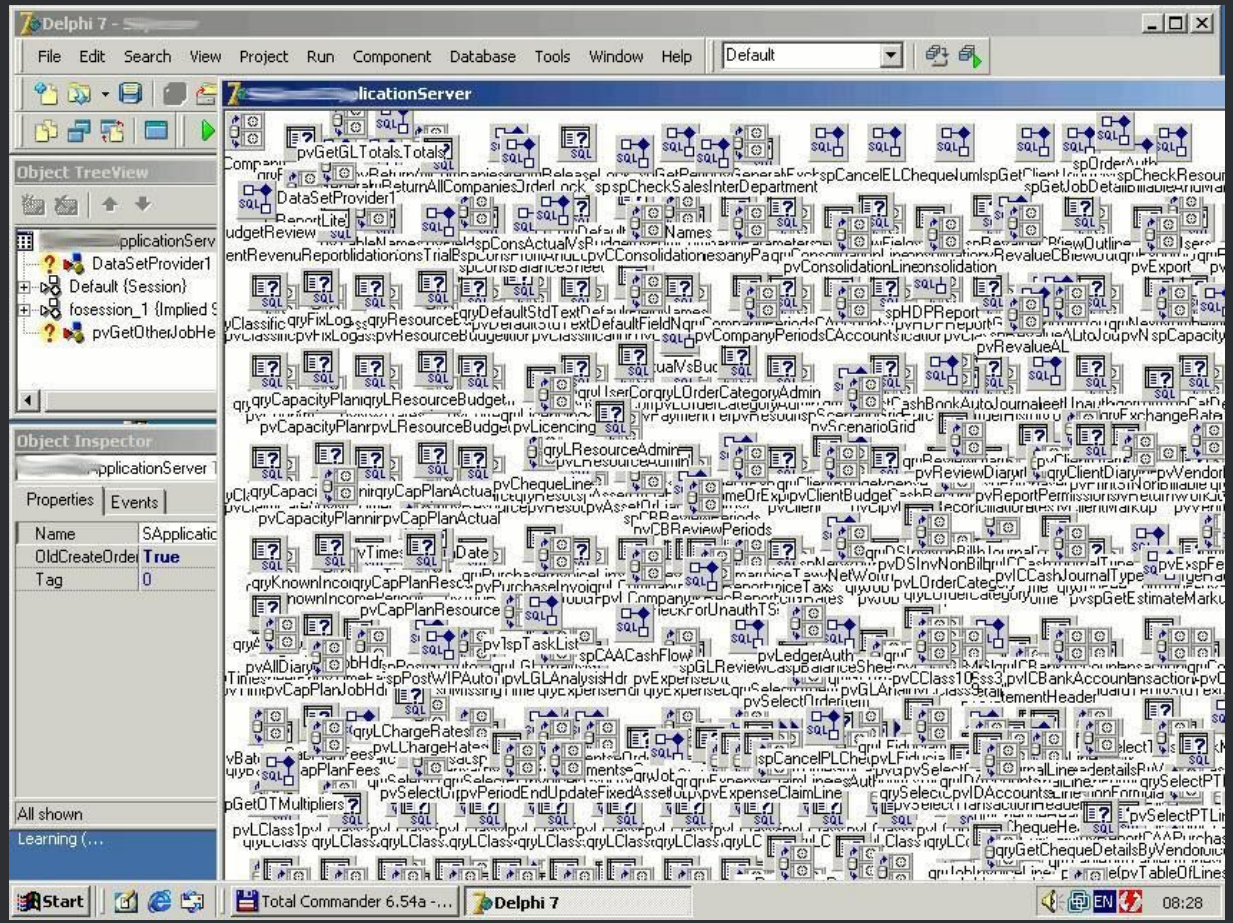
CLEAN CODE

“Qualquer tolo pode escrever um código para um computador entender, bons programadores escrevem código que outros humanos entendem”

__ Martin Fowler



Um código bom é chato, ele é previsível

Por que códigos ruins acontecem.

1 - Entregar resultados rápidos demais
2 - Deadlines apertadas
3 - A tarefa é grande demais
4 - O escopo mudou muito ao longo do projeto
5 - As pessoas não entendem o que você faz

Um código limpo é aquele que parece que a pessoa se importou em fazer.

PEQUENOS DETALHES

```
function TForm3.NomeProduto(aId: Integer): String;
begin
  if aId <= 0 then
    Result := 'Produto Inválido'
  else
    Result := 'Produto ' + IntToStr(aId);
end;
```

```
function TForm3.NomeProduto(aId: Integer): String;
begin
  Result := 'Produto Inválido';

  if aId > 0 then
    Result := 'Produto ' + IntToStr(aId);
end;
```

```
function TForm3.Dividir(aValue1, aValue2: Double): Double;
begin
  if aValue2 = 0 then
    Result := 0
  else
    Result := aValue1 / aValue2;
end;
```

```
function TForm3.Dividir(aValue1, aValue2: Double): Double;
begin
  Result := 0;

  if aValue2 > 0 then
    Result := aValue1 / aValue2;
end;
```

Precisamos mensurar o custo do que fazemos

As empresas perdem muito dinheiro em horas incontáveis de **códigos mal escritos**

Um erro no seu projeto é um erro na sua trajetória profissional

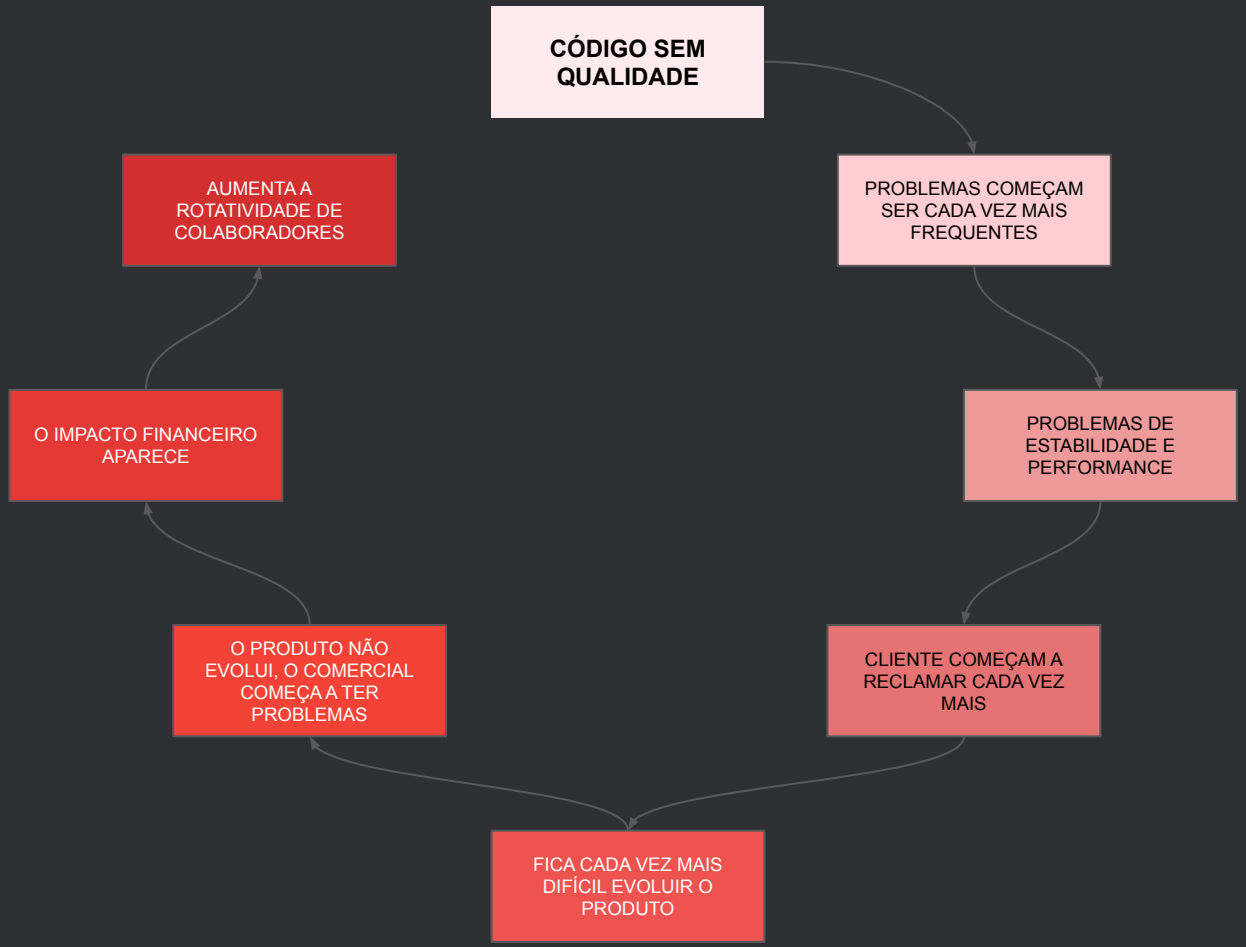
Não é o cliente que responde pelos seus erros

Escrever um código legível é tão importante quanto escrever um código que funciona

CICLO DA IMPRODUTIVIDADE

Problemas de Clean Code vão além dos problemas de tecnologia, são problemas de gestão.

Sem padrão não existe estimativa



Falta de padrão leva a desmotivação

OS DETALHES DO CLEAN CODE

NOMENCLATURAS

- Tudo tem nome então dar nomes é importante
- Não use abreviaturas
- Coloque nomes buscáveis

MÉTODOS E FUNÇÕES

- Devem ser pequenos
- Evitar Métodos Aerodinâmicos
- Cuidado com as abstrações
- Evitar passagem de muitos parâmetros
- Evitar retornos boolean
- Opte por Gets e Sets

COMENTÁRIOS

- Evite comentários para explicar o código
- Se precisa explicar é porque está mal feito
- Comentários ficam desatualizados rapidamente
- Comentários devem:
  - Informar caminhos que foram tentados em outras refatorações
  - Explicar regras de negócio por trás do código
  - Gerar documentação

FORMATAÇÃO

- Código deve estar estruturado
- Respeite a Indentação que você definir
- Se possível respeite a Style Guide do Delphi
  https://edn.embarcadero.com/article/10280

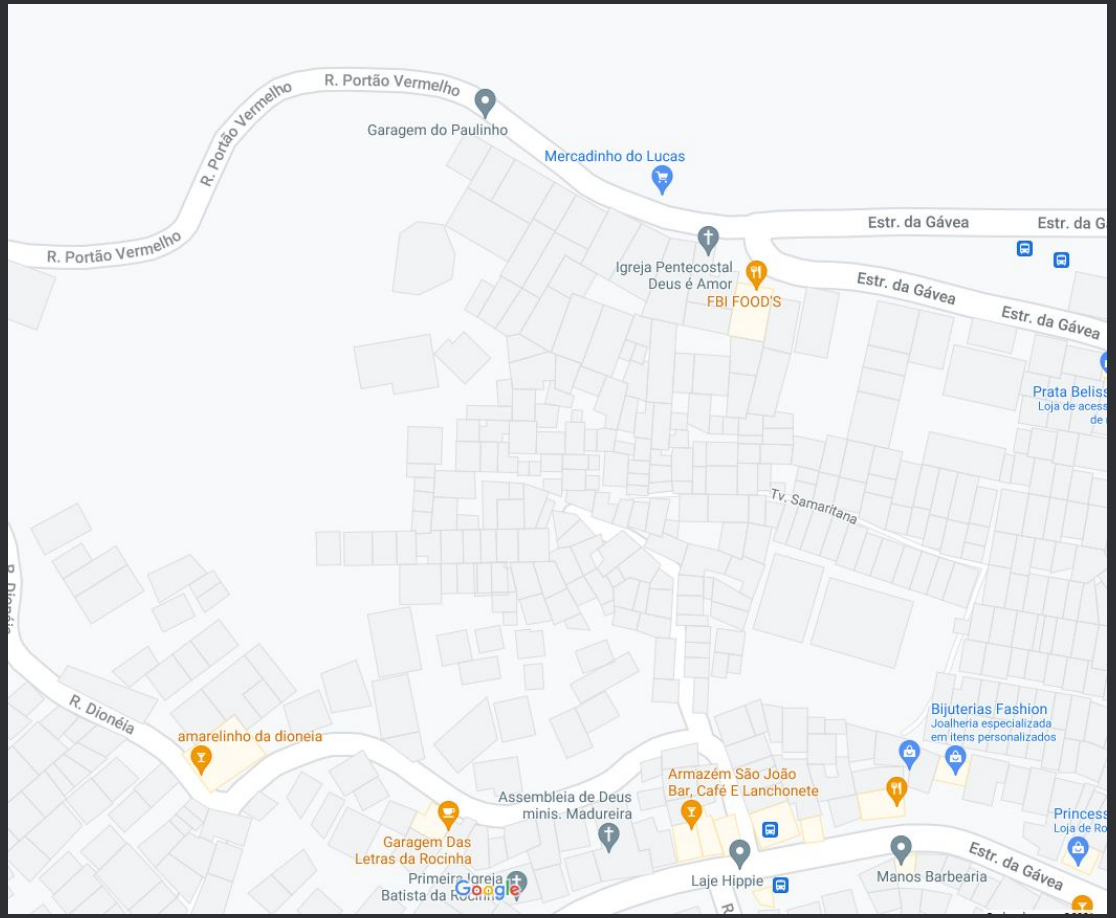
OBJETOS E ESTRUTURAS DE DADOS

- Encapsule seus atributos (Cuidado com Property)
- Evitar criar uma sacola de gets e sets

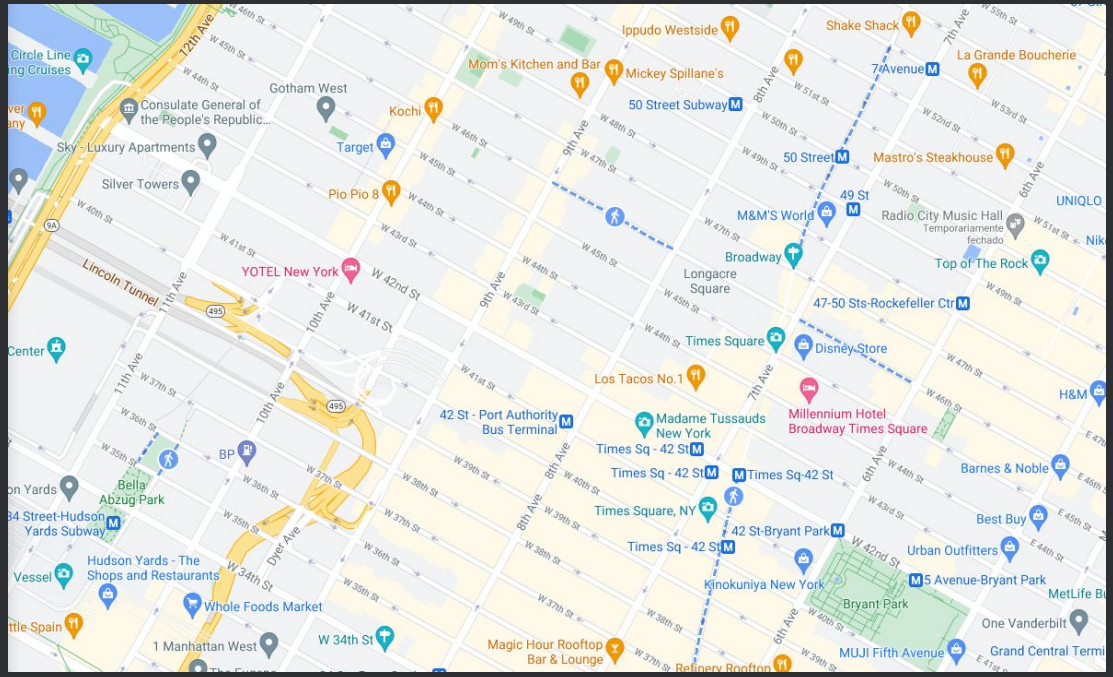
É SÓ UM IF'ZINHO

Não confunda prototipar com projetar

Planejar antes de executar



Planejar antes de executar



A teoria das Janelas Quebradas

FOCO NA MANUTENÇÃO

Total Productive Maintenance

O foco do controle de qualidade está na manutenção e não na produção

Quanto mais lógica dentro de um método maior a quantidade de bugs não isolados

```
function TACBrNFe.EnviarEvento(idLote: integer): Boolean;
var
  i, j: integer;
  chNfe: String;
begin
  if EventoNFe.Evento.Count <= 0 then
    GerarException(ACBrStr('ERRO: Nenhum Evento adicionado ao Lote'));

  if EventoNFe.Evento.Count > 20 then
    GerarException(ACBrStr('ERRO: Conjunto de Eventos transmitidos (máximo de 20) ' +
      'excedido. Quantidade atual: ' + IntToStr(EventoNFe.Evento.Count)));

  WebServices.EnvEvento.idLote := idLote;

  {Atribuir nSeqEvento, CNPJ, Chave e/ou Protocolo quando não especificar}
  for i := 0 to EventoNFe.Evento.Count - 1 do
  begin
    if EventoNFe.Evento.Items[i].InfEvento.nSeqEvento = 0 then
      EventoNFe.Evento.Items[i].infEvento.nSeqEvento := 1;

    FEventoNFe.Evento.Items[i].InfEvento.tpAmb := Configuracoes.WebServices.Ambiente;

    if NotasFiscais.Count > 0 then
    begin
      chNfe := OnlyNumber(EventoNFe.Evento.Items[i].InfEvento.chNfe);

      // Se tem a chave da NFe no Evento, procure por ela nas notas carregadas //
      if NaoEstaVazio(chNfe) then
      begin
        For j := 0 to NotasFiscais.Count - 1 do
        begin
          if chNfe = NotasFiscais.Items[j].NumID then
            Break;
        end ;

        if j = NotasFiscais.Count then
          GerarException( ACBrStr('Não existe NFe com a chave ['+chNfe+'] carregada') );
      end
      else
        j := 0;

      if trim(EventoNFe.Evento.Items[i].InfEvento.CNPJ) = '' then
        EventoNFe.Evento.Items[i].InfEvento.CNPJ := NotasFiscais.Items[j].NFe.Emit.CNPJCPF;

      if chNfe = '' then
        EventoNFe.Evento.Items[i].InfEvento.chNfe := NotasFiscais.Items[j].NumID;

      if trim(EventoNFe.Evento.Items[i].infEvento.detEvento.nProt) = '' then
      begin
        if EventoNFe.Evento.Items[i].infEvento.tpEvento = teCancelamento then
        begin
          EventoNFe.Evento.Items[i].infEvento.detEvento.nProt := NotasFiscais.Items[j].NFe.procNFe.nProt;
        end;
      end;
    end;
  end;

  Result := WebServices.EnvEvento.Executar;

  if not Result then
    GerarException( WebServices.EnvEvento.Msg );
end;
```

É mais vantajoso um desenvolvedor gastar mais tempo para otimizar uma rotina do que 10 trabalhando em uma rotina improdutiva.

5 ETAPAS PARA MANTER UM CÓDIGO LIMPO

1 - Ordenação
2 - Sistematização
3 - Limpeza
4 - Padronização
5 - Disciplina

DIVIDIR PARA CONQUISTAR

Separação de funções auxiliam a leitura do código

```
procedure TForm1.Button1Click(Sender: TObject);
var
  vText : PChar;
  vRetorno : String;
begin

  vText := PChar(Edit1.Text);
  while (vText^ <> #0) do
  begin
    if CharInSet(vText^, ['0'..'9']) then
      vRetorno := vRetorno + vText^;

    Inc(vText);
  end;

  if Length(vRetorno) <> 8 then
    raise Exception.Create('Entrada Inválida');


  Edit2.Text := Copy(vRetorno, 1, 2) + '.' + Copy(vRetorno, 3, 3) + '-' + Copy(vRetorno, 6, 3);
end;
```

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  TCommand
    .New
      .&then(removeLettersAndSpecialCharacters)
      .&then(validateZipCode)
    .resolve(EditInputZipCode);
end;
```

Quando uma função é pequena ele é especialista e pode ser reutilizada

Se uma nova implementação precisar ser feita em uma regra de negócio, ela não pode afetar outras áreas da regra de negócio.

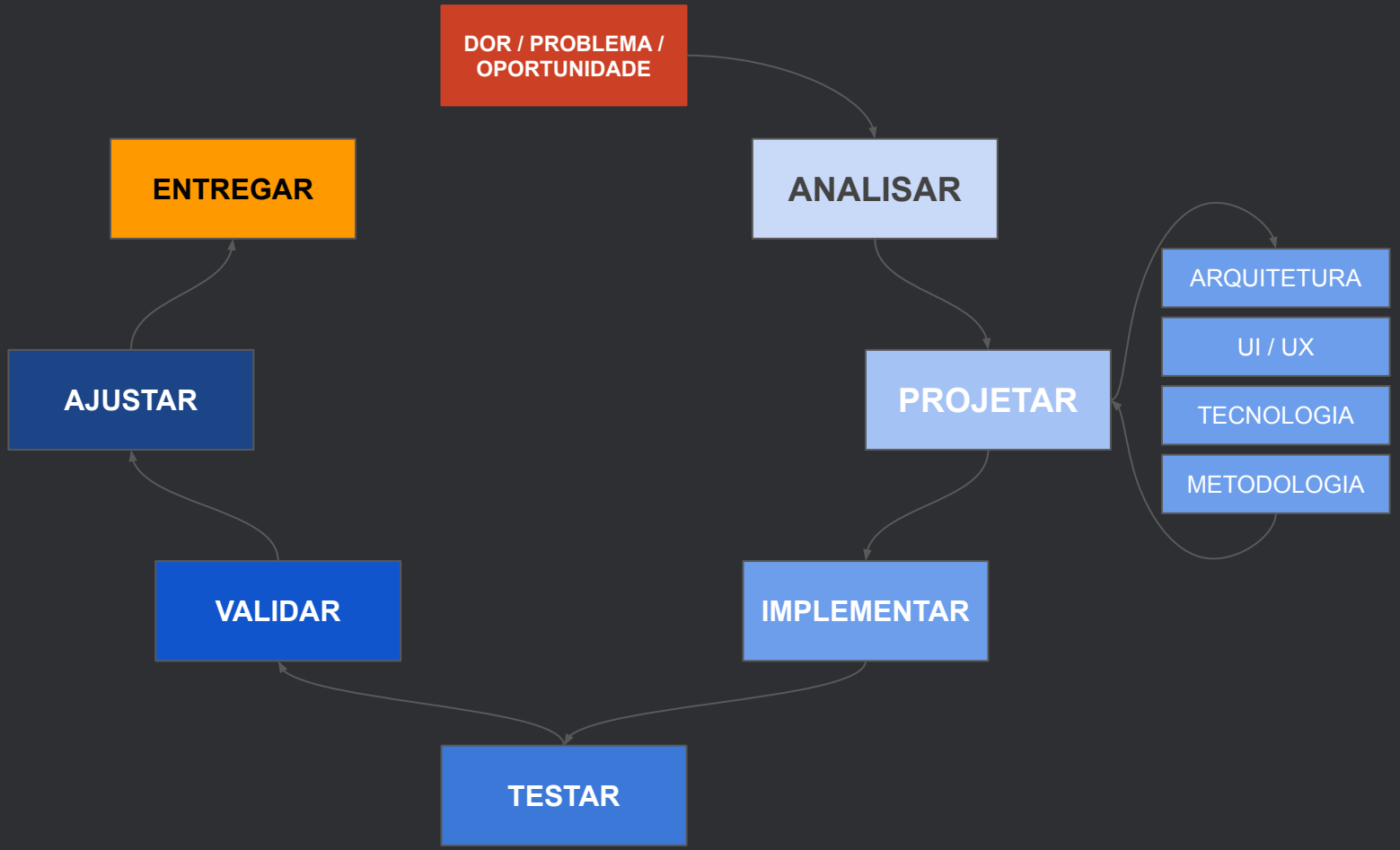
TODO TRABALHO DEVE TER UM ESCOPO FECHADO

Quanto mais o tempo passa mais IMPRODUTIVO fica

80% do que fazemos em desenvolvimento de software é manutenção

Empresário Brasileiro tem pouco tempo para planejar e muito dinheiro para gastar com manutenção!
(ironia)

Todo projeto precisa ter começo, meio e fim



CÓDIGO RUIM PODE DESTRUIR O MUNDO

Problemas no Mariner (1962)

Custo: 18,5 milhões dólares

Desastre: Mariner, um foguete com uma sonda espacial para Vênus, foi desviado de seu percurso de voo logo após o lançamento. O controle da missão destruiu o foguete 293 segundos após a decolagem.

Causa: Um programador, ao passar para o computador uma fórmula que haviam lhe entregado escrita manualmente, se esqueceu de uma barra. Sem ela, o software tratava variações normais de velocidade como se fossem sérios problemas, causando falhas por tentativas de correções que acabaram por enviar o foguete fora do curso.

3ª Guerra Mundial (Quase!) (1983)

Custo: Quase toda a humanidade

Desastre: O sistema de alerta precoce soviético falsamente indicou que os Estados Unidos tinham lançado cinco mísseis balísticos. Felizmente, o oficial de serviço soviético tinha uma “sensação esquisita no estômago” e fundamentalmente, se os EUA estavam realmente atacando, eles lançariam mais de cinco mísseis, por isso ele relatou o aparente ataque como um alarme falso.

Causa: Um bug no software soviético falhou ao detectar reflexos solares como falsos mísseis.

Linhas da AT&T “morrem” (1990)

Custo: 75 milhões de ligações perdidas e 200 reservas aéreas perdidas

Desatre: Um switch dos 114 centros de swiches da AT&T sofreu um problema mecanico que fez com que todo o seu centro fosse desligado. Quando o seu centro voltou a ativa, enviou uma mensagem aos outros, o que causou o desligamento dos outros centros e deixou a empresa parada por 9 horas.

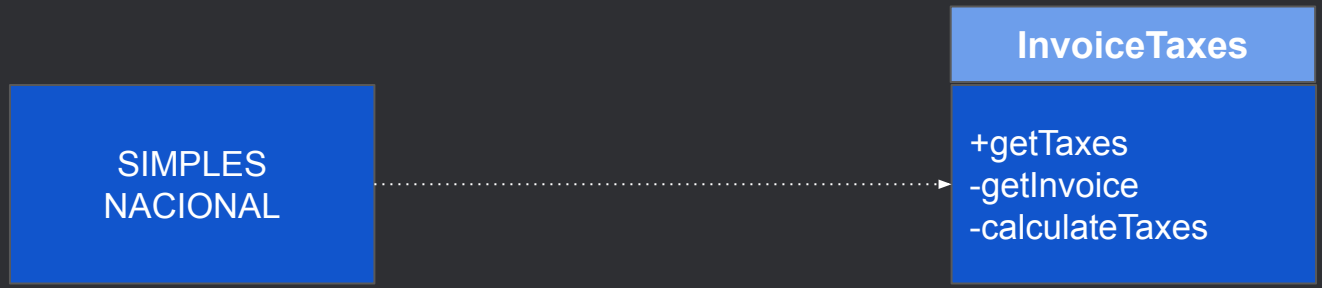
Causa: Uma única linha de código em uma atualização de software implementada para acelerar chamadas causou um efeito cascata que desligou a rede.

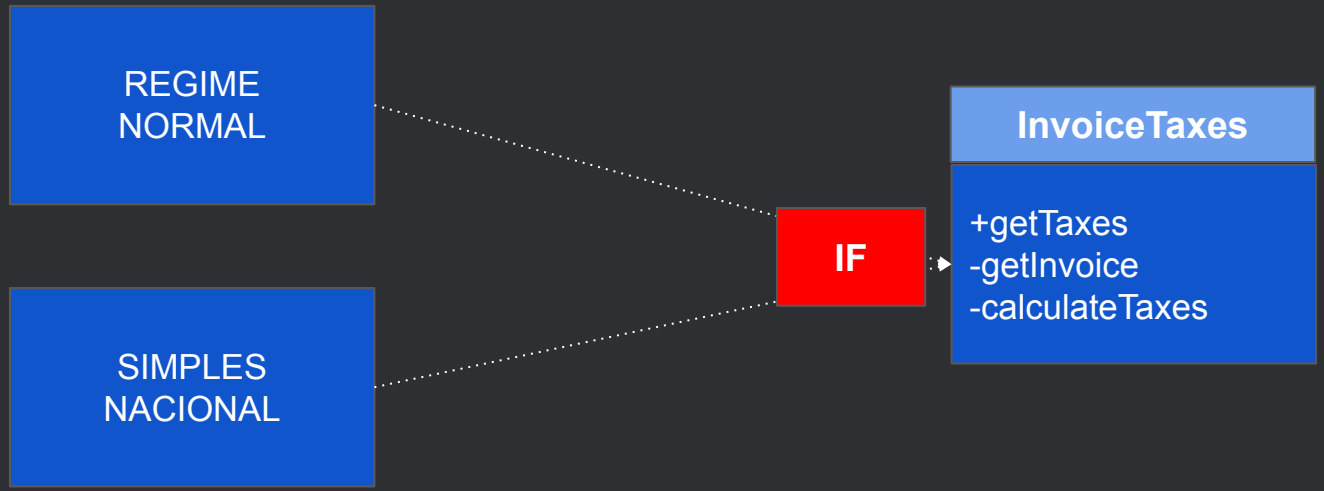
Patriot Acaba com Soldados (1991)

Custo: 28 soldados mortos e 100 feridos.

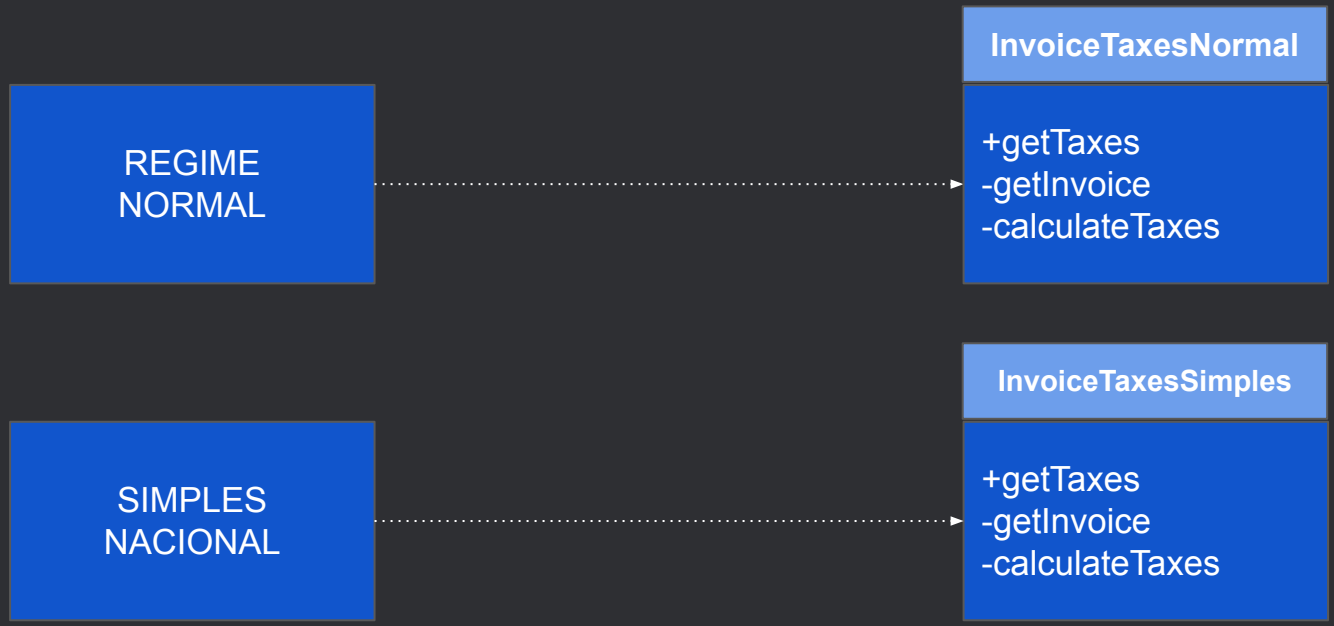
Desastre: Durante a primeira Guerra do Golfo, um sistema (Patriot) americano de mísseis na Arábia Saudita falhou ao interceptar um míssel vindo do Iraque. O míssel destruiu acampamentos americanos.

Causa: Um erro de arredondamento no software calculou incorretamente o tempo, fazendo com que o sistema Patriot ignorasse os mísseis Scud de entrada.

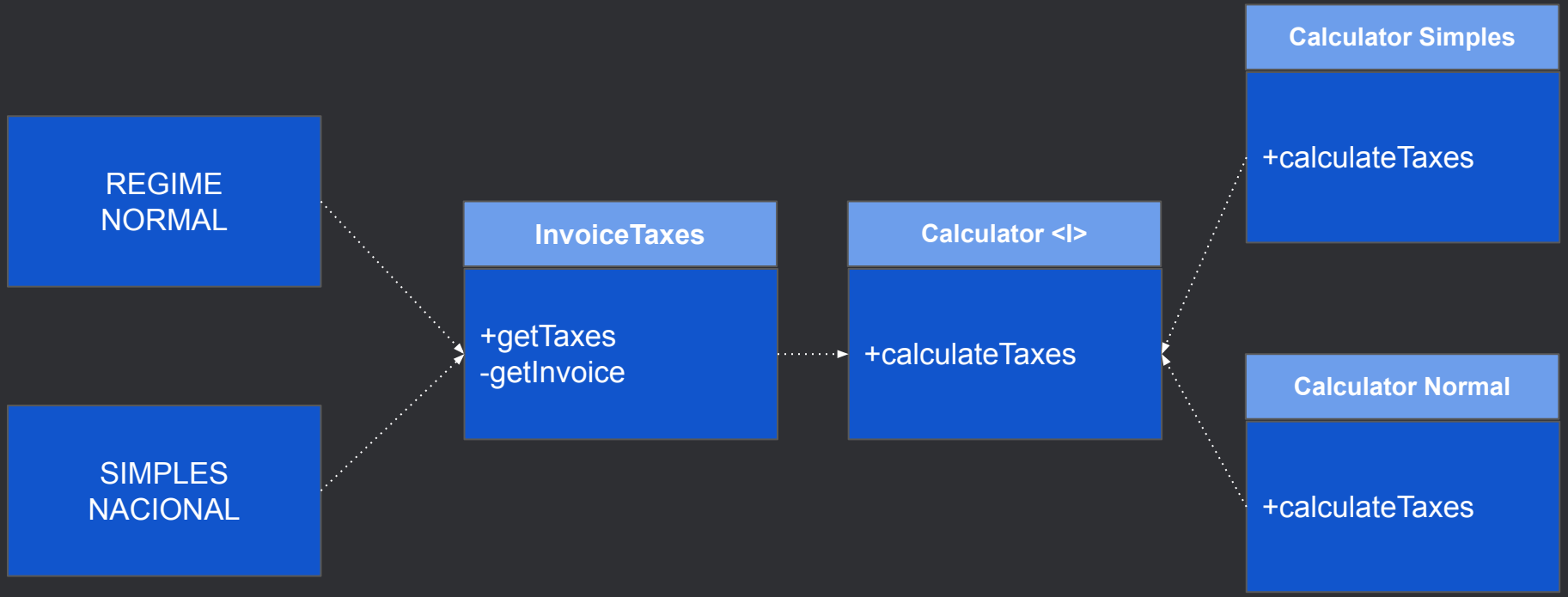




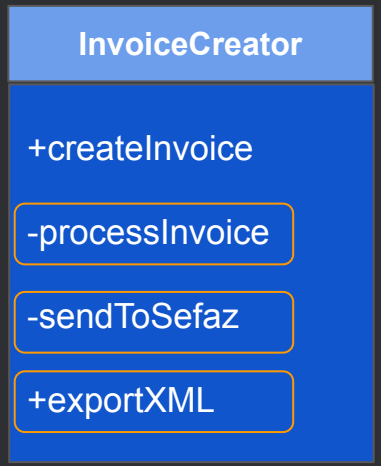
Responsabilidade Única
Código Duplicado



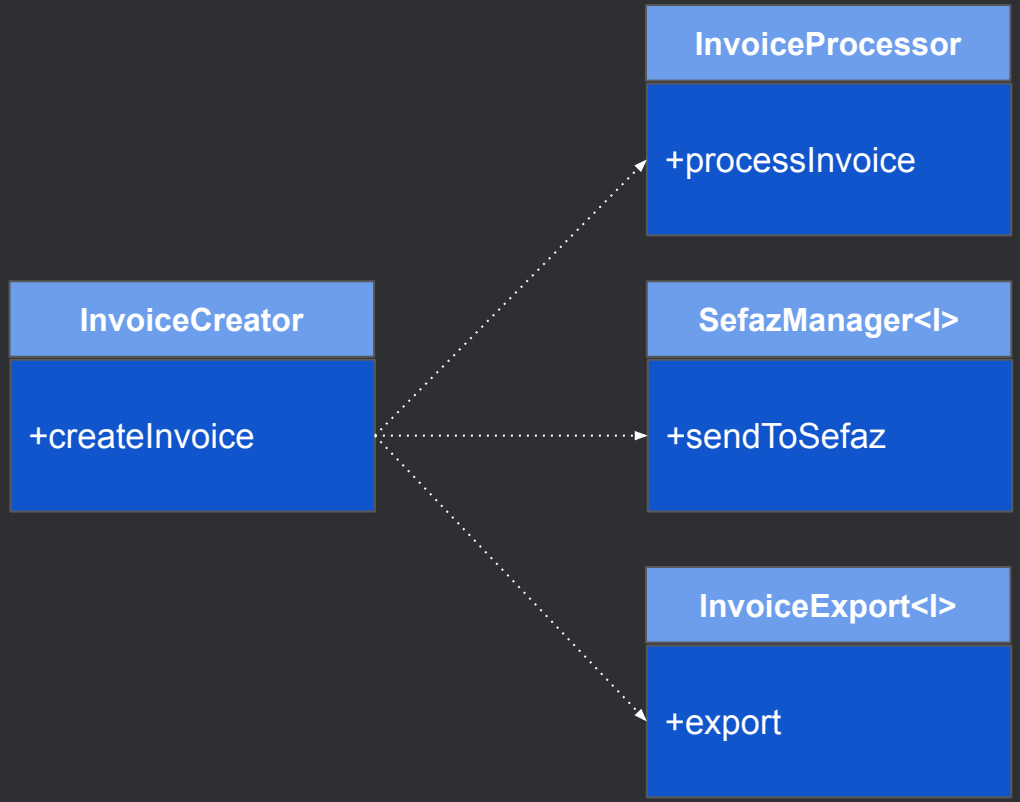
SOLID e DRY



Baixa Coesão



Alta Coesão



Alta Coesão
Inversão de Dependência