[TOC]

MANUAL DE DESENVOLVIMENTO EMPRESA INFOCEPT

Este manual se refere a um conjunto de boas práticas na escrita de software que você vai aplicar para obter uma maior legibilidade e manutenibilidade do seu código. Onde qualquer programador da loja, que precise realizar uma manutenção, conseguirá de forma simples e eficiente.

REQUISITOS PROGRAMADORES WEB

FRONT-END

BÁSICO

- HTML
- CSS
- JAVASCRIPT / TYPESCRIPT
- GIT
- NOÇÕES BÁSICAS DE BACK-END (PHP)
- CONHECIMENTO DO CONSOLE DO NAVEGADOR
- DOCUMENTAÇÃO

CONHECIMENTO EM:

- REACT
- BOOTSTRAP / TAILWIND CSS (SENSAÇÃO DO MOMENTO)
- UI (User Interface)
- UX (User Experience)
- CONHECIMENTO NO DOCKER
- CONSUMO DE API
- JSON
- SASS

BACK-END

BÁSICO

- NOÇÕES BÁSICAS DO FRONT-END (HTML, CSS E JAVASCRIPT)
- GIT
- BACK-END (PHP)
- ESTRUTURA MVC
- BANCO DE DADOS
- DOCUMENTAÇÃO
- CONSUMO DE API

CONHECIMENTO EM:

- FOCO NA SEGURANÇA
- REACT
- BANCO DE DADOS RELACIONAS E NÃO RELACIONAIS
- LARAVEL (ESTÁ SE TORNANDO UM REQUISITO BÁSICO)

FULLSTACK

BÁSICO

- DOMÍNIO DO FRONT-END
- DOMÍNIO DO BACK-END

CONHECIMENTO EM:

- DEVOPS
- NOÇÃO DE GERENCIAMENTO DE PROJETOS EM TODAS ETAPAS

[TOC]

FERRAMENTAS UTILIZADAS

remos realizar uma list de programas e ferramentas que geralmente são utilizadas em todos projetos não sendo obrigatório o uso em todos projetos:

- Laragon - (servidor local)
- VSCode - (editor de códigos)
- FileZila Client - (acesso FTP)
- Mind Meister - (criação do mapa mental, auxilia na visão sistêmica do projeto)
- Trello - (Utilizado para controle de tarefas)
- Discord - (Conversas, criação de grupos de estudo e projeto além de reuniões)
- Whatsapp - (Conversas curtas e envio de audio)
- Chrome Developer - (Navegador que permite abrir nossas aplicações web)
- Git - (atualmente estamos realizando tomada de decisões sobre essa ferramenta, sua funções e os versionamentos do projeto)
- Docker - (software que permite testas ambientes (Desenvolvimento,Teste,Homologação,Produção...) com o objetivo de tirar aquela frase "Na minha máquina funciona!")
- Marvel - (Criação de wireframe)
- Bootstrap Studio - (Criação de wireframe e front-end)

IREI CRIAR UM MENU PARA CADA FERRAMENTAS, SEMELHANTE AO QUE FIZ COM VSCODE, E POSTERIORMENTE VIDEOS AULAS SOBRE CADA UMA.

VSCODE

O VScode é um editor de códigos da Microsoft sua grande vantagem é ser leve, sua versão gratuita permite usar sem muitas limitações além disso conseguimos instalar plugins com o objetivo de melhorar ou acrescentar novas funcionalidades, abaixo vamos listar alguns desses plugins:

DOWNLOAD E INSTALAÇÃO

video

1. ACESSAR O SITE OFICIAL VS CODE
2. Acessar a aba de download e escolher o tipo de sistema operacional que será instalado
3. Aguarde até a conclusão do download
4. Instale o programa apenas avançando as janelas

5. Abra o programa realizando as configurações iniciais
6. Não é um passo obrigatório, mas realizando o login na conta do vscode, você consegue salvar suas configurações e sincronizar em qualquer outro computador.
7. Instalações de PLugins

EXTENSÕES / PLUGINS

Os plugins tem a função de auxiliar com novas funcionalidades, muitos deles ficam obsoleto com o tempo, sempre necessário adicionar e atualizar, conforme o tempo e conhecimento adquirido.

- Markdown All in One - (documentações de projeto)
- Dictionary Completion - (auxiliar em palavras, por padrão vem com o dicionário em inglês.)
- Paste Image - Para colar imagens captura de telas direto no vscode
- PlantUML - (auxilia em criação de fluxograma) instalação
- Auto Rename Tag - Ao mudar uma tag de abertura, automaticamente a tag de fechamento é alterada.
- Blockman - Highlight Nested Code Blocks - auxilia na visualização de blocos (qual tag está dentro de outra)
- Bookmarks - Com esse plugin podemos marcar determinados trechos com comentário para possíveis mudanças ou explicações
- Bootstrap 5 Quick Snippets - Extensão com snippets(atalhos ou autocomplete) do bootstrap 5
- Spelling Checker for Visual Studio Code - auxilia com correção e sugestão de palavras, podemos adicionar vários idiomas inclusive o brasileiro, adicionando a extensão do idioma desejado.
- Color Picker - auxilia na visualização e criação de cores CSS
- Composer - permite utilizar o composer dentro do vscode
- Dictionary Completion - auxilia em sugestões de palavras por padrão do idioma inglês.
- Docker - instalei mais ainda não sei usar kkk mas sei que o docker atualmente é algo primordial, irei estudar accidental
- Format HTML in PHP - formatação de html dentro do PHP
- GitLens — Git supercharged - permite visualizar ações do git diretamente no vscode
- Dracula Oficial - Tema de cores (não obrigatório)
- Laravel Extension Pack - plugin com kit plugins para laravel
- Material Icon Theme - auxilia na visualização dos arquivos com ícones

ATALHOS

Abaixo temos alguns atalhos importantes sobre o programa VSCODE

- **CTRL H**: permite abrir o recurso localizar e substituir
- **CTRL D** : ao selecionar uma palavra e ir apertando CTRL D, ele vai selecionar todas outras palavras com a mesma escrita, permitindo alterar em uma única vaporize
- **CTRL ;**: aplica ao trecho selecionado um comentário

CHROME DEVELOP

O CHROME DEVELOP é um navegador que tem funções próprias para desenvolvedores web, isso vai nós auxiliar durante os projetos, além disso conseguimos instalar plugins com

o objetivo de melhorar ou acrescentar novas funcionalidades, abaixo vamos listar alguns desses plugins:

DOWNLOAD E INSTALAÇÃO

video

1. ACESSAR O SITE OFICIAL CHROME DEVELOP
2. Acessar a aba de download e clicar no botão correto.
3. Aguarde até a conclusão do download
4. Instale o programa apenas avançando as janelas
5. Abra o programa realizando as configurações iniciais
6. Não é um passo obrigatório, mas realizando o login na conta do google, você consegue salvar suas configurações e sincronizar em qualquer outro computador.
7. Instalações de PLugins

EXTENSÕES / PLUGINS

Os plugins tem a função de auxiliar com novas funcionalidades, muitos deles ficam obsoleto com o tempo, sempre necessário adicionar e atualizar, conforme o tempo e conhecimento adquirido.

- Fake Filler - (preenchimento automático de formulário)
- Nimbus Screenshot & Screen Video Recorder - (captura de tela e vídeos)
- JSON Formatter - (formata arquivos json, que facilita navegação e visualização)

TECNOLOGIAS / RECURSOS(LIBs) UTILIZADAS

video Linguagens de programação e frameworks atualmente utilizada por nossos desenvolvedores:

- HTML - Linguagem de marcação de texto (estrutura de um desenvolvimento web ou "esqueleto")
- CSS - Linguagem de marcação de estilo (permite mudar ou melhorar a aparência das tags HTML)
- JS / JQUERY - Recurso da parte dinâmica do lado cliente (front-end), atualmente na loja usamos JS e Jquery.
- PHP - em nossos projetos o PHP é usado para o lado servidor (back-end), a intenção e que todos desenvolva habilidades nessa linguagem e posteriormente no framework laravel
- Laravel - Framework PHP que permite facilitar, torna mais seguro e escalável a parte back-end.
- Bootstrap - atualmente na versão 5, usamos esse framework com objetivo de facilitar as partes responsivas de nossas aplicações.
- validation - utilizando para validar campos do formulário, juntamente com jquery
- fontawesome - pacotes de ícones utilizados nos projetos
- CKEDITOR - editor com recursos de formatação em campos textarea

INSTALAÇÃO

BOOTSTRAP

1. Download do arquivo

2. Extração do arquivo
3. Copie o arquivo booststrap.min.css (para a pasta css do projeto) e boostrap.bundle.min.js (para a pasta js do projeto) por convenção essas pastas estão dentro de assets
4. No template chamamos duas tags `<link>` e `<script>` Dentro da tag head fazemos o comando abaixo:

```
<link href="<?= BASE_URL; ?>assets/css/bootstrap.min.css" rel="stylesheet">
```

Na parte inferior do template usamos o comando abaixo (se trabalharmos com JQUERY, coloque abaixo dele):

```
<script src="<?= BASE_URL; ?>assets/js/bootstrap.bundle.min.js"></script>
```

FONT AWESOME

5. Acesse o link de download
6. Clique em free for web
7. Extração do arquivo
8. Mova a o arquivo all.css para dentro de css do projeto Vamos colocar entre a tag `<head>` usamos o comando abaixo:

```
    <link href="<?= BASE_URL; ?>assets/css/all.css" rel="stylesheet">
```

9. Para consultar os icones usamos o link Icons

JQUERY

1. Acesse o link de download
2. Aperte Ctrl + S para Salvar a página do jquery
3. Vamos salvar dentro de assets > js
4. No template chamamos usamos a tag `<script>`
5. Vamos colocar entre a tag `<head>` , para garantir que o jquery carregue antes de outros javascript que necessite dele, usamos o comando abaixo:

```
    <script src="<?= BASE_URL; ?>assets/js/jquery-3.6.1.min.js"></script>
```

VALIDATION

1. Download do arquivo
2. Extração do arquivo
3. Mova o arquivo jquery.validate.min.js para dentro de nosso projeto, por convenção colocamos dentro de **assets > js**
4. No template chamamos a tag `<script>` Na parte inferior do template fazemos o comando abaixo:

```
<script src="<?= BASE_URL; ?>assets/js/jquery.validate.min.js"></script>
```

Lembrete: Precisamos chamar abaixo do jquery.

CKEDITOR

1. Acessar o site oficial
2. Personalizar o ckeditor (se necessário)
3. Clicar em download
4. Extrair o arquivo
5. Mova o arquivo ckeditor.js para dentro de nosso projeto, por convenção colocamos dentro de **assets > js**
6. No `<textarea class="class_name"></textarea>` que precisamos do ckeditor, inserimos uma classe ou id para posteriormente usar no JavaScript.
7. Criamos um arquivo externo com o comando abaixo javascript:

```
<script src="<?= BASE_URL; ?>assets/js/ckeditor.js">
ClassicEditor
  .create(document.querySelector('.class_name'), {
        licenseKey: '',
    })
    .then(editor => {
        window.editor = editor;
    })
    .catch(error => {
        console.error('Oops, something went wrong!');
        console.error('Please, report the following error on
https://github.com/ckeditor/ckeditor5/issues with the build id and the error stack
trace:');
        console.warn('Build id: pu35lwblkpul-n4nfthb3g72i');
        console.error(error);
    });
```

LIBs e TEMPLATES INTERNOS

- MVC
- LARAVEL
- EXCEL TO XML

ETAPAS DOS PROJETOS

Vamos padronizar as formas com que os projetos aconteceram com prazos e entregas semanais, além disso os projetos serão trabalhados de forma coletiva, não será mais único como estava acontecendo.

- Análise econômica
- Levantamento e análise de requisitos
- Design do projeto
- Implementação
- Teste
- Documentação
- Suporte, manutenção e atualização

Análise econômica

A primeira etapa do desenvolvimento de software é a análise econômica, ou análise financeira do projeto. **(Na maioria dos casos essa etapa pode vim após a coleta de requisitos)**

Aqui, profissionais da administração e desenvolvimento fazem avaliações sobre a viabilidade do software a ser desenvolvido, sua compatibilidade com os canais existentes de distribuição, os requisitos exigidos pela demanda, qual o custo e o tempo necessário para o desenvolvimento, entre outras coisas.

Ferramentas como excel ou sistema de cálculos podem ser necessário.

Levantamento e análise de requisitos

Nessa etapa, a equipe de desenvolvimento elabora uma lista com todos os requisitos que o software em questão deve apresentar.

Por isso, a análise de requisitos é uma etapa extremamente fundamental. Afinal, a grande maioria dos clientes não detêm o conhecimento técnico em torno de como os softwares são desenvolvidos. Sendo assim, eles podem possuir uma visão simplista ou abstrata do projeto.

Dessa forma, os profissionais de TI mais experientes buscam identificar, ainda nesta fase, requisitos que podem se mostrar incompletos, ambíguos ou até mesmo contraditórios.

Ferramentas como mapa mental geralmente são utilizadas nessa parte

Design do projeto

Estabelecidos os requisitos para o software, é iniciada a fase de design do projeto. Nessa etapa de desenvolvimento, buscamos definir um template ou se será necessário criar todo projeto do zero.

Ferramentas como marvel (wireframe), envato elements (templates) ou boostrap studio, podem ser utilizados

Implementação

Nessa etapa de implementação é aquela em que os desenvolvedores começam a escrever, de fato, o código de programação do software, utilizando como base para isso os projetos elaborados na fase anterior.

Iremos criar um git ou uma forma de deixar "subprogamas" que são funções já feitas por outro desenvolvedor quando necessário para reutilizar agilizado algumas partes do projeto.

1. Criar o trello do projeto (gestou ou programador responsável) com as tarefas separadas de forma aplicável e fácil entendimento.
2. Criar o repositório no git seguindo as etapas pré estabelecidas pelos gestores de projeto.
3. Documentação modelo de todas tarefas, a cada semana um membro será responsável por criar a documentação do que foi feito na semana atual e atualizar o trello.
4. A cada 15 dias teremos uma reunião pontual para discutirmos melhorias e ajustes necessários.

Teste

O teste de software, como o próprio nome sugere, é a etapa do processo de desenvolvimento em que a equipe busca identificar possíveis erros, bugs e/ou defeitos.

Os testes são feitos tanto interna quanto externamente, contando para tanto com o auxílio de usuários. Em alguns casos, esses usuários testam o programa voluntariamente; em outros, a empresa pode contratar analistas de testes de software, também conhecidos como testadores de software.

Documentação

A documentação é fundamental para auxiliar com a manutenção do software, bem como ajudar programadores que não estiveram envolvidos com o processo de desenvolvimento a entenderem melhor o projeto em si.

Como mensurado na etapa da implementação, iremos realizar a documentação semanal.

Suporte, manutenção e atualização

Por fim, após ter codificado, testado e aprovado o software, entra em cena a etapa de suporte. Isso inclui processos que vão da instalação até a personalização do programa, para deixá-lo com os parâmetros ideais para o cliente.

Além disso, essa fase costuma incluir o treinamento para o software, visando instruir o usuário final a utilizar propriamente o produto.

Já a etapa de manutenção e atualização envolve a correção de bugs que foram descobertos após o lançamento e o desenvolvimento de novos requisitos.

Essa etapa será incluso nas Análise econômica de acordo com a dificuldade de cada projeto.

REGRAS GERAIS

SIGA AS CONVENÇÕES

Ao iniciar um projeto, siga-as regras! Se nas definições utilizamos por exemplo constantes em maiúsculo, variaveis separadas com underline (_) não importa! Siga sempre os padrões do projeto.

KISS - KEEP IT STUPID SIMPLE (MANTENHA ISTO ESTUPIDAMENTE SIMPLES - KISS)!

Mantenha as coisas simples! Normalmente tendemos a complicar as coisas que poderiam ser muito mais simples. Recursos na programação tem que ser algo percetivo e com facilidade, métodos e variaveis por exemplo coloquem o nome representando o que ele realmente é.

REGRA DE ESCOTEIRO

"Deixe sempre o acampamento mais limpo do que você encontrou!" Nossos projetos usa um método de trabalho em equipe, isso quer dizer que sempre podemos indicar melhorias nos projetos e novos métodos, as reuniões mensais tem esse objetivo.

CAUSA RAIZ

Sempre procure a causa raiz do problema, nunca resolva as coisas superficialmente. No dia-a-dia, na correria, tendemos a corrigir os problemas superficialmente e não adentrar neles, o que muitas vezes causa o re-trabalho! Tente sempre procurar a causa raiz e resolver assim o problema de uma vez por todas!

EVITE CONFIGURAÇÕES DESNECESSÁRIAS

Evite deixar configurações no sistema só por que alguém ainda não definiu como aquilo deve ser. Isto polui o código e traz uma complexidade desnecessária. Não crie ou deixe algo que não vá usar.

```
<?php
    public void ConfiguraUsoOracle()
    {
        // ainda não sabemos se vamos ou não suportar Oracle também
        throw new NotImplementedException();
    }
?>
```

SEJA CONSISTENTE

Se você executa algo de uma forma, execute todo o resto desta mesma forma. Seja consistente na forma com que aplica o código. Siga sempre o padrão definido.

```
// Codificando em inglês
public function CustomerRepository { ... }

// Agora mudou para "portuglês"
public function ProdutoRepository { ... }

// Agora é português
public function RepositorioUnidadeMedida { ... }
```

UTILIZE VARIÁVEIS CONCISAS

Escolher para classes, variáveis e métodos concisas, mesmo que resultem em um nome maior. Elas devem ser auto-explicativas, sem a necessidade de comentários ou informações adicionais. Lembre-se que se você precisa explicar seu código, então algo pode ser melhorado nele.

```
// Total do que?
decimal total = 0;

// Total do carrinho de compras
decimal shoppingCartTotal = 0;
```

FAÇA DISTINÇÕES SIGNIFICANTES

Utilize sempre nomes nos quais quem estiver lendo seu código possa diferenciar seu significado de outros possíveis nomes.

```
<script>
Exemplo
// Evite
var salario = 7500M;

// Tem um significado maior
var salarioEmReais = 7500M;
</script>
```

EVITE USO EXCESSIVO DE STRING

Quem nunca perdeu horas procurando um BUG que era apenas um problema de comparação de string? Evite digitar a mesma string várias vezes, utilize constantes para isto.

```
<?php
    Exemplo
    // Evite
    if(environment == "PROD")
        ...

    // Utilize
    const string ENV = "PROD";

    if(environment == ENV)
        ...
?>
```

REGRAS PARA FUNÇÕES OU MÉTODOS

Pequenas e com apenas um objetivo Mantenha suas funções ou métodos o menor possível. É mais fácil ter métodos menores e reutilizáveis do que tudo dentro de um método só.

Exemplo

```
// Evite
public void RealizarPedido()
{
    // Cadastra o cliente
    // Aplica o desconto
    // Atualiza o estoque
    // Salva o pedido
}

// Utilize
public void SaveCustomer() { ... }
public void ApplyDiscount() { ... }
```

```
public void UpdateInventory() { ... }
public void PlaceOrder() { ... }
```

NÃO TOME DECISÕES DESNECESSÁRIAS

Não utilize os famosos "flags" para tomar decisões dentro dos métodos, divida-os em vários métodos ou até mesmo outras classes.

Exemplo

```
// Evite
public class CustomerRepository
{
    public void CreateOrUpdate(Customer customer, bool create)
    {
        if(create)
            ...
        else
            ...
    }
}
```

```
// Utilize
public class CustomerRepository
{
    public void Create(Customer customer) { ... }
    public void Update(Customer customer) { ... }
}
```

REGRAS DE COMENTÁRIOS

UM CÓDIGO BOM É EXPRESSIVO

Teoricamente, se você precisa comentar uma parte do seu código, é por que algo está errado com ele, ele não está expressivo o suficiente.

NÃO SEJA REDUDANTE

Evite comentários que não fazem sentido algum ao contexto ou cenário.

Exemplo

```
// Função principal do sistema
public void Main() { ... }
```

EVITE CÓDIGOS COMENTADOS (APENAS PARA TESTE)

Não deixe sujeira em seu código, ao invés de deixar algo comentado, remova ele. Hoje temos versionadores de código (GIT), você pode "voltar no tempo" facilmente.

Exemplo

```
// Evite
public void MinhaFuncao()
{
    // string texto = "1234";
    // public void Metodo() {... }
}
```

INTENÇÃO

Um bom uso de comentários é sobre a intenção de um método, classe ou variável (Variável nem tanto).

Exemplo

```
// Utilize

// Retorna a lista de produtos inativos
// para o relatório de fechamento mensal
public IList<Product> ObtemProdutosInativos()
{
    ...
}
```

ESCLARECIMENTO

Outro uso interessante para os comentários são esclarecimentos sobre o código.

Exemplo

```
// Utilize
public void CancelarPedido()
{
    // Caso o pedido já tenha sido enviado
    // ele não pode mais ser cancelado.
    if(DataEnvio > DateTime.Now)
    {
        AddNotification("O pedido já foi enviado e não pode ser cancelado");
    }
}
```

Consequências Podemos utilizar comentários para alertar sobre trechos do código que podem ter consequências mais sérias. Neste caso recomendo o uso de um comentário em mais elaborado.

Exemplo

```
// Utilize

// ATENÇÃO: Este método cancela o pedido e estorna o pagamento
public void CancelarPedido()
{
    ...
}
```

ESTRUTURA DO CÓDIGO

SEPARE CONCEITOS VERTICALMENTE

Mantenha uma estrutura de pastas saudável e organizada. Não precisa criar uma pasta para cada arquivo, mas pode haver uma separação por contexto ou tipo.

- meu_projeto
  - assets
    - upload
      - video
      - image
      - audio

DECLARE VARIÁVEIS PRÓXIMAS DE SEU USO

Não crie todas as variáveis juntas, no começo da class ou método, defina-as próximas de onde serão utilizadas.

Exemplo

```
// Evite
var total = 0;

public void CreateCustomer() { ... }

public void CreateOrder() { ... }

public void UpdateCustomer() { ... }

public void CalculateTotal()
{
    total = 250; // <- Só é utilizada aqui
}
```

```
// Utilize
public void CreateCustomer() { ... }

public void CreateOrder() { ... }

public void UpdateCustomer() { ... }

var total = 0;
public void CalculateTotal()
{
    total = 250;
}
```

AGRUPE FUNCIONALIDADES SIMILARES

Se uma função pertence a um grupo dentro de um objeto, mantenha-as sempre por perto.

Exemplo

```
// Evite
public void CreateCustomer() { ... }

public void CheckInventory() { ... }

public void CreateOrder() { ... }

public void UpdateCustomer() { ... }

public void CalculateTotal() { .. }
```

```
// Utilize
public void CreateCustomer() { ... }
public void UpdateCustomer() { ... }

public void CheckInventory() { ... }
public void CreateOrder() { ... }
public void CalculateTotal() { .. }
```

NÃO QUEBRE A IDENTAÇÃO

Este item dispensa comentários. Um código não identado não pode ser enviado para o projeto.

Exemplo

```
// Evite
public class MinhaClasse{
var valor=12;
Console.WriteLine(valor);
}
```

BANCO DE DADOS

- Inglês
- Nome intuitivo
- Separar com underline (_)
- Nome tabela no plural
- TABELA ACCOUNT
    - ID
    - LOGIN (EMAIL OU TELEFONE)
    - PASSWORD (PASSOWRD_HASH - 255 CARACTERES)
    - PERMISSION (QUANDO NECESSÁRIO))
- CHAVE ESTRANGEIRA
    - nome_tabela_id
    - ligação nome: info_nome_tabela_origem_nome_tabela_atual EX: info_student_certificate

VARIÁVEIS

- Inglês
- Nome intuitivo
- Separar com underline (_)
- Vamos basear geralmente semelhante aos campos do Banco de dados

FUNCTION

- Inglês
- Nome intuitivo
- CamelCase Ex: getAll, getDocumentStudent
- return true / false ou array

MVC

CONTROLLERS

- homeController
  - Responsável pelas páginas estaticas ou que pouco se alteram
- notfoundController
  - Será chamada sempre que um link não existir ou estiver errado
- ajaxController
  - Usado para recursos via ajax (quando queremos realizar uma operação no servidor sem atualizar a página)

MODEL

- SITES
  - Responsável pelos recursos de processamento de dados que não estão ligados diretamente em uma única tabela, ou seja, ele pode ser usados por várias tabelas ou controllers. Ex: gerador de slug, upload de arquivos...
- ACCOUNTS
  - Responsável pelas informações da conta do usuário (login, CRUD)

CONSIDERAÇÕES FINAIS

Uma boa leitura da documentação da empresa e de extrema importância, pois baseado nela você vai desenvolver os projetos, permitindo que futuras manutenções ou correções sejam feitas de formas padronizadas.