

# Embarcadero Conference 2024

---

Inovação faz parte do nosso DNA!



{Ivan Souza

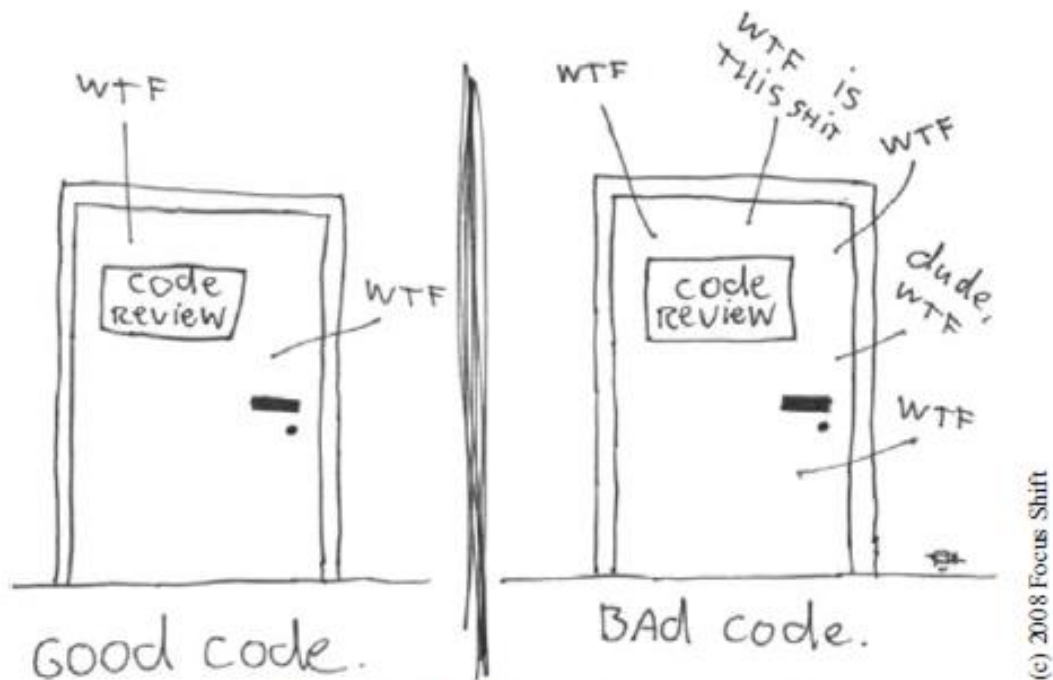
# **Delphi com Clean Architecture**

## **Do planejamento aos testes unitários**

Nesta palestra, veja como organizar seu projeto de forma a ter units desacopladas, independentes e dimensionadas para serem usadas em multi-plataforma, multi-tarefa, reativas a testes unitários e disponíveis para pipelines em CI/CD



The ONLY VALID MEASUREMENT  
OF CODE QUALITY: WTFs/MINUTE



Reproduced with the kind permission of Thom Holwerda.  
[http://www.osnews.com/story/19266/WTFs\\_m](http://www.osnews.com/story/19266/WTFs_m)

# Métricas de um Código de Qualidade

## What Is Clean Code?

There are probably as many definitions as there are programmers. So I asked some very well-known and deeply experienced programmers what they thought.

**Bjarne Stroustrup, inventor of C++  
and author of *The C++ Programming Language***

*I like my code to be elegant and efficient. The logic should be straightforward to make it hard for bugs to hide, the dependencies minimal to ease maintenance, error handling complete according to an articulated strategy, and performance close to optimal so as not to tempt people to make the code messy with unprincipled optimizations. Clean code does one thing well.*



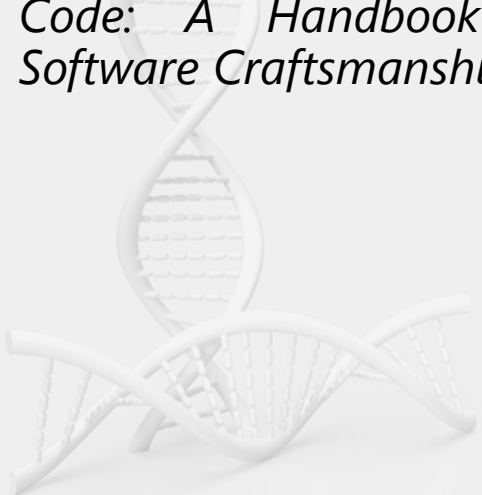
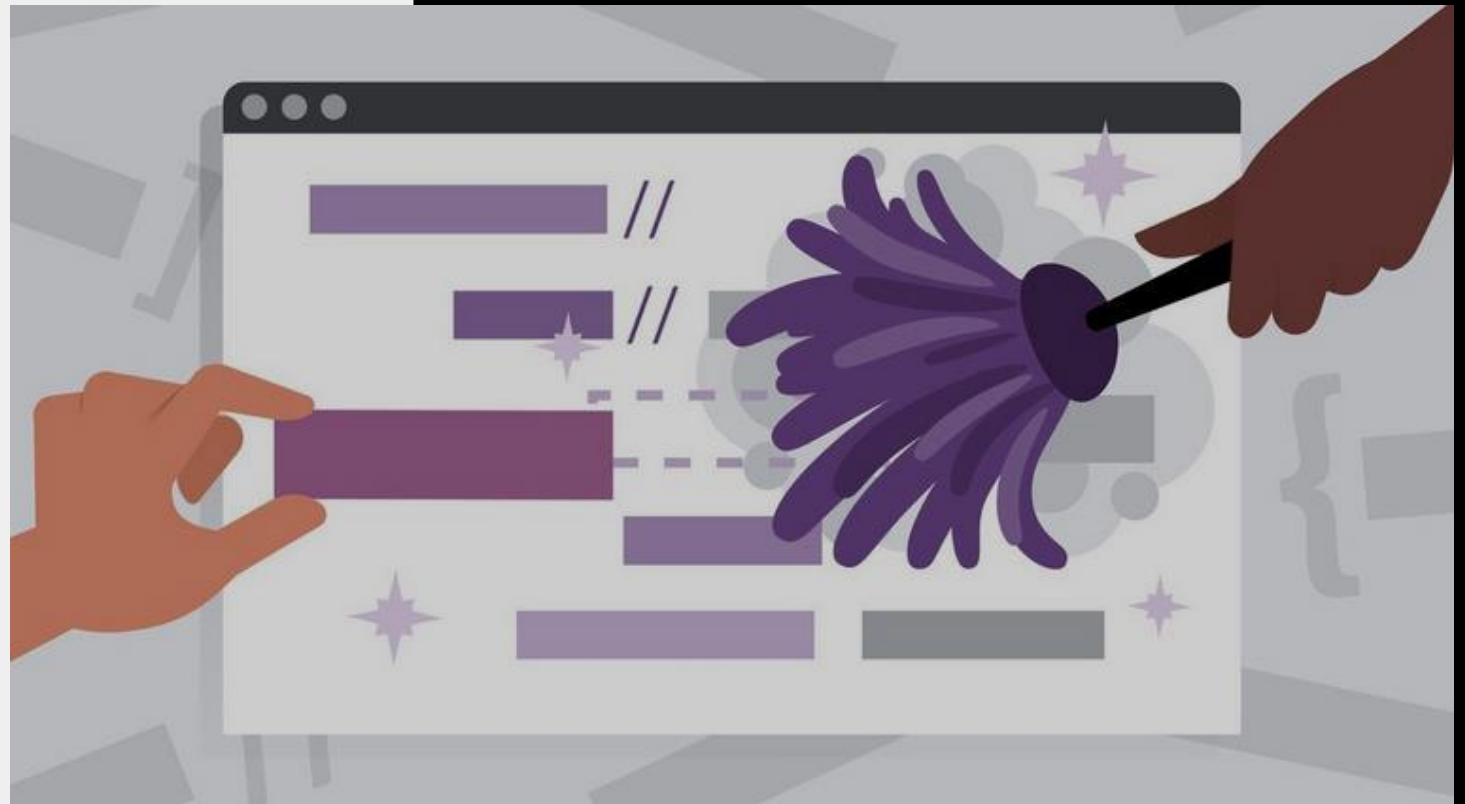
# Clean Code



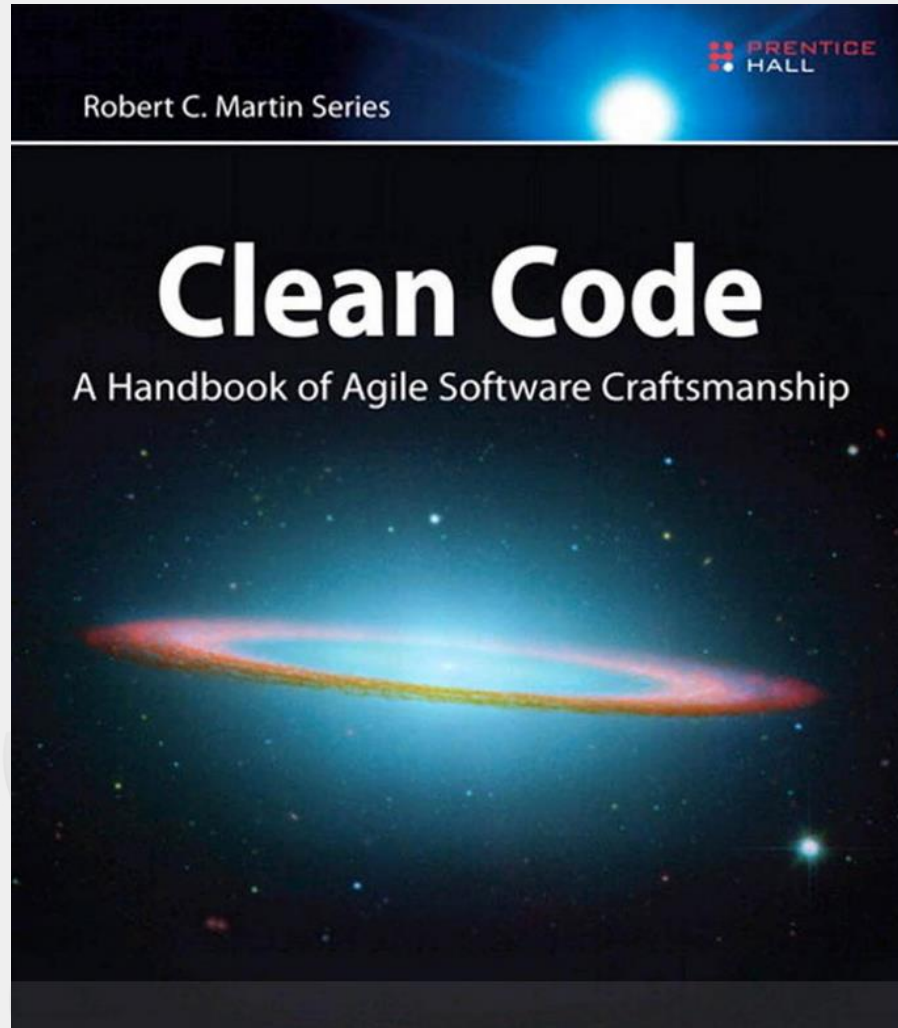
# Clean Code

"Clean Code" é um conceito que se refere a práticas de programação que resultam em um código que é fácil de ler, entender, e manter.

O termo foi popularizado por Robert C. Martin (também conhecido como "Uncle Bob") em seu livro intitulado *Clean Code: A Handbook of Agile Software Craftsmanship*.







**Fonte:**  
**uncle Bob**

# **Nomes Significativos**

**Funções Pequenas e Simples**  
(KISS – Keep It Simple, Stupid)

**Evite Comentários Excessivos**

**Formatação Consistente**

**Reduzir Repetição**  
(DRY – Don't Repeat Yourself)

**Controle de Fluxo Simples**

**Manutenção Simples**

**Trate Exceções**



# **Alguns Princípios do Clean Code**

**Facilidade de Leitura**

**Manutenção e Extensão**

**Colaboração**

**Menos Erros**

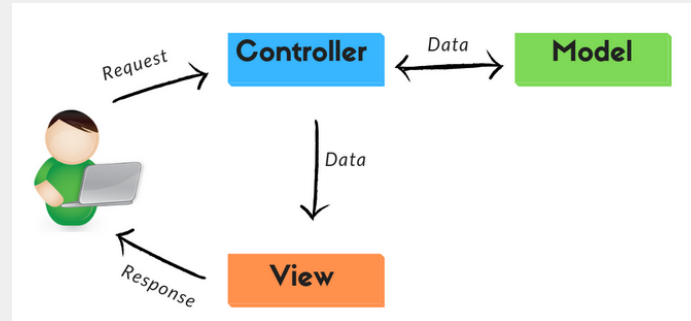
Clean Code não é uma prática específica para uma linguagem de programação; é um conjunto de princípios universais aplicáveis a qualquer linguagem.

Implementar Clean Code pode requerer mais esforço inicial, mas os benefícios em termos de manutenção e escalabilidade são significativos.

# **Benefícios do Clean Code**



# MVC – Model View Controller



## Solid Principles



**Extra: REDIS**

**Padrões  
no  
Clean Code**

## O que é o Padrão MVC?

O padrão **MVC** (**Model-View-Controller**) é uma arquitetura de software que ajuda a organizar e estruturar o código, o que se alinha com os princípios de **Clean Code** ao promover uma separação clara de responsabilidades. Isso torna o código mais fácil de entender, manter e testar.

O padrão **MVC** divide uma aplicação em três componentes principais:

### Model (Modelo)

- Representa os dados e a lógica de negócios.
- Gerencia o estado da aplicação e responde a comandos para manipular esses dados.

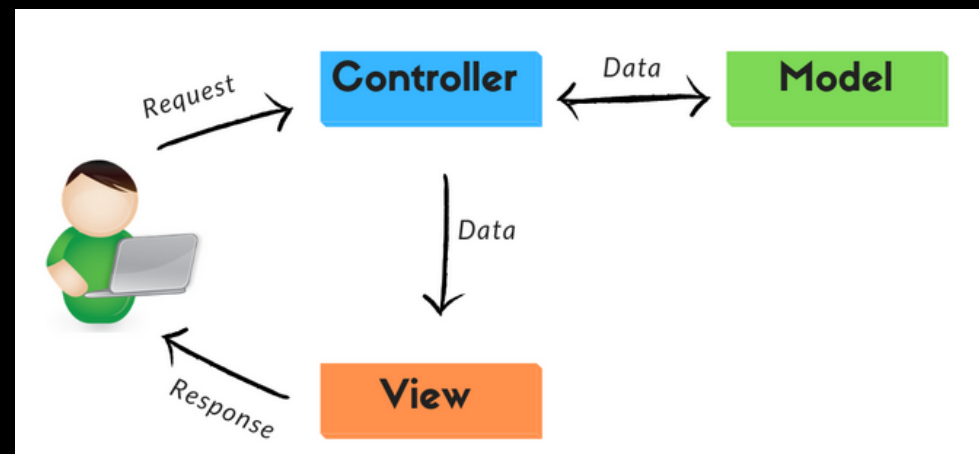
### View (Visão)

- Responsável pela interface do usuário e pela apresentação dos dados.
- Exibe as informações do **Model** para o usuário e envia as entradas do usuário para o **Controller**.

### Controller (Controlador)

- Atua como intermediário entre o **Model** e a **View**.
- Recebe entradas do usuário através da **View**, processa essas entradas (geralmente manipulando o **Model**), e atualiza a **View** de acordo.

# Padrão MVC



## Separação de Responsabilidades

**MVC** promove uma separação clara das responsabilidades: **Model** cuida da lógica e dados, **View** cuida da apresentação, e **Controller** lida com a lógica de interação. Isso evita o código "espaguete".

## Facilidade de Manutenção e Escalabilidade

Ao separar as preocupações, o padrão **MVC** facilita a manutenção e a expansão da aplicação.

## Testabilidade

Com o **MVC**, é mais fácil testar cada componente de forma isolada.

## Reusabilidade

**Models e Controllers** podem ser reutilizados em diferentes partes da aplicação ou até mesmo em outras aplicações.



# Como o MVC se relaciona com o Clean Code?

O **SOLID** é um conjunto de princípios de design de software criado para ajudar os desenvolvedores a escrever código mais eficiente, manutenível e escalável. Cada letra da palavra "SOLID" representa um dos cinco princípios:

1. **Single Responsibility Principle (Princípio da Responsabilidade Única)**

Uma classe deve ter uma única responsabilidade ou motivo para mudar. Em outras palavras, uma classe deve fazer apenas uma coisa e fazê-la bem.

2. **Open/Closed Principle (Princípio do Aberto/Fechado)**

As entidades de software (como classes, módulos e funções) devem estar abertas para extensão, mas fechadas para modificação.

3. **Liskov Substitution Principle (Princípio da Substituição de Liskov)**

Objetos de uma classe derivada devem poder substituir objetos da classe base sem alterar a correção do programa.

4. **Interface Segregation Principle (Princípio da Segregação de Interfaces)**

Uma classe não deve ser forçada a implementar interfaces que não usa. Em vez de ter uma interface grande e genérica, é preferível dividir essa interface em várias interfaces menores e mais específicas.

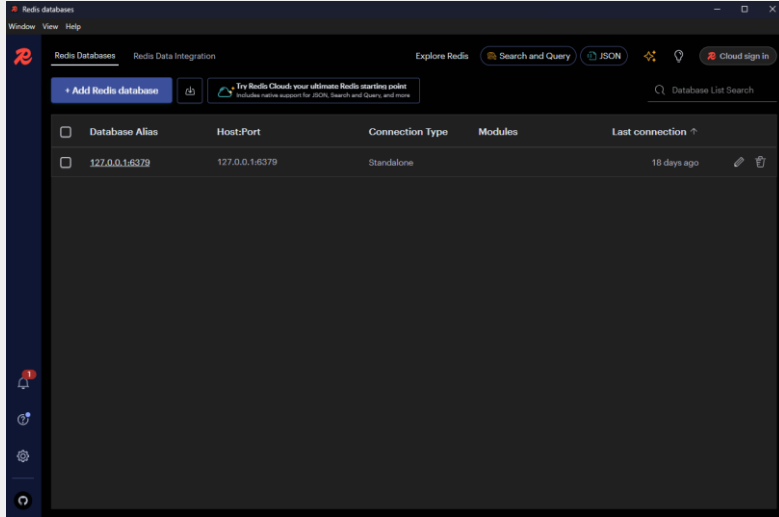
5. **Dependency Inversion Principle (Princípio da Inversão de Dependência)**

Módulos de alto nível não devem depender de módulos de baixo nível; ambos devem depender de abstrações (interfaces ou classes abstratas).

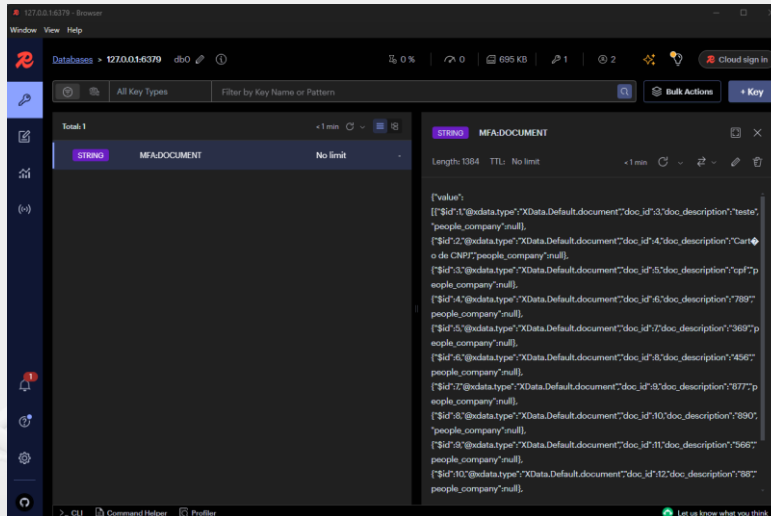
Esses princípios foram popularizados por Robert C. Martin (também conhecido como Uncle Bob) e são amplamente adotados na engenharia de software para melhorar a qualidade e a sustentabilidade do código.

# SOLID Principles





# Extra: REDIS

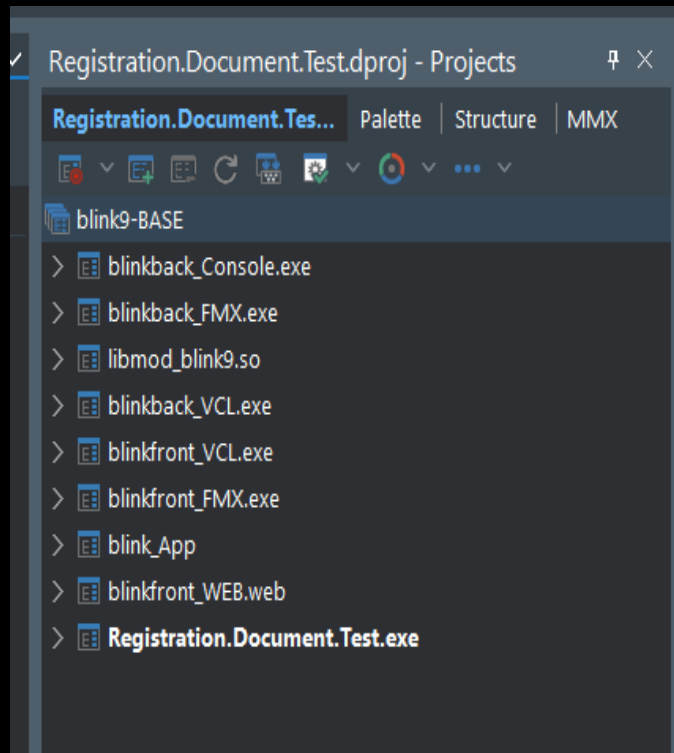


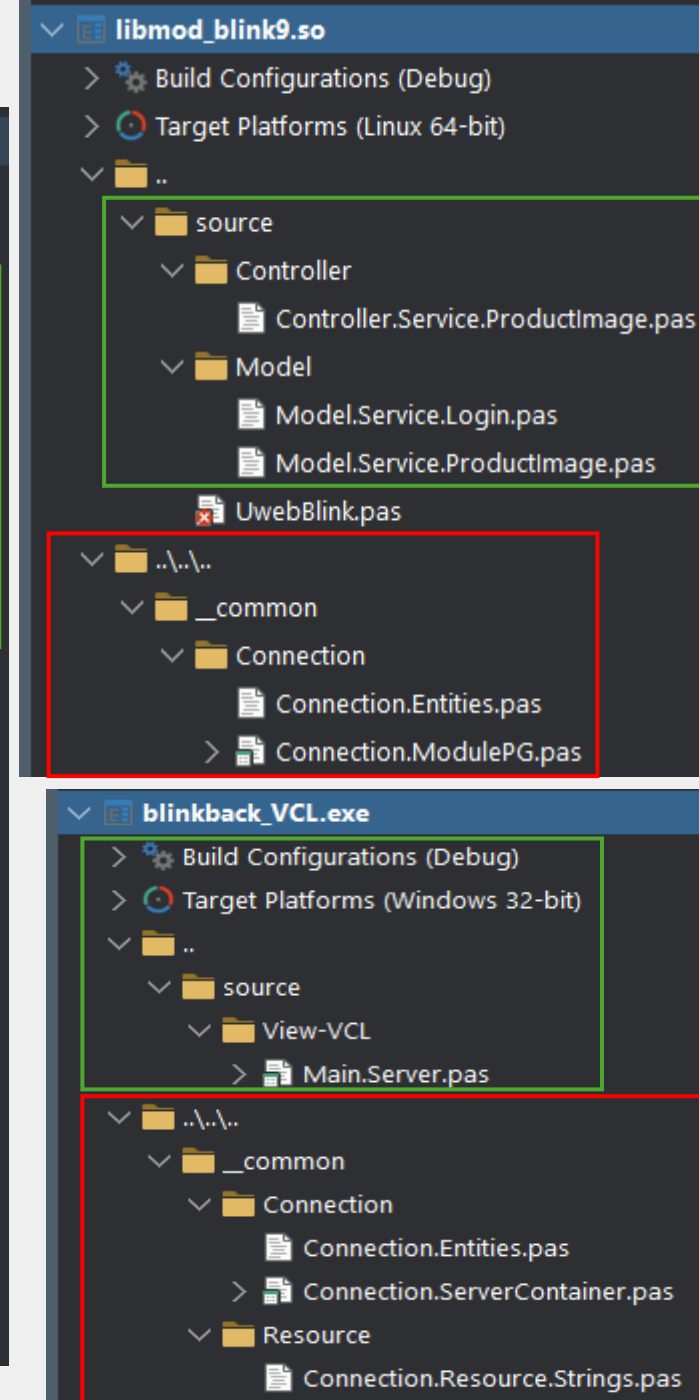
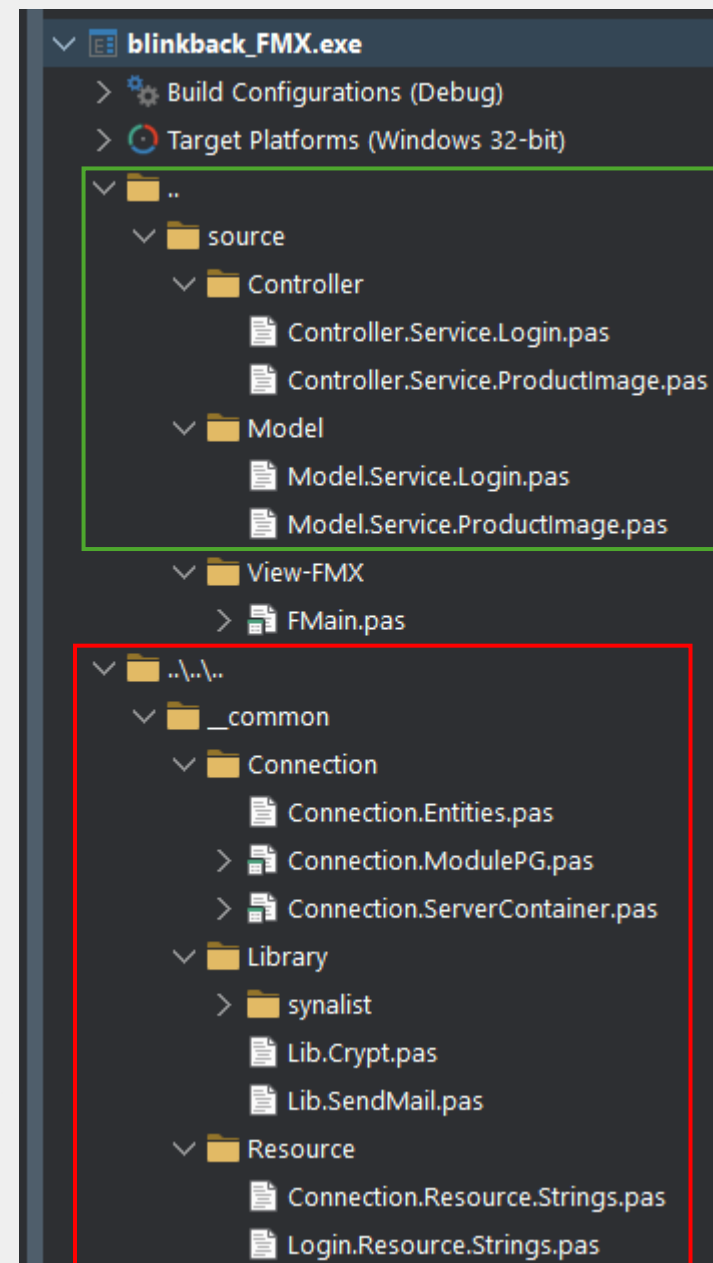
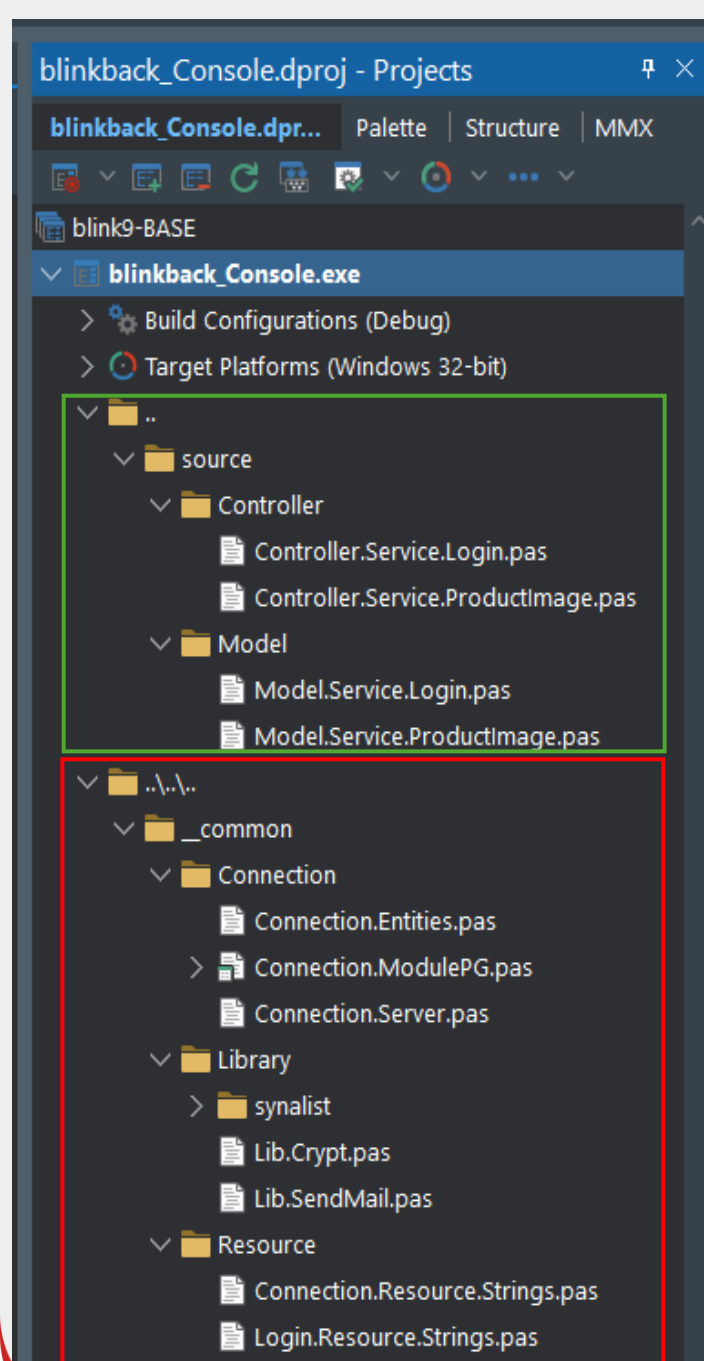
# Aplicando no Delphi



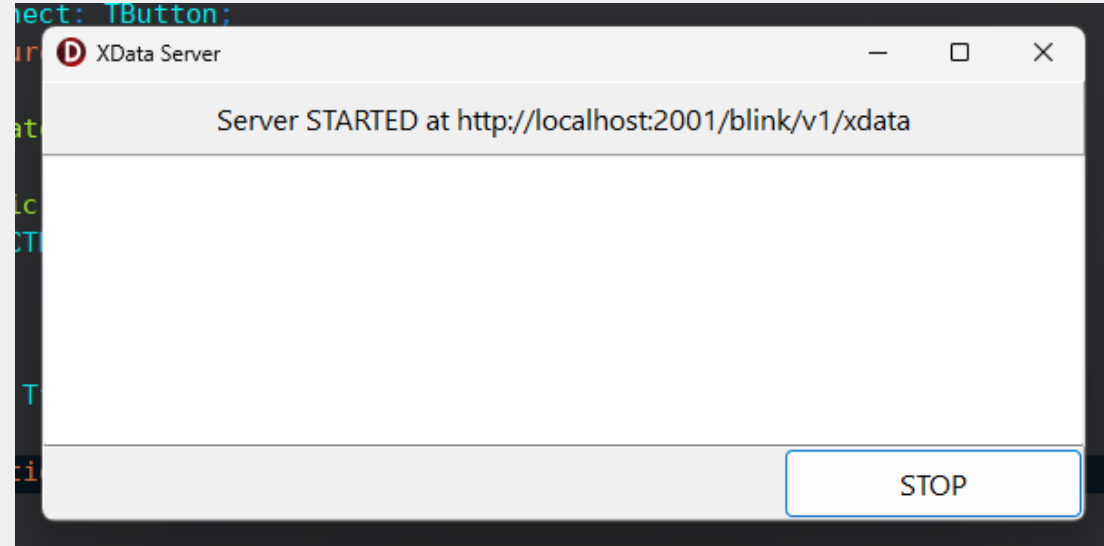
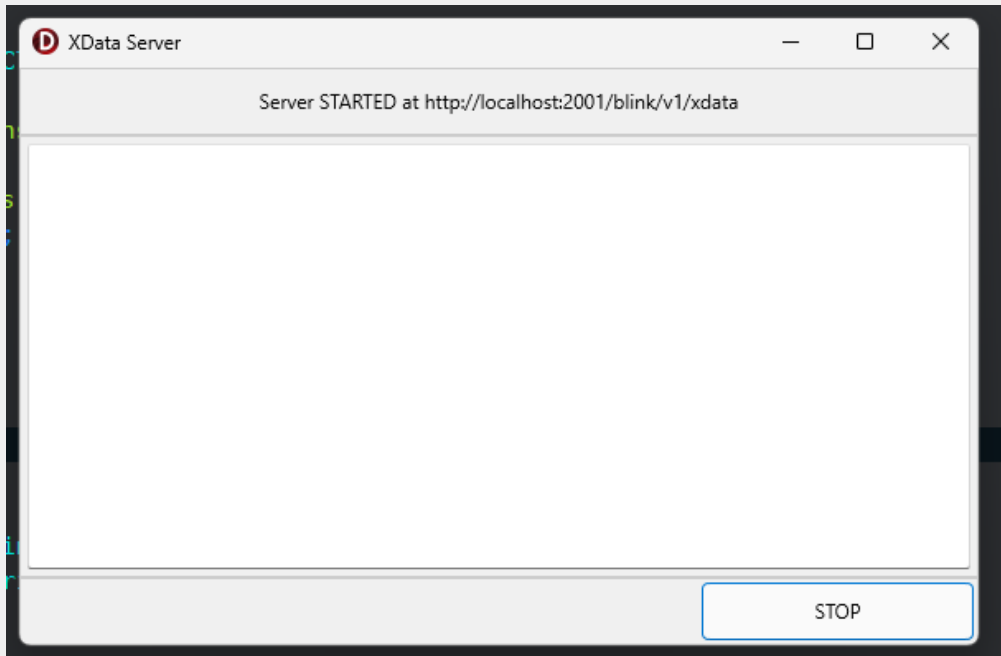
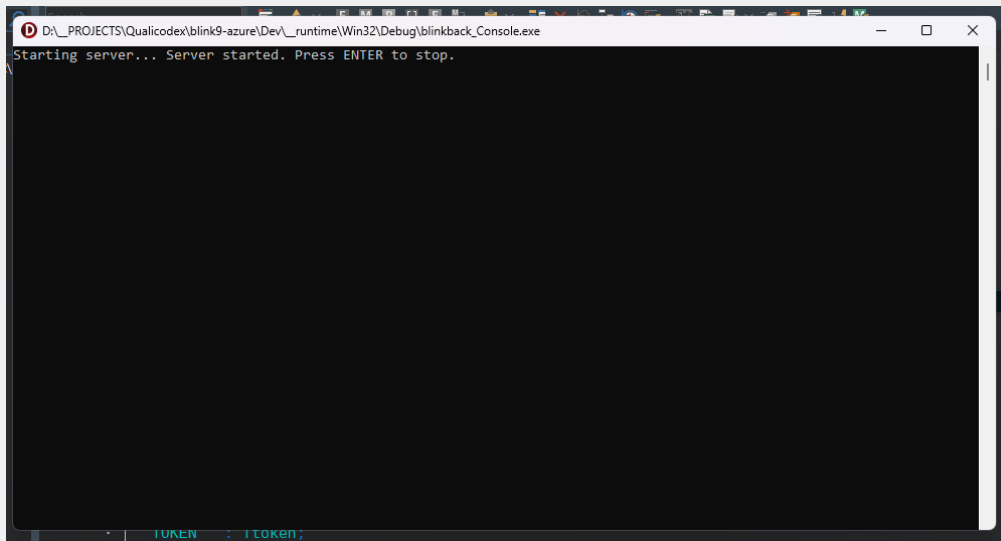


# Definições





Back - end



**B  
a  
c  
k  
-  
e  
n  
d**

- > blinkback\_VCL.exe
- ▼ blinkfront\_VCL.exe
  - > Build Configurations (Debug)
  - > Target Platforms (Windows 32-bit)
  - ▼ ..
    - ▼ source
      - > Controller
      - > Model
      - > Resource
      - > View-VCL
    - ▼ ..\..\..
      - ▼ \_common
        - > Connection
        - > Resource
  - > blinkfront\_FMX.exe
  - > blink\_App
  - > blinkfront\_WEB.web
  - > **Registration.Document.Test.exe**

- ▼ blinkfront\_VCL.exe
  - > Build Configurations (Debug)
  - > Target Platforms (Windows 32-bit)
  - ▼ ..
    - ▼ source
      - ▼ Controller
        - ConfigUI.Controller.pas
        - Configuration.Controller.pas
        - Document.Controller.pas
        - FirstAccess.Controller.pas
        - MainMenu.Controller.pas
        - Notification.Single.Controller.pas
        - People.Controller.pas
        - People.Document.Controller.pas
        - Registration.Card.Controller.pas**
        - Registration.Document.Controller.pas
        - Registration.People.Controller.pas
        - UserLogin.Controller.pas
  - > Model
  - > Resource
  - > View-VCL

- ▼ blinkfront\_VCL.exe
  - > Build Configurations (Debug)
  - > Target Platforms (Windows 32-bit)
  - ▼ ..
    - ▼ source
      - > Controller
      - ▼ Model
        - Document.Model.pas
        - FirstAccess.DTO.pas
        - Configuration.Model.pas
        - MainMenu.DTO.pas
        - Notification.Single.pas
        - People.Document.Model.pas
        - People.Model.pas
        - Registration.Card.DTO.pas**
        - Registration.Document.DTO.pas
        - Registration.People.DTO.pas
        - UserLogin.Model.pas
      - > Resource
      - > View-VCL
    - ▼ ..\..\..
      - ▼ \_common
        - > Connection
        - > Resource
  - > blinkfront\_FMX.exe

- ▼ blinkfront\_VCL.exe
  - > Build Configurations (Debug)
  - > Target Platforms (Windows 32-bit)
  - ▼ ..
    - ▼ source
      - > Controller
      - > Model
      - > Resource
      - ▼ View-VCL
        - > Configurion.View.pas
        - > FirstAccess.View.pas
        - > Login.View.pas
        - > MainMenu.View.pas
        - > Notification.Single.View.pas
        - > Registration.Base.View.pas
        - > **Registration.Card.View.pas**
        - > Registration.Document.View.pas
        - > Registration.People.View.pas
    - ▼ ..\..\..
      - ▼ \_common
        - > Connection
        - > Resource
  - > blinkfront\_FMX.exe
  - > blink\_App
  - > blinkfront\_WEB.web
  - > **Registration.Document.Test.exe**



# Front-end



```

uses
  System.Generics.Collections,
  Connection.Entities;

type
  // FirstAccess DTO
  IFirstAccessDTO = Interface['{57B58518-3691-45A6-AA71-565E366CCC3D}']
  // Main
  function GetMain : Boolean;
  procedure SetMain(aValue : Boolean);

  // DTO functions
  function DocumentLoadData : TList<Tdocument>;
  procedure DocumentSaveData;
  procedure PeopleSaveData(aPeopleName,
    aPeopleAddress,
    aPeopleNumber,
    aPeopleComplement,
    aPeopleNeighborhood,
    aPeopleTown,
    aPeopleState,
    aPeoplePostalCode,
    aPeopleEmail : String);

  // Other Functions
  function MainExists : Boolean;
  function MainUpdate : Boolean;
end;

implementation

end.

```

## Model

```

procedure TFirstAccessController.PeopleSaveData(aPeopleName,
  aPeopleAddress,
  aPeopleNumber,
  aPeopleComplement,
  aPeopleNeighborhood,
  aPeopleTown,
  aPeopleState,
  aPeoplePostalCode,
  aPeopleEmail : String);

const
  _ID = -1;
var
  _People : TPeopleController;
begin
  _People := TPeopleController.Create;

  _People.Name      := aPeopleName;
  _People.Address   := aPeopleAddress;
  _People.Number    := aPeopleNumber;
  _People.Complement := aPeopleComplement;
  _People.Neighborhood := aPeopleNeighborhood;
  _People.Town      := aPeopleTown;
  _People.State     := aPeopleState;
  _People.PostalCode := aPeoplePostalCode;
  _People.Email     := aPeopleEmail;

  if not _People.PeopleExists(_ID) then
  begin
    _People.PeopleUpdate(_ID);
  end;
end;

```

## Controller

```

procedure TfrmMainFirstAccessView.btnUserSaveClick(Sender: TObject);
begin
  FFirstAccessController.PeopleSaveData(edtUserName.Text,
    edtUserAddress.Text,
    edtUserNumber.Text,
    edtUserComplement.Text,
    edtUserNeighborhood.Text,
    edtUserTown.Text,
    edtUserState.Text,
    edtUserPostalCode.Text,
    edtUserEmail.Text);
end;

```

## View

```

//
function TFirstAccessController.DocumentLoadData : TList<Tdocument>;
var
  _RedisString : TRedisString;
  _Exists       : Boolean;

  _Entity : TEntity<Tdocument, TList<Tdocument>>;
  JsonResult, JSONString : string;
begin
  _RedisString := TRedisString.Create('');
  _RedisString.Value := '';

  // if REDIS is on
  if Assigned(FRedisClient) then
  begin
    _Exists := FRedisClient.EXISTS(FRedisKeyMFADocument);

    if _Exists then
    begin
      _RedisString.Value := FRedisClient.GET(FRedisKeyMFADocument).Value;
    end;
  end;

  // if REDIS key exists
  if not SameStr(_RedisString.Value, '') then
  begin
    // Create Empty Entity
    FDocuments := TList<Tdocument>.Create;
    _Entity := TEntity<Tdocument, TList<Tdocument>>.Create;

    try
      // Create Entity from JSON value from Redis String
      FDocuments.AddRange(_Entity.CreateEntityFromJSON(_RedisString.Value));
    finally
      _Entity.Free;
    end;
  end;
end;

```

## Controller

Form1

Usuário  Empresa

Senha

Token

BLINK - LOGIN

Username


Password

Main Menu

Ínício Cadastro Fluxo de Caixa Relatório Procurar Configuração Ajuda

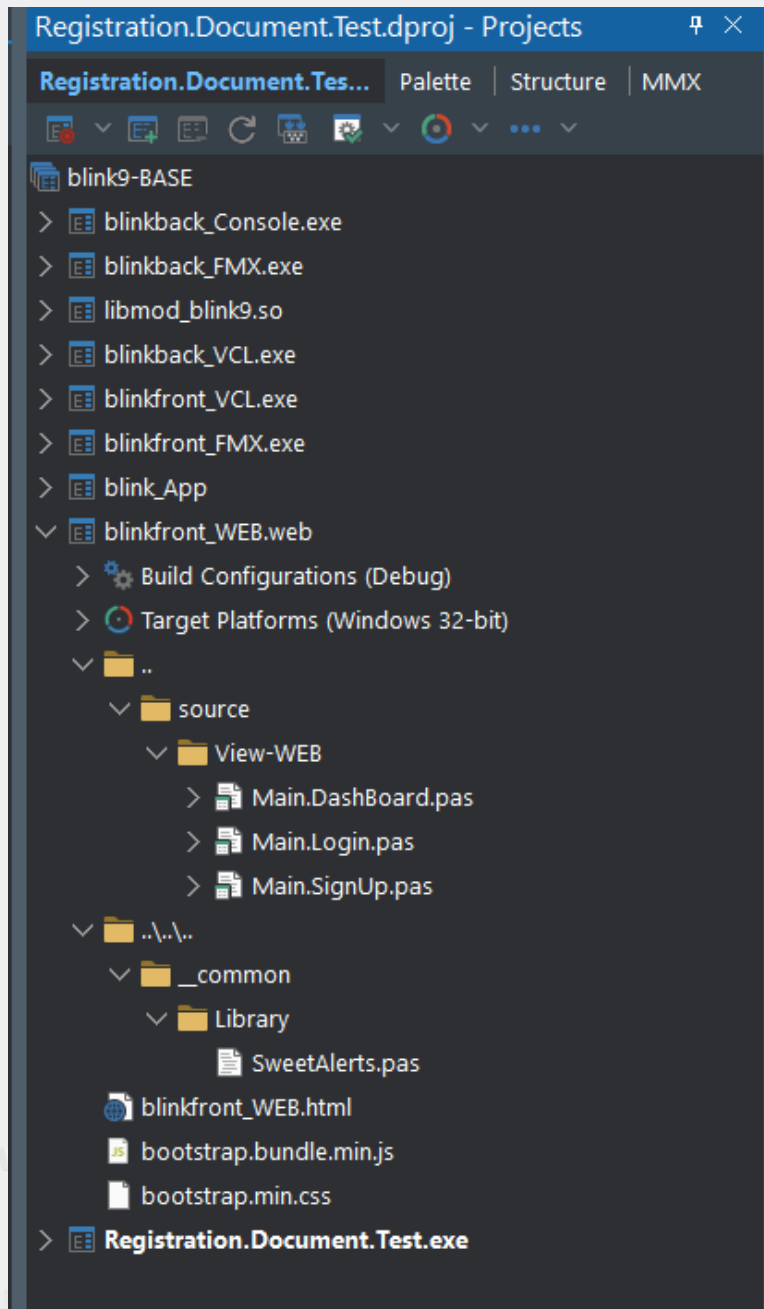
New Open Save Print Paste Copy Select All

File Clipboard Início

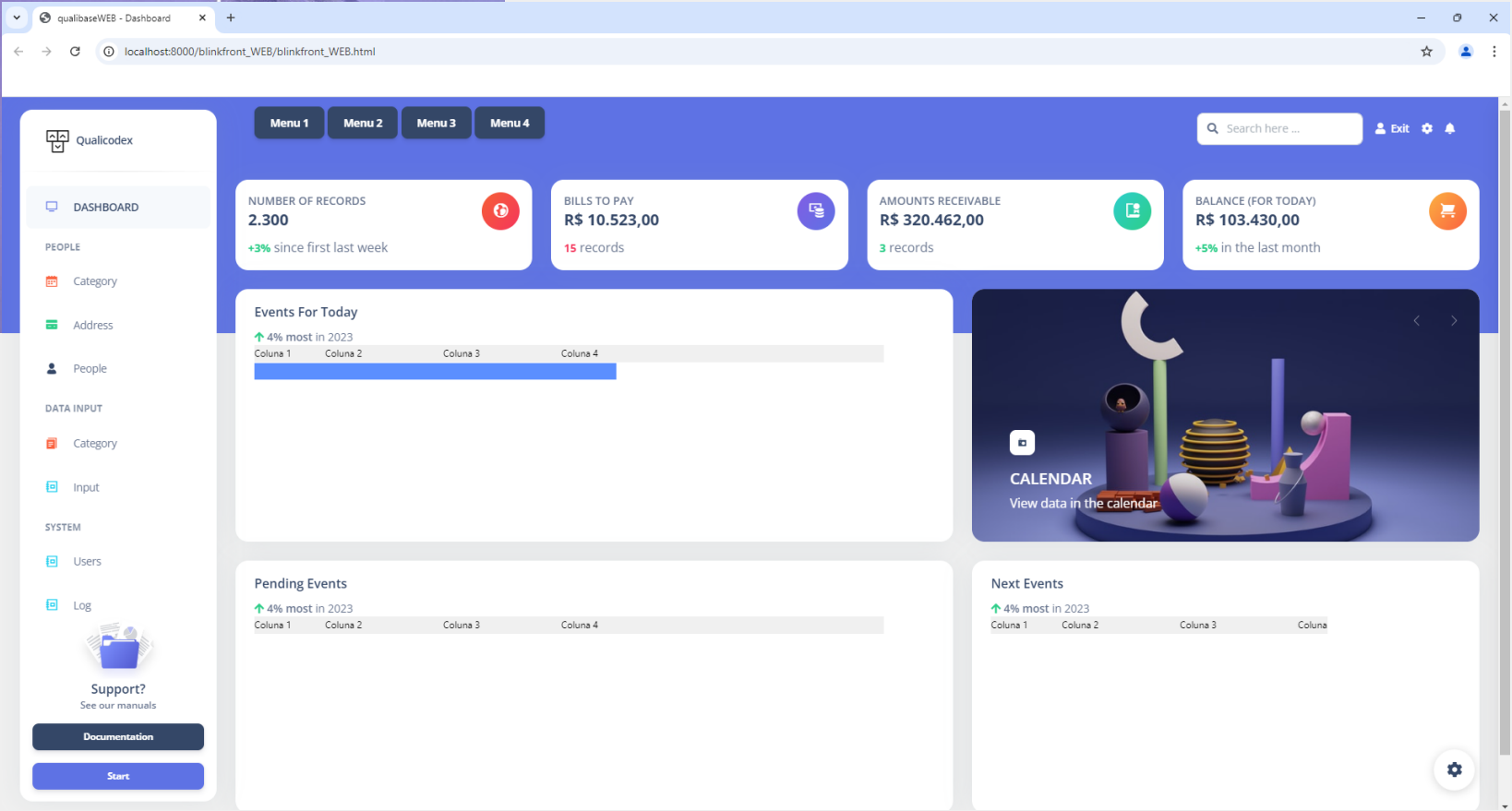
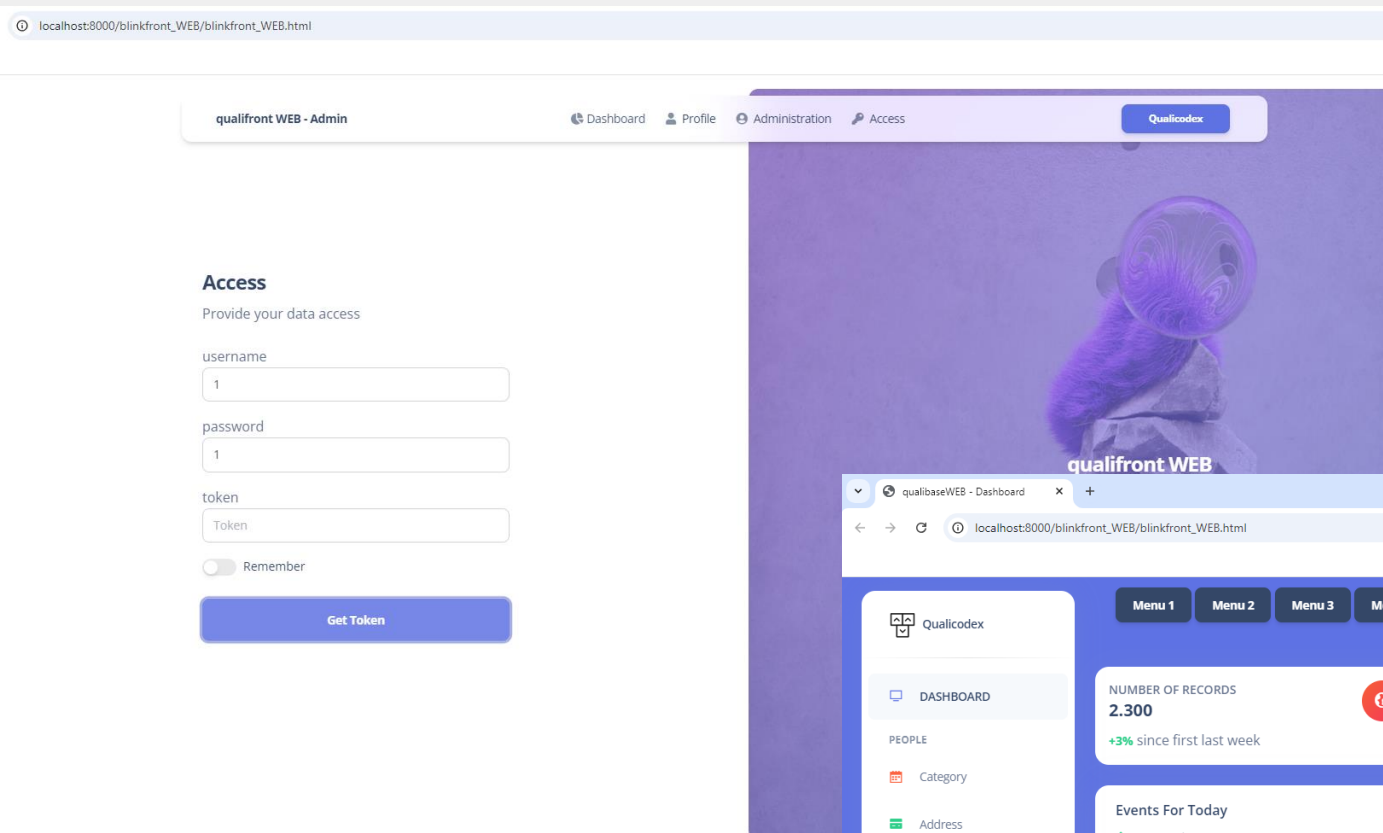


Front-end

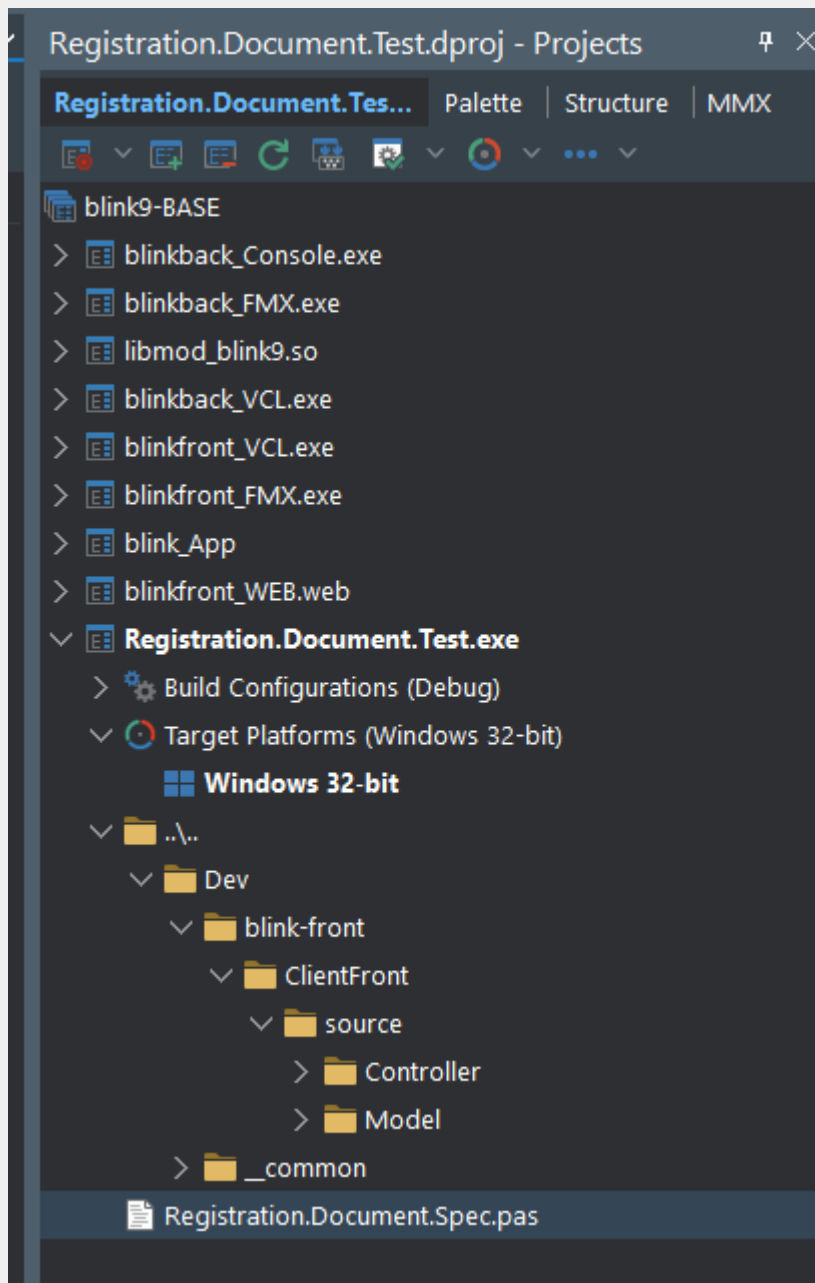




# Front-end Web

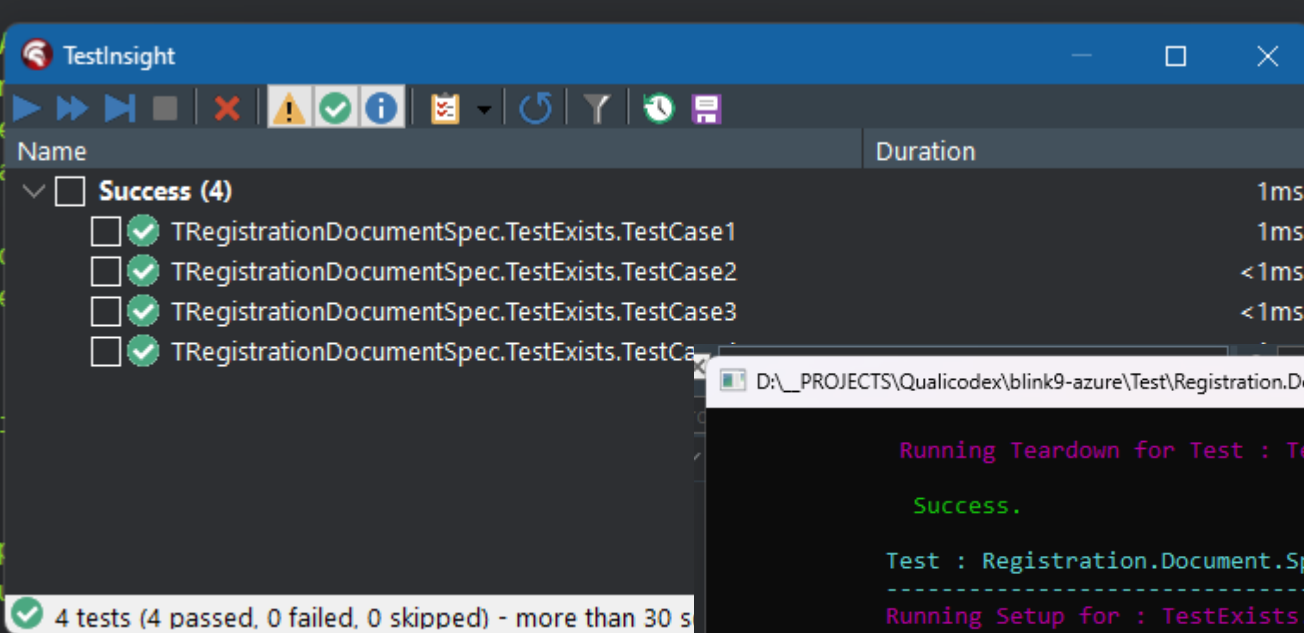


Front-end  
Web



# Testes Unitários

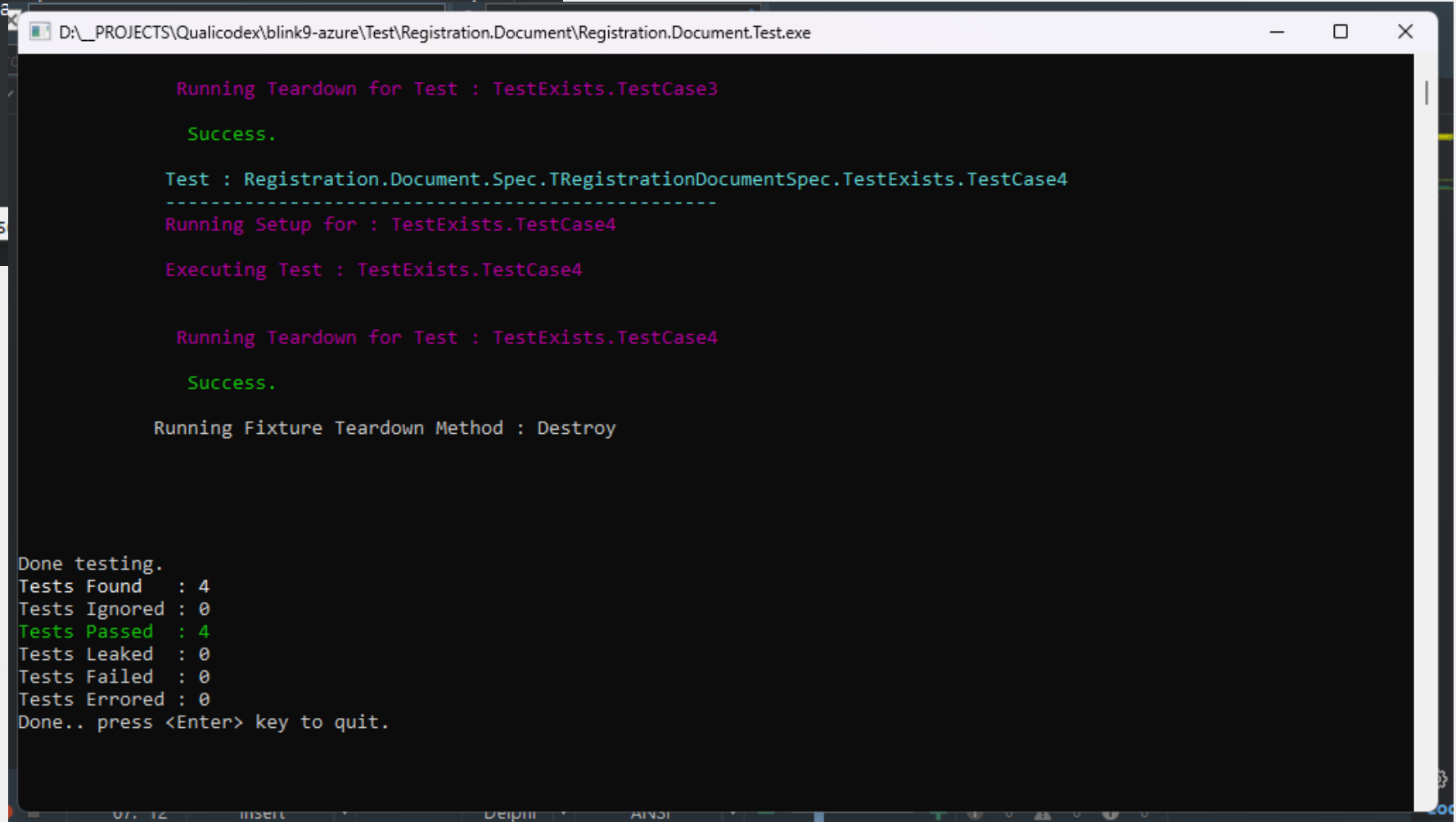
# Testes Unitários



The screenshot shows the TestInsight application window. At the top, there is a toolbar with various icons for test execution and management. Below the toolbar, a table displays the test results. The table has two columns: 'Name' and 'Duration'. The first row is a summary row labeled 'Success (4)' with a duration of '1ms'. The subsequent rows list individual test cases, all of which passed, with durations of '1ms' or '<1ms'. At the bottom of the window, a status bar indicates '4 tests (4 passed, 0 failed, 0 skipped) - more than 30 s'.

Name	Duration
✓ Success (4)	1ms
✓ TRegistrationDocumentSpec.TestExists.TestCase1	1ms
✓ TRegistrationDocumentSpec.TestExists.TestCase2	<1ms
✓ TRegistrationDocumentSpec.TestExists.TestCase3	<1ms
✓ TRegistrationDocumentSpec.TestExists.TestCase4	<1ms

✓ 4 tests (4 passed, 0 failed, 0 skipped) - more than 30 s



The screenshot shows a command prompt window with the title 'D:\\_PROJECTS\Qualicodex\blink9-azure\Test\Registration.Document\Registration.Document.Test.exe'. The output of the test execution is displayed in a monospaced font. The output includes messages for running teardown, success, test execution, setup, and teardown for TestCase3 and TestCase4. It also shows a summary of the test results at the bottom, indicating that all tests passed.

```
Running Teardown for Test : TestExists.TestCase3
Success.

Test : Registration.Document.Spec.TRegistrationDocumentSpec.TestExists.TestCase4
-----
Running Setup for : TestExists.TestCase4
Executing Test : TestExists.TestCase4

Running Teardown for Test : TestExists.TestCase4
Success.

Running Fixture Teardown Method : Destroy

Done testing.
Tests Found : 4
Tests Ignored : 0
Tests Passed : 4
Tests Leaked : 0
Tests Failed : 0
Tests Errored : 0
Done.. press <Enter> key to quit.
```

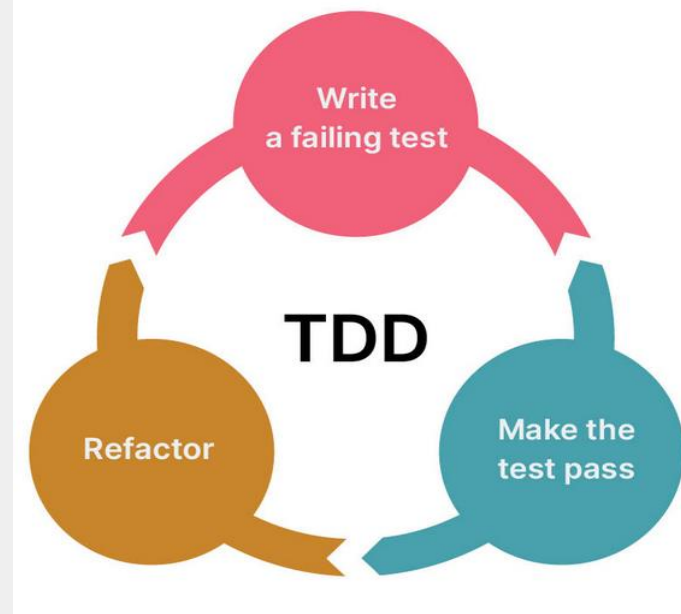
# TDD

## Test-driven development

### *Desenvolvimento Orientado por Testes (TDD)*

*É um processo de desenvolvimento de software onde os desenvolvedores escrevem casos de teste automatizados antes de escrever o código funcional.*

*O principal objetivo do **TDD** é garantir que o software atenda aos requisitos e se comporte conforme o esperado*



**1. Escreva um Teste:** Antes de escrever qualquer código funcional, o desenvolvedor escreve um teste para a próxima funcionalidade a ser implementada. O teste é baseado nos requisitos e garante que a função se comporte conforme o esperado.

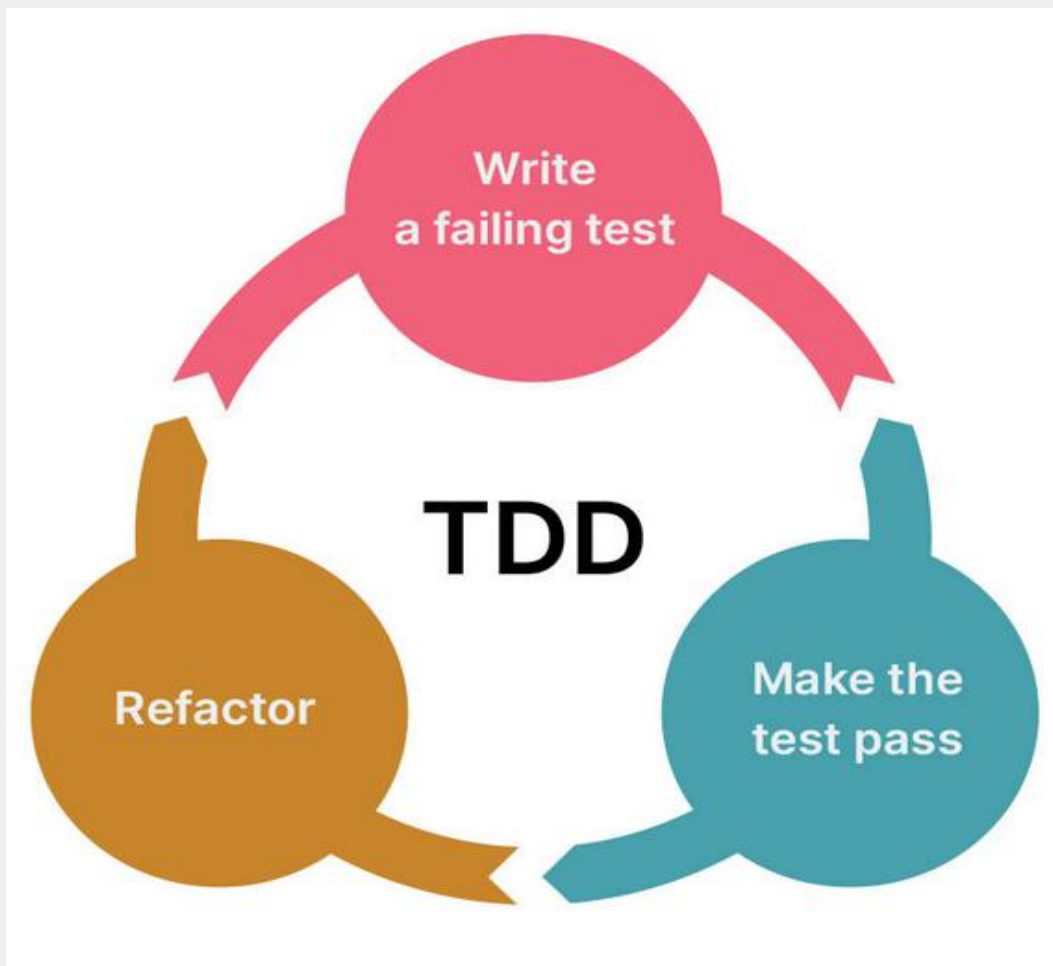
**2. Execute o Teste:** O teste recém-escrito é executado para verificar se ele falha. Este é um passo importante porque valida que o teste está funcionando corretamente e que a funcionalidade ainda não foi implementada.

**3. Escreva o Código:** O desenvolvedor escreve a quantidade mínima de código necessária para que o teste passe. Este código geralmente é muito simples e direto.

**4. Execute os Testes:** A suíte de testes é executada novamente para verificar se todos os testes passam, incluindo o novo. Se o novo teste passar e nenhum outro teste falhar, o código funciona como esperado para o novo requisito.

**5. Refatoração:** O desenvolvedor procura por quaisquer melhorias que possam ser feitas no código. A refatoração tem como objetivo melhorar a estrutura e a manutenibilidade do código sem alterar seu comportamento. Após a refatoração, todos os testes são executados novamente para garantir que o código ainda funcione conforme o esperado.

**6. Repita:** O processo é repetido para a próxima funcionalidade.



## Benefícios do TDD

- **Melhoria na Qualidade do Código:** Ao escrever testes primeiro, os desenvolvedores garantem que seu código seja testável e que eles tenham uma compreensão clara dos requisitos.
- **Refatoração com Confiança:** Como os testes são executados frequentemente, os desenvolvedores podem refatorar o código com confiança, sabendo que qualquer regressão será detectada pelos testes.
- **Documentação:** Os testes servem como uma forma de documentação que descreve como o código deve funcionar.
- **Redução do Tempo de Depuração:** Ao detectar problemas cedo por meio dos testes, os desenvolvedores passam menos tempo depurando.

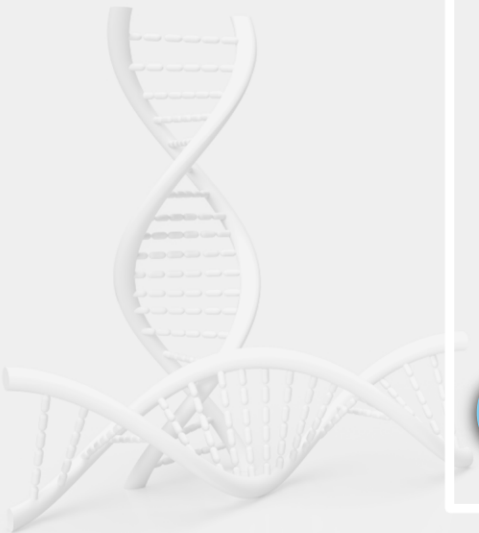


begin

// Let's code

{ ... }

end





*Dica!*



## SmartPointerClass.pas

```
unit SmartPointerClass;
```

```
interface
```

```
uses
```

```
    Generics.Defaults;
```

```
type
```

```
    TSmartPointer<T: class, constructor> = record
```

```
    strict private
```

```
        FValue: T;
```

```
        FFreeTheValue: IInterface;
```

```
        function GetValue: T;
```

```
    private
```

```
        type
```

```
            TFreeTheValue = class (TInterfacedObject)
```

```
            private
```

```
                fObjectToFree: TObject;
```

```
            public
```

```
                constructor Create(anObjectToFree: TObject);
```

```
                destructor Destroy; override;
```

```
            end;
```

```
    public
```

```
        constructor Create(AValue: T); overload;
```

```
        procedure Create; overload;
```

```
        class operator Implicit(AValue: T): TSmartPointer<T>;
```

```
        class operator Implicit(smart: TSmartPointer <T>): T;
```

```
        property Value: T read GetValue;
```

```
    end;
```

```
procedure TFormSmartPointers.btnImplicitCreateClick(Sender: TObject);
var
    smartP: TSmartPointer<TStringList>;
begin
    // smartP.Create;
    smartP.Value.Add('foo');
    Log ('Count: ' + IntToStr (smartP.Value.Count));
end;
```

# Smart Pointers

```
implementation
```

```
{ TSmartPointer<T> }
```

```
constructor TSmartPointer<T>.Create(AValue: T);
```

```
begin
```

```
    FValue := AValue;
```

```
    FFreeTheValue := TFreeTheValue.Create(FValue);
```

```
end;
```

```
procedure TSmartPointer<T>.Create;
```

```
begin
```

```
    Create (T.Create);
```

```
end;
```

```
function TSmartPointer<T>.GetValue: T;
```

```
begin
```

```
    if not Assigned(FFreeTheValue) then
```

```
        Create;
```

```
    Result := FValue;
```

```
end;
```

```
class operator TSmartPointer<T>.Implicit(smart: TSmartPointer<T>): T;
```

```
begin
```

```
    Result := Smart.Value;
```

```
end;
```

```
class operator TSmartPointer<T>.Implicit(AValue: T): TSmartPointer<T>;
```

```
begin
```

```
    Result := TSmartPointer<T>.Create(AValue);
```

```
end;
```

```
{ TSmartPointer<T>.TFreeTheValue }
```

```
constructor TSmartPointer<T>.TFreeTheValue.Create(anObjectToFree: TObject);
```

```
begin
```

```
    fObjectToFree := anObjectToFree;
```

```
end;
```

```
destructor TSmartPointer<T>.TFreeTheValue.Destroy;
```

```
begin
```

```
    fObjectToFree.Free;
```

```
    inherited;
```

```
end;
```

tmssoftware.com



develop faster

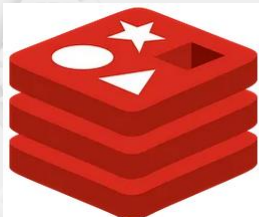


# Frameworks usados

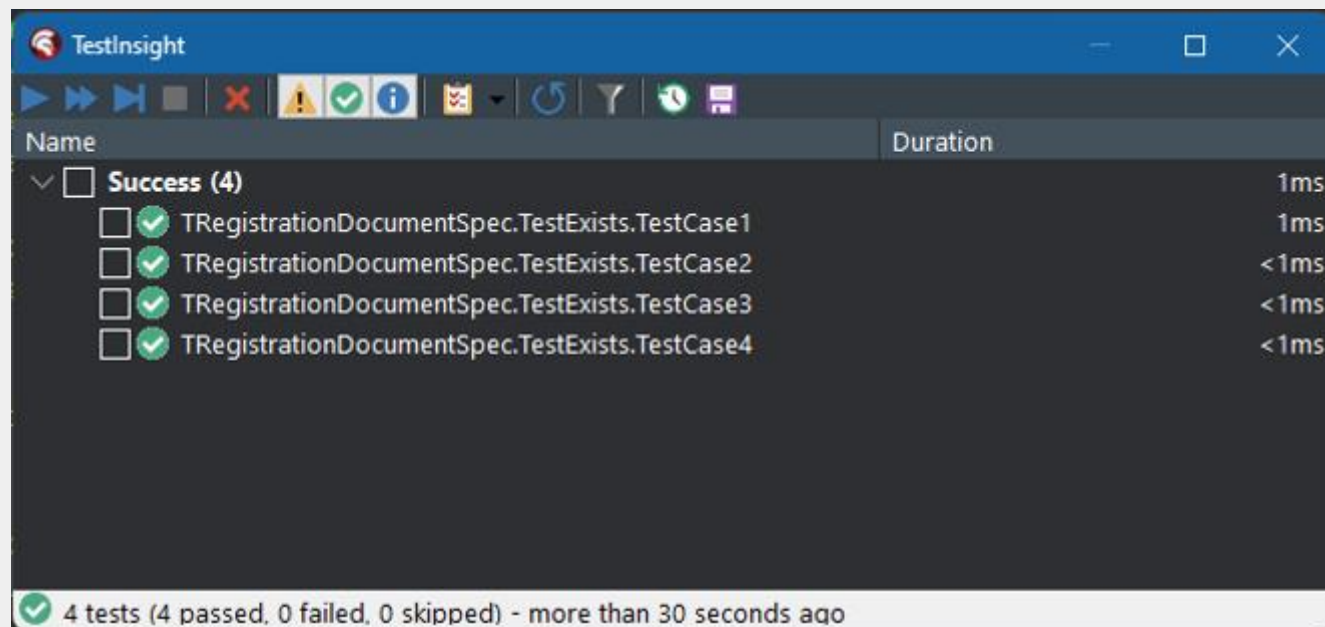
FNC

TMS FNC Component Studio

Supported operating systems/browsers/IDEs:



redis



## **FNC**

<https://www.tmssoftware.com/site/fnc-products.asp>

## **WebCore**

<https://www.tmssoftware.com/site/tmswebcore.asp>

## **TMS BIZ**

<https://www.tmssoftware.com/site/tmsbiz.asp>

## **Redis**

<https://redis.io/>

## **Testinsight**

<https://bitbucket.org/sglienke/testinsight/wiki/Home>

## **Test-driven development**

[https://en.wikipedia.org/wiki/Test-driven\\_development](https://en.wikipedia.org/wiki/Test-driven_development)

## **Smart Pointers – Marco Cantù**

<https://www.marcocantu.com/code/dh2009/SmartPointers.htm>

# Links Úteis

# Embarcadero Conference 2024


Inovação faz parte do nosso DNA!





Quer me ver na  
**#ECON25?**  
Acesse o QRCode  
e avalie minha palestra!




**Ivan Souza**

 [@ivansouza2012](https://www.instagram.com/ivansouza2012)

 [ivanlsouza](https://www.linkedin.com/in/ivansouza)

 [ilsouza@gmail.com](mailto:ilsouza@gmail.com)

 [\(43\) 99102 5152](tel:(43)991025152)

