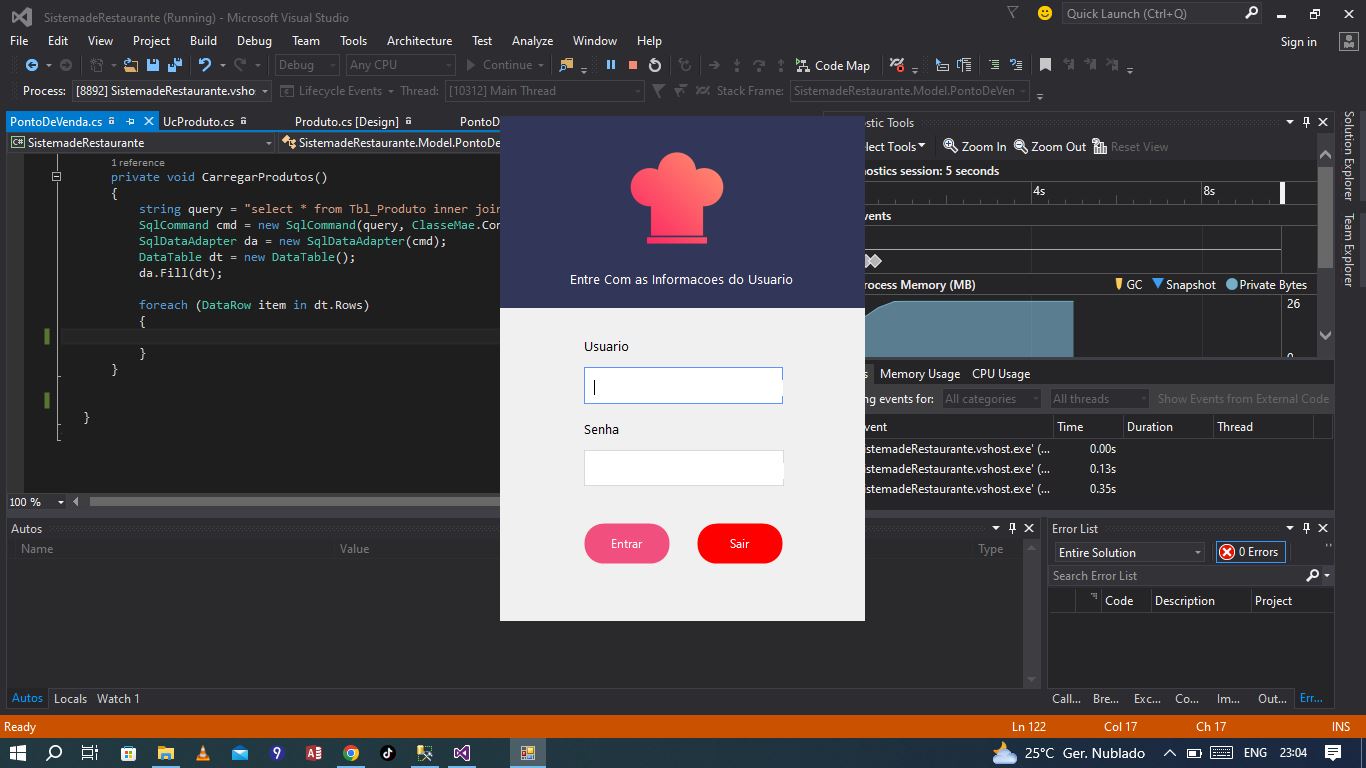


Figura 12 – Login - parte DESKTOP

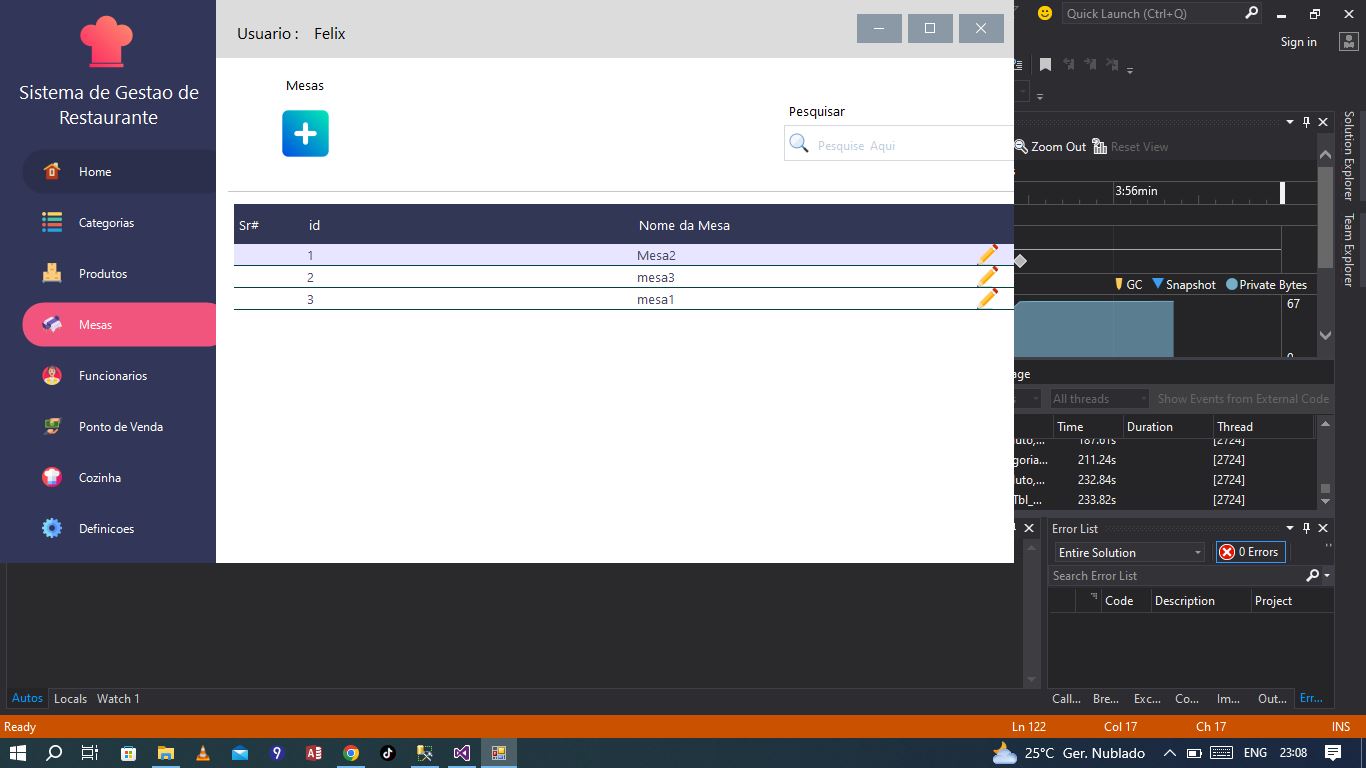


Figura 13 - Menu inicial - parte Desktop

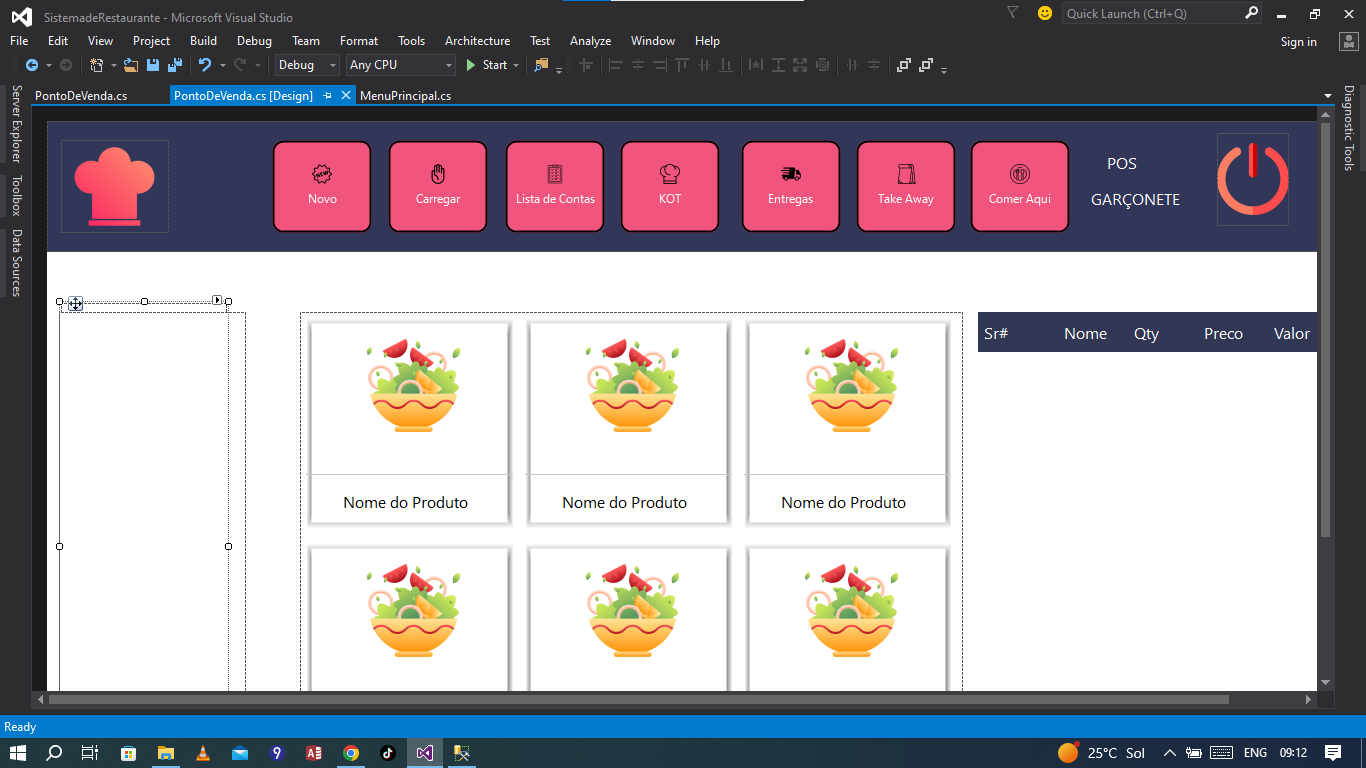


Figura 14 – Ponto de Venda – parte Desktop

# Conclusão

Concluí que com a criação deste site, haverá uma grande melhoria no que concerne á Gerenciamento de Restaurante Flor do Sequele.

Este trabalho foi desenvolvido com o intuito de Apesar do principal objetivo de um restaurante ser entregar uma comida de qualidade e um serviço acolhedor, tal objetivo não advém somente de um bom cozinheiro. Assim, precisa de um gerenciamento adequado de todos os seus recursos, sejam eles físicos, humanos ou financeiros.

# Referencias Bibliográficas

Electron. Livro “BuildingCrossPlatfrom-DesktopAplication-WithElectron.pdf”, consultado em 2020.

* Electron. Livro “DC-BA\_-\_ Supercharge\_with\_Electron.pdf”, consultado em 2020.
* Electron. Livro ”Electron-sample- Sandeep Kumar Patel”, consultado em 2020.
* Base de Dados. Devimedia, 2020. Disponível em: **https://www.devmedia.com.br/conceitos-fundamentais-de-banco-de-dados/1649**.
* Base de Dados, **https://pt.wikipedia.org/wiki/Base\_de\_dados\_em\_rede**.
* Base de Dados, **https://www.lucidchart.com/pages/database-diagram/database-models**.
* Web Page. Wikipédia, 2020. Disponível em: **https://pt.wikipedia.org/wiki/webpage**. Acesso em: 03 de ago. De 2020.
* Javascript. Wikipédia, 2020. Disponível em: **https://pt.wikipedia.org/wiki/JavaScript**. Acesso em: 03 de ago. De 2020.
* PHP. Wikipédia, 2020. Disponível em: **https://pt.wikipedia.org/wiki/php**. Acesso em: 03 de ago. De 2020.