**CENTRO PAULA SOUZA**

**ETEC PEDRO FERREIRA ALVES**

**Ensino Médio com Habilitação Profissional de Técnico em Desenvolvimento de Sistemas**

**Arthur Candido de Melo**

**Avalone Silva Cabrera**

**Christian Martins Teixeira**

**Gabriel José Pires de Avila**

**Gustavo Gutierrez**

**João Henrique de Oliveira**

**Leonardo Henrique dos Santos**

**MOGI MIRIM**

**2022**

**Arthur Candido de Melo**

**Avalone Silva Cabrera**

**Christian Martins Teixeira**

**Gabriel José Pires de Avila**

**Gustavo Gutierrez**

**João Henrique de Oliveira**

**Leonardo Henrique dos Santos**

**Easy Chefs**

Trabalho de Conclusão de Curso apresentado ao Curso Técnico do Ensino Médio com Habilitação Profissional de Técnico em Desenvolvimento de Sistemas da Etec Pedro Ferreira Alves, orientado pelos Prof. M. Sc. Marcos Roberto de Morais e Prof. Esp. Sinzomar Gomes de Melo como requisito parcial para obtenção do título de técnico em Ensino Médio com Habilitação Profissional de Técnico em Desenvolvimento de Sistemas.

**MOGI MIRIM**

**2022**

O trabalho de conclusão de curso apresentado neste documento procura contribuir contra o **desperdício de alimentos**. Frente ao grande desperdício de alimentos que enfrentamos em nossa atual sociedade, o projeto EasyChefs traz uma das formas de solucionar este problema de maneira digital e intuitiva em uma plataforma online que permite aos usuários a busca de receitas a partir de ingredientes que o mesmo encontra em sua residência. A proposta desse projeto, além de reduzir o desperdício de alimentos, também visa a interatividade entre os usuários da plataforma de modo em que a causa do projeto se torna mais “descontraída”, divertida e “saborosa”. Utilizando diversas tecnologias back-end e front-end desenvolvemos a plataforma de maneira organizada utilizando métodos de desenvolvimento ágil de software, partindo desde as pesquisas e prototipagem até os testes finais em grupo.

Palavras-chave: Desperdício. Alimentos. Digital. Plataforma. Online. Receitas. Ingredientes. Tecnologias. Software.The presented work in this document seeks to contribute against the waste of food. Faced to the great waste of food in the society, the EasyChefs project bring one of the solution ways to solve this problem in a digital and intuitive method. A online plataform that allows the users to search recipes out of ingredients that himself finds in his house. The proposal of this project, besides the food waste, is to prioritize the interativity between the plataform users in way that the project cause turns into more "relaxed", funny and "delicious". Using various technologies back-end and front-end we developed the plataform in a organized way using agile software development methods, leaving from the searchings and prototyping to the final tests in group.

Keywords: Waste. Foods. Digital. Platform. Online. Revenues. Ingredients. Technologies. Software.

Sumário

[1 INTRODUÇÃO 7](#_Toc118461493)

[2 REVISÃO BIBLIOGRAFICA 8](#_Toc118461494)

[2.1 Análise de Projetos de Sistemas 8](#_Toc118461495)

[2.1.1 Elicitação de requisitos 8](#_Toc118461496)

[2.1.2 Mockup 9](#_Toc118461497)

[2.1.3 DIAGRAMA 11](#_Toc118461498)

[2.2 Paradigmas de Programação 16](#_Toc118461499)

[2.2.1 Paradigma sequencial 17](#_Toc118461500)

[2.2.1 Paradigma estruturado 18](#_Toc118461501)

[2.2.2 Orientado a Objetos 20](#_Toc118461502)

[2.3 Plataformas de desenvolvimento 36](#_Toc118461503)

[2.3.1 Plataforma Web 40](#_Toc118461504)

[2.3.2 Plataforma Desktop 49](#_Toc118461505)

[2.3.2 Plataforma Mobile 63](#_Toc118461506)

[2.4 Banco de dados 68](#_Toc118461507)

[2.4.1 Modelagem de Dados 69](#_Toc118461508)

[2.4.2 SGBDS 82](#_Toc118461509)

[2.4 Tecnologias Diversas 87](#_Toc118461510)

[2.5.1 Bootstrap 87](#_Toc118461511)

[2.5.2 Laravel 92](#_Toc118461512)

[2.5.3 Formik 94](#_Toc118461513)

[2.5.4 Sequelize 98](#_Toc118461514)

[2.5.5 Yup 109](#_Toc118461515)

[3 Desenvolvimento 114](#_Toc118461516)

[3.1 Levantamento de Requisitos 114](#_Toc118461517)

[3.1.1 Entrevista/coleta de dados 114](#_Toc118461518)

[3.1.2 Cronograma de Atividades 114](#_Toc118461519)

[3.2 Protótipo de sistema 116](#_Toc118461520)

[3.2.1 aplicativo 116](#_Toc118461521)

[3.2.1 Site 123](#_Toc118461522)

[3.3 Apresentação da Solução 127](#_Toc118461523)

[4 Considerações finais 127](#_Toc118461524)

[5 Referências 128](#_Toc118461525)

# INTRODUÇÃO

A Alimentação é algo indispensável para os seres vivos, ela é necessária para assimilar o alimento para a realização das atividades vitais, sem ela não é possível que seres vivos desempenhem funções básicas do organismo, como crescer e se reproduzir. Com a evolução da humanidade as pessoas passaram a produzir diversas variedades de alimentos, porem com esse aumento de variedade, as pessoas passaram a comprar uma quantidade de alimento mais do que necessárias, essa compra resulta com o desperdício de alimentos, que muitas vezes não passam nem a ser abertas, porem ficam na geladeira e estragam.

De acordo com (Paúl, 2021) “O chamado "Índice de Desperdício de Alimentos 2021" apresenta um número quase assustador: em 2019, foram 931 milhões de toneladas de alimentos desperdiçados. Isso sugere que 17% da produção total de alimentos do mundo foram para o lixo”. Segundo (Jesus, 2021) “Em média, cada brasileiro desperdiça 41,6 kg de alimentos por ano. “

O objetivo do presente trabalho de conclusão de curso é desenvolver um aplicativo cujo conceito principal seja evitar o desperdício de alimentos, e incentivar à pratica da culinária, reaproveitando alimentos da sua geladeira e evitando que eles sejam mal utilizados, oferecendo um grande leque de opções culinárias.

Nos últimos anos com o aumento do consumismo na sociedade, percebe-se que muitas famílias compram mais coisas do que realmente necessitam e isso vale para a alimentação também, e com essas compras mais que necessárias percebe-se que houve um grande aumento do desperdício de alimentos.

Diante desse contexto o presente tema para esse Trabalho de Conclusão de Curso é: “A Redução do desperdício de alimentos, e prática da culinária através de uma solução híbrida de tecnologia”

Espera-se que com o uso do aplicativo desenvolvido cause uma boa solução para o desperdício de alimentos novos/não utilizados seja oferecer um software que permita que a pessoa indique seus alimentos que estão prestes a estragarem e a partir desses ingredientes recomendar receitas culinárias típicas/praticas para o cliente de forma acessível permitindo que ele reutilize esses alimentos ao invés de desperdiça-los e ainda irá incentiva-lo a praticar a culinária.

# REVISÃO BIBLIOGRAFICA

Neste capitulo apresentaremos, de forma geral, conceitos da área da tecnologia de informação, e também nossas revisões de pesquisas e discussões sobre outros autores.

## 2.1 Análise de Projetos de Sistemas

Nesta seção será exibida modelos de desenvolvimento de projetos, para verificar sua funcionalidade, preparar e executar soluções para diversos outros problemas distinguido no período de análise.

2.1.1 Elicitação de requisitos

A Elicitação de requisitos tem sua etapa de extrema importância para um software obter seu sucesso pelo fato de ser um projeto de desenvolvimento pois a Elicitação é responsável pela extração de necessidade e desejos dos stakeholders que serão traduzidas em funcionalidades que serão implementadas em sua solução final de desenvolvimento.

A Elicitação é importante para a engenharia de requisitos pois lá é onde o ponto de partida para a identificação e definição os requisitos de um software. E, portanto, quanto melhor for executado as atividades dessa Elicitação melhor será definido os requisitos de software.

A Elicitação é a fase em que o profissional de TI atende a necessidade do cliente e o orienta sobre aquilo que o mesmo deseja. E também é o momento de conversa com o usuário, de sentimento sobre o que este espera que seja entregue a ele depois de finalizado e pronto para o uso.

Na Elicitação de requisitos são percebidas as necessidades do sistema e as características que esse sistema deve ter. Apesar de parecer uma etapa relativamente simples do projeto, não é, pois, existem suas determinadas fazer que apenas um erro pode comprometer todo o seu sistema. De acordo com o autor

[...] as técnicas mais comuns utilizadas em Elicitação de requisitos são as de introspecção, entrevista e análise de protocolos. A técnica de introspecção se baseia em imaginar que tipo de sistema eu iria querer se eu estivesse executando esta tarefa, utilizando este equipamento, entre outros (maxwell, 2008)

2.1.2 Mockup

O Mock-up é um modelo produzido mais para a produção do trabalho. Ele serve para mudar ideias em funcionalidades e ajudar um freguês a expandir e entender o quanto ele necessita.

Normalmente, temos uma ideia genérica do quão queremos, porém não temos certeza quão aos detalhes da produção, quanto por exemplo a posição de botões e de conteúdo, ou sobre quanto as páginas e funcionalidades irão interagir entre si.

Além disso, pode viver a espera de que um programador/desenvolvedor ou web designer contratado irá pensar nessas estruturas, em que época muitas vezes ele pretende levemente exibir um trabalho, de contrato com instruções pré-definidas. Essa dessemelhança de expectativas muitas vezes acaba em disputa.

Figura - exemplo Mockup



Fonte: https://blog.workana.com/pt/sin-categorizar/a-importancia-do-mock-up-nos-projetos-de-ti/

Ao se trabalhar com um mock-up, o freguês expõe o profissional ao que ele tem em mente e consegue perceber melhor o coeficiente de dificuldade do trabalho. Isto também facilita a ligação e a produção do projeto depois da contratação. (Gabrielle Lopes, 2021)

O mock-up também permite que seu fabricante seja testemunhado por um empenho demasiado menor. Imagine que você descobre que os seus clientes não se adaptaram com saúde a uma peça do seu programa levando ao encerramento do trabalho. O valor e época para alterações nessa fase é enorme e isto pode evitar com uma fase de testes por meio do mock-up.

Dessa forma, você pode iniciar uma parte da programação neste instante sabendo que seu produto tem boa usabilidade e que foi admitido pelo seu freguês final. A maior peça dos gerentes de projetos trabalha dessa forma. (Lopes, 2021)

**Como fazer seu mock-up:**

Agora sabemos que a consideração do mock-up, é essencial para saber que irão desenvolvê-lo. Há muitas ferramentas que possibilitaram quão você “brinque” com as suas ideias e simule as funcionalidades. Balsamiq e Pencil são bons exemplos.

Seu mock-up não terá planificação ou integração com fundamento de dados ou redes sociais, senão pode se igualar a vida isso tudo.

O caso de haver simplicidade ainda permite que você o crie, próprio sem saber programar, ou que pague alguém para fazê-lo por um custo mais baixo. De qualquer forma, é essencial que o freguês se envolva o máximo possível nesta etapa, pois só ele sabe o que realmente ele deseja.

Também aconselhamos que você não se preocupe com o layout nesse momento, deixe isso para um profissional de web layout. A ideia agora é armar a armação do seu projeto, não os detalhes.

Também não se preocupe se você não conceber seu produto final 100%. Faça o quanto você puder, e certamente esse material ajudará o profissional contratado a saber de forma melhor, também irá reduzir seus riscos de problema.

Esse caso também possibilitará que você mude de ideia quantas vezes quiser, envolvendo pouco que não te agrade. Muitas vezes, isso não é provável quanto o pedaço de programação que foi iniciada. (Lopes, 2021)

2.1.3 DIAGRAMA

**CASO DE USO:**

Bom nós iremos começar com o caso de uso que vai auxiliar você no levantamento dos seus requisitos funcionais do seu sistema, ele irá descrever o seu conjunto de funcionalidade do sistema e sua interação com os seus elementos externos entre si.

Cenários: quando nós falamos sobre o caso de uso, nós temos que ter em mente o conceito de um cenário, que no caso seriam as instâncias de uso.

O cenário poderá ser compreendido com a sequência dos passos que irá descrever uma interação entre o usuário do sistema.

**Quando usar o diagrama de caso de uso:**

Diagrama

Descrição gerada automaticamenteEste diagrama de caso de uso não oferece muitos detalhes e não espere, por exemplo, que ele mostre uma ordem em que os seus passos serão executados.

Figura - Exemplo caso de uso1

Fonte: https://www.lucidchart.com/pages/pt/diagrama-de-caso-de-uso-uml#:~:text=um%20diagrama%20UML-,O%20que%20%C3%A9%20diagrama%20de%20caso%20de%20uso%3F,de%20s%C3%ADmbolos%20e%20conectores%20especializados

Em vez disso, esse diagrama de caso será adequado e trará uma visão geral deste relacionamento entre um caso de uso como, atores e sistema.

Geralmente os profissionais desta área recomendam usar o diagrama de uso para você poder completar um caso de uso descrito em um único texto.

O diagrama de caso de uso também é ideal para:

* Poder representar as suas metas de uma interação feita no sistema do usuário.
* Define e organiza o requisito funcional no sistema.
* Específica o contexto e o seu requisito do seu próprio sistema.
* Modela o seu fluxo básico de um evento no seu caso de uso.

Figura – Exemplo caso de uso 2

Diagrama

Descrição gerada automaticamente

Fonte: https://www.lucidchart.com/pages/pt/diagrama-de-caso-de-uso-uml#:~:text=um%20diagrama%20UML-,O%20que%20%C3%A9%20diagrama%20de%20caso%20de%20uso%3F,de%20s%C3%ADmbolos%20e%20conectores%20especializados

**DIAGRAMA DE CLASSE:**

**DIAGRAMA DE CLASSE (UML)**

COMO SURGIU?

A criação da UML foi criada oficialmente em outubro de 1994, quando Rumbaugh acabou se juntando a Booch na Rational. O foco inicial foi do projeto e era a unificação dos métodos Booch e o OMT (Furlan, 1988). Esse esboço é de uma versão 0.8 de um método unificado e foi lançado em outubro de 1995.

**PARA QUE SERVE O DIAGRAMA DE CLASSE UML?**

Na versão UML, o seu diagrama de classe são as 6 versões do diagrama estrutural, os seus diagramas de classe são importantes para a etapa de uma modelagem de objeto e vai modelar a estrutura estática do sistema. E na verdade depende do nível de complexidade do sistema, também é possível usar uma imagem do diagrama de classe para você modelar este sistema inteiro ou em diversos diagramas de classe para você poder modelar todos os componentes do sistema.

Os diagramas de classe são todas as cópias do sistema ou do subsistema. Também é possível utilizar o diagrama de classe para você modelar os objetos que fazem o seu sistema, e também para poder exibir o relacionamento entre os objetos.

Benefício de diagrama de classe (UML)

* Ilustrar os modelos de dados para um sistema de uma informação, não importa o nível de dificuldade.
* Pode entender melhor a visão geral de um esquema da aplicação
* Expressa visualmente a necessidade principal do sistema e divulga a informação para a empresa

Figura - exemplo caso de uso 3

Diagrama

Descrição gerada automaticamente

Fonte: https://www.lucidchart.com/pages/pt/diagrama-de-caso-de-uso-uml#:~:text=um%20diagrama%20UML-,O%20que%20%C3%A9%20diagrama%20de%20caso%20de%20uso%3F,de%20s%C3%ADmbolos%20e%20conectores%20especializados

**MARCAÇÁO DA SEQUÊNCIA (UML).**

Ele pode facilmente construir um diagrama na sequência a partir do seu texto. A marcação segue com as normas comuns de uma empresa para que você consiga criar o seu diagrama com facilidade e muita eficiência.

Figura - exemplo caso de uso4

Diagrama, Desenho técnico

Descrição gerada automaticamente

Fonte: https://www.lucidchart.com/pages/pt/diagrama-de-caso-de-uso-uml#:~:text=um%20diagrama%20UML-,O%20que%20%C3%A9%20diagrama%20de%20caso%20de%20uso%3F,de%20s%C3%ADmbolos%20e%20conectores%20especializados

**Exemplo é marcação do (UML)**

Modificador de acesso de membro da (UML).

Geralmente as classes possuem vários níveis no acesso, e depende do seu modificador do seu acesso (visibilidade). Agora veja os níveis de acesso com os determinados símbolos.

* Público (+)
* Privado (-)
* Protegido (#)
* Derivado (/)
* Estático (sublinhado)

# 2.2 Paradigmas de Programação

Paradigma de programação é um estilo, modelo ou uma metodologia. Ele não é tratado como uma linguagem, mas sim a forma de como você resolve seus problemas utilizando uma linguagem de programação determinada.

Neste mundo muitas linguagens de computação são conhecidas, porém como tudo, essas linguagens precisam seguir algumas regras quando são implementadas, e estas regras são os paradigmas (Paradigma é visto como modelo para “padrão”).

Com isto, quando uma nova linguagem de programação é inventada, conforme seus atributos, tende-se a enquadrar a um paradigma ou até multiparadigma (Quando se suporta um aglomerado de paradigmas), sendo o JavaScript um usado como exemplo mais claro. (Silva, 2020)

**Surgimento**

O primeiro paradigma de computação originou-se entre as décadas de 50 e 60 (1950 – 1960), uma época que a capacidade de computadores era enfraquecida, e a dificuldade para o envio de instruções ao computador era elevada. No início eram de programação imperativa e declarativa, sendo tratados na época com “abordagens de programação” e apenas na década de 70 (1970) foram chamadas de paradigmas.

E conforme o tempo a forma de programar foi evoluindo, sendo eles originários dos dois iniciais. Atualmente existem muitos paradigmas, sendo alguns deles mais antigos e já não são mais utilizados e outros pouco afamado. Dentre os diversos existentes podemos citar os:

* Estruturado;
* Procedural;
* Orientado objeto;
* Orientado a eventos;
* Orientado a Aspectos;
* Concorrente;
* Distribuído;
* Funcional;
* Lógico;
* Reativo.

O que Paradigma de Programação tem a ver com Linguagens de Programação?

Todas linguagens são baseadas em algum paradigma, e também possui linguagens baseadas em várias, sendo validas nesses casos multiparadigmas, alguns exemplos são Java, JavaScript e Python.

Em desenvolvimento de softwares possui debates de qual seria o melhor paradigma, porém paradigmas se referem em como humanos podem ler e escrever instruções enviadas ao computador, então o melhor paradigma seria aquele que facilitará o entendimento e o progresso de seu código.

Programação está sempre associado ao tema de conhecer formas possíveis de enviar instruções ao computados, logo, entender paradigmas irá facilitar o entendimento e aprendizado de quaisquer linguagens, a ponto de tornarem elas menos importantes. (Silva, 2020)

2.2.1 Paradigma sequencial

Paradigma sequencial foi o primeiro tipo de paradigma a surgir, e ele ainda acaba sendo dominante nos dias de hoje. Programas que são centrados em conceito de estado (Modelado por variáveis) e atitudes (comandos) visam mudar seu estado.

Computação tem seu significado de transformar valores da memória de um computador. Suas regras serão definidas por passos que serão executados:

* Declarações explicitas de variáveis e alocações de memorias;
* Presenciar interatividade e laços;
* Programa executado significa executar uma series de comandos;

O estado de um programa será mantido em suas variáveis de programas. Estas variáveis serão associadas a localizações de memorias caracterizadas por endereços e valores armazenados no endereço. (Testa, 2014)

Também é possível acessar um valor diretamente, lendo seu valor da variável, e o valor da variável é alterado por meio das atribuições.

Como exemplo (usando pascal):

**X :=5;**

Seu termo na esquerda do operador da atribuição (:=) é a variável na qual o valor estará sendo alterado e na direita seu novo valor. O “X” da variável tem seus valores diferentes antes e depois da sua atribuição. A noção (Efeito) de um programa dependerá da ordem na qual suas atribuições serão escritas e executadas. (Emanuelle Testa, 2014)

2.2.1 Paradigma estruturado

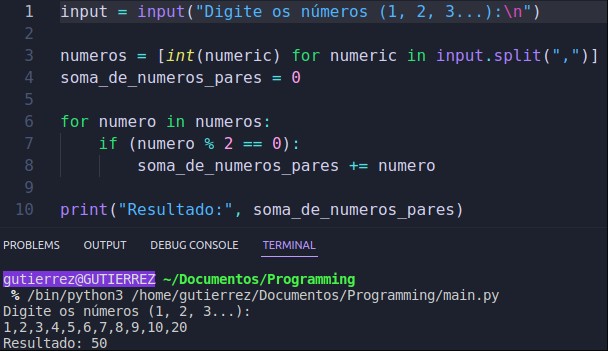
O paradigma estruturado acaba determinando as instruções que são enviadas ao computador onde são formadas três estruturas, sendo elas:

* Sequência: Suas instruções serão codificadas na sequência ao serem executados.
* Condição: É um bloco de código onde apenas é executado se uma condição for verdadeira (IF’s/ELSE’s, SWITCH com CASE’s).
* Repetição: Trecho do código que pode ser executado inúmeras vezes. (FOR’s, WHILE’s, recursividade).

A programação estruturada acaba permitindo maior ação ao software, com isto é possível coletar dados e não estar preso a um conjunto estático.

Figura - Exemplo paradigma estruturado 1

Figura - Exemplo paradigma estruturado 2

Texto

Descrição gerada automaticamente

Fonte: Próprio Autor

O código da esquerda age por cima de um conjunto estático de dados, o que irá fixar a execução para um intervalo de 1 a 10.

Enquanto o código da direita acaba ampliando sua capacidade do software e permite que agora ele acabe sendo aplicado a qualquer aglomerado de números informados na sua execução. O que mostra a relevância do paradigma estruturado para o progresso e no desenvolvimento de software.

Também neste paradigma a capacidade de modularizar ou criar sub-rotinas. Este paradigma tornou-se popular por propor uma forma de raciocínio intuitivo em que cada bloco de código é legível e compreensível, e também, em certa medida, contra o uso do GOTO (Do inglês go to, em português ir para). A programação estruturada dominou a programação até os anos 90 (1990), quando foi substituída pela popularidade do OO (Orientação a Objetos), mas ainda é ótimo para pessoas que estão aprendendo a programar porque é uma maneira mais fácil de programar. (Souza, 2021)

Diversos softwares possuem codificações seguindo o paradigma estruturado, o que não se caracteriza sendo uma má ou boa codificação, porém naturalmente não condizem com as intenções dos desenvolvedores. Isto porque diversas vezes estes creem estarem usando paradigmas como procedural ou Orientação Objeto, porém o código continua seguindo o seu modelo estruturado. Segue exemplos:

* Blocos grandes de códigos estruturados fazendo diversas coisas distintas de um único método.
* Rotinas inteiras encapsuladas em classes, porém não representa o objeto de fato
* Métodos/classes com variadas responsabilidades, onde o que será determinado é o código para ser executado onde ainda são condicionadas as repetições, muitas vezes utilizadas em excesso.

O problema com os pontos acima é que muitas equipes de desenvolvimento não conseguem distinguir a diferença entre o uso Estruturado, Procedimental e OO, o que na prática leva a baixa qualidade de implementação, confusão de código e altos custos de manutenção. A programação estruturada é útil e aplicável, mas usá-la sem considerar paradigmas complementares resulta em código menos legível. (Souza, 2021)

2.2.2 Orientado a Objetos

O paradigma orientado a objetos é um dos paradigmas mais utilizados no mercado de trabalho. Além de ser um dos primeiros paradigmas que encontramos quando começamos a olhar para o desenvolvimento de software, a maioria das linguagens de uso comum na indústria tem uma forte base orientada a objetos, o que torna o domínio desse paradigma crucial. Aqui examinaremos quais são os principais pontos do paradigma orientado a objetos.

Objetos em linguagem orientada é organizada a maior parte do código em estruturas chamas “Classes”. (Campomori, 2022)

**Classes**

A classe pode ser entendida como o molde inicial, o molde muitas vezes pode representar certa estrutura do mundo real na qual o código vai lidar. Imagine que trabalhemos com aplicações que envolvam carros. Muito provável, que a aplicação lidará com carros e, por isto, vamos necessitar um molde para definir o que é um carro dentre nosso código. O molde vai ser responsável para estabelecer o que seria um carro e o que ele pode fazer.

Como todos imaginamos, quando falamos de carro as primeiras coisas que veem a mente é:

* Placa;
* Modelo;
* Marca;
* Cor

Também podemos fazer ligações entre as seguintes ações:

* Ligar;
* Acelerar;
* Desligar

Nota-se que, quando se fala sobre o que um carro é, falamos sobres suas características, assim seja: características que definem tal carro. Porém quando conversamos com o que um carro pode fazer, discutimos sobre ações na qual a carros desempenha.

Incluindo para termos técnicos, podemos designar as características do carro como atributos, enquanto designamos ações como métodos. Então no final, métodos e atributos ficaram agrupados em uma classe.

Estes atributos e métodos podem ser representados por meio de uma linguagem de modelagem chamado UML (Unified Modeling Language). Com a UML ela pode prever alguns diagramas que buscam ajudar no processo de modelagem de um software. Entre esses diagramas, temos o diagrama de classe. (Campomori, 2022)

A classe carro, se fosse utilizada a linguagem Java para poder representar nossa classe, ficaria da seguinte maneira:

Figura - Exemplo com POO 1

Texto

Descrição gerada automaticamente

Fonte: Próprio Autor

Neste exemplo, contém a classe Carro realizando o papel do nosso molde. As variáveis Fusca e Monza são objetos de Carro ou, se usando termos técnicos da forma correta, Fusca e Monza são instancias de Carro. Pois elas derivam de Carro, o objeto tem todos estes atributos e métodos previstos em Carro.

Note que a partir deste molde podemos especificar vários carros... poderíamos ter um Chevrolet Monza roxo com placas JDF-8729, um Volkswagen Fusca azul com placas KPA-9830, ou mesmo um Ford Escort Xr3 cinza com placas MRG-3562. São carros porque vêm do mesmo molde: todos têm marca, modelo, cor e placa, e além de poder ligar, desligar, frear e acelerar, os atributos e métodos são determinados pela classe carro criada.

Nesse caso, Chevrolet Monza, Volkswagen Fusca e Ford Escort Xr3 são objetos porque são "feitos" a partir de um molde que definimos, a classe Carro. Objetos são como variáveis ​​que criamos para usar nossas classes, seja para definir suas propriedades ou para chamar seus métodos. A ligação coordenada entre a escrita e leitura de propriedades e a invocação de métodos dá o tom para aplicações escritas em linguagens orientadas a objetos. (Campomori, 2022)

**Encapsulamento**

Sendo levado em consideração ainda o exemplo da classe Carro, podemos imaginar algo que indique verificar se o carro está ligado ou desligado. Com isto poderia dar mais “capacidade” á Carro. Poderíamos, por exemplo, certificar que o carro só pode acelerar ou frear caso estiver ligado. Com esse indicador poderia ainda ser alterado pelos métodos ligar (); e desligar ();

Com esta ideia em mente a classe Carro ficaria desta forma:

Figura - Exemplo com POO 2

Texto

Descrição gerada automaticamente

Fonte: Próprio Autor

Observe que, além de ser adicionado o atributo chamado ligado com o tipo booleano (um tipo de dado lógico que pode ter apenas um de dois valores possíveis: verdadeiro ou falso) sendo (true) para ligado e (false) para desligado, e ainda se possui o trecho de código abaixo:

Figura - Exemplo com POO 3



Fonte: Próprio Autor

Com este trecho determinamos que ele é um construtor, ele é invocado ao iniciarmos um objeto a partir de uma classe. De maneira geral, quando ele é invocado chamamos a criação da instancia com a palavra-chave new. Com isto, se obtém este código:

Figura - Exemplo com POO 4



Fonte: Próprio Autor

Chamamos este método justamente de construtor. Quando reescrito este método, impomos uma “customização” na inicialização dos objetos da classe Carro: qualquer carro criado que possuir o indicador ligado como false, ou seja, o carro começará como desligado como seu padrão.

Então, poderia ser definido, como um exemplo se chamando acelerar em nosso Carro. Podemos certificar que o atribui ligado para “fiscalizar” melhor o método: um carro, claramente, só pode acelerar se estiver com partida (ligado).

Figura - Exemplo com POO 5

Texto

Descrição gerada automaticamente

Fonte: Próprio Autor

Desta maneira, a classe Carro é utilizada da seguinte forma:

Figura - Exemplo com POO 6

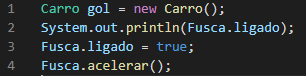
Texto

Descrição gerada automaticamente

Fonte: Próprio Autor

Supostamente, na classe Carro funciona corretamente. Porém, temos um imprevisto: o atributo ligado é acessível para todos, da mesma maneira na qual os atributos modelo e marca por exemplo. Com isto dizemos que podemos “adulterar” seu comportamento de Carro. E poderia, como exemplo, alterar de forma pratica seu conteúdo do atributo ligado antes de chamar o método ligar, assim permitindo acelerar um carro que estiver em sua tese desligado:

Figura - Exemplo com POO 7



Fonte: Próprio Autor

Com certeza isto é uma situação problema, pois nosso carro agora possui uma brecha para funcionar de maneira distinta de como foi planejado.

Para resolver esse problema, precisamos recorrer ao pilar da orientação a objetos: o encapsulamento. O encapsulamento é projetado para ocultar propriedades e métodos de uma classe que não deve ser acessada por outras estruturas. Isso é exatamente o que precisamos: a propriedade vinculada teoricamente deveria ser acessível apenas pela própria Carro, o que permitiria apenas que os métodos ligada e desligada alterassem as métricas de saúde do carro da maneira correta. Isso nos impediria de acessar a propriedade de fora, fazendo com que o código de que estamos falando aqui falhe.

Encapsulamento em linguagens orientadas a objetos é definir através de algo chamado de atributo de visibilidade. Esse atributo de visibilidade estabelece justamente o quanto é acessível nossos atributos e métodos são com relação as deveras estruturas do nosso código, de maneira geral, teremos três tipos de atributos de visibilidade básico e comum à linguagem orientada a objeto em geral:

* Public: a estrutura será visível de qualquer lugar do seu código, incluindo em outros tipos de classes, fora a classe que definirá o atributo ou método em si;
* Private: uma estrutura será apenas visível pela classe que caracterize uma estrutura em si, estruturas externas não irão conseguir acessar o seu método de atributo que esteja marcado com este atributo de visibilidade;
* Protected: somente é visível a classe-mãe e nas classes-filhas.

Ao considerar Carro, podemos observar o problema acontecendo devido o atributo ligado esteja definido como Public, se tornando acessível em qualquer lugar. Notamos que este é o problema, porque o atributo ligado não poderá acessar a partir de qualquer lugar, ele só pode ser acessível dentro dos métodos ligar e desligar, ambos em Carro. O atributo ligado não é encapsulado.

Pode-se encapsular se tornar private, fazendo que ele fosse acessível somente em Carro.

Figura - Exemplo com POO 8

Texto

Descrição gerada automaticamente

Fonte: Próprio Autor

Desta forma, o erro no qual vimos antes não irá acontecer mais, devido ao atributo ligado agora é apenas acessível dentro da própria classe Carro.

Figura - Exemplo com POO 9



Fonte: Próprio Autor

Então pode-se dizer que o atributo ligado de Carro está encapsulado. E se quiser ler o valor de ligado externamente (pois já que alterar ele externamente estará totalmente errado), então se cria um método no qual desenvolve o valor ligado.

Figura - Exemplo com POO 10

Texto

Descrição gerada automaticamente

Fonte: Próprio Autor

Podemos ver pelo menos se o carro estará ligado ou desligado

Figura - Exemplo com POO 11

Texto

Descrição gerada automaticamente

Fonte: Próprio Autor

Esse método é chamado geralmente de método de acesso, pois ele equipa algum tipo de acesso indireto a um atributo encapsulado. Com relação a encapsulamento, temos dois tipos de método de acesso:

* **Get**: permite ver o valor de um atributo;
* **Set**: permite alterar o valor de um atributo.

Uma prática comum em linguagens orientadas a objetos (principalmente Java) é envolver todas as propriedades com métodos de acesso como get e set, evitando o acesso direto às propriedades. Embora esta seja uma prática comum, é importante dizer que o fato de usarmos métodos de acesso não garante nenhum encapsulamento de uma estrutura. O que garante esse encapsulamento é a definição da visibilidade correta de cada propriedade e método e o estabelecimento correto do método de acesso. Por exemplo, não criamos um método set para a propriedade link porque ele só pode ser alterado pela própria classe. (Campomori, 2022)

Aplicando esta regra em Carro, teremos isto:

Figura - Exemplo com POO 12

Texto

Descrição gerada automaticamente

Fonte: Próprio Autor

**Herança**

Reaproveitar um código e a possibilidade de se evitar que ele esteja duplicado é o objetivo de linguagens orientadas. Imagine que precisemos criar uma classe para definir outro tipo de veículo, como uma bicicleta. Ela possui os atributos parecidos com um carro, ambos possuem marca e seu modelo. Além, que ambos são um tipo de veículo.

Este seria um código para definir a classe bicicleta sem considerar a classe carro:

Figura - Exemplo com POO 13

Texto

Descrição gerada automaticamente

Fonte: Próprio Autor

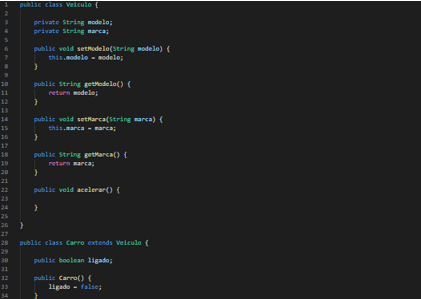
Nota-se que o código foi duplicado relativamente com marca e modelo nas classes carro e bicicleta, o que pode ser bem ruim em termos de manutenção do código a longo prazo. Porém, só teríamos uma maneira de evitar duplicidade, podemos utilizar o conceito da herança. (Campomori, 2022)

Se fosse aplicar o conceito de herança, teríamos três classes:

* A classe carro, com tudo aquilo que um carro possui de ações e características;
* A classe bicicleta, com tudo aquilo que uma bicicleta possui de ações e características;
* E uma nova classe declarada de veículo, com tudo que possa existir entre carros e bicicletas.

Então as classes Carro, Bicicleta e veículo pode ser definida dessa forma:

Figura - Exemplo com POO 14



Fonte: Próprio Autor

Figura - Exemplo com POO 15

Texto

Descrição gerada automaticamente

Fonte: Próprio Autor

Vendo que todas as semelhanças entre as classes Carro e Bicicleta vão para a classe Veículo. As classes Carro e Bicicleta agora herdam da classe Veículo para reutilizar essas estruturas comuns, além do fato de Carro e Bicicleta serem tipos de veículos. É importante que esse relacionamento "existe" ou "existe" exista entre classes quando vamos usar herança.

Neste caso, podemos fazer Carro herdar Veículo, porque um carro é um veículo; assim como podemos fazer Bicicleta herdar Veículo, porque a bicicleta é um meio de transporte. Com a herança, podemos evitar a duplicação de código e facilitar a manutenção, além de manter a consistência, desde que a regra "ser" seja devidamente aplicada.

E quando se possui herança sendo utilizada, as classes podem assumir um dos dois papeis:

-Classe-Base: serve de base para demais classes. No nosso exemplo, veículo seria nossa Classe-Base;

-Sub-Classe: Herdando de outra determinada classe. No nosso exemplo, carro e bicicleta são Sub-Classes. (Campomori, 2022)

**Abstração**

Quando se lida com Orientação a objetos, acaba sendo comum que tentando sempre escrever um código baseado em abstrações, onde busca trazer flexibilidade a um código.

Em exemplos passados, houve um problema de abstração: qualquer veículo é capaz de acelerar, sendo o veículo um carro ou uma bicicleta. Assim, não podemos remover o método acelerar () da classe veículo, onde todo veículo tem este comportamento. Porém, a classe veículo em não “reconhece” como se acelera.

Quem reconhece como acelerar é a classe Carro (onde sabe como o carro acelera) e a classe Bicicleta (onde sabe como a bicicleta acelera). Vamos resolver esta situação, podemos tornar o método acelerar () abstrata: isto vai acabar desabrigando a classe Veículo para ser definida uma implementação para o método (onde a classe Veículo não reconhece como acelerar), porém obriga as Sub-Classe (em caso, classes Carro e Bicicleta) a serem definidas os seus comportamentos de aceleração. (Campomori, 2022)

E ainda, parando para analisarmos, não há logica instanciarmos objetos da classe veículo, já que ela acaba servindo como uma Sub-Classe nesse panorama. Poderíamos também instanciar objetos das classes Carro e Bicicleta, porém não da classe veículo. Onde queremos evitar que objetos sejam instanciados da classe veículo, já que ela deve ser somente uma Classe-Base, pois também poderemos definir que veículo é uma classe abstrata.

Figura - Exemplo com POO 16

Texto

Descrição gerada automaticamente

Fonte: Próprio Autor

Figura - Exemplo com POO 17

Texto

Descrição gerada automaticamente

Fonte: Próprio Autor

Usando abstração neste caso, sabemos que qualquer novo tipo de veículo criado a partir da classe veículo deve implementar o comportamento de aceleração. Isso faz sentido porque todo carro precisa ser capaz de acelerar. (Campomori, 2022)

**Polimorfismo**

As linguagens orientadas a objetos ainda fornecem suporte para a criação de estruturas polimórficas. Uma estrutura polimórfica é uma estrutura que pode mudar seu comportamento interno sob certas circunstâncias. Essa mudança de comportamento pode acontecer de algumas maneiras, como por meio de substituição de método e por meio do LSP (Princípio de Substituição de Liskov).

No exemplo anterior, tivemos um exemplo de implementação de polimorfismo por meio de sobreposição de métodos: tínhamos a classe veículo, que possuía o método abstrato acelerar (). O método é abstrato porque a classe veículo só sabe que deve conter esse comportamento, mas não como deve acontecer.

No entanto, as classes Carro e Bicicleta são forçadas a implementar o método acelerar () estendendo a classe veículo. Cada uma dessas classes implementa o método acelerar () da maneira que melhor se adapta a cada tipo de veículo. Aqui, já temos um exemplo de polimorfismo: as classes Carro e Bicicleta são polimórficas porque possuem um ancestral comum (classe veículo), o que as obriga a implementar o método acelerar (), mas cada uma delas é O método mais correto em cada veículo.

Essa mudança de implementação não precisa modificar o código da classe veículo, e a implementação do método acelerar () em cada classe é isolada: a implementação do método acelerar () na classe Carro não afeta a implementação do mesmo na classe Bicicleta e vice-versa.

Outro exemplo de polimorfismo é a aplicação do Princípio da Substituição de Liskov (também conhecido como LSP - Princípio da Substituição de Liskov). O LSP faz parte de um conjunto de cinco práticas de codificação chamadas SOLID. Esses princípios são projetados para produzir código de alta qualidade que esteja em conformidade com os princípios das linguagens orientadas a objetos. (Campomori, 2022)

Para entender o LSP, considere o seguinte código:

Figura - Exemplo com POO 18



Fonte: Próprio Autor

O código não parece diferente: temos um objeto chamado veículo do tipo Carro. Porém, como Carro herda da classe veículo, podemos inferir que objetos do tipo Carro também podem ser considerados do tipo veículo; afinal, agora todo Carro também é um veículo por herança. Assim, podemos escrever o código acima da seguinte forma:

Figura - Exemplo com POO 19



Fonte: Próprio Autor

Podemos definir um objeto de veículo como do tipo Veículo, mas criá-lo com base em Car. Nesse momento, o objeto veículo se comportará como um carro. No exemplo acima, dissemos que veículo é abstrato e Carro é concreto. Se quisermos mudar nosso tipo de veículo para bicicleta, basta apenas mudar a forma de realização.

Todo o código abaixo ainda funcionará porque Bicicleta também é Veículo. Isso torna nosso código mais flexível (se quisermos alterar o comportamento do código, precisamos apenas alterar a retificação) e à prova de falhas ao escrever o código (porque a abstração garantirá que o código esteja logo abaixo da definição a retificação continuará a funcionar de forma transparente).

Figura - Exemplo com POO 20



Fonte: Próprio Autor

No exemplo, por mais de todas as vantagens que podemos notar no que diz respeito à qualidade e manutenção do código, podemos dizer que o objeto veículo é um objeto polimórfico, pois a concretização está sendo capaz de alterar seu comportamento. Porém, o código subsequente não é afetado por essa troca. (Campomori, 2022)

## Plataformas de desenvolvimento

“Plataforma de Desenvolvimento” é um termo usado para denominar plataformas que ajudam o usuário a construir e implementar de forma eficaz aplicativos personalizados com menos codificações do que o desenvolvimento tradicional.

Criado em 1970 por Winston Royce o modelo de desenvolvimento tradicional também conhecido como modelo cascata foi baseado em ordens de padrões derivados de diversos segmentos de engenharias dos anos 70.

Figura - Exemplo plataforma de desenvolvimento

Diagrama

Descrição gerada automaticamente

Fonte: https://www.devmedia.com.br/processo-de-teste-agil-x-tradicional/36854#:~:text=Metodologia%20tradicional,-Nesse%20contexto%20temos&text=Foi%20chamado%20de%20desenvolvimento%20tradicional,pequeno%2C%20m%C3%A9dio%20e%20grande%20porte

Foi chamado de desenvolvimento tradicional, pois é a base para diversos modelos utilizados há décadas pela indústria de software e é considerado um modelo rígido de pouca flexibilidade, adaptabilidade e versatilidade, sendo utilizado em projetos de pequeno, médio e grande porte.

**Plataforma de desenvolvimento de baixo código**

Por meio de interações e fragmentos de códigos pré-construídos ou blocos, uma plataforma de baixo código permite que pessoas com pouco conhecimento sobre programação consigam desenvolver um aplicativo ou site de maneira especificada com uma interface visual de arrastar e soltar.

Dois sites que ganharam seu espaço entre plataformas de baixo código são o Google Sites e o Wix.com

Figura - DOC

Ícone

Descrição gerada automaticamente

Fonte: https://maisgeek.com/como-usar-o-google-sites

O Google Sites é uma ferramenta de criação de página da web incluída como parte do conjunto gratuito de editores de documentos do Google com base na web oferecido pelo Google. O “Google Docs., Google Drawings, Google Forms e Google Keep, Google Sheets, Google Slides.”

Figura - WIZ

Teclado de computador

Descrição gerada automaticamente

Fonte: https://blog.ajasolucoesdigitais.com.br/a-verdade-sobre-wix/

Wix.com foi fundada em 2006, em Israel, por Avishai Abrahami, Nadav Abrahami e Giora Kaplan. Consiste em uma plataforma online de criação e edição de sites, que permite aos usuários criar sites em HTML5 e sites Mobile sem necessidade de conhecimento de conhecimento prévio em programação ou design.

Além disso, a plataforma elimina a necessidade de criar estruturas, vincular bancos de dados e outras tarefas normalmente incluídas no desenvolvimento tradicional.

E torna o desenvolvimento mais fácil para que mesmo pessoas sem habilidades de programação possam desenvolver aplicativos por conta própria.

Com plataformas de baixo código, suas equipes de TI podem desenvolver aplicativos com mais rapidez e menos erros do que a codificação tradicional. Elas não precisam gastar meses construindo um aplicativo do zero, pois essas plataformas fornecem componentes padronizados, como formulários, modelos de relatório e fragmentos de código prontos para uso.

Recursos a serem considerados em uma plataforma de baixo código

**Modelagem visual**

Basicamente a modelagem visual é o uso de notações gráficas e textuais semanticamente ricas para capturar o design de um software.

A interface de modelagem visual de plataformas de baixo código permite que você veja a aparência do seu aplicativo conforme você o constrói.

**Arrastar e soltar**

É um processo aonde a formatação e programação de um software é padronizada e facilitada para que não-programadores tenham acesso ao desenvolvimento de um website.

**Segurança**

A parte de segurança está presente em todos os pilares de um processo de desenvolvimento de software mantendo sua privacidade e projetos privados de forma segura e confiável.

**Implantação de vários dispositivos**

Hoje em dia a diversidade de marcas tecnológicas está presente no mercado popular então um software de fácil acessibilidade entre diversas máquinas como Windows, Mac e Linux é essencial. É essa questão também se encaixa nos dispositivos mobile como Android, IOS e Windows Phone.

**Escalabilidade**

Conforme seus projetos vão crescendo funções vão sendo atribuídas a usuários o tornando administradores, especialistas e dessa forma seu projeto tende a crescer cada vez mais de forma segura e controlada.

**Desenvolvimento mais rápido**

Negócios em rápida evolução exigem tecnologias que acompanhe as tendências de mercado que estão em constante mudança. E plataformas de desenvolvimento de baixo código pode ser essencial nessas situações. A vantagem que o desenvolvimento de baixo código oferece é sua capacidade de impactar significativamente a entrega de um novo software.

 Os aplicativos desenvolvidos com plataformas de baixo código ajudam as organizações a se tornarem mais ágeis.

**Produtividade aprimorada**

Tarefas manuais demoradas como atualizar os dados sempre que houver uma mudança dando assim mais tempo para o desenvolvedor se concentrar na codificação onde a engenhosidade humana é realmente necessária.

Como resultado, os projetos serão implementados com mais rapidez, eficiência e padrões mais elevados.

**Gerenciamento de riscos**

Seus dados se tornam mais seguros permitindo o controle de todos os processos de gerenciamento de riscos em uma plataforma centralizada.

Simplificando assim a sua manutenção pois atualizar um aplicativo de baixo código para segurança de dados é fácil e economiza tempo.

**Flexibilidade**

Uma plataforma de desenvolvimento de baixo código facilita a mudança de aplicativos e sua adaptação a novos requisitos.

Quer seja uma tarefa simples ou um fluxo de trabalho complexo, os usuários podem alcançá-lo sem escrever uma linha infinita de códigos. O baixo código fornece recursos que permitem aos usuários implementar novas estruturas ou modelos de sistema operacional rapidamente.

**Experiência do cliente**

Aplicativos com um design agradável e criativos agradam seus clientes por ser visualmente atraentes, trazendo assim um bônus ao usuário final.

Também aumentando a compreensão e a capacidade de resposta do cliente diante ao seu aplicativo ou projeto.

2.3.1 Plataforma Web

Assim como o nome já diz a plataforma web se baseia em uma relação harmônica de troca de informações entre quem produz com que consome podendo ser usada para fins acadêmicos, profissionais ou lazer.

**Como criar uma plataforma web**

**Principais passos**

Os passos a seguir não devem ser vistos como algo a se seguir à risca pois cada projeto tem sua particularidade como também sua finalidade e funcionalidades desejadas

**Identifique os requisitos do projeto**

Identificar os requisitos do projeto não é nada mais doque manter constantemente certa interação com seu cliente afim de saber como ele idealiza e deseja sua web site por exemplo. Saber e identificar funcionalidades como a página de login ou até mesmo a barra de pesquisa.

**Construir protótipo não funcional**

Também conhecido como wireframe, o protótipo pode ser definido como um esqueleto visual da plataforma, uma espécie de versão inicial ou então o termo “beta” a onde o protótipo oferece a base ao início do projeto.

**Desenvolver a solução**

O Processo de desenvolvimento da solução é parcialmente dividida em 3 etapas

Figura - Processos de desenvolvimento

Diagrama

Descrição gerada automaticamente

Fonte: Próprio Autor

**Etapa 1: Elaborar as especificações das funções**

Implementar a interface de programação de aplicação (API) nessa etapa é fundamental por ter a possibilidade de integrar com outras plataformas, como mobile e desktop.

É função da API desempenhar toda a lógica do produto, assim como manter o domínio sobre os dados. É ela também que permite a integração com sistema de terceiros, como ocorre quando há a necessidade de processamento de pagamentos ou do consumo de dados públicos.

O desenvolvimento de uma API pode ser aplicado por diversas linguagens de programação, mas Javascript e engine node são muito utilizadas e ganham preferência em diversas empresas como na Cubos. (Cubos.io)

**Etapa 2: Elaborar e implantar banco de dados**

Planejar como os dados serão armazenados no banco e em como eles serão expostos ao usuário deve ser visto e revisto por conter informações privadas e pessoais. Também deve ser pensado no formato de otimização para que haja uma interação ágil entre o sistema e o banco de dados.

**Etapa 3: Programar a Solução**

A partir do wireframe introduzir o layout por meio do front-end entram em cena para eu possam estruturar a navegabilidade da plataforma e fazer sua responsividade.

Juntando assim as peças e atributos já prontos e dando um passo para o fim do projeto.

**Testar e corrigir solução**

A melhor forma de encontrar bugs que passaram despercebidos no momento do desenvolvimento é por meio de testes e atribuições mantendo assim a integridade do projeto durante o desenvolvimento.

Assim como apps de games tem sua versão de teste conhecida como “versão beta” as plataformas web também tem sua versão de testes que são entregues a usuários que não fazem parte da equipe de desenvolvedores com a intenção de encontrar novos bugs ou falhas.

Simultaneamente à execução esses testes, os bugs serão corrigidos e as melhorias serão implementadas de acordo o feedback advindo dos testes.

**Publicar**

Publicar sua plataforma da forma correta é essencial para que ela atinja o público que você desenvolvedor deseja atingir, mas mesmo depois de passar por todos os testes de correção a plataforma ainda pode apresentar bugs, portanto manter seu projeto sempre atualizado e em constante desenvolvimento é essencial.

**Front-end e Back-end**

Duas formas de desenvolvimento que se completam como um quebra-cabeça, mas com duas funções totalmente diferentes uma da outra.

Front-end: Assim como o nome já diz (parte dianteira) a programação em front-end se baseia no design e projeção do seu projeto. Em um site por exemplo os códigos em front-end vão modelar editando as cores, a fonte, imagens entre outras funcionalidades

Figura - Figura com HTML, CSS e JAVASCRIPT

Gráfico

Descrição gerada automaticamente com confiança média

Fonte: https://meunegocio.uol.com.br/blog/html-css-ou-javascript-qual-aprender-primeiro/

**HTML**

**-Surgimento**

Criado na Suíça no ano de 1991 o HTML foi projetado para interligar instituições de pesquisa próximas e facilitar o compartilhamento de documentos com facilidade

**-Proposito**

O HTML não é considerado uma linguagem de programação e sim uma linguagem de marcação que tem como proposito a modelagem e definições de uma página na web. Seus códigos se baseiam em comandos para definirem o tipo de letra, qual o tamanho, cor, espaçamento, e vários outros aspectos do site ou página da web

**-Exemplos**

“Alguns códigos em HTML e suas funções:

<title> Define o título da página.

<script> Permite adicionar funções em páginas com script, podendo assim adicionar códigos em [Javascript](https://www.infoescola.com/informatica/javascript/) (Este código permite que alguns sites em HTML tenham joguinhos ou animações, verificações de formulários antes do envio para o servidor, entre outras funcionalidades)

<style> Define formatação em [CSS](https://www.infoescola.com/informatica/cascading-style-sheets-css/).”

(trecho pesquisado no site Infoescola <https://www.infoescola.com/informatica/html/> )

**-Motivos para aprender**

HTML é considerada uma das linguagens mais reconhecidas no mercado de desenvolvimento de sites e mesmo tendo diversos comandos e funções é uma linguagem simplificada e de fácil entendimento.

**CSS**

**-Surgimento**

Criado pelo norueguês Håkon Wium Lie em outubro de 1994 com o intuito de facilitar a programação de sites onde se era considerada muito complexa

**-Proposito**

Teve como proposito inicial facilitar a programação transformando códigos gigantescos em pequenas frases ou compostos por uma única palavra, mas com a notoriedade do HTML o CSS também conhecido como (Cascading Style Sheets ) ficou à disposição para os Web designers por ser uma linguagem que determinava o layout de uma página web

**-Exemplos**

Figura - Exemplo selector

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: https://developer.mozilla.org/pt-BR/docs/Learn/Getting\_started\_with\_the\_web/CSS\_basics

P  Seletor abre colchetes para a apresentação das declarações e propriedades

{}  Colchetes servem para caracterizar as propriedades e declarações

Color  Comando da propriedade para anteceder a o comando da cor

Red  valor da propriedade indica vermelha na seguinte declaração

**-Motivos para aprender**

O CSS vem para facilitar a vida de nós programadores, trazendo assim um melhor manuseio de formatação e melhorando nossa forma de programar deixando-a mais eficiente e de forma eficaz

**Java Script**

**-Surgimento**

Criado pelo americano Brendan Eich desenvolvedor da empresa Netscape em meados de 1995 o JS é uma linguagem de programação utilizada inicialmente para implementar itens complexos a uma página da web.

**-Proposito**

JavaScript é uma linguagem de programação utilizada não só para incrementar arquivos e complexidades a sua página web como também é utilizado para o desenvolvimento de aplicativos para a Web.

**-Exemplos**

<Script>  Utilizado para iniciar a programação em JS

</Script>  Utilizado para finalizar a programação em JS

//  Indica um comentário

If (...) -> usado para comparar uma variável com uma string e verificar se a condição seja verdadeira (TRUE) ou se uma condição é falsa (FALSE)

Else (...) -> trabalha junta ao comando “if(...)” identificando condições falsas

**-Motivos para aprender**

O Grande diferencial da linguagem JS é que ele permite ser desenvolvido dentro do código HTML, sendo assim conforme você avança no desenvolvimento do seu site basta colocar o comando “<script” para iniciar a programação em JS e ao finalizar basta utilizar o código “</script>”.

Back-end: Assim como o próprio nome sugere, a programação em Back-end é baseada na parte por traz de um site ou aplicação, fazendo assim a ponte entre os dados abordados pelo navegador e o banco de dados na maioria dos casos.

Usuários comuns não tem acesso livre a Back-end, assim garantindo assim que não haja uma manipulação dos dados ou acesso direto ao banco de dados aonde este usuário em questão teria acesso a informações pessoais de clientes catalogados.

**Linguagens adeptas a programação em Back-end:**

Figura - Exemplo PHP

Diagrama

Descrição gerada automaticamente

Fonte: https://blog.matters.tech/migrating-from-php-to-javascript-with-node-js-155534498b58

**PHP**

**-Surgimento**

Criado pelo dinamarquês Rasmus Lerdorf no ano de 1995 com a proposta de ser um simples conjunto de binários Commoon Gateway Interface (CGI) escrito em linguagens de programação C.

**-Versões**

PHP/FI 2.0: Lançada em 1997 o PHP2 teve diversos bugs a serem resolvidos, portanto, passou muito tempo em sua versão beta.

PHP3: Criada por Andi Gutmans e Zeev Suraski, também em 1997, mas lançada em 1998 o PHP3 com sua alta capacidade de extensibilidade teve seu grande diferencial sendo está versão a chave para o PHP que utilizamos hoje.

PHP4: Lançado nos anos 2000, o PHP 4 foi projetado através de uma base Zend Engine, esta versão do PHP se popularizou pela sua interação com outras funções permitindo assim a interação entre a linguagem e outros componentes web.

PHP5: Após 4 anos do lançamento do PHP4 o PHP5 chegou ao mercado de inovações trazendo uma base na Zend Engine 2.0 entre outras inovações, mantendo sua ideia de automodificação que surgiu no PHP3.

**-Motivos para aprender**

Hoje o PHP é uma linguagem muito relevante no mercado do desenvolvimento de softwares pesquisas feitas pela academia da tecnologia ACADTEC apontam que 80% dos sites na internet rodam PHP e que a tendencia deste número é crescer ainda mais nos próximos anos.

**Node.js**

**-Surgimento**

Lançado no ano de 2009 pelo pesquisador Ryan Dahl, a criação do node JS veio como uma necessidade pelo fato do Java Script não suportar uma sobrecarga de requisições, o que muitas das vezes geravam o bloqueio ao carregamento de arquivos ou processamentos pesados.

**-Proposito**

Sua proposta inicial era facilitar o modo que usuários lidavam com os recursos da máquina e consumir apenas os recursos necessários de desenvolvimento. Assim acelerando o desempenho de suas aplicações.

Hoje você consegue desenvolver em node JS através de diversas plataformas como Windows, Mac e Linux possibilitando assim diversos tipos diferentes de aplicações Web em apenas um código Java script.

O Node.Js proporciona ao desenvolvedor diferentes possibilidades de desenvolvimento a serem feitos na Web, portanto, o Node.js é apreciado como uma plataforma colaborativa, ágil com as seguintes características:

Responde às requisições das aplicações;

Tecnologia gratuita e livre de licenças;

[Front-end e Back-end](https://www.x-apps.com.br/o-que-e-front-end-e-back-end/);

Código aberto: possibilita uma comunidade ativa que contribui com o projeto;

Programação JavaScript: linguagem popular da web;

Performance: baixo consumo de memória e aproveitamento da CPU” (X-apps, 2019)

**-Motivos para aprender**

Um dos motivos para se aprender Node.js é pelo seu fácil manuseio, facilitando assim sua interação e permitindo programar em Javascript no lado cliente e servidor.

Essa questão é muito significativa no mercado do desenvolvimento de projetos, pelo fato de que a maioria dos apps e sites são desenvolvidos por uma equipe e o node.js facilita a interação e a comunicação entres membros da equipe e apenas esse diferencial mudou a percepção do mercado sobre Node.js

A utilização do V8 JavaScript Engine e seu design assíncrono permitem uma otimização considerável com respostas imediatas ao seu projeto, concluindo assim projetos sofisticados sem necessariamente aumentar a complexidade do seu código.

2.3.2 Plataforma Desktop

As aplicações desktop são aplicações/softwares que podem ser instalados e executados em um computador usado para realizar tarefas específicas. Algumas aplicações feitas para Desktop podem ser utilizadas por diversos usuários em um único ambiente de rede. Entretanto o desenvolvimento de aplicações web começou a tomar conta, substituir ou completar as aplicações de Desktop, que eram difíceis de serem vendidas, mantidas, compradas, atualizadas.

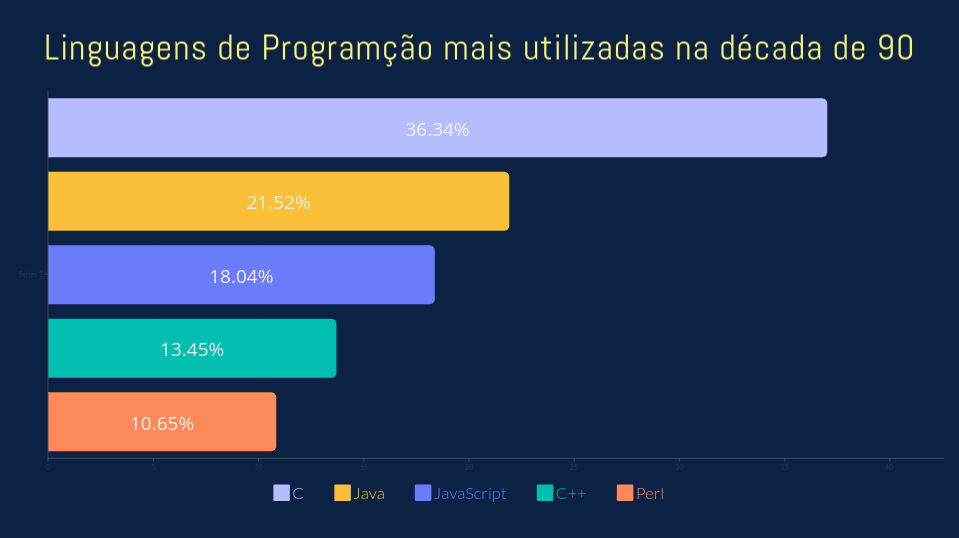
**Java**

Java é uma linguagem de programação de computadores que foi criada na década de 90 (noventa) pela equipe que a Sun Microsystems criou chamada Green Team, liderada por James Gosling. As linguagens de programação como Pascal e C (linguagens que eram bastante utilizadas na época), precisavam de compiladores (Um compilador é um programa de computador que traduz o código fonte, dependendo da linguagem de programação utilizada, e transforma em um arquivo executável, que pode ser utilizado em maquinas virtuais e físicas) específicos dependendo da plataforma que fosse utilizada. Sem isso, os programas só seriam executáveis para um único sistema.

A partir disso o projeto Green Talk foi criado e mais para a frente virou a linguagem Oak. A linguagem Oak foi criada para interligar dispositivos diferentes e realizar comunicações entre si. Porém mais para frente a equipe teve um problema de direitos relacionados com o nome “Oak” e teve que mudar o nome da linguagem em homenagem a uma gíria relativa a “Café quente”: Java Coffe.

Em 1994 (mil novecentos e noventa e quatro), devido ao advento da linguagem HTML (HyperText Mackup Language) e da WWW (World Wide Web), com isso o objetivo do grupo Green Talk foi reestabelecido com o objetivo de criar um Navegador capaz de compilar programas em Oak. Surge então o navegador HotJava e a linguagem Java. A partir daí vários projetos surgiram para popularizar a linguagem como o Java Ring e sondas espaciais feitas pela NASA. Em 2008 a Sun Microsystems foi adquirida pela Oracle. Abaixo um gráfico com as linguagens de programação mais utilizadas na década de 90:

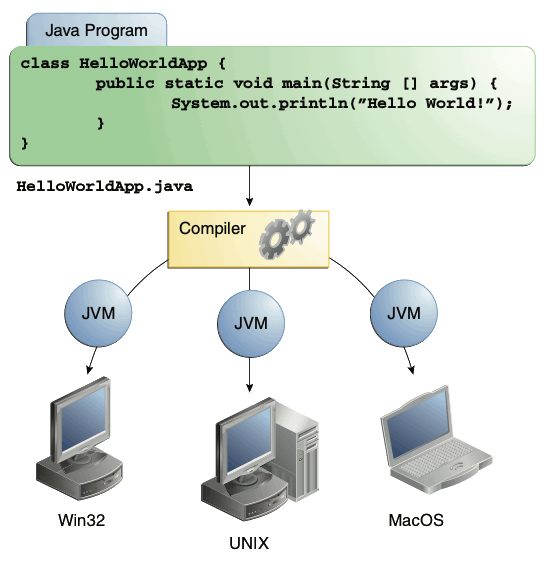
Figura - Linguagens



Fonte: https://blog.matters.tech/migrating-from-php-to-javascript-with-node-js-155534498b58

A tecnologia Java é tanto uma linguagem de programação quanto uma plataforma. O objetivo inicial da linguagem foi a questão de sua portabilidade e segurança. Na linguagem de programação Java, todo código fonte obtém uma versão de si mesmo em bytecodes (Linguagem de máquina da Java Virtual Machine ou “JVM”) que são arquivos criados com a extensão de arquivo “.class” e quando o Java utiliza “.java”. Como o Java VM está disponível em muitos sistemas operacionais diferentes é possível compilar o “.class” no Windows, MacOS, Linux e outros.

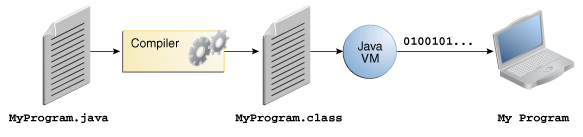
Figura - Java



Fonte: https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html

Diferente de outras linguagens, o código fonte de Java não é compilado nativamente e executado diretamente pela máquina e sim em um código intermediário gerado em “bytecode” que através do compilador é gerado um arquivo com extensão “.class” e depois traduzido para linguagem de máquina (binário) através da Java VM e finalmente o resultado do programa é gerado no computador. 

Figura - Java compiler



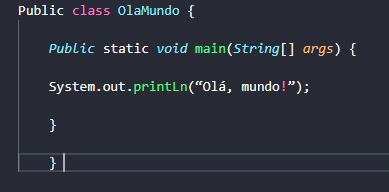
Fonte: https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html

Desta forma o código fonte se torna altamente portátil sendo possível compila-lo em qualquer computador que tenha Java VM instalado.

**Sintaxe do Java:**

**Exemplo de código que exibe mensagem “Olá, mundo! ”:**

Figura - Exemplo com "Olá, mundo"



Fonte: Próprio Autor

**Principais características da linguagem:**

1. Orientado a objetos: adota o paradigma da programação orientada a objetos.
2. Portabilidade: Escreva uma vez, execute em qualquer lugar.
3. Distribuído: Fornece uma grande variedade de bibliotecas padrões para que os programas consigam ser distribuídos.
4. Garbage Collection: Processo do qual o Java executa o gerenciamento de memória otimizando a execução e a limpeza da memória.

**Principais componentes do Java:**

1. JVM (Java Virtual Machine): é uma camada intermediária entre o sistema e a aplicação.
2. JRE (Java Runtime Environment): é utilizado para compilar aplicações feitas com Java.
3. JDK (Java Development Kit): é o kit de desenvolvimento Java, sendo utilizado para aqueles que desenvolvem aplicações em Java.

**Principais ambientes de desenvolvimento Java:**

1. Java SE (Java Standart Edition)
2. Java EE (Java Enterprise Edition)
3. Java ME (Java Micro Edition)

**Onde posso programar em Java?**

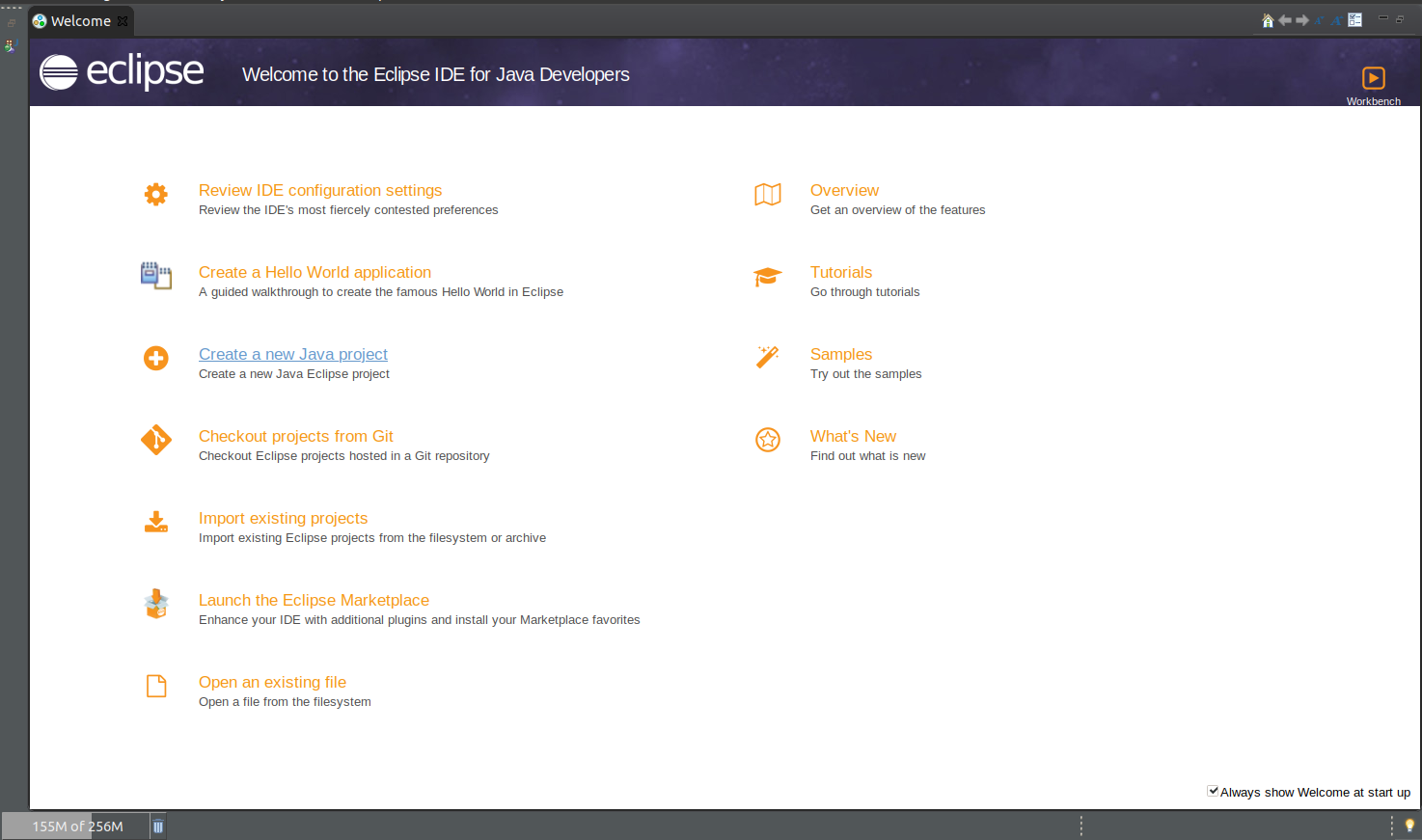
Existem muitas plataformas ou ides que podem lhe ajudar a programar em Java, mas atualmente as mais populares no mercado são: Eclipse, NetBeans e o IntelliJ IDEA. Antes de explicarmos melhor como funciona o processo de desenvolvimento de uma aplicação Java dentro de uma plataforma de desenvolvimento integrado, vamos entender o que é uma IDE (Integrated Development Environment).

Uma IDE ou Ambiente de Desenvolvimento Integrado é uma plataforma da qual você pode editar um código, acessar um terminal, executar um script, debugar e compilar usando um único ambiente. Uma IDE é repleta de funcionalidades para lhe ajudar no processo de desenvolvimento de um programa/aplicação, além das funcionalidades citadas anteriormente ela também oferece extensões, plugins e facilita o processo de escrita utilizando um sistema de inteligência artificial que autocompleta o código e sugere alterações, agiliza o processo de desenvolvimento do programador melhorando assim seu desempenho e agilidade, facilitando assim sua vida. Além das próprias IDEs existem alguns editores de código que se comportam da mesma maneira como o Visual Studio Code ou VSCode para os mais íntimos que é o editor de código mais utilizado no mundo atualmente, é claro que existem IDEs focadas em certas linguagens em específico, mas normalmente os desenvolvedores optam por desenvolver seus projetos em editores de código mais avançados como o VSCode por sua facilidade de gerenciamento de projetos em múltiplas linguagens diferentes, por exemplo, no VSCode você é capaz de editar códigos em Python e JavaScript simultaneamente e compilar, debugar, etc.

Como posso desenvolver minha aplicação Java em uma IDE ou editor de código?

Antes de começar a realmente programar em Java, precisamos de duas coisas: O JDK e uma IDE. Como aviamos citado antes o JDK é o kit de desenvolvimento em Java, ou seja, todo desenvolvedor Java precisa instalar esse kit em sua máquina para poder programar e compilar arquivos Java, já a IDE que iremos utilizar nesse exemplo será o Eclipse. Logo após a instalação do JDK e a IDE Eclipse é só criar um novo projeto Java e está tudo pronto!

Figura - Eclipse



Fonte: Próprio Autor

**Python**

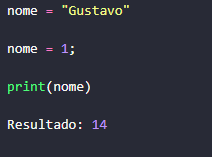
Python foi criado no final dos anos oitenta por Guido Van Rossum no Centro de Matemática e tecnologia da Informação (CWI), na Holanda. O nome da língua veio em referência a uma série de televisão chamada Monty Python do qual o criador gostava. Van Rossum é o principal autor de Python. Python é uma linguagem de programação interpretada da qual o objetivo é focado em trazer uma sintaxe que favoreça a legibilidade do código, além de ser “free”. Atualmente, a linguagem é gerenciada pela Python Software Foundation. Além da sintaxe simples, a linguagem de programação é também multi-paradigma, pois suporta orientação de objeto, programação imperativa e, em menor escala, programação funcional. Usa uma tipagem forte e dinâmica, o que permite o próprio interpretador entenda o tipo da variável sem que o usuário precise declarar.

**O que é uma linguagem com tipagem forte e dinâmica?**

Tipagem dinâmica significa que o tipo de dado que a variável irá receber pode mudar ao longo do código, já a linguagem com tipagem estática é justamente o contrário, uma vez que o valor for declarado como inteiro ele não pode mudar o tipo no decorrer do código, a não ser que haja uma intervenção do desenvolvedor, como por exemplo:

Tipagem dinâmica:

Figura - código tipagem dinâmica

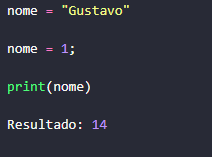


Fonte: Próprio Autor

*Resultado: 14*

*Tipagem estática:*

Figura - código tipagem estática

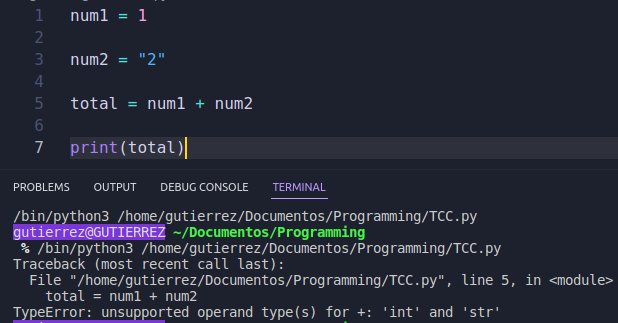


Fonte: Próprio Autor

Resultado: Erro, o valor “nome” já foi definido como String, linha 1.

Python além de ter uma tipagem dinâmica ele também é uma linguagem fortemente tipada, ou seja, caso o desenvolvedor, no código, queira realizar uma operação matemática entre dois valores com tipos distintos lhe resultará em erro ou exceções, portanto essas operações somente podem ser realizadas através de variáveis do mesmo tipo ou com a realização da conversão (casting) de valores. Por exemplo:

Figura – Exemplo código soma de variável do tipo String e Number



Fonte: Próprio Autor

Resultado: Erro

Para o compilador é impossível somar um número com uma variável de texto.

Python também é uma linguagem multiplataforma, podendo ser utilizada e desenvolvida em Windows, GNULinux, UNIX e MacOS/Apple, assim como outras linguagens, indo desde supercomputadores até telefones celulares. Além de ter alta portabilidade ela é bem poderosa tendo uma infinidade de bibliotecas onde você pode encontrar coisas desde criações de jogos até web development. E o melhor de tudo, Python é uma linguagem de programação gratuita, assim como as principais encontradas dentro do mercado. Python foi feita tendo base na linguagem ABC, parte da sintaxe derivada do C e compreensão de listas, funções anônimas e função map derivadas da linguagem Haskell.

**Principais características da linguagem:**

1. Fácil de aprender
2. Fácil leitura e compreensão
3. Fácil manutenção
4. Multiplataforma
5. Modo interativo

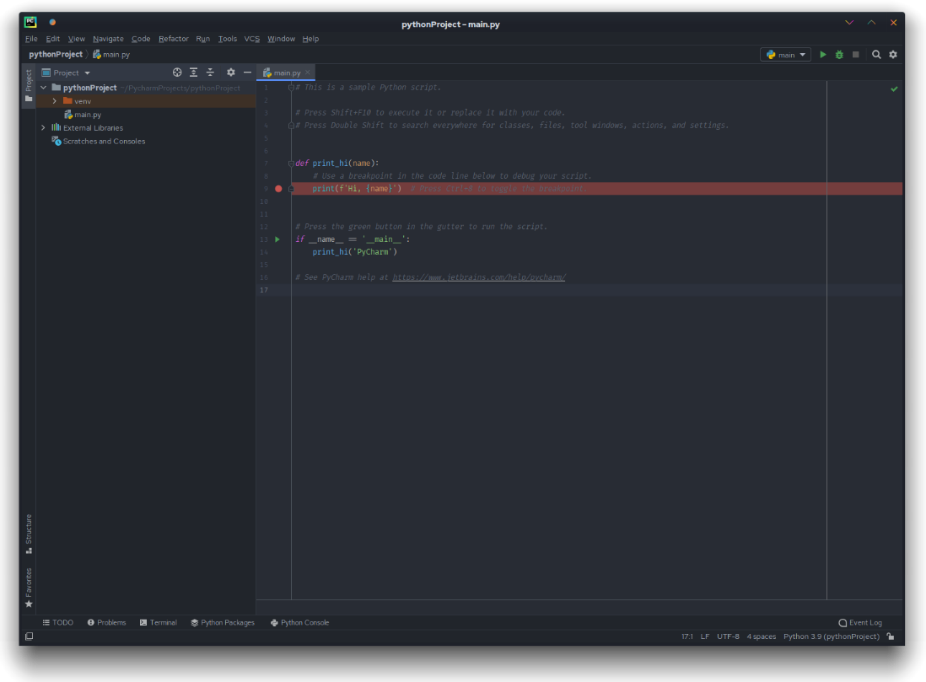
**Curiosidades sobre a linguagem:**

1. Fácil de ler, escrever e ser mantida;
2. StackOverflow considerou Python a linguagem de renome que mais cresce, e a linguagem que os programadores mais desejam aprender;
3. Eficiente para fazer protótipos;
4. Altamente escalável e poderosa;
5. Open-Source;
6. Não utiliza ponto e vírgula (;) para finalizar uma instrução;
7. Uma variável pode armazenar diferentes tipos de dados;
8. Não há chaves ({}) para delimitar o início e final de um bloco de código.

**Onde eu posso programar em Python?**

Python pode ser instalado diretamente do site oficial da própria organização (python.org), assim como Java, Python pode ser desenvolvido dentro de ide’s feitas especificamente para a linguagem como o JetBrains PyCharm:

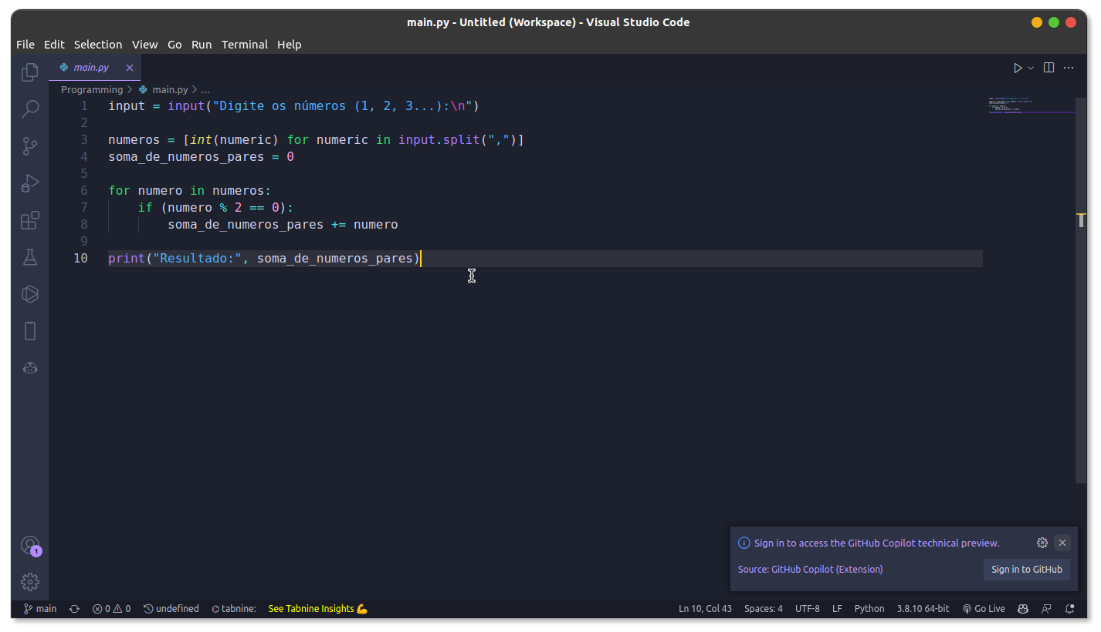
Figura - PyCharm



Fonte: Próprio Autor

E é claro em editores de código como o VSCode:

Figura - VSCode



Fonte: Próprio Autor

**Motivos para aprender Python:**

1. Facilidade: Python além de ser uma linguagem altamente poderosa ela é bem simples. Utilizando uma sintaxe simples, ela permite realizar operações com muito mais facilidade do que linguagens complexas como Java ou C++;
2. Aumento de produtividade: Python é uma linguagem bastante intuitiva e consequentemente, irá gerar menos erros no código, tornando assim o processo mais rápido. Além disso, com um conhecimento prévio/básico de Python, você já consegue automatizar tarefas e processos que teria de fazer a mão.
3. Comunidade ampla e popularidade: Python é uma linguagem cuja comunidade é altamente ativa e consequentemente os materiais didáticos são mais amplos e acessíveis na internet. Sem contar que existem diversas bibliotecas comunitárias feitas para Python disponibilizadas gratuitamente por diversos usuários que possibilitam executar funções ainda mais elaboradas.
4. Utilização versátil: Python nos permite executar diversas operações integradas com outras linguagens de programação (cross-language), como Java, bibliotecas C e C++ e componentes .NET.

1. Automação: Além de tudo Python é uma ótima escolha para aqueles que querem automatizar tarefas no seu dia-a-dia. Uma vantagem é que os scripts de Python, pode ser usado por outros scripts, como se emprestassem funções para outros projetos, importando-as como bibliotecas.

Sintaxe Python:

Exemplo de programa que mostre a mensagem “Olá, mundo! ”

Figura – Exemplo de código pra exibir um “Olá, Mundo!”



Fonte: Próprio Autor

**C#**

A convite da Microsoft, Anders Hejlsberg formou uma equipe de programadores para desenvolver uma nova linguagem de programação. Em 2001, a ECMA (European Computer Manufacturers Association) aprovou o C# e a linguagem recebeu a especificação ECMA-334. Mais tarde, em 2003, tornou-se padrão também da ISO, recebendo a especificação de ISO/IEC 23270. O C# surgiu com a finalidade de proporcionar a criação de aplicações sobre a plataforma da .NET Framework, com o objetivo de tornar menos rígido o desenvolvimento de aplicativos. Dessa maneira, o desenvolvedor não desenvolve apenas para um dispositivo em específico, e sim para toda a plataforma .NET Framework.

A linguagem C# foi desenvolvida baseada na sintaxe de programação semelhantes a C, C++ e Java. Assim, desenvolvedores habituados com alguma dessas linguagens conseguem desenvolver soluções sem dificuldades, pois ela mantém recursos base destas linguagens e apenas adiciona outros exclusivos da própria linguagem.

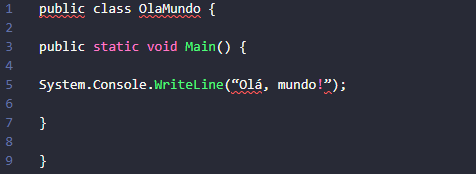
Em relação à linguagem C, o C# tem objetivo na compilação de alto nível, já o C é voltado para o desenvolvimento de baixo nível. De acordo com MSDN (2012), o C#, torna o C++ mais simples e introduz elementos que não estão disponíveis em Java, como: tipos primitivos com valores nulos, delegações, expressões lambda e acesso direto à memória.

O C# é uma linguagem de programação criada pela Microsoft, sendo a que mais se destaca da plataforma .NET. É uma linguagem que suporta diversos paradigmas como orientação a objetos, ela suporta conceitos básicos como herança, encapsulamento e polimorfismo. Além de se tratar de uma linguagem fortemente tipada e case-sensitive, ou seja, diferencia letras maiúsculas e minúsculas.

**Sintaxe do C#:**

Exemplo de programa que mostre a mensagem “Olá, mundo! ”:

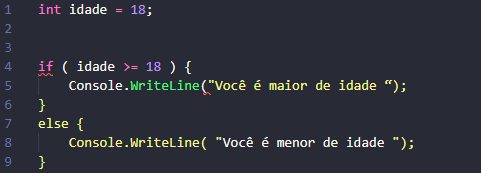
Figura – Sintaxe de “Olá, Mundo!” Em C#



Fonte: Próprio Autor

**Exemplo de utilização de if e Else:**

Figura – Exemplo de IF e Else



Fonte: Próprio Autor

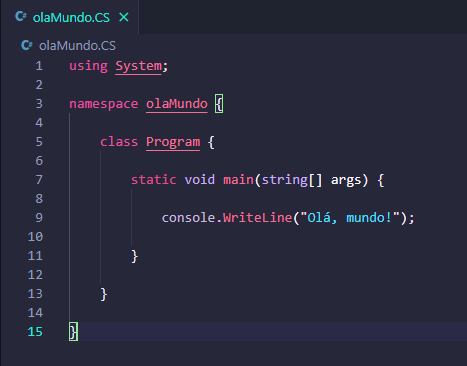
**Características do C#:**

1. A maioria das classes do .NET foram criadas para linguagem C#;
2. É case-sensitive, ou seja, faz diferenciação entre letra minúsculas e maiúsculas;
3. Consiste em uma linguagem totalmente orientada a objetos;
4. É uma linguagem fortemente tipada, o que evita erros oriundos de uma manipulação de tipos ou atribuições;
5. Os projetistas de C# costumam dizer que essa linguagem é tão poderosa quanto o C++ e tão simples quanto o Visual Basic.

Onde eu posso programar em C#?

Assim como as outras linguagens de programação, C# pode ser desenvolvido através de IDEs e Editores de códigos como VSCode. Entretanto existe uma IDE dedicada ao desenvolvimento de software das linguagens .NET como: Visual Basic, C, C++, C# e F#, chamada Visual Studio. Muito cuidado para não confundir Visual Studio com Visual Studio Code, apesar das duas terem o nome bem semelhante e serem da mesma empresa (Microsoft), elas são bem distintas em relação a usabilidade, interface, ferramentas e objetivo.

Figura – Visual Studio



Fonte: Próprio Autor

A maior diferença entre as duas plataformas é o tipo de ferramenta em que elas consistem, enquanto Visual Studio Code é apenas um editor de texto, o Visual Studio é uma IDE dedicada ao desenvolvimento de aplicações .NET. Outra diferença notável é que enquanto o Visual Studio Code é gratuito, o Visual Studio é pago.

**C# para jogos**

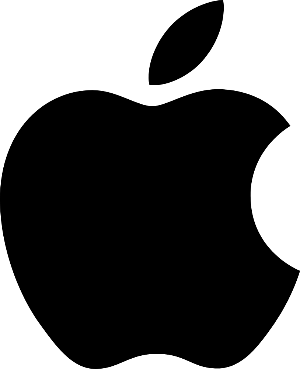
Muitos jogos famosos hoje em dia utilizam a linguagem C# para seu desenvolvimento como: Animal Crossing: Pocket Camp, Escape from Tarkov e muitos outros. Além de ser umas das plataformas mais populares de desenvolvimento de jogos no mercado, a Game Engine Unity utiliza C# como uma de suas línguas nativas para o desenvolvimento de seus projetos, 50% (cinquenta por cento) de todos os jogos mobile foram desenvolvidos utilizando o C#. Com uma alta compatibilidade com o C# cujo os scripts são feitos utilizando a sintaxe do C#, porém com APIs e Frameworks são fornecidos pela própria Unity para a elaboração do projeto.

2.3.2 Plataforma Mobile

Desenvolvimento mobile não se trata de apenas construir aplicativos para celulares e sim de fazer qualquer tipo de desenvolvimento para qualquer dispositivo móvel. As principais características desse tipo de desenvolvimento é a portabilidade, a conveniência e a praticidade.

**Principais plataformas do desenvolvimento Mobile**

Figura - Apple



Fonte: Imagem: https://pt.m.wikipedia.org/wiki/Ficheiro:Apple\_logo\_black.svg

**iOS**

O iOS é um sistema operacional móvel criado e desenvolvido pela Apple. A cada ano é lançado uma nova versão do sistema operacional, que atualmente (2022) está na sua versão iOS 15.4.1. A empresa também não permite que o Sistema operacional seja utilizado em terceiros. A Apple foi a primeira empresa a introduzir o uso dos dedos para gerir a interface, descartando o uso de canetas para isso. Foi a primeira plataforma a trazer o desenvolvimento móvel para os dias atuais renovando a ideia do que é um dispositivo móvel e software.

**O que é preciso para desenvolver em iOS?**

Conhecimento básico em Swift

Swift é uma linguagem de programação desenvolvida pela Apple focada em desenvolvimento mobile com exclusividade do sistema operacional IOS. Com muitas características modernas, a linguagem de programação Swift é a linguagem de programação muito expressiva e flexível. Além disso aceita funções como objetos de primeira classe, ou seja, você pode utilizar outras funções como parâmetros de outras funções.

**Familiaridade com ides**

Como a Swift tem acesso a bibliotecas da Cocoa, interface de desenvolvimento de apps para computadores da Apple, é necessário ter conhecimento e experiência para trabalhar em plataformas e ferramentas como Xcode ou AppCode, por exemplo. Além de trabalhar com a Swift para o desenvolvimento de aplicações mobile IOS, você também pode desenvolver as mesmas em Objective-C. Uma linguagem orientada a objetos, também uma das linguagens mais antigas quando o assunto se trata de programação, estruturada e desenvolvida pela StepStone em 1980.

**Ambiente de desenvolvimento IOS**

Além de algumas opções de ides para desenvolvimento de aplicações IOS a mais utilizada dentre os programadores de aplicativos IOS, o Xcode é a IDE mais utilizada, mas para ser utilizada é necessário que você possua a máquina específica para o sistema iOS, que no caso, podem ser dispositivos e modelos OS X, Macbook Pro Macbook Air, Macbook Mini e derivados.

**Pontos negativos**

* Custos;
* A própria linguagem;
* Tempo.

Infelizmente para que seja possível o desenvolvimento de aplicações iOS com Swift e Xcode é necessário que você tenha ao menos um dispositivo da Apple como algum Macbook ou Macintosh (iMac), o que pode gerar alguma barreira socioeconômica e na maioria das vezes menos popular. Normalmente os custos para adquirir um Macbook por exemplo, pode variar de 5 a 9 mil reais dentre os mais antigos, enquanto os mais recentes ficam na faixa de 9 a 18 mil reais.

Figura - Android



Fonte: https://commons.wikimedia.org/wiki/File:Android\_logo\_2019.png

**Android**

A é o nome do sistema operacional baseado em Linux que pode ser utilizado em celulares, netbooks e tables. É desenvolvida pela Open Handset Alliance, um grupo organizado por várias empresas, dentre elas a Google.

A principal concorrente do iOS, o Android foi lançado em 2008, mas o Sistema Operacional mesmo começou muito antes disso, em outubro de 2003. Possui serviços bem completos com a marca Google, por exemplo, quando você compra algum produto do qual o sistema operacional é Android, todos os aplicativos são interconectados com outros sistemas da Google, além disso o dispositivo já vem com alguns apps pré-instalados como o Google Play, Google Play Filmes e outras funcionalidades exclusivas da google, facilitando a vida do usuário com a experiência de uso.

Além do fato de ser um sistema operacional open-source o Android traz muitas vantagens para seus utilizadores tais como: Armazenamento-extra, variedade de aparelhos, Atualizações rápidas, Integração com aplicativos Google, Backup fácil.

Onde desenvolver aplicações Android?

O Android Studio é uma da IDEs mais populares quando o assunto é desenvolvimento mobile para Android. Ela oferece muitas ferramentas que contém o SDK, debuggers e emuladores necessários para o desenvolvimento, além disso, outra vantagem de utilizar o Android Studio é que você pode utilizar em qualquer sistema operacional, sendo Mac, Windows ou Linux. Junto à JetBrains o Android está desenvolvendo uma nova linguagem para aplicações em android chamado Kotlin.

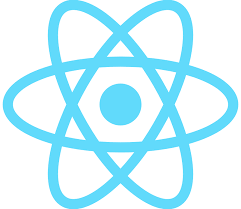
**Vantagens:**

* Sistema operacional Open-Source;
* Funciona de forma leve junto aos aparelhos com mais hardware;
* Possui touch sensível em análise a outros aparelhos antigos;
* Aplicativos grátis na Android Market;
* Conexão à rede social;
* Conexão à internet;
* Aparelhos mais baratos nas operadoras.

**Desvantagens:**

* Vírus na loja Android Market;
* Problemas na atualização;
* Aplicativos funcionam para alguns aparelhos e para outros não funcionam;
* Falta de atualização do sistema operacional para o aparelho;
* Aplicativos pagos mais caros que em outras lojas.

Figura - React

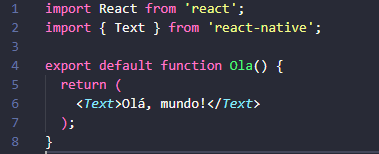


Fonte: https://reactnative.dev/

**React Native**

React Native é uma biblioteca JavaScript criada pelo Facebook. É usada para desenvolver aplicativos para os sistemas Android e iOS de forma nativa. Criada em 2015 sobre a licença MIT, o React Native é um Framework de JavaScript para facilitar o desenvolvimento de aplicações mobile, facilitando muito a vida de muitos desenvolvedores que atuam na área. Com o React Native é possível criar aplicações híbridas, ou seja, com um único código o desenvolvedor pode criar um app para diferentes plataformas, ela permite acessar a interface e os recursos nativos do Android e iOS utilizando o próprio JavaScript, o código é produzido de forma semelhante ao React para web e possuí a base de conhecimento compartilhada entre o desenvolvimento mobile e front-end.

Figura - Código React 1



Fonte: Próprio Autor

**Vantagens de utilizar React Native para o desenvolvimento de aplicações mobile:**

* Desenvolvimento Econômico;
* Entrega mais rápida de projetos de aplicativos;
* JavaScript;
* Requer equipes menores;
* Open-Source;
* Hot Reloading;
* Comunidade de desenvolvedores ativa;
* Excelente desempenho do aplicativo;
* Aparência Nativa;
* Design Modular.

**Desvantagens:**

* Desafios de depuração e compatibilidade;
* Requerimento de desenvolvedores nativos;
* Depende do Facebook;
* Gerenciamento de memória não é excelente;
* Problemas com licença e patente;
* Adoção lenta das funções mais recentes;
* Desafios de segurança de JavaScript;
* Desempenho vs. Problemas de recursos.

# 2.4 Banco de dados

Banco de dados de acordo com (Laundon, et al., 2014) é caracterizado como uma coleção de dados interligados de maneira organizada que se relacionam para dar mais eficiência durante uma pesquisa ou estudo científico, seu principal objetivo como o próprio nome diz é armazenar e salvar os dados utilizados.

No começo, cada entidade era um arquivo de dados que vinha com um software bem básico apenas para manipular os dados dessa entidade, nesses softwares era possível realizar operações de cadastro, alteração, exclusão e consulta de arquivos digitais

O primeiro projeto de banco de dados foi projetado pela IBM (International Business Machines Corporation), na década de 60, época na qual eles fizeram grande investimento para solucionar problemas desse banco de dados digital primitivo, nesse momento surgiram vários modelos de banco de dados

2.4.1 Modelagem de Dados

A modelagem de dados ou modelagem do banco de dados consiste em aplicar técnicas teóricas e práticas para a realização de um banco de dados consistente, não redundante e que se encaixe em qualquer tipo de Sistema Gerenciador de Banco de Dados. Ao realizar essa modelagem você criará uma representação visual dos dados conforme são usados (Instituto Federal Santa Catarina, 2015)

A modelagem de dados está dividida em 3 fases: Modelo Conceitual, Modelo Lógico e Modelo Físico, abaixo segue a definição de cada uma:

**Modelo Conceitual**

O Modelo Conceitual é o nível mais alto da abstração, seu foco é descrever as entidades envolvidas em um negócio, definindo seus atributos e o modo que se relaciona com outras entidades. O DER (Diagrama Entidade Relacionamento) é um exemplo de Modelo Conceitual criado em forma gráfica, na qual se deve identificar todas as entidades e relacionamentos de um determinado “universo”. (DevMedia, 2022)

Sua principal função é capturar os requisitos de informação e regras de negócio sob o ponto de vista do negócio, por isso o Modelo Conceitual deve ser feito na fase de levantamento de requisitos onde o cliente está inserido.

Abaixo estão os principais componentes do modelo conceitual e suas respectivas definições de acordo com (Lopes, 2022):

**Entidades**

Uma entidade é a representação de um objeto que existe e é distinguível de outros, por exemplo entidades podem ser pessoas, lugares, objetos, profissões etc. As entidades geralmente são representadas por um Retângulo

Figura – Representação Entidade Forte e Fraca

Diagrama

Descrição gerada automaticamente

Fonte: https://static.wixstatic.com/media/e523ba\_51dbdffd232d4e6c944f0da6b316e3e9.jpg/v1/fill/w\_525,h\_226,al\_c,lg\_1,q\_90/e523ba\_51dbdffd232d4e6c944f0da6b316e3e9.webp

As entidades podem ser classificadas em dois tipos de Entidades: Entidades Fortes e Entidades Fracas

As entidades fortes são aquelas que são independentes, que não dependem de nenhuma outra entidade para a identificação para serem criadas, elas dependem apenas delas mesmas. Já a entidade fraca é aquela que depende da existência de outra entidade para ela existir/ se identificar, um exemplo pode ser um Dependente de um Funcionário de uma Empresa, pois só vai haver “Dependente” se tiver um funcionário

**Relacionamentos**

Relacionamentos é a maneira que as entidades se ligam, elas geralmente são ações que associam/conectam as entidades. Assim como as entidades, os relacionamentos devem expressar um significado dentro do “universo” modelado.

Os relacionamentos podem ser representados através de um losango, eles também são divididos em 2 relacionamentos: Relacionamento forte x Relacionamento fraco

Figura – Representação de Relacionamento Forte e Fraco

Diagrama

Descrição gerada automaticamente

Fonte: https://static.wixstatic.com/media/e523ba\_df6d9ff350be4d1696adaeb596d301cc.jpg/v1/fill/w\_671,h\_385,al\_c,lg\_1,q\_90/e523ba\_df6d9ff350be4d1696adaeb596d301cc.webp

Relacionamento fraco irá existir quando houver uma entidade fraca, isso se deve pelo fato de tanto a entidade quanto o relacionamento dependem de outra entidade para sua “existência”, já o relacionamento forte ele não depende de entidade para existir, ele é apenas usado para conectar as entidades

Ao criarmos relacionamentos no modelo conceitual devemos aplicar 1 tipo de conceito na hora da criação do mesmo: a cardinalidade

**Cardinalidade**

Cardinalidade corresponde a representação de quantas vezes um determinado elemento de uma entidade pode estar associado a um relacionamento e a outras entidades. Ela pode ser definida em cada um dos lados do relacionamento e é representada por um conjunto (x, y) onde x é a cardinalidade mínima e o y a máximo, porém em alguns mapeamentos é comum que apareça apenas cardinalidade máxima.

A cardinalidade mínima é utilizada para representar uma obrigatoriedade/opção do relacionamento. Enquanto isso a cardinalidade máxima define a quantidade máxima de vezes que o elemento da entidade pode estar associado no relacionamento.

O uso da cardinalidade é obrigatório, pois é a partir da cardinalidade que se refletem regras que devem ser representadas em um modelo conceitual e até mesmo no mapeamento do modelo logico.

Existem 3 tipos de Cardinalidade:

0: Geralmente usado na cardinalidade mínima indica a opção do elemento da entidade no relacionamento;

1: Quando na cardinalidade mínima indica a obrigatoriedade de no mínimo 1 elemento da entidade no relacionamento, e para cardinalidade máxima indica que o máximo vai ser elementos da entidade que esse relacionamento vai ter é 1;

N: Geralmente usado na cardinalidade máxima indica que pode haver mais de 1 elemento da entidade no relacionamento;

A partir desses 3 tipos de cardinalidade, existem 3 tipos de relacionamento

Relacionamento 1 para 1 (1:1): onde apenas 1 elemento da entidade A se relaciona apenas com 1 elemento da entidade B

Relacionamento 1 para vários (1:N): onde apenas 1 elemento da entidade A relaciona-se com vários elementos da entidade B

Relacionamento vários para vários (N:N): onde vários elementos da entidade A relaciona-se com vários elementos da entidade B

Exemplo de Cardinalidade em um DER

Figura – Sentido da Leitura de Cardinalidade

Diagrama

Descrição gerada automaticamente

Fonte: https://static.wixstatic.com/media/e523ba\_bca6c4a57c1c440e9a3fa41f0b0c9518.jpg/v1/fill/w\_525,h\_186,al\_c,lg\_1,q\_90/e523ba\_bca6c4a57c1c440e9a3fa41f0b0c9518.webp

A figura 55 explica que um empregado pode trabalhar e trabalha em apenas uma empresa e que uma empresa pode ter nenhum ou vários empregados, e mostra também o sentido da leitura da cardinalidade.

**Atributos**

Um atributo pode ser considerado uma “característica” da entidade, eles são responsáveis por atribuírem informações que vão caracterizar a entidade, geralmente são representados por pequenos círculos que são ligados a entidade na qual eles pertencem.

Atributos podem ser divididos em 4 tipos:

**Atributo Identificador**

O atributo identificador, como o próprio nome diz é responsável por identificar uma entidade, geralmente são dados únicos como: Códigos ou Registros. O mesmo é representado por um círculo preenchido no Modelo Conceitual

**Atributo Simples**

O Atributo não identificado ou atributo simples, corresponde a maioria dos atributos, ele não possui nenhuma característica especial, exemplos de atributos simples são: nome, datas. Ele é representado por um círculo sem preenchimento no Modelo Conceitual

**Atributo Multivalorado**

Os atributos multivalorados, são usados quando você pode ter vários itens para um mesmo campo, como é o caso do número telefone que uma pessoa pode ter mais de um número, esse atributo é recomendado apenas no Modelo Conceitual para abstrair os dados necessários. O Atributo Multivalorado é representado por um círculo com um asterisco do lado.

**Atributo Composto**

Utilizamos o atributo composto, quando o atributo que estamos a usar pode ser subdividido em partes menores, um exemplo simples, um atributo endereço, pode ser subdividido em Rua, Cidade, Bairro etc. Igual ao Atributo multivalorado seu uso é recomendado apenas no Modelo Conceitual, e é representado por um atributo que recebe outros atributos “menores”.

Figura – Representação dos Atributos

Diagrama

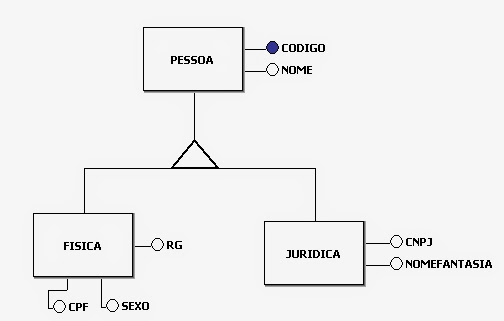
Descrição gerada automaticamente

Fonte: https://static.wixstatic.com/media/e523ba\_43d4d62fda874612abc4ddddd2ed960f.jpg/v1/fill/w\_378,h\_465,al\_c,lg\_1,q\_90/e523ba\_43d4d62fda874612abc4ddddd2ed960f.webp

**Generalização e especialização**

A Generalização e Especialização consiste em separar hierarquicamente entidades que terão subdivisões, dependendo do caso de uso podemos notar que uma entidade poderá ser subdividida, onde elas ainda serão a mesma entidade, porém com atributos e nomes diferentes, exemplo, em determinado caso de uso temos uma entidade pessoa e no próprio caso é dito que as pessoas ou são pessoas jurídicas, ou pessoas físicas, logo teríamos que dividir os “tipos” de pessoa

Figura – Representação de Especialização e Generalização



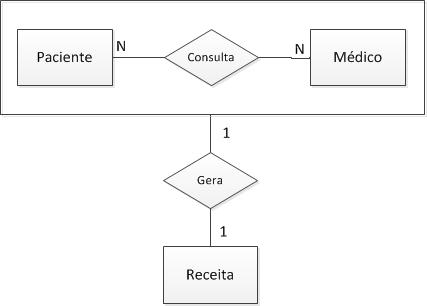
Fonte: https://lh6.googleusercontent.com/-6UrPk6Hpytg/U0IYZ5okm1I/AAAAAAAAANI/GLbESkpw2Pw/s504/genespetotal.jpg

A figura 57 representa como é feita divisão, as pessoas físicas e jurídicas passaram a ter atributos próprios, porém continuam sendo uma pessoa e herdando os atributos daquela entidade “pai” Pessoa.

**Agregação**

A Agregação é feita a partir de um relacionamento entre entidades, a partir desse relacionamento criamos uma espécie de “nova entidade”, que pode ser usada para outros relacionamentos, isso ocorre pois não podemos relacionar dois relacionamentos, então usamos a agregação para criar essa “exceção”.

Figura – Exemplo de Agregação



Fonte: https://douglasgaspar.files.wordpress.com/2013/10/mer-8.jpg

A Figura 58 nos mostra um exemplo prático, onde o relacionamento consulta entre Médico e Paciente, é utilizado para gerar uma receita, nesse caso a Consulta é uma agregação pois ela está utilizando uma relação para se relacionar com outra entidade.

**Modelo Lógico**

A Modelagem lógica ajuda na compreensão visual das informações que serão armazenadas que serão usadas para cumprir as tarefas necessárias e suas funções. Para criarmos um modelo lógico é recomendável que tenhamos um modelo conceitual, pois a partir dele iremos introduzir os elementos associados, as definições e teremos um melhor contexto para nossa estrutura de dados, pois nessa fase é que iremos aplicar os conceitos das chaves primarias, estrangeiras, da cardinalidade e dos relacionamentos.

A principal função do modelo lógico é visualizar os elementos de dados, ver como eles se relacionam e detalhar e definir o tipo dos atributos. Ao fazermos o mapeamento para o modelo lógico já iremos utilizar “tabelas” para exibir os dados (Erwin Quest Software, 2022)

Figura – Representação da Tabela do Modelo Lógico

Gráfico

Descrição gerada automaticamente  
Fonte: http://mz.pro.br/modelagem/conceitual\_logico.pdf

Na figura 59 podemos notar uma tabela, essa tabela nada mais é que uma entidade do Modelo Conceitual onde o nome da tabela, é o nome da entidade, a chave primaria é o atributo identificador, e os outros atributos são o restante dos atributos do modelo conceitual

Essa é uma conversão “simples” de apenas uma entidade, porém para fazermos o mapeamento completo devemos levar em conta a tudo que aquela entidade está relacionada e todo o “universo” do Modelo Conceitual.

Os principais relacionamentos que requerem “cuidados” na hora de serem mapeados são:

**Relacionamento um para um (1:1)**

Onde a entidade dominada irá receber o atributo identificador ou chave primária da entidade dominante, como chave estrangeira que é responsável por interligar as tabelas

Figura – Exemplo de Relacionamento 1:1

Gráfico, Diagrama, Gráfico de caixa estreita

Descrição gerada automaticamente

Fonte: https://sites.google.com/site/uniplibancodedados1/\_/rsrc/1348510710650/aulas/aula-7---tipos-de-relacionamento/aula\_7\_relacionamento3.bmp

Na figura 60 mostra o seguinte caso um gerente, gerencia um departamento, onde o gerente é a entidade dominante e departamento a entidade dominada, ao fazer o mapeamento pode se notar que o Departamento recebe o código do gerente.

**Relacionamento um para muitos (1:N)**

Para realizarmos o mapeamento desse caso primeiro precisamos identificar a entidade que recebe a cardinalidade de muitos, o N no caso e a Entidade com valor de 1

Figura – Exemplo Relacionamento 1:N

Imagem em preto e branco

Descrição gerada automaticamente

Fonte: http://mz.pro.br/modelagem/conceitual\_logico.pdf

Na figura 61 mostra um caso em que um departamento tem vários funcionários e um funcionário pertence a UM departamento, aqui ao realizarmos a conversão o funcionário irá receber a chave primária do departamento como chave estrangeira, igual mostra a figura abaixo

Figura - Tabela Departamento e Funcionários Relacionamento 1:N

Diagrama

Descrição gerada automaticamente  
Fonte: http://mz.pro.br/modelagem/conceitual\_logico.pdf

Na figura 62 mostra as 2 tabelas já convertidas e mostra também que a tabela funcionário recebeu o código do departamento, que é a chave primária da entidade departamento, como sua chave estrangeira

**Relacionamento muitos para muitos (N:N)**

Nesse tipo de relacionamento iremos criar uma tabela para a relação entre as entidades, e pegar a chave primária de cada uma das entidades

Figura - Exemplo Relacionamento N:N

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente  
http://mz.pro.br/modelagem/conceitual\_logico.pdf

Nesse caso o aluno pode cursar várias disciplinas e a disciplina “pertence” a vários alunos, logo a tabela aplicando o conceito de Relacionamento N para N, irá ficar da seguinte forma

Figura - Representação Tabelas Modelo Logico Relacionamento N:N

Diagrama

Descrição gerada automaticamente  
Fonte: http://mz.pro.br/modelagem/conceitual\_logico.pdf

Pode se notar que foi criada uma tabela recebendo as chaves primárias das tabelas aluno e disciplina como chave estrangeira.

**Relacionamento Generalização/ Especialização**

No caso de Generalização/ Especialização devemos criar uma tabela para cada entidade “filha”, e uma tabela para a entidade pai, e consequentemente as tabelas filhas irão receber a chave primária da tabela pai como chave primária e estrangeira.

Figura - Exemplo Relacionamento Generalização e Especialização

Diagrama

Descrição gerada automaticamente  
Fonte: http://mz.pro.br/modelagem/conceitual\_logico.pdf

Nesse caso a entidade veículo é categorizada em 3 tipos: terrestre, aquático e aéreo. Para esse caso iremos mapear da seguinte maneira por se tratar de uma generalização e especialização

Figura - Tabela em Caso de Generalização/Especialização

Diagrama

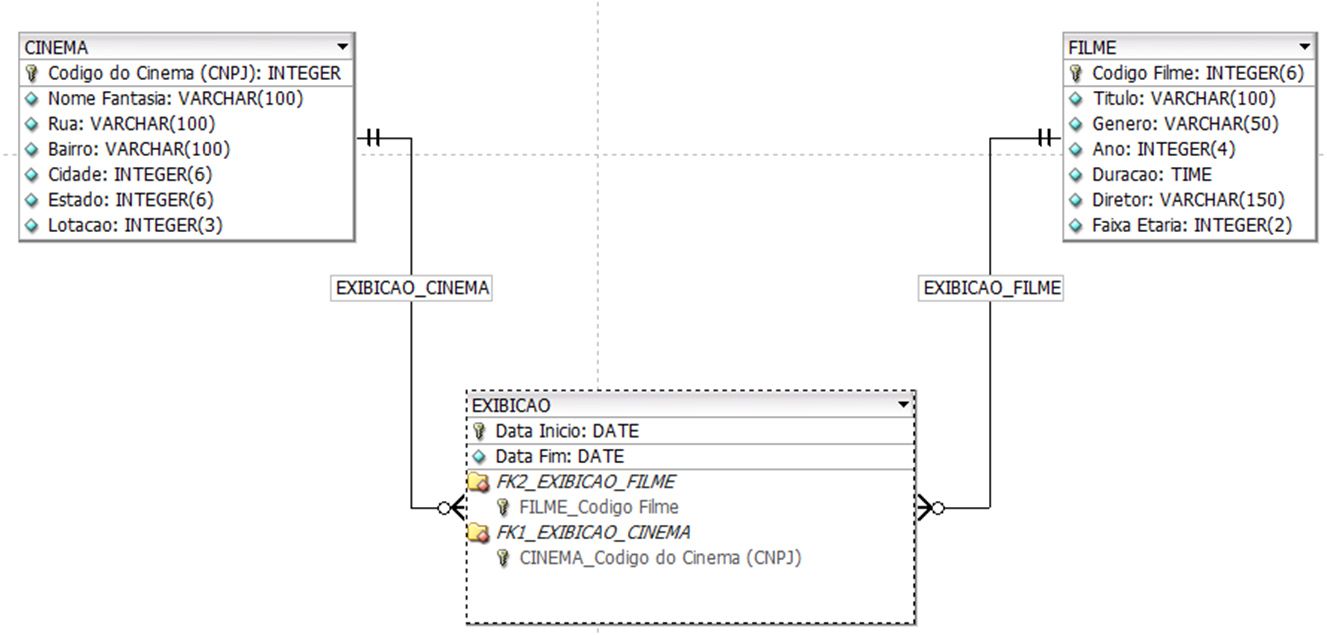
Descrição gerada automaticamente  
Fonte: http://mz.pro.br/modelagem/conceitual\_logico.pdf

Conforme mostra a figura 66 todas as entidades filhas receberam uma tabela para cada, e a entidade pai também recebeu a tabela, todas as tabelas filhas receberam a chave primária da tabela veículo, como chave primária e estrangeira

**Modelo Físico**

O Modelo físico é a última etapa da modelagem de dados, nesse modelo devemos ter uma linguagem de banco de dados definida e faremos a construção de tabelas, e definição dos tipos dos atributos no banco de dados a partir de comandos SQL/ ou NOSQL dependendo do modelo escolhido ou até mesmo por interface gráfica caso estiver usando um SGBD (Sistema Gerenciador de Banco de Dados)

Figura - Representação do Diagrama de Modelo Físico

  
Fonte: https://livrodigital.uniasselvi.com.br/GTI11\_principios\_de\_banco\_de\_dados/unidade2.html?topico=1

Como mostra a figura 67 nessa etapa da modelagem de dados que definimos o tipo dos atributos criados e seu limite de caracteres, são usados vários comandos para a criação dessas tabelas

Entre os principais comandos da Linguagem de Definição de tabelas e do CRUD (Linguagem de manipulação) estão:

Create: Usado para criação de tabelas e bancos de dados;

Primary Key: Utilizados para definição de chaves primárias;

Foreign Key: Utilizado para Definição de chave estrangeira;

Insert: Utilizado para Inserção de dados na tabela;

Update: Utilizado para atualizar dados de uma tabela;

Delete: Utilizado para exclusão de dados da tabela;

Select: Exibir Registros da tabela;

Alter: Alterar campos de uma tabela ou uma tabela;

Figura - Exemplo da Criação de uma tabela de banco de Dados no modelo físico

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Fonte: Próprio autor

A Figura x.x mostra o processo feito para a criação de uma tabela no modelo físico, onde como citado utilizamos o create table, para criamos uma tabela, e na hora de inserirmos os atributos da mesma, devemos definir seu nome, seu tipo, máximo de caracteres e Características existentes em alguns bancos de dados, nesse exemplo temos o NOT NULL que significa que o valor não pode ser nulo e o AUTO\_INCREMENT que adiciona um valor automaticamente caso ele não seja definido

2.4.2 SGBDS

Um SGBD ou Sistema Gerenciador de Banco de Dados, como o próprio nome diz é um software utilizado para o gerenciamento de Banco de Dados, nele é possível criar, modificar, eliminar, inserir dados e eliminar dados das bases de dados.

“O sistema de gerenciamento de bancos de dados (SGBD) é um conjunto de programas que gerenciam a estrutura do banco de dados e controlam o acesso aos dados armazenados. Até certo ponto, o banco de dados se assemelha a um arquivo eletrônico com conteúdo muito bem-organizado com a ajuda de um software poderoso, conhecido como sistema de gerenciamento de banco de dados. “

“O SGBD serve como intermediário entre o usuário e o banco de dados. Sua estrutura é armazenada como um conjunto de arquivos e o único modo de acessar os dados nesses arquivos é por meio do SGBD.” (Rob, et al., 2010 pp. 6, 7)

Os bancos de dados só começaram a ser operados por SGBDS (Sistemas Gerenciadores de Banco de Dados) a partir de 1970, quando o pesquisador Edgar Frank “Ted” Codd da IBM(International Business Machines Corporation ) apresentou o Modelo Relacional no Artigo “A Relational Model of Data for Large Shared Data Banks”, nesse artigo explicava para técnicos sem experiência como armazenar e extrair grandes quantidades de dados de um banco de dados, a partir dessa ideia a própria IBM decidiu criar o primeiro projeto de Sistema Gerenciador de banco de dados chamado “System R”. Antes disso eram operados por sistemas de arquivos do sistema operacional utilizado na máquina.

Hoje em dia dentre os principais modelos de SGBDS estão os Modelos Relacionais (SQL – Standard Query Language) e Modelos Não Relacionais (NoSQL - Not Only Standard Query Language), abaixo seguem suas definições e exemplos:

**Modelo Relacional (SQL)**

Como dito no anteriormente, o Modelo Relacional foi criado por Edgar Frank “Ted” Codd que trabalhava na IBM, por volta da década de 70 e consistia. Esse primeiro modelo relacional era operado pelo sistema chamado “System R” da IBM que utilizava a linguagem SQL (Standard Query Language) para a consulta do famoso System R, porém era utilizada apenas para o Sistema da IBM.

A Primeira versão padrão do SQL foi desenvolvida apenas em 1984 quase uma década após ter sido implementada no System R, pois era necessária uma padronização da linguagem, já que após ela ter sido utilizado no System R outros desenvolvedores estavam criando outras versões da linguagem.

Nos dias de hoje, a Linguagem SQL é uma das mais utilizadas no ramo de banco de dados, dentre os principais SGBDS que utilizam a linguagem estão: Oracle, SQL Server e MySQL

**SGBD Oracle**

O Oracle SGBD foi criado no ano de 1977 por Lawrence Ellison, Bob Miner e Ed Oates, com o avanço das tecnologias de banco de dados nos anos 70 e 80, eles perceberam uma oportunidade de comercializar um Sistema que gerenciasse o banco de dados relacional na época, já que nenhuma empresa estava a comercializar um sistema desse tipo.

A Linguagem Utilizada pela Oracle é PL/SQL que é uma extensão da linguagem SQL padrão, além de ser utilizado para o SGBD ela é utilizada em muitos outros softwares da empresa.

Os principais fatores que levaram o SGBD da Oracle a ser tão popular são os fatos de que oferecem uma segurança avançada de várias camadas, implementação de criptografia, estabilidade, flexibilidade de gerenciamento, suporte entre outros (Andrade, 2022)

**MySQL**

O MySQL é um SGBD Open-source que utiliza a linguagem SQL para criação e busca de banco de dados. O MySQL foi criado em 1994 na Suécia pelos finlandeses: David Axmark, Allan Larsson e Michael "Monty" Widenius, que futuramente vieram a criar a MySQL AB, desenvolvedora do SGBD, porém a empresa foi comprada pela Oracle no ano de 2010, passando a ser detentora do MySQL

De acordo com a (2022) é um dos SGBDS mais utilizados no mercado de trabalho. Um dos motivos que levam o MySQL a ser um dos mais utilizados é pelo fato de ser Open-source (código aberto), por pode se conectar a outros bancos de dados como DB2 e Oracle, e por ter muitas funcionalidades, além de uma interface gráfica ótima e intuitiva. (Souza, 2020)

**SQL Server**

O SQL Server é um SGBD que em 1988 foi criado pela Sybase, em parceria com a Microsoft, ele utiliza as Linguagens T-SQL e ANSI SQL que derivam da linguagem SQL padrão. Diferente do MySQL ele não é open-source, porém de acordo com (2022) continua sendo um dos SGBDS mais utilizados do mercado.

Os principais motivos da popularidade do SQL Server se devem ao fato de possuíram facilidade na análise de dados, sua linguagem é flexível, possuem cloud storage (armazenamento em nuvem), emitem relatórios com alta gestão de informação e têm ótima segurança e criptografia de dados. (Flexus, 2021)

**Modelo Não Relacional (NoSQL)**

Os modelos de banco de dados NoSQL (Not Only Standard Query Language) de acordo com (Amazon, 2022) são criados para modelos de dados específicos e têm esquemas flexíveis para a criação de aplicativos modernos. Os bancos de dados NoSQL são amplamente reconhecidos por sua facilidade de desenvolvimento, funcionalidade e performance em escala.

O Modelo NoSQL possibilita uma variedade de modelos de dados, como chave-valor, documento e gráficos que são optimizados para performance. Nesse modelo o registro que seria salvo em uma tabela em um banco de dados relacional, vai ser salvo como um arquivo JSON (JavaScript Object Notation), esse arquivo irá conter todos seus atributos em um único documento permitindo desenvolvimento intuitivo e escalabilidade horizontal. (Amazon, 2022)

O principal motivo de utilizaram Banco de Dados NoSQL é o fato dele ter Flexibilidade, que é o fato de permitir um desenvolvimento rápido e iterativo, a escalabilidade que faz com sejam projetados para serem escalados horizontalmente usando clusters distribuídos de hardware, em vez de escalá-los verticalmente adicionando servidores caros e robustos e Alta Performance, que traz uma maior otimização dos dados utilizados, além de ser Altamente Funcional. (Amazon, 2022)

Entre os principais SGBDS NoSQL estão o: MongoDB, Redis e Cassandra

**Mongo DB**

O MongoDB é um banco de dados orientado a documentos open-source lançado no ano de 2009 pela MongoDB Inc., seu código foi desenvolvido em C++, C, GO e JavaScript. O MongoDB é utilizado para grande quantidade de dados e que tenha bom funcionamento e performance com grandes volumes de dados. Ele oferece suporte para quase todas as linguagens de programação utilizadas atualmente.

Seu processo é mais fluido do que a de um banco de dados relacional, e seus elementos têm identificação única, os itens não são relacionados entre si e sua hierarquia é flexível e suas informações diferente de um banco de dados relacional que são armazenadas em tabelas, são armazenadas em documentos e coleções (que são subpartes do banco de dados, independentes).

As principais vantagens de usar o MongoDB são que não é necessário projetar um esquema do banco de dados para trabalhar com ele, fornece grande flexibilidade para os campos do documento, não requer injeções SQL e é facilmente integrável com Big Data Hadoop que é utilizado para armazenar dados e executar aplicações em clusters de hardwares comuns.

**Redis**

Segundo a (Amazon, 2022) o Redis é um banco de dados de chave-valor mais conhecidos atualmente de código aberto e distribuído na memória, foi lançado em 2009 pela Redis Labs e é compatível com diversas linguagens de programação. Ele oferece um conjunto de estrutura versátil, permitindo a criação de várias aplicações personalizadas. Seu nome REDIS significa Remote Dictionary Server, ou traduzindo para português servidor de dicionário remoto.

“Por conta da sua velocidade e facilidade de uso, o Redis é uma escolha em alta demanda para aplicações web e móveis, como também de jogos, tecnologia de anúncios e IoT, que exigem o melhor desempenho do mercado. ” (Amazon, 2022)

Ainda de acordo com a (Amazon, 2022) as principais vantagens de usar o REDIS são desempenho extremamente rápido, versatilidade e fácil uso, replicação e persistência, além da compatibilidade com diversas linguagens de programação no mercado

**Cassandra**

Cassandra é um banco de dados open-source não relacional e colunar, criado pelo Facebook em 2008, foi inspirado no DynamoDB da Amazon e no BigTable da Google. O banco de dados colunar utilizado na Cassandra é bem similar ao utilizado em um banco de dados relacional.

De acordo com o (DevMedia, 2022) Cassandra utiliza a Linguagem CQL (Cassandra Query Language) que é uma linguagem utilizada para facilitar a criação e manipulação de estruturas e realizar manipulações de dados, sua sintaxe é bem similar ao SQL.

Entre as principais vantagens do Cassandra é de que ele possui uma alta performance, é escalável, possui uma boa segurança e sua linguagem CQL é similar ao SQL que pode ser útil para usuários acostumados ao banco de dados relacional.

## Tecnologias Diversas

A tecnologia é um produto da ciência e da engenharia envolvendo uma série de instrumentos, métodos e técnicas projetados para resolver problemas. É a aplicação prática do conhecimento científico em vários campos de estudo.

2.5.1 Bootstrap

Bootstrap é um framework para a linguagem CSS que foi desenvolvido no Twitter nos meados de 2010, com seu principal objetivo de padronizações de ferramentas da empresa. Desde então ele passou por diversas atualizações, e atualmente ela está na sua versão 4.4.

Este framework pode combinar com JavaScript e CSS com objetivo de estilizar elementos de uma página em HTML. Ele consegue expandir suas opções além de mudar cores e formatos de letras.

Consiste em ser uma ferramenta onde proporciona interagir na página, pois acaba oferecendo series de componentes que acabam facilitando a comunicação com o usuário, como menus de navegação, controle de páginas, barra de progresso e etc.

Além que a funcionalidade que o framework pode acabar oferecendo, onde seu principal foco é permitir construção de site responsivo para dispositivo móvel. Significa que páginas podem desenvolver para funcionar em desktops, tabletes e smartphones, de formas mais simples e mais organizadas. (Souza, 2019)

**Como surgiu:**

Originalmente criado por designers e desenvolvedores no Twitter, o Bootstrap se tornou um dos frameworks de front-end e projetos de código aberto mais populares do mundo. Bootstrap foi criado no Twitter em meados de 2010 por @mdo (**Mark Otto**) e @fat (**Jacó Thornton**). Antes de se tornar uma estrutura de código aberto, o Bootstrap era conhecido como Twitter Blueprint. Ele foi desenvolvido como um guia de estilo dentro da ferramenta interna da empresa por mais de um ano antes de seu lançamento público e continua até hoje.

Originalmente lançado na sexta-feira, 19 de agosto de 2011, houve mais de 20 lançamentos desde então, incluindo duas grandes reescritas para v2 e v3. No Bootstrap 2, eles adicionaram a funcionalidade responsiva como uma folha de estilo opcional a todo o framework. Com base no Bootstrap 3, eles reescreveram a biblioteca novamente para torná-la responsiva por padrão com uma abordagem mobile-first. E no Bootstrap 4, reescreveram o projeto novamente para abordar duas importantes mudanças na arquitetura: migrar para Sass e migrar de CSS para flexbox.

**Como Bootstrap funciona:**

Bootstrap é formado por uma série de arquivos CSS e JS (JavaScript) que serão responsáveis em atribuir caracteres específicos a elementos da sua página. Tendo nele um arquivo principal em seu framework, designado de Bootstrap.css, onde contém definições para todos os estilos utilizáveis. Sendo mais direto, a estrutura framework acaba sendo formada por dois diretórios (uma estrutura para acabar organizando arquivos de computador ou que tenha referencias de outros arquivos):

- CSS: que contenha arquivos necessários para a estilização dos elementos e uma alternativa a um tema único;

- JavaScript: contém as versões do registro bootstrap.js (original e minificado), que é causante por dar as aplicações de feitio que necessitam de manuseamento interativa.

Para pôr uma característica a um constituinte, basta instruir a classe adequada na expressão “class” do constituinte a haver estilizado. Veja o exemplo:

Figura – Exemplo de class em tag HTML com Bootstrap



Fonte: Próprio Autor

Neste exemplo, foi imputado o conteúdo “rounded-sm” para a caráter da imagem. Trata-se de um feitio quão adiciona cantos arredondados no elemento. Portanto, ao tender a imagem, as características referentes a essa aula reunião aplicadas no elemento. (Souza, 2019)

**Como baixar o Bootstrap:**

Existem diferentes formas de se baixar o framework. Uma delas é executar o download da versão compilada dos códigos CSS e JavaScript por meio da página do Bootstrap (https://getbootstrap.com/docs/5.1/getting-started/download/). Há também a opção de baixar o código-fonte do framework, no instante que ele é uma ferramenta de código aberto.

Quem não quiser baixar os arquivos para poder ter acesso ao framework sem a essencialidade de instalar ele no servidor. Na prática, os arquivos de instalação ficam em outro domínio, ou seja, em outro DNS. Para isso, é apenas usar os links para acessar o BootstrapCDN e, dessa forma, adicionar as referências para os arquivos necessários para a sua utilização.

Segundo o site (<https://rockcontent.com/br/blog/bootstrap/>) – *“**Outra forma de baixar o framework é por meio de gerenciadores de pacotes. É importante dizer que o Bootstrap pode ser utilizado com diferentes linguagens de programação”*

Por este motivo, ele pode acabar sendo baixado com RubyGems, npm, do Node.js, Nuget ou Composer, e sendo utilizado para se criar um site em WordPress, em sites que são desenvolvidos em Ruby on Rails, em Asp.Net entre tais coisas. (Souza, 2019)

**Como Configurar e utilizar o Bootstrap:**

Existem diferentes formas de se configurar o framework para utilizá-lo em uma prática web. Entretanto, para que ele acabe funcionando de forma mais adequada, é necessário que adicione as bibliotecas JQuery e Popper.js, que serão necessárias para a prática de alguns componentes do Bootstrap.

Para se iniciar a empregar o Bootstrap em uma página, é necessário adicionar o registro dos principais arquivos do framework na página matriz da administração. Confira, a seguir, o código de uma página HTML com todas as referências necessárias para que o framework funcione.

Figura – Código HTML “linkando” com o Bootstrap

Texto

Descrição gerada automaticamente

Fonte: Próprio Autor

É vital seguir sua ordem de referência conforme o exemplo. Então, sua primeira referência deverá ser o arquivo Bootstrap.css localizado na tag **<head>**. Já suas referências a arquivos JavaScript deverão ser colocadas no final de sua página, antes mesmo de encerrar a tag **</body>**. A ordem de scripts deve ser: JQuery, Popper.js e Bootstrap.js.

É necessário comentar que a forma de configurar o Bootstrap vai variar de acordo com o tipo de ambiente que ele estará presente. Em WordPress, como exemplo, é possível se adicionar o framework por meio de alterações do arquivo **functions.php** do templante. (Souza, 2019)

**Funcionalidades do Bootstrap:**

O Bootstrap pode oferecer uma vasta série de funcionalidades que poderão implementar seu site. Observe agora o que se pode se fazer com a ferramenta.

**Layout Responsivo:**

Uma das funcionalidades principais do Bootstrap é permitir a adaptação do layout da sua página sendo feita de acordo com o dispositivo utilizado. Para se garantir está responsivo, o framework trabalhará a estilização com o elemento **<div>** e com utilização da classe **container**.

Colocando em prática, o elemento **<div>** funcionará criando uma série de grades, se assemelhando com uma tabela, onde será capaz de estruturar uma página com mais aptidão. Tabelas eram utilizadas antigamente para se criar layouts responsivos, porém, possuía limitações onde a função da definição de largura das colunas, o que acabou invalidando sua utilização em dispositivos menores, como celulares.

Enquanto o elemento **<div>** é mais flexível, porque permitirá que a largura será definida e redimensionada com mais facilidade. O Bootstrap atribui o elemento **<div>** a característica da classe **container**, que funcionará para definir dimensões adequadas para elementos que serão inseridos no espaço. Em resumo, o framework trabalhará com três tipos de container:

* Container: em contíguo com a caráter max-width, que determina qual é o comprimento do quadro ideal para que haja a apropriação do layout da página;
* Container-Fluid: considera toda a vastidão do quadro do dispositivo para a firmação do layout. Para isso, é considerada a caráter width:100% em todos os limites de comprimento de quadro;
* Container (Breakpoint): considera width:100% mesmo que um determinado comprimento seja atingido.

**Biblioteca de componentes:**

Outra praticidade curiosa do framework é a porção de componentes que podem ser utilizados para proporcionar maior interatividade e corrigir a ligação com o usuário. Confira, a seguir, alguns dos principais componentes do Bootstrap. (Souza, 2019)

**Alerts:**

Com o Bootstrap existe a possibilidade de configurar de forma mais simples e dinâmica tipos de alertas, com especificas cores, conforme a situação. Para demonstrar um alerta ao usuário indicar atenção, como exemplo, basta ser utilizado a classe. alert-danger e será exposto uma caixa com texto com seu fundo em vermelho. Observe a seguir:

Figura – Código para Alert do Bootstrap



Fonte: Próprio Autor

**Carousel:**

Um componente bastante empregado no Bootstrap é o Carousel. Trata-se de um slideshow, ou seja, um instrumento que permitirá a amostra de imagens de aparência responsiva. Ele também possibilitará a introdução de efeitos especiais para a trajeto da registro e controles de amostra, como os indicadores de rente e anterior.

**NavBar:**

Outro forte componente do framework é o NavBar, que permite a edificação de um sistema de navegação responsivo. É provável caracterizar diferentes formas de apresentação do menu, como tomar entre o posicionamento lateral ou superior, e limitar a forma de apresentação, que pode existir estendida ou contraída.

Também é provável estipular a forma de mostra dos links do menu, que pode existir em forma de botão, link, menu suspenso, entre outras configurações, para conceder a implementação da navegação do site. (Souza, 2019)

2.5.2 Laravel

Laravel é um framework (facilitador para o desenvolvimento de aplicações) grátis, feito para web e criado em PHP. O criador Taylor Otwell tem como principal objetivo desenvolver aplicações web com padrão de arquitetura MVC. Algumas características nativas do Laravel são sua sintaxe simples e precisa, um sistema modular com gerenciador de dependências dedicado, várias formas de acesso a banco de dados relacionais e vários utilitários indispensáveis no auxílio ao desenvolvimento e manutenção de sistemas.

O código fonte do Laravel está alojado no GitHub e licenciado sob os termos da licença MIT.

**Porque utilizar o framework LARAVEL?**

Atualmente empresas utilizam para o desenvolvimento de sistemas web, e para um melhor desempenho diariamente, seu principal objetivo é permitir que os programadores que o utilizam trabalhem de forma rápida, mantendo o código exuto e funcional.

**3 razões para utilizar Laravel:**

Seguro: Comparando com os demais frameworks php, o Laravel possui uma ampla reputação de segurança,

Popularidade: Desde a data de lançamento Laravel vai crescendo ainda mais sua popularidade como repositório do Github, tendo quase o triplo de estrelas que o segundo colocado Symfony, no segmento PHP.

Agilidade: Diminuir o tempo de esforço é uma de suas propostas. A sua estrutura simplifica muitas tarefas simples de projetos web, incluindo o cache, roteamento, sessões e autenticação

**Propósitos:**

O Laravel evolui com você. Se você está aprendendo o básico no desenvolvimento web, a grande biblioteca de documentação, guias e vídeo aulas te ajudará a aprender o básico sem ficar cansado. Ele é incrivelmente escalável (Crescer rápido). A amigável natureza de escalonamento do PHP e o suporte adicionado para sistemas de cache rápido e distribuído, como o Redis, que ajuda muito. Na verdade, **os** aplicativos Laravel foram facilmente selecionados para lidar com milhares de requisições por segundo. O Laravel combina os melhores pacotes do PHP para oferecer o framework bonito e amigável disponível para o desenvolvedor. Além disso, vários desenvolvedores talentosos do mundo todo ajudaram para sua estrutura.

**Conclusão:**

O Laravel é um ótimo Framework para desenvolvimento de aplicações web. Neste arquivo destaquei alguns de seus principais temas, características, funcionalidades, que se torna uma ótima escolha para a criação dos seus projetos.

2.5.3 Formik

Formik é uma biblioteca que auxilia desenvolvimento de formulários, simplificando sua forma de utilização dos campos de texto, numérico, checkbox entre outros. Nele obtemos a possibilidade de criarmos poucas linhas de formulários robustos que poderão ou não conter regras de validações.

Para instalarmos o Formik é necessário irmos até o prompt de comado e escrever o seguinte comando:

(yarn add formik)

Logo após podemos começar a criar nosso formulário. Para fazermos isto, incluir um Formik componente no nosso aplicativo, que podemos usar nossos componentes de controle do nosso formulário dentro dele. (Santos, 2019)

Então o estado será atualizado sem mesmo escrever muito código.

Figura 72 - Exemplo Formik (1)



Fonte: Próprio Autor

Figura 73 - Exemplo Formik (2)



Fonte: Próprio Autor

Nenhum código acima está adicionando o componente Formik, que possui diversos acessórios.

O initialValue nos permitirá especificar valores dos campos do nosso formulário.

O validate permitirá especificar regras da nossa validação.

É uma função que pegará valores e retornar um objeto erros com a chave do valor do formulário e o valor sendo a mensagem do erro.

onSubmit é uma função que executara quando clicar em Enviar.

Tudo o que estamos fazendo é exibir uma caixa de notificação com os valores introduzidos para simplificar as coisas neste exemplo.

setSubmitting é uma função para informarmos o Formik se estamos enviando ou não.

Se concluído, passaremos false para setSubmitting para o que fizermos.

A função que passaremos para onSubmit não será chamada para todos os campos do formulário que tenha dados validos. (Santos, 2019)

Dentro do nosso Formik tem uma função com parâmetros de objetos com diversas propriedades.

E retorna um elemento de formulário com dois controles de entrada de texto.

handleChange é um parâmetro que possui um manipulador que altera o que precisamos para lidarmos com quaisquer alteração no valor do formulário.

handleSubmit possui um onBlur manipulador para elementos do formulário.

Exibiremos objetos de erros depois do controle de entrada é tocado e o objeto error é preenchido.

isSubmitting é um parâmetro deveras útil que nos permitirá saber se o formulário está ou não sendo enviado.

Quando ele é enviado, podemos desabilitar o botão enviar para que não podemos reenviar ele.

Values terá o valor do formulário que poderemos definir com o valor de nossos elementos de entrada.

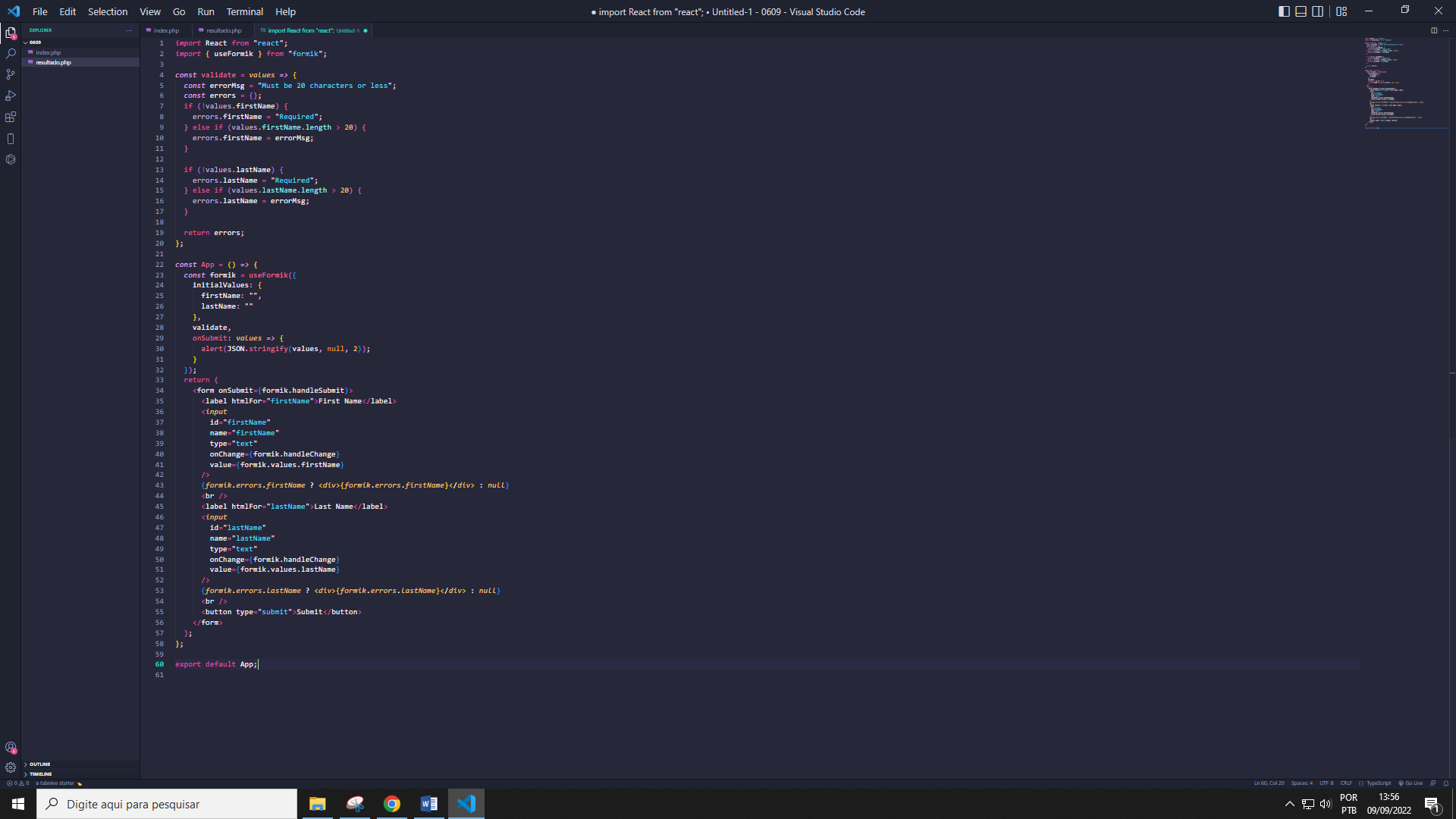
**Hooks API**

Como bibliotecas React são mais populares, o Formik também vem com sua API baseada em Hooks (série de técnicas para se modificar ou melhorar o comportamento de um sistema operacional).

Formik possui o hook useFormik, que retorna uma função que poderemos estar usando em nossos próprios formulários para manipularmos alterações de valor de formulário, exibição de mensagens de erros e enviar formulários. (Santos, 2019)

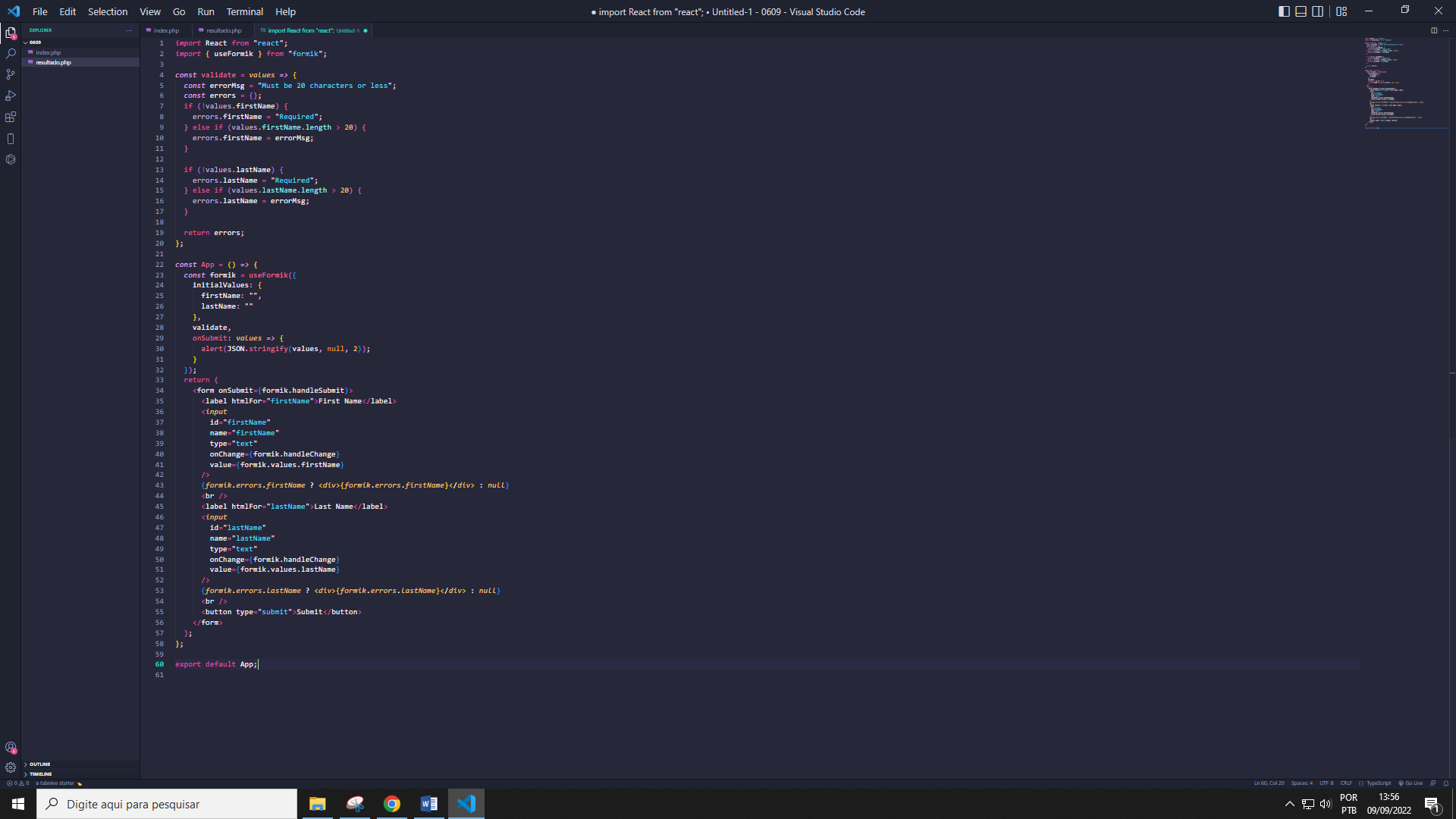
Podemos utilizar o hook useFormik da tal maneira:

Figura 74 - Exemplo Formik (3)



Fonte: Próprio Autor

Figura 75 - Exemplo Formik (4)



Fonte: Próprio Autor

Aqui separamos a função validate movendo ela para fora do App componente.

Os mecanismos de validações ainda são os mesmos. Validamos o valor dos campos presentes no formulário especificando como chaves da propriedade initialValue.

Em App, indicamos o hook useFormik passando um objeto com as propriedades initialValues, nossa função onSubmit, onde serão chamadas se todos os componentes inseridos forem validos.

Ele retornará o objeto Formik, que tem propriedades handleSubmit, handleCharge, erros e values, onde são todos os mesmos como nos exemplos anteriores. (Santos, 2019)

O uso do useFormik é beneficiado pois podemos mover a parte validate externa para fora do nosso componente.

Além disso, utilizamos os métodos de retorno e propriedades em nossos formulários. Isso significa que não precisamos escrever tudo explicitamente, nem usar os componentes internos do Formik, que são bastante restritivos.

2.5.4 Sequelize

**O que é?**

Sequelize é um tipo de ORM (Object/Relational Mapper) sendo baseada em Promise para o Node.js e io.js, suportando os dialetos PostgreSQL, MySQL, MariaDB, SQlite e MSSQL também recursos de transação, replicação de leitura, relacionamentos e entre tais coisas.

Possuindo um mecanismo de migração bem poderoso que transformará esquemas existentes de bancos de dados em novas versões, fornecendo também mecanismos de sincronização de bancos de dados que geram estruturas de bancos de dados especificando a estrutura do modelo.

Estando disponível via NPM, e sua instalação poderá ser executada no Node.js com o comando: npm install –save sequelize. (Macorati, 2014)

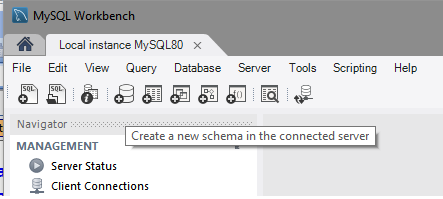
**Criando o schema MySQL:**

Antes de começar é necessário criar um novo schema no MySQL, usando o MySQL Workbench conforme a seguir:

1º

- Abra seu MySQL Workbench e clique no ícone https://www.macoratti.net/17/01/node_sequelize12.png para ser criado um novo schema;

Figura 76 - Exemplo Sequelize (1)

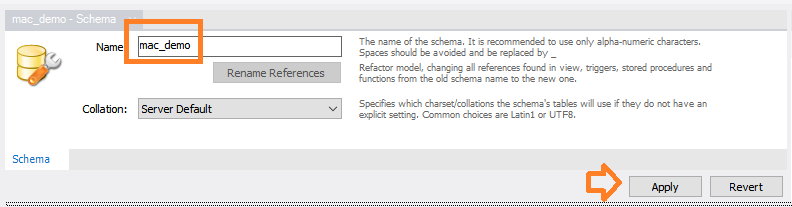


Fonte: macoratti.net

2º

- Em seguida informe o nome do seu schema, no exemplo usamos mac\_demo, e após clique no botão Apply;

Figura 77 - Exemplo Sequelize (2)

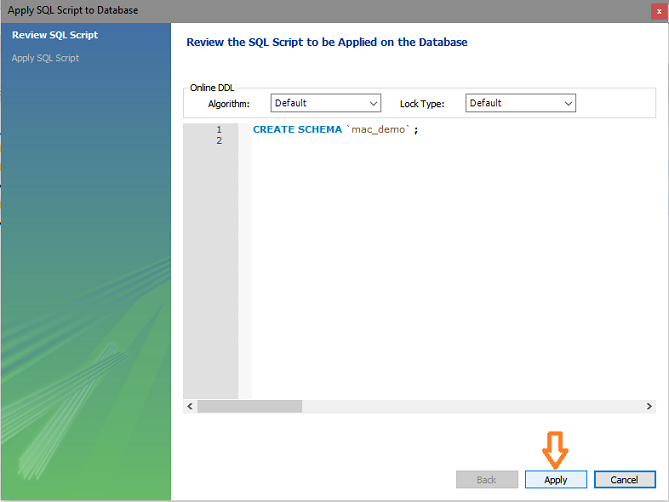


Fonte: macoratti.net

3º

- Em seguida, clique novamente em Apply e logo depois em Finish;

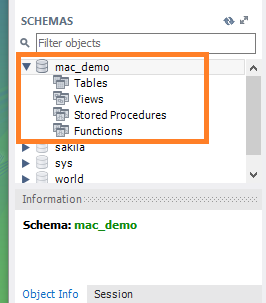
Figura 78 - Exemplo Sequelize (3)



Fonte: macoratti.net

Logo no final, você deverá ter em sua área (SCHEMAS) do MySQL Workbench o schema (mac\_demo) criado e pronto para ser utilizado:

Figura 79 - Exemplo Sequelize (4)



Fonte: macoratti.net

**Criando Aplicação Node e Instalando o Sequelize:**

Aqui você já deve ter o Node.js e o NPM instalados em sua máquina para prosseguir.

Crie uma pasta na no local onde será criada sua aplicação node, e onde vamos estar instalando o Sequelize e seus pacotes necessários.

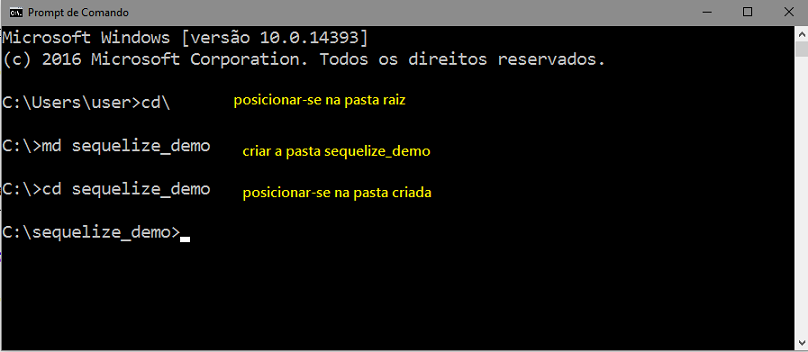
Com isto em mente, abra seu prompt de comando, e com a janela de comandos aberta digite os comandos a seguir:

cd\ - **para posicionar na pasta raiz**

md sequelize\_demo – **para criar a pasta sequelize\_demo**

cd sequelize\_demo – **para posicionar na pasta sequelize\_demo**

Figura 80 - Exemplo Sequelize (5)

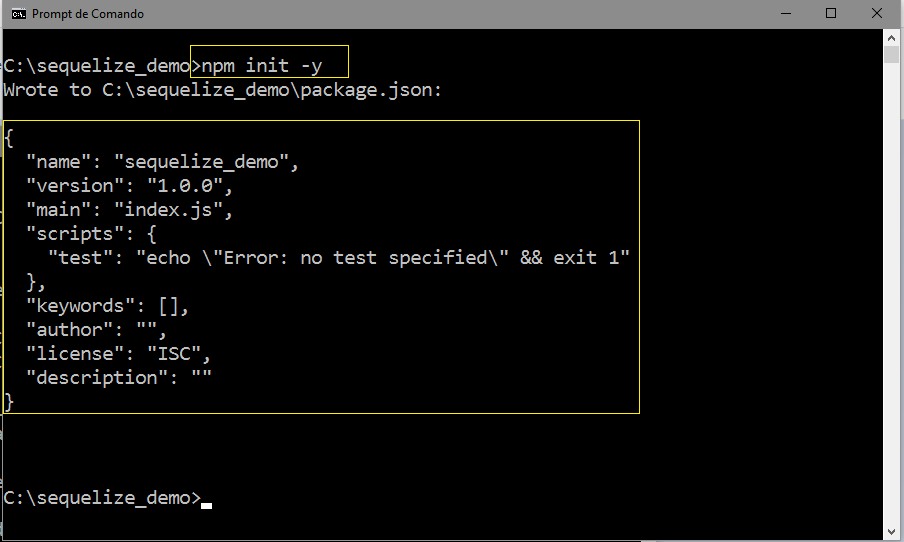


Fonte: macoratti.net

Vamos estar iniciando a aplicação criando o arquivo package.json que será utilizado para definirmos os pacotes que vamos usar na aplicação.

Digite no seu prompt o comando: npm **init-y**

Figura 81 - Exemplo Sequelize (6)



Fonte: macoratti.net

Será criado o arquivo package.json e seu conteúdo será exibido conforme a figura apresentada.

Agora para se instalar o Sequelize, digite o seguinte comando: **npm install –save mysql (**o –save irá salvar o arquivo package.json a dependência ao sequelize)

Em seguida vamos instalar o Connector para o MySQL digitando o comando: **npm install –save mysql** (o –save salva no arquivo package.json a dependência ao conector MySQL)

Abrindo o arquivo package.json podemos ver que as dependências do MySQL e do Sequelize informadas. Isso é importante pelo fato que com apenas na informação deste arquivo podemos gerar todas as suas dependências geradas usando o NPM através do **npm install**. (Macorati, 2014)

Figura 82 - Exemplo Sequelize (7)



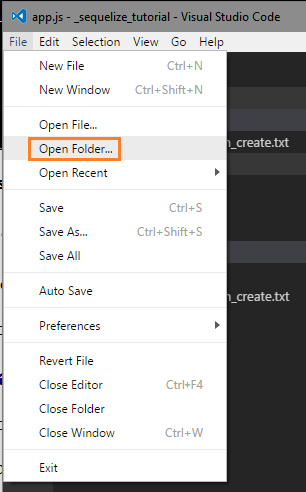
Fonte: macoratti.net

Aqui já temos o Sequelize e seu conector para o MySQL em nossa aplicação, agora iremos primeiro estabelecer uma conexão e testamos se conseguimos acessar o schema **mac\_demo** criado.

Se utilizando o VS code (Visual Studio Code é um editor de código-fonte desenvolvido pela Microsoft) e criando o arquivo app.js da aplicação

Vamos agora utilizar o VS code para criarmos o arquivo app.js na pasta sequelize\_demo na nossa aplicação. Abrindo o VS code, indo no menu File clique na opção Open Folder.

Figura 83 - Exemplo Sequelize (8)



Fonte: macoratti.net

Selecione a pasta sequelize\_demo na qual criamos para nosso projeto. Devemos visualizar apenas a pasta node\_modules que foi criada no nosso projeto, onde contém todos os arquivos de dependências necessárias para executar nossa aplicação.

Deverá também ter o arquivo package.json. Clique no ícone New File para criarmos um novo arquivo no projeto e coloque o nome app.js:

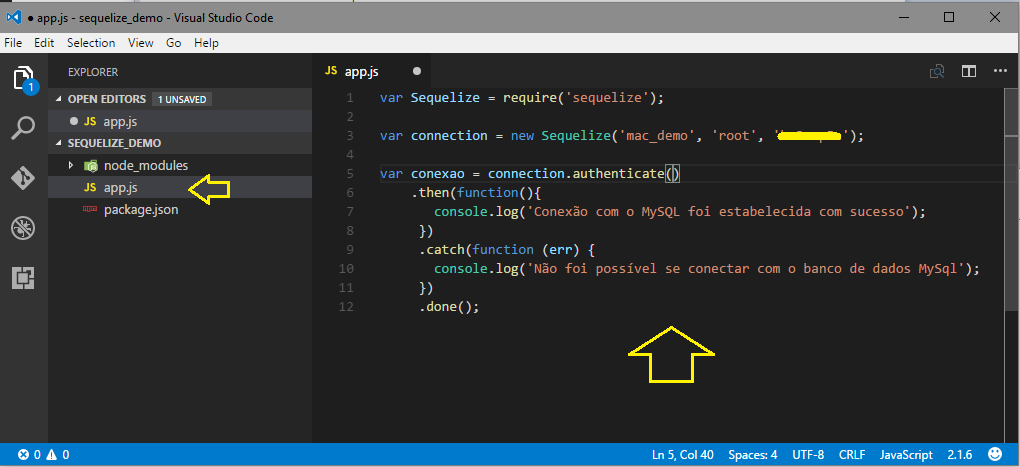


Fonte: macoratti.net

Figura 84 - Exemplo Sequelize (9)

Em seguida digite o código em seguida no arquivo app.js:

Figura 85 - Exemplo Sequelize (10)



Fonte: macoratti.net

Vamos procurar entender este código:

No código a primeira coisa que se fez foi importar um modulo sequelize usando o require (‘sequelize’). Após Sequelize representar uma variável de referência ao Sequelize. Não se precisa importar nada referente ao conector com o banco de dados, pois o Sequelize cuidará disto.

A seguir criamos uma nova instancia do sequelize utilizando o seu construtor que possui a seguinte sintaxe:

New Sequelize (database, [username=null], [options={}])

Aonde:

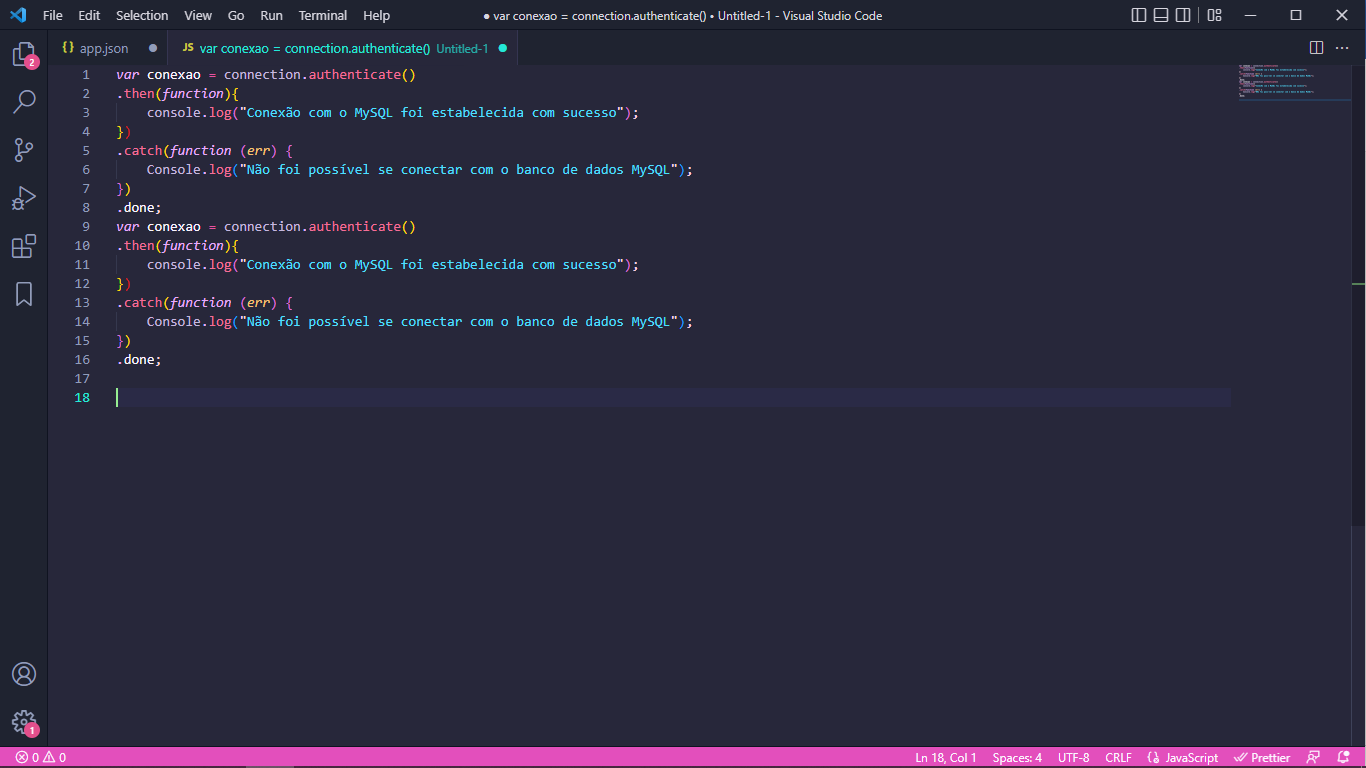
Database é o nome do schema do banco de dados que iremos acessar

1. **Username** é o nome do usuário do acesso do MySQL, no exemplo utilizado é usuário **root**;
2. **Password** é a senha do nosso usuário root. Para o exemplo é usado a senha do usuário root;

A seguir se utiliza um parâmetro authenticate que retornará uma **Promise** que verificará a conexão com o banco de dados mac\_demo do MySQL:

Nota: Promise é objeto usados para um processamento assíncrono. Um Promise (“promessa”) representa um valor onde pode estar disponível no momento (Macorati, 2014)

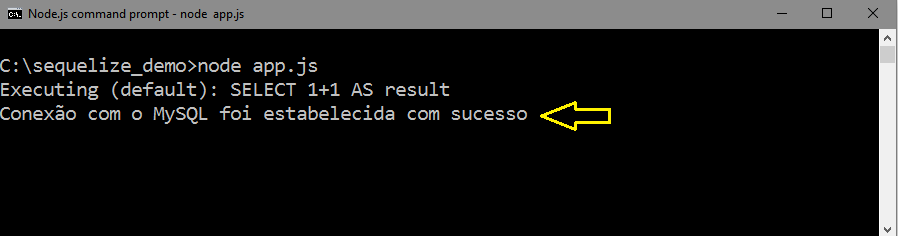
Figura 86 - Exemplo Sequelize (11)



Fonte: macoratti.net

Agora iremos rodar nossa aplicação digitando: **node app.js**

Figura 87 - Exemplo Sequelize (12)



Fonte: macoratti.net

Note-se que a mensagem indica que a conexão foi estabelecida com sucesso.

Agora iremos criar uma tabela **Artigos**, neste schema e definir dois campos: **título e assunto**. Logo em seguida incluiremos um registro na tabela e para concluirmos vamos exibir o registro incluído.

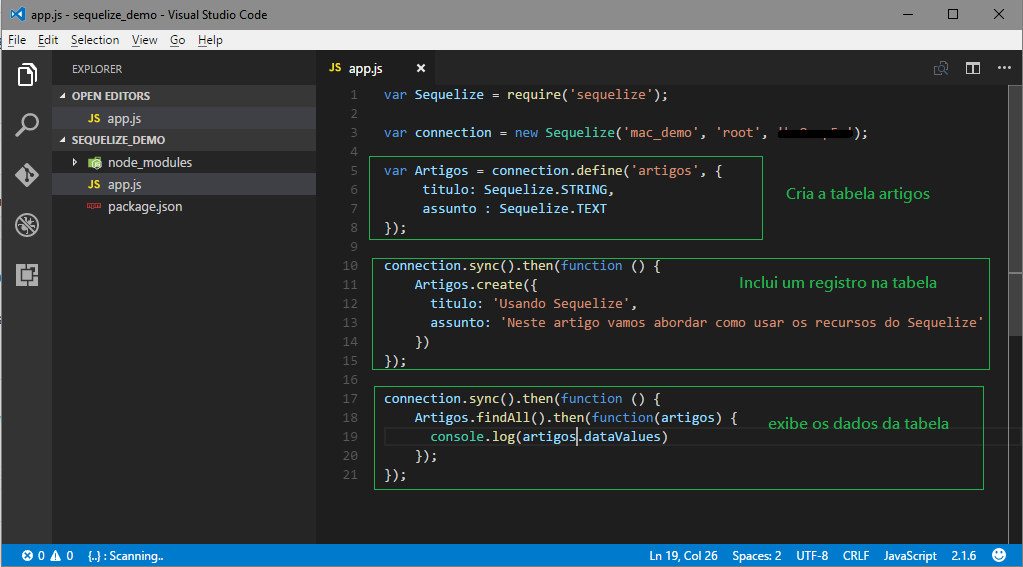


Figura 88 - Exemplo Sequelize (13)

Fonte: macoratti.net

Para isto vamos mudar o código do arquivo **app.js** conforme mostra a seguir:

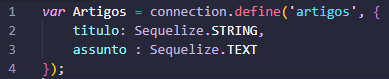
Buscando entender o código:

Usamos o método **define** para se definir mapeamentos entre modelos e tabelas. Com este método o Sequelize automaticamente adicionara os seguintes atributos:

* **createdAt**
* **updateAt**

Assim será capaz de sabermos quando foi a entrada dentro do Banco de Dados e quando pela última vez foi atualizado.

Figura 89 - Exemplo Sequelize (14)



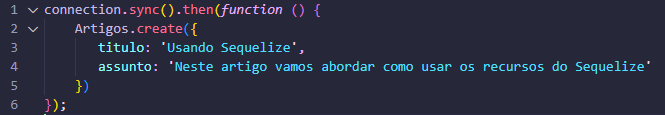
Fonte: Próprio Autor

Então o método **define**, criara a tabela **Artigos** com os campos:

* **Título – tipo de dados em STRING**
* **Assunto – tipo de dados em TEXT**
* **createdAd**
* **updatedAt**

Logo após estaremos utilizando o método **sync()** para sincronizarmos os modelos para nosso banco de dados com a tabela e usando o método **create()** para incluirmos registros nos campos título e assunto. (sync() retorna uma Promise)

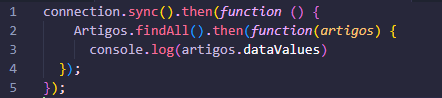
Figura 90 - Exemplo Sequelize (15)



Fonte: Próprio Autor

E finalmente usamos o método **findAll()** para listarmos todos os registros da tabela artigos.

Figura 91 - Exemplo Sequelize (16)



Fonte: Próprio Autor

Para testarmos, basta digitarmos o comando: **node app.js**

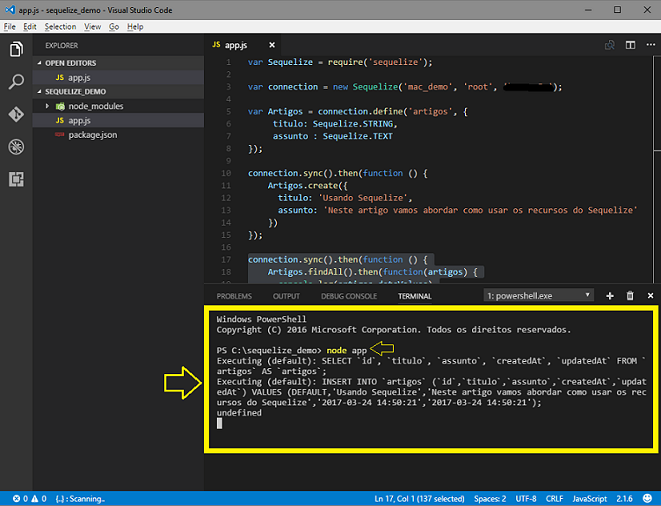
Figura 92 - Exemplo Sequelize (17)



Fonte: macoratti.net

Também podemos executar o comando no VS Code. Basta pressionar **CTRL+SHIFT+’**. Para se abrir a janela de comandos digite: **node app**

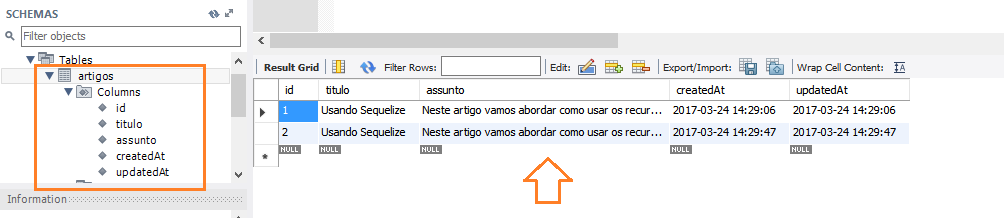
Figura 93 - Exemplo Sequelize (18)



Fonte: macoratti.net

Como visto, temos o registro já incluído e exibido no nosso console. Para constarmos iremos abrir o **MySQL Workbench** e verificarmos a tabela e seus dados:

Figura 94 - Exemplo Sequelize (19)



Fonte: macoratti.net

Se observa que no padrão os campos createdAt e updatedAt foram criados e já atualizados. O registro está duplicado porque executamos duas vezes o comando: **node app.js**.

Assim, já acessamos o banco de dados MySQL, criamos uma tabela e inserimos um registro. (Macorati, 2014)

2.5.5 Yup

**O que é?**

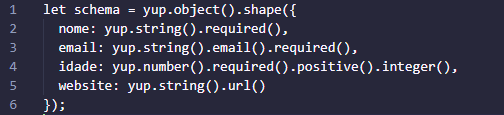
Yup é um construtor de esquemas para validações de campos e transformações de valores no JavaScript. Resumindo, com o Yup teremos o conceito de schema, que não é mais nada que o formato que os seus campos deverão estar seguindo, ou seja, se obtivermos um objeto, onde dentro teremos três campos que deverão ser em textos, com o Yup definimos o schema para isto. Os schemas são muito flexíveis e acabam permitindo modelares complexas validações, correlacionadas ou não correlacionadas, e até mesmo transformação de valores. (Martins, 2021)

Imagine que estes são os valores de um formulário:

* **Nome**: texto <obrigatório>;
* **Idade**: numérico <obrigatório>;
* **E-mail**: texto <obrigatório> <e-mail>;
* **Website**: texto <url>

Com estas informações, poderemos estar criando um schema no Yup para estar validando estes campos:

Figura 95 - Yup (1)



Fonte: blog betrybe

Temos aqui a criação do nosso schema para validarmos os campos que já foram passados. Logo, com poucas linhas e de forma mais intuitiva conseguimos estar criando um esquema para validarmos nossos dados.

A forma que o Yup está escrito é bastante inspirada em Joi (linguagem de descrição de esquema e validador de dados para JavaScript), a única diferença apresentada, é que o Yup é bem focado na parte de desenvolvimento de front-end, e na biblioteca temos a separação de dois fluxos:

**Cast**: método de ajuda na transformação e de serialização de valores;

**IsValid**: verifica se a entrada do valor está modelada de forma correta. Com isto, podemos estar separando estes dois passos ou até junta-los. (Martins, 2021)

**Porque usar Yup em seu projeto React?**

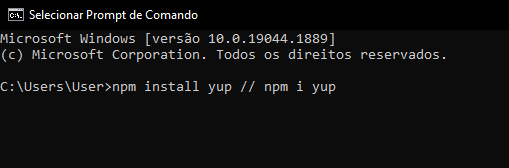
A validação de formulários é uma tarefa muito difícil e muitas vezes fazer a parte manual é exaustiva. Imagine que vamos ter que criar todo tipo de regra, exemplo: verificar se o campo está vazio, se o e-mail está correto, se contém o tamanho mínimo de caracteres e entre outros casos.

Com isto, percebemos que a validação se tornou muito repetitiva e os códigos estão cada vez mais sujos, e no fim, descobrir que seria mais simples utilizar o Yup em nossa aplicação. Com isto, ele poderia simplificar a forma de validação dos campos e parar de repetir tanto código. Também um suporte incrível à internacionalização das mensagens de erros.

**Instalação e importação do Yup**

Para instalação do Yup apenas rode um destes comandos em seu prompt de comando:

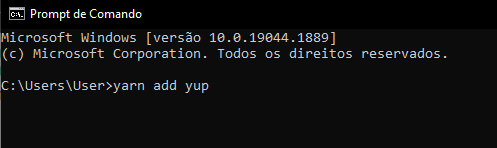
Figura 96 - Exemplo Yup (2)



Fonte: blog betrybe

Ou

Figura 97 - Exemplo Yup (3)

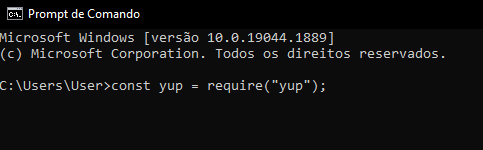


Fonte: blog betrybe

Com isto podemos seguir para sua importação

Para a importação do pacote, devemos estar utilizando o **require** quando estamos usando o NodeJs ou então, o import se estamos usando o ambiente do navegador.

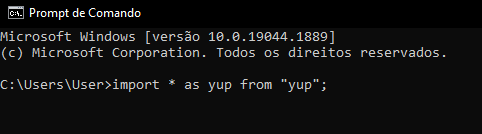
Figura 98 - Exemplo Yup (4)



Fonte: blog betrybe

Ou

Figura 99 - Exemplo Yup (4)

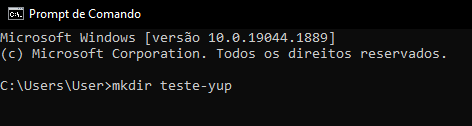


Fonte: blog betrybe

Exemplos de uso de Yup

Antes de utilizarmos o Yup, iremos criar primeiro nosso projeto. Para isso precisaremos primeiro estar baixando o NodeJs na máquina. Vamos estar criando um novo diretório (no prompt de comando), no nosso exemplo vamos chama-la de teste-yup:

Figura 100 - Exemplo Yup (5)

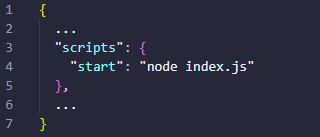


Fonte: blog betrybe

Após, entre na pasta que criamos e inicie um novo projeto. E na nossa pasta do projeto, iremos estar criando um novo arquivo em JavaScript chamado **touch index.js.**

Logo após, vamos no nosso package.json e adicionaremos um novo comando para facilitarmos a execução do arquivo.

Figura 101 - Exemplo Yup (6)



Fonte: Próprio Autor

**Criando base de dados e schema de validação**

Iremos agora estar importando o Yup neste arquivo e a base de dados que vamos validar:

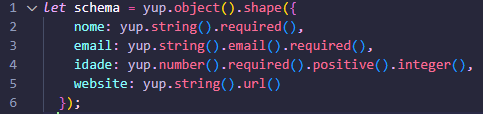
Figura 102 - Exemplo Yup (7)



Fonte: Próprio Autor

Um novo objeto foi criado no qual temos um nome, e-mail e um campo de idade. Para estramos validando estes valores, devemos criar um schema de validação:

Figura 103 - Exemplo Yup (8)



Fonte: Próprio Autor

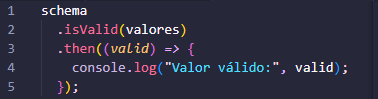
Na declaração da variável schema temos a utilização da biblioteca do nosso Yup para definirmos um objeto por meio da função **object()** e logo após escrevemos qual será o formato do objeto. Dentro disso teremos que:

* O nome é texto obrigatório;
* O e-mail tem de ser valido e também obrigatório;
* A idade tem de ser um número, obrigatório, positivo e inteiro.

**Validando dados através do schema**

Para estarmos validando nossos valores, temos que utilizar outros métodos de Yup:

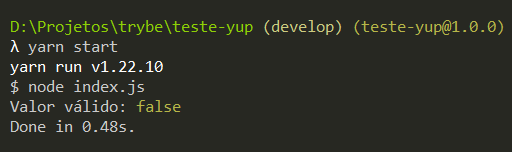
Figura 104 - Exemplo Yup (9)



Fonte: Próprio Autor

Aqui iniciaremos a validação de um valor por método IsValid(). Passaremos então o objeto criando posteriormente com o valor da idade incorreto. Por fim, depois do método then(), mostraremos o resultado da validação em uma string, para facilitar visualizações. Rodando o comando (npm run start) teremos o resultado da nossa validação.

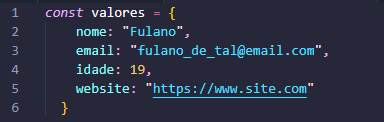
Figura 105 - Exemplo Yup (10)



Fonte: blog betrybe

Podemos então arrumar os valores que passamos:

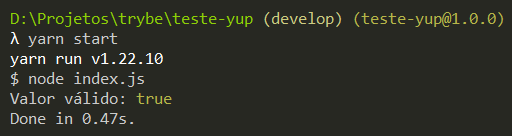
Figura 106 - Exemplo Yup (11)



Fonte: Próprio Autor

Se rodarmos novamente, perceberemos um novo resultado:

Figura 107 - Exemplo Yup (12)



Fonte: blog betrybe

Desta maneira, conseguimos estar entendendo a força que o Yup está nos proporcionando. (Martins, 2021)

# Desenvolvimento

No terceiro capitulo iremos abordar a fase de desenvolvimento do TCC, onde apresentaremos, de forma geral, a explicação e discussão do tema e os seus resultados finais.

## Levantamento de Requisitos

O levantamento de requisitos será a parte mais importante do processo, onde resultará no desenvolvimento final do TCC. Entender o que o cliente desejará, ou aquilo que o cliente se acredita que necessite e as regras do negócio ou os processos dele.

3.1.1 Entrevista/coleta de dados

Na nossa coleta de dados preferimos buscar através de Documentos (Imagens, textos e material de análise), optamos fazermos nossa coleta desta maneira pois seria um método mais prático e rápido, invés de fazermos entrevistas com grupos de pessoas ou entre outras coisas.

* + 1. Cronograma de Atividades

O nosso cronograma foi feito totalmente no site Trello (Um aplicativo de gerenciamento de projeto baseado na web originalmente desenvolvido em 2011 pela Fog Creek Software). Nesse aplicativo é possível utilizar-se desde cartões e quadros para auxiliar a equipe ou um único usuário em seu planejamento de projeto.

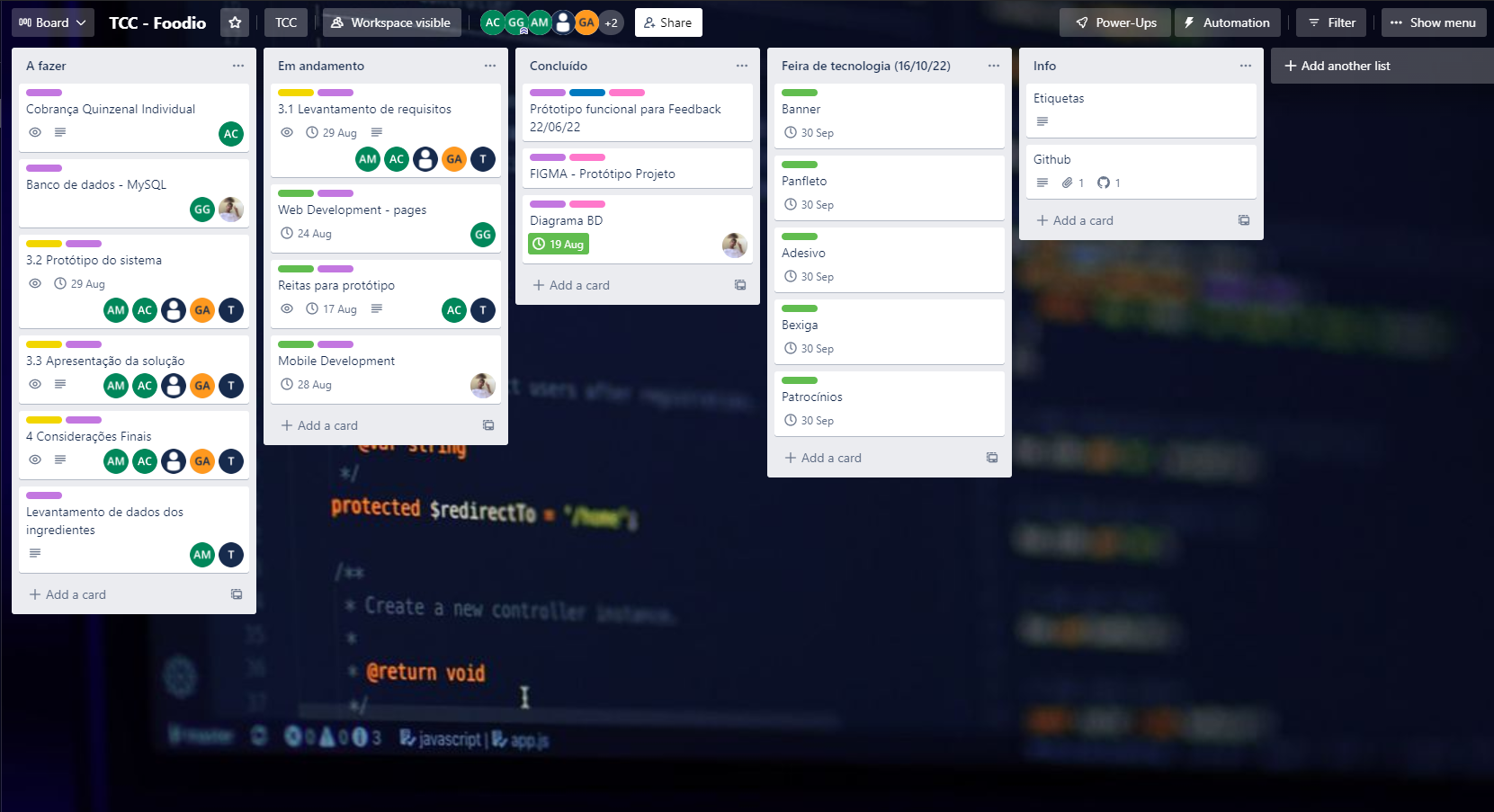
Com isto nossa equipe criou 5 quadros para o nosso auxilio, sendo estes os quadros:

* A fazer: deixando marcando os itens pendentes em nosso TCC
* Em andamento: marcado para atividades que a equipe está realizando no momento
* Concluído: Deixar claro as atividades que já foram concluídas pela equipe

Agora dois outros quadros que criamos para o auxílio e entendimento da nossa equipe:

* Feira de tecnologia (16/10/22): Uma feira de Ciências da Etec Pedro Ferreira Alves onde a nossa principal ideia é apresentarmos o site e o app do Easy Chef.
* Info: Com informações para ajudar a montar os cartões e o nosso Github

Figura 108 – Calendário Trello



Fonte: Próprio Autor

Com o Trello podemos definir aos membros da nossa equipe o que é designado para que cada um deverá fazer dentro de um prazo de tempo.

Ex. Em andamento, os membros designados no cartão (3.1 Levantamento de requisitos) devem entregar dentro da data de 29/08/2022

Para entendimento de todos, é possível colocar etiquetas em cima dos cartões para entendimento de todos sobre o nível da atividade, sendo essas etiquetas:

* Verde: Baixa prioridade (2 semanas do prazo exigido)
* Amarela: Documentação (Itens que deverão entrar na parte de documentação do TCC)
* Laranja: Alta prioridade (1 semana do prazo exigido)
* Vermelha: Urgente (48 horas do prazo exigido)
* Roxa: TCC (cartões que pertencem a área do TCC)
* Rosa: Concluído (atividades marcadas como concluídas)

## Protótipo de sistema

Para a criação do protótipo do nosso site e aplicativo, buscamos encontrar sites que pudessem ajudar na parte do funcionamento e na parte visual, mas o mais importante também, que ajudasse a termos uma concepção melhor dos nossos pensamentos e ideias.

Então buscamos utilizar um site chamado (Figma) ele é um editor gráfico de vetores e prototipagens de projetos de design, sendo principalmente baseado em navegadores da web e tudo nele pode ser feito de forma online no navegador.

Com esta ferramenta em mãos conseguimos criar protótipos do nosso aplicativo e site de formas mais produtiva e ágil, pois com ele também conseguimos fazer nossa correção de erros durante o processo de testes do usuário.

3.2.1 aplicativo

Para termos uma ideia mais generalizada sobre como seria nosso aplicativo, começando definindo a sua base de cores, sendo elas branco, preto e um tom de verde mais claro para ter como ideia de como iria se continuar o aplicativo.

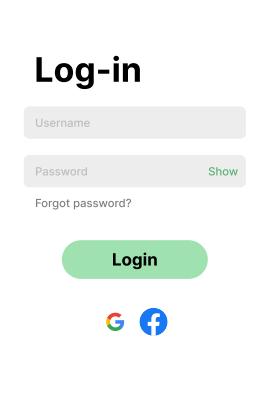
Figura 109 – Login/Register App



Fonte: Próprio Autor

Com a paleta de cores já criada procuramos começando pela nossa tela de Login sendo encontrado nela o título do aplicativo, e os botões de Login e register, para aqueles que acabaram de chegar terem a chance de se cadastrar.

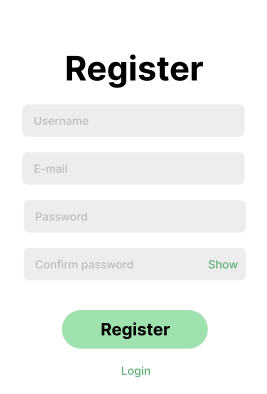
Figura 110 – Login App



Fonte: Próprio Autor

Nossa tela de Login, podendo se cadastrar tanto pelo Facebook e Google e também tendo a opção de recuperar a sua senha.

Figura 111 – Register App



Fonte: Próprio Autor

A nossa tela de (Register) podendo o usuário criar uma conta no nosso aplicativo com seu próprio endereço de e-mail.

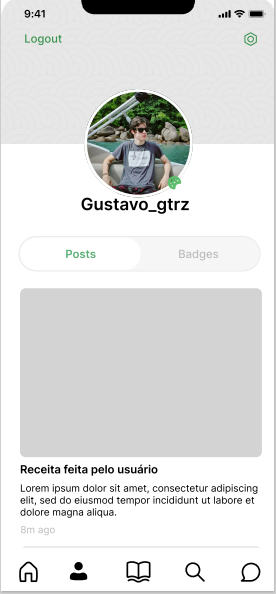
Figura 112 – Feed App



Fonte: Próprio Autor

A primeira tela que será apresentada ao usuário irá ser o (Feed) sendo apresentado nele receitas feitas por outros usuários, sendo capaz de você curti-las, compartilhar, comentar e denunciar receitas, caso elas não seguirem os padrões do aplicativo, gerando um retorno do público e engajamento na comunidade.

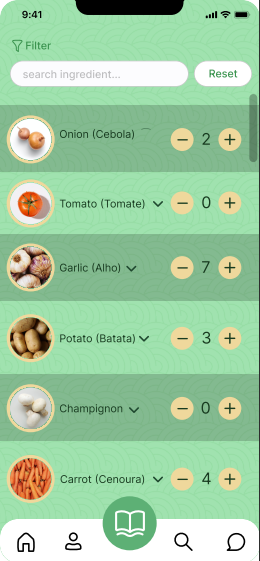
Figura 113 – Usuário App



Fonte: Próprio Autor

Contém uma tela para o usuário sendo possível você publicar receitas para próprio engajamento.

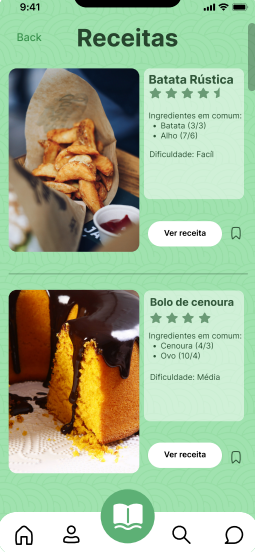
Figura 114 – Filtragem Ingredientes App



Fonte: Próprio Autor

Ao clicar no ícone de livro, presente na área inferior do aplicativo, vamos ao foco principal do aplicativo, onde é você inserir os ingredientes presentes dentro da sua residência. Também presente os botões de Reset, para zerar todos os itens selecionados pelo usuário e o botão de pesquisa de alimentos, para facilitar a procura do usuário.

Figura 115 – Resultado Receitas App



Fonte: Próprio Autor

Após selecionar os ingredientes desejados, o usuário será direcionado para a tela de receitas já prontas, com aqueles ingredientes já selecionados. Sendo também apresentado a dificuldade do prato.

3.2.1 Site

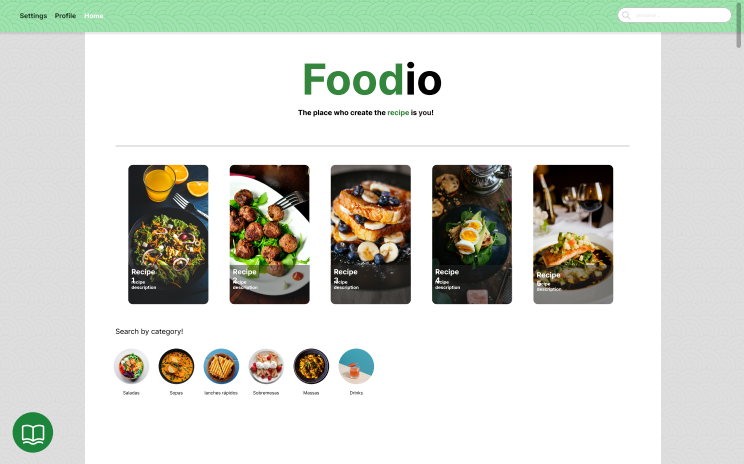
Para termos uma ideia mais generalizada sobre como seria nosso site, pegamos

Como base as paletas de cores do nosso aplicativo, para seguirmos o padrão de cores e esquematização do site, para não fugir do nosso padrão de cores e familiarização.

Figura 116 – Login/Register Site

Fonte: Próprio Autor

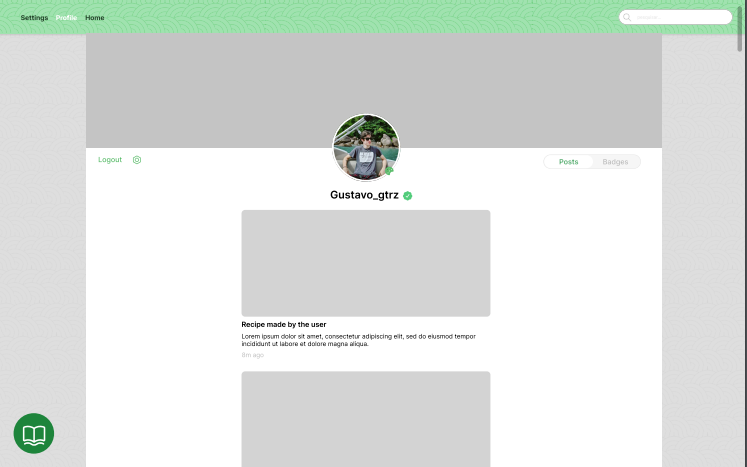
Nossa página inicial, onde poderá ser feita o Login do usuário, seu cadastro caso não tenha uma conta no nosso site, a opção de Login com o Google e a opção de recuperar a sua senha caso tenha perdido ou esquecido.



Fonte: Próprio Autor

Figura 117 – Home Site

Logo após o usuário realizar o cadastro ele será redirecionado para a página inicial do site, onde será apresentado as receitas mais feitas durante a semana, em baixo categorias de receitas que poderão ser feitas pelo usuário, desde saladas á drinks. E na barra superior a barra de pesquisa, os botões para as páginas de configurações, perfil do usuário e a pagina Home.

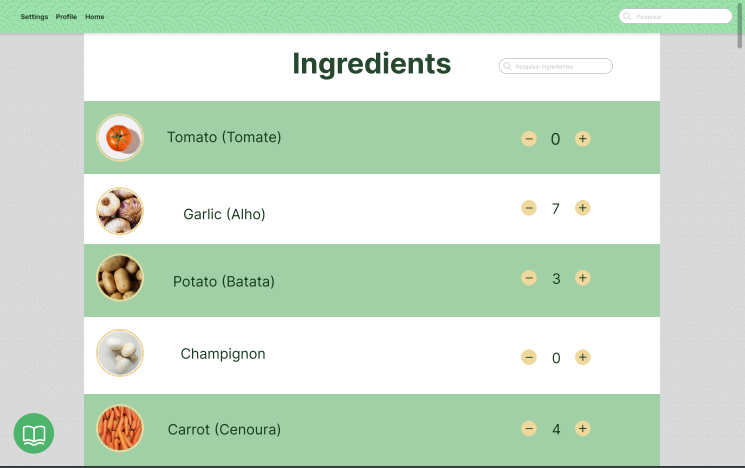


Fonte: Próprio Autor

Figura 118 – Usuário Site

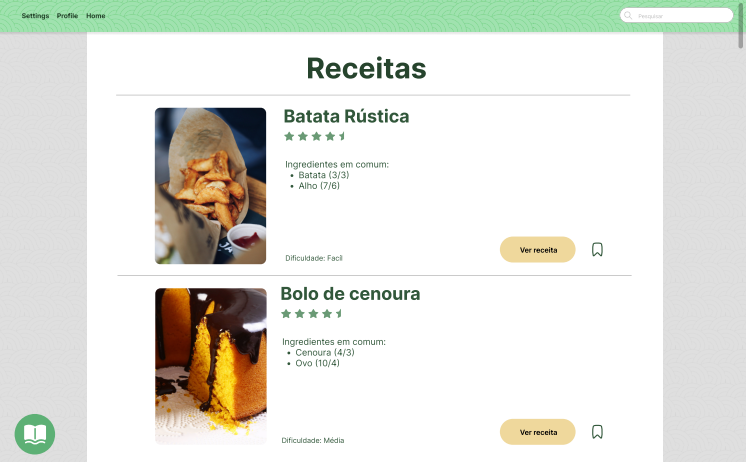
Na página de perfil do usuário irá funcionar quase como a do aplicativo, sendo capaz você visualizar as suas receitas que foram feitas e receitas que você postou no dentro do Easy Chef. E a barra superior também continuará a mesmo presente na página Home. E presente no canto inferior esquerdo um ícone de livro, sendo ele para o usuário informar seus ingredientes para no final gerar a receita para o usuário.

Figura 119 – Filtragem de Ingredientes Site



Fonte: Próprio Autor

Essa seria nossa página onde o usuário informaria os ingredientes presentes dentro da sua residência, para no final fornecer as receitas para o usuário.



Fonte: Próprio Autor

Figura 120 – Resultado Receitas Site

Esta seria a página de retorno para o usuário, sendo apresentado a ele as receitas com os ingredientes que ele selecionou, sendo também apresentado a dificuldade do prato.

3.3 Apresentação da Solução

Nosso ideal desde do inicio, era diminuir o disperdicio de alimento presente em muitas residencias pelo mundo todo. O foco principal era visar em uma solução para redução deste problema, e percebemos que a maioria dos alimentos jogados fora eram sobras de ingredientes dentro da geladeira, como poucas quantidades de alimentos e dentre outras coisas. Então para reduzir este problema, observamos que poderiamos utilizar uma forma criativa para incentivar os outros a utilizarem estes ultimos alimentos presentes em sua residencia, sempre visando em reduzir disperdicios de alimentos ainda pereciveis.

# Considerações finais

Visando nosso objetivo inicial onde era de se esperado que com o uso do nosso aplicativo desenvolvido, causa-se uma solução ideal para o desperdício de alimentos, sendo eles novos ou não utilizados, oferecendo um software onde permita que a pessoa indique seus alimentos que estão prestes a estragarem e a partir desses ingredientes recomendar receitas culinárias típicas/praticas, para o cliente de forma acessível permitindo que ele reutilize esses alimentos ao invés de desperdiça-los e ainda incentiva-lo a praticar a culinária.

Pode-se dizer que com o Easy Chefs, conseguimos visar nosso principal objetivo, a redução de alimentos presentes nas residências das pessoas, também tentando deixar nosso projeto mais chamativo para o público, transformando-o em uma rede social, para usuários interagirem entre si, para que assim nosso projeto alcance cada vez mais usuários, e assim estarem ajudando sempre a evitar o desperdício de alimento.

# Referências

**Amazon. 2022.** O que é NoSQL? *Amazon Web Services.* [Online] 05 de 2022. https://aws.amazon.com/pt/nosql/.

**—. 2022.** O que é o Redis? *Amazon Web Services.* [Online] 05 de 2022. https://aws.amazon.com/pt/elasticache/what-is-redis/.

**Andrade, Ana Paula de. 2022.** Os principais SGBDs relacionais. *Treina Web.* [Online] Abril de 2022. https://www.treinaweb.com.br/blog/os-principais-sgbds-relacionais.

**Campomori, Cleber. 2022.** Os pilares da orientação a objetos. [Online] 05 de 2022. https://www.treinaweb.com.br/blog/os-pilares-da-orientacao-a-objetos.

**2022.** DB-Engines Ranking. *DB-ENGINE.* [Online] Maio de 2022. https://db-engines.com/en/ranking.

**DevMedia. 2022.** Introdução ao banco de dados NoSQL Cassandra. *DevMedia.* [Online] Maio de 2022. https://www.devmedia.com.br/introducao-ao-banco-de-dados-nosql-cassandra/30533.

**—. 2022.** MER e DER: Modelagem de Bancos de Dados. *DevMedia.* [Online] 04 de 2022. https://www.devmedia.com.br/mer-e-der-modelagem-de-bancos-de-dados/14332.

**Erwin Quest Software. 2022.** Modelagem lógica de dados. *Erwin.* [Online] 04 de 2022. https://www.erwin.com/br-pt/solutions/data-modeling/logical.aspx.

**Flexus. 2021.** Microsoft SQL Server é uma boa opção para a minha empresa? *Flexus.* [Online] Maio de 2021. https://www.flexussolucoes.com.br/microsoft-sql-server/.

**Instituto Federal Santa Catarina. 2015.** PI S8 DSW I DouglasARS. *Enciclopédia Instituto Federal Santa Catarina.* [Online] Novembro de 2015. https://wiki.ifsc.edu.br/mediawiki/index.php/PI\_S8\_DSW\_I\_DouglasARS.

**Jesus, Agnes Diniz Gomes de. 2021.** O DESPERDÍCIO DE ALIMENTOS NO BRASIL. *VERAKIS.* [Online] 25 de Maio de 2021. https://verakis.com/o-desperdicio-de-alimentos-no-brasil/.

**Laundon, Kenneth e Laundon, Jane. 2014.** *Sistemas de Informações Gerenciais.* s.l. : Pearson Universidades, 2014.

**Lopes, Bergson. 2022.** Modelo Conceitual de Dados - Aprenda a utilizar os principais mecanismos de abstração. *BLRDATA.* [Online] 04 de 2022. https://www.blrdata.com.br/single-post/2016/03/19/modelo-conceitual-de-dados-aprenda-a-utilizar-os-principais-mecanismos-de-abstra%C3%A7%C3%A3o.

**Lopes, Gabrielle. 2021.** *A importância do mock-up nos projetos de TI.* [Online] 2021. https://blog.workana.com/pt/sin-categorizar/a-importancia-do-mock-up-nos-projetos-de-ti/.

**Macorati. 2014.** Macoratti.net. [Online] 2014. https://www.macoratti.net/17/01/node\_sequelize1.html#:~:text=O%20Sequelize%20%C3%A9%20um%20ORM,de%20leitura%20e%20muito%20mais.

**Martins, Vinicius. 2021.** blog betrybe. [Online] 2021. https://blog.betrybe.com/desenvolvimento-web/yup/.

**maxwell. 2008.** *Elicitação de Requisitos .* [Online] 2008. https://www.maxwell.vrac.puc-rio.br/15760/15760\_3.PDF.

**Paúl, Fernanda. 2021.** Os efeitos do desperdício chocante de alimentos no mundo. *BBC News Mundo.* [Online] 20 de Março de 2021. https://www.bbc.com/portuguese/internacional-56377418#:~:text=O%20chamado%20%22%C3%8Dndice%20de%20Desperd%C3%ADcio,mundo%20foram%20para%20o%20lixo.

**Rob, Peter e Coronel, Carlos. 2010.** *Sistemas de banco de dados: Projeto, implementação e administração.* s.l. : Cengage Learning, 2010.

**Santos, Luan. 2019.** Medium . [Online] 2019. https://medium.com/reactbrasil/formik-como-criar-um-formul%C3%A1rio-com-campos-din%C3%A2micos-a150d60a8610.

**Silva, Eduardo. 2020.** Quais são os paradigmas de programação mais importantes? [Online] 2020. https://blog.geekhunter.com.br/quais-sao-os-paradigmas-de-programacao/.

**Souza, Isaac Felisberto de. 2021.** *Paradigmas de Programação .* [Online] 2021. https://guia.dev/pt/pillars/languages-and-tools/programming-paradigms.html.

**Souza, Ivan de. 2019.** *Bootstrap: saiba neste guia para iniciantes o que é, por que e como usá-lo.* [Online] 2019. https://rockcontent.com/br/blog/bootstrap/.

**Souza, Ivan. 2020.** MariaDB ou MySQL: saiba qual tecnologia de banco de dados escolher. *Rockcontent.* [Online] Abril de 2020. https://rockcontent.com/br/blog/mariadb/.

**Testa, Emanuelle. 2014.** *Curso Sistemas de Informação Disciplina: Arquitetura de Software.* [Online] 2014. https://slideplayer.com.br/slide/365071/.

**X-apps. 2019.** *https://www.x-apps.com.br/node-js.* [Online] 2019.