# ESPECIFICAÇÃO E COMPILAÇÃO DE UMA LINGUAGEM O.O. PARA A PLATAFORMA .NET

Gustavo Z. Leyendecker

Joyce Martins

# Conteúdo

- introdução
  - objetivos do trabalho
  - porque a plataforma .NET
- revisão bibliográfica
  - linguagens de programação
  - plataforma .NET
  - compiladores
- características da linguagem
- desenvolvimento do compilador
- resultados e conclusão
  - extensões
- apresentação do protótipo

# Introdução

- objetivos
  - especificar uma linguagem de programação
  - implementar um compilador (4 módulos)
  - novas funcionalidades

# Introdução

- plataforma .NET
  - multi-linguagens
  - Microsoft Intermediate Language (MSIL)

# Revisão bibliográfica

- linguagens de programação
  - modelo imperativo X modelo declarativo
  - características: legibilidade, redigibilidade, confiabilidade, eficiência, facilidade de aprendizado, ortogonalidade, reusabilidade, modificabilidade e portabilidade

# Revisão bibliográfica

- linguagens de programação
  - modelo imperativo X modelo declarativo
  - características: **legibilidade**, redigibilidade, confiabilidade, eficiência, facilidade de aprendizado, ortogonalidade, reusabilidade, modificabilidade e portabilidade

```
do {
   if (Something) {
      // Do something
      goto endLoop;
   }
} while(someCondition);
endLoop:
   //...
```

```
do {
   if (Something) {
      // Do something
      break;
   }
} while(someCondition);
   //...
```

# Revisão bibliográfica

- linguagens de programação
  - modelo imperativo X modelo declarativo
  - características: legibilidade, **redigibilidade**, confiabilidade, eficiência, facilidade de aprendizado, ortogonalidade, reusabilidade, modificabilidade e portabilidade

```
var
   c: integer;
begin
   for c := 1 to 10 do
   begin
      {...}
   end
end.
```

```
for (int c = 1; c <= 10; i++) {
   //...
}
```

# Revisão bibliográfica

- linguagens de programação
  - modelo imperativo X modelo declarativo
  - características: legibilidade, redigibilidade, **confiabilidade**, eficiência, facilidade de aprendizado, ortogonalidade, reusabilidade, modificabilidade e portabilidade

# Revisão bibliográfica

- linguagens de programação
  - modelo imperativo X modelo declarativo
  - características: legibilidade, redigibilidade, confiabilidade, **eficiência**, facilidade de aprendizado, ortogonalidade, reusabilidade, modificabilidade e portabilidade

# Revisão bibliográfica

- linguagens de programação
  - modelo imperativo X modelo declarativo
  - características: legibilidade, redigibilidade, confiabilidade, eficiência, **facilidade de aprendizado**, ortogonalidade, reusabilidade, modificabilidade e portabilidade

# Revisão bibliográfica

- linguagens de programação
  - modelo imperativo X modelo declarativo
  - características: legibilidade, redigibilidade, confiabilidade, eficiência, facilidade de aprendizado, **ortogonalidade**, reusabilidade, modificabilidade e portabilidade

```
//Java
int x, y = 2, z = 3;
byte a, b = 2, c = 3;
x = y + z;
a = b + c;
```

# Revisão bibliográfica

- linguagens de programação
  - modelo imperativo X modelo declarativo
  - características: legibilidade, redigibilidade, confiabilidade, eficiência, facilidade de aprendizado, ortogonalidade, **reusabilidade**, modificabilidade e portabilidade

# Revisão bibliográfica

- linguagens de programação
  - modelo imperativo X modelo declarativo
  - características: legibilidade, redigibilidade, confiabilidade, eficiência, facilidade de aprendizado, ortogonalidade, reusabilidade, **modificabilidade** e portabilidade

# Revisão bibliográfica

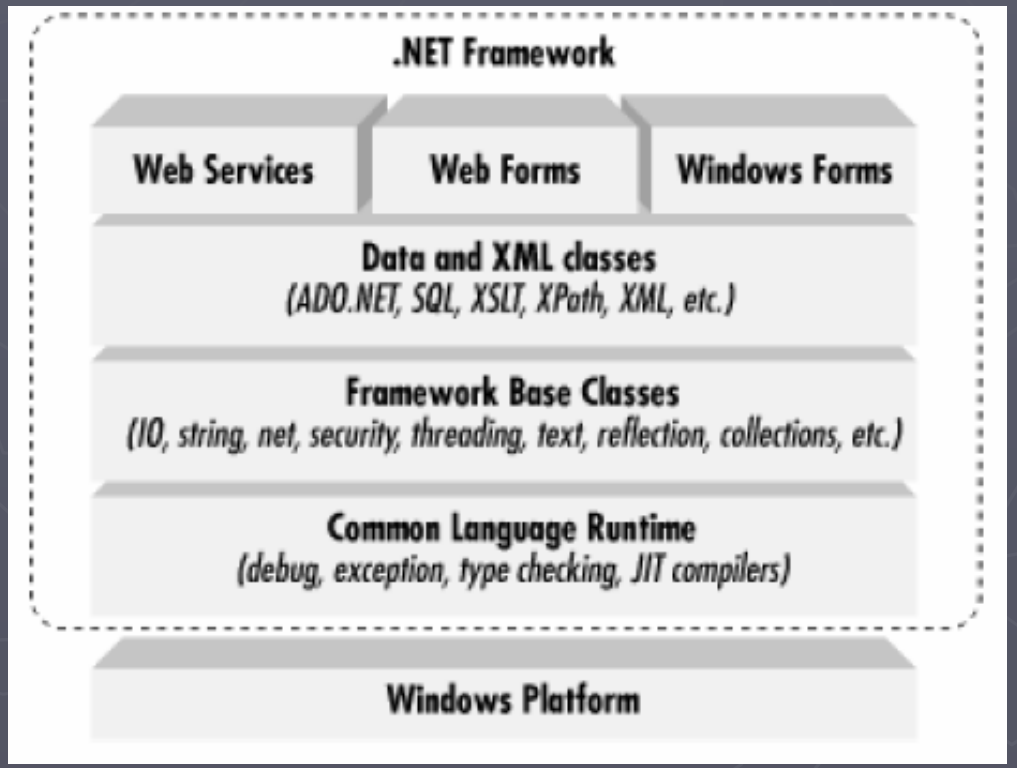
- linguagens de programação
  - modelo imperativo X modelo declarativo
  - características: legibilidade, redigibilidade, confiabilidade, eficiência, facilidade de aprendizado, ortogonalidade, reusabilidade, modificabilidade e **portabilidade**

# Revisão bibliográfica

- plataforma .NET
  - *Common Language Specification* (CLS)
  - *Common Type System* (CTS)
  - *Microsoft Intermediate Language* (MSIL)
  - *Assembly* .NET
  - *Common Language Runtime* (CLR)
  - *Framework* .NET

# Revisão bibliográfica

- *Framework* .NET




Fonte: Thai e Lam (2001, p. 12)

# Revisão bibliográfica

- compiladores
  - largos ou estreitos
  - controle de fluxo
  - módulos

# Revisão bibliográfica

- compiladores
  - largos ou estreitos
  - controle de fluxo
  - módulos
    - *front-end* (léxico, sintático, semântico e código intermediário)
    - *back-end* (outros)

# Características da linguagem

- herança simples
- sobrecarga de métodos
- *case-sensitive*
- semelhante ao C (fluxo), Java e C# (orientação a objetos)

# Características da linguagem

- produtiva para o desenvolvimento da camada de negócio de uma aplicação

```
if (valor.CompareTo(new DateTime(2005, 08, 06)) < 0)
{
   //Este bloco é executado apenas quando a data contida
   //na variável valor for menor que 06/08/2005.
}
```

```
if (valor < datetime'06/08/2005')
{
   //Este bloco é executado apenas quando a data contida
   //na variável valor for menor que 06/08/2005.
}
```

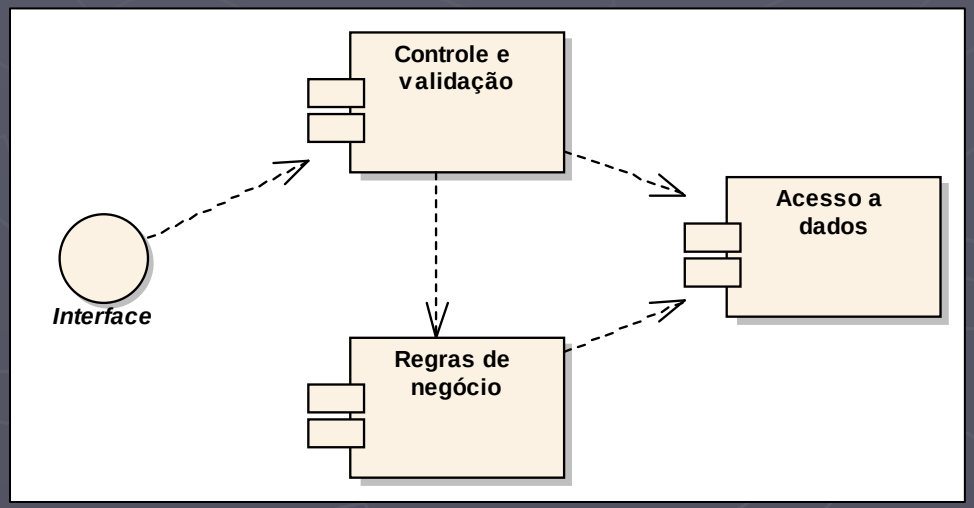
# Características da linguagem

- produtiva para o desenvolvimento da camada de negócio de uma aplicação

```
if (!(cpf like "???.???.???/??"))
{
   Console.WriteLine(“Formato do CPF inválido");
}
```

# Características da linguagem

- uso de rotinas de outras LPs .NET
- outras linguagens .NET podem usar as rotinas escritas na LP proposta

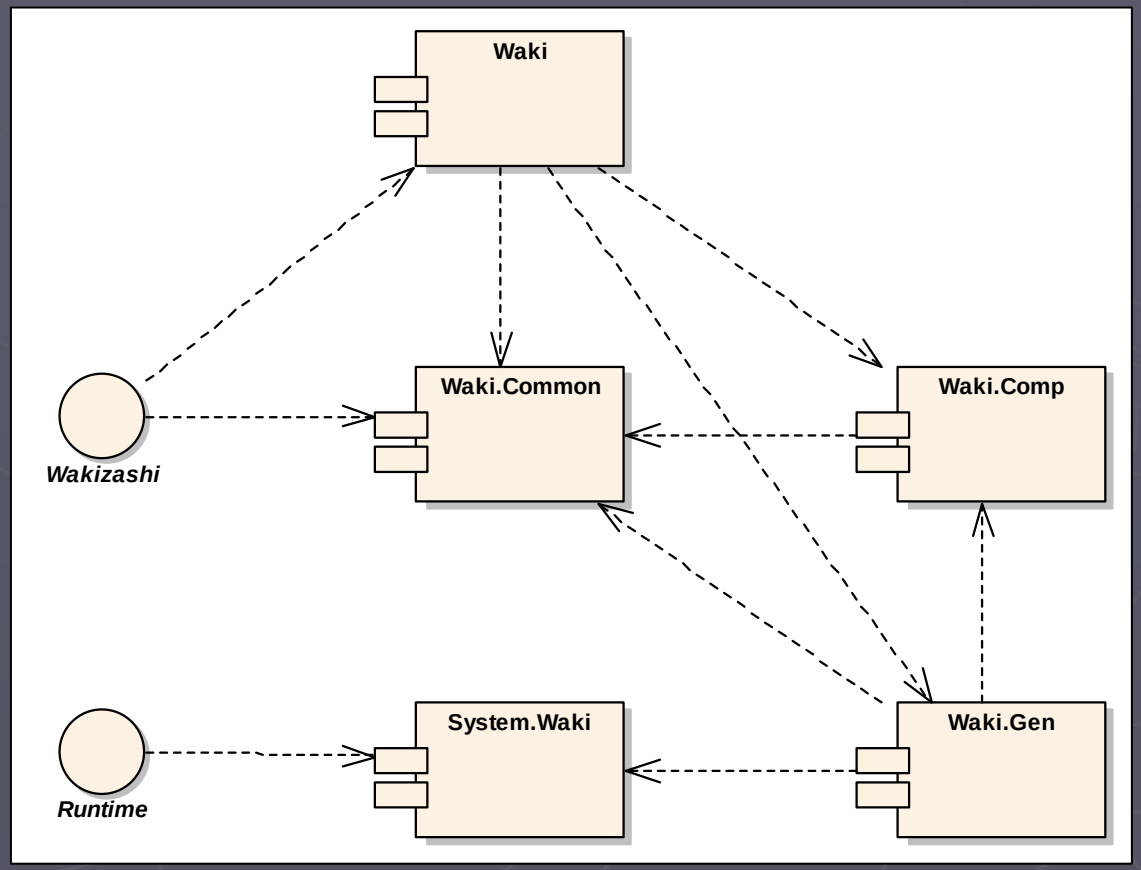


# Desenvolvimento do compilador

- requisitos
  - reportar erros
  - gerar código MSIL
  - gerar *assembly* .NET
  - ser implementado em C#

# Desenvolvimento do compilador

- projetos (.NET)
  - Wakizashi
  - Waki
  - Waki.Common
  - Waki.Comp
  - Waki.Gen
  - System.Waki

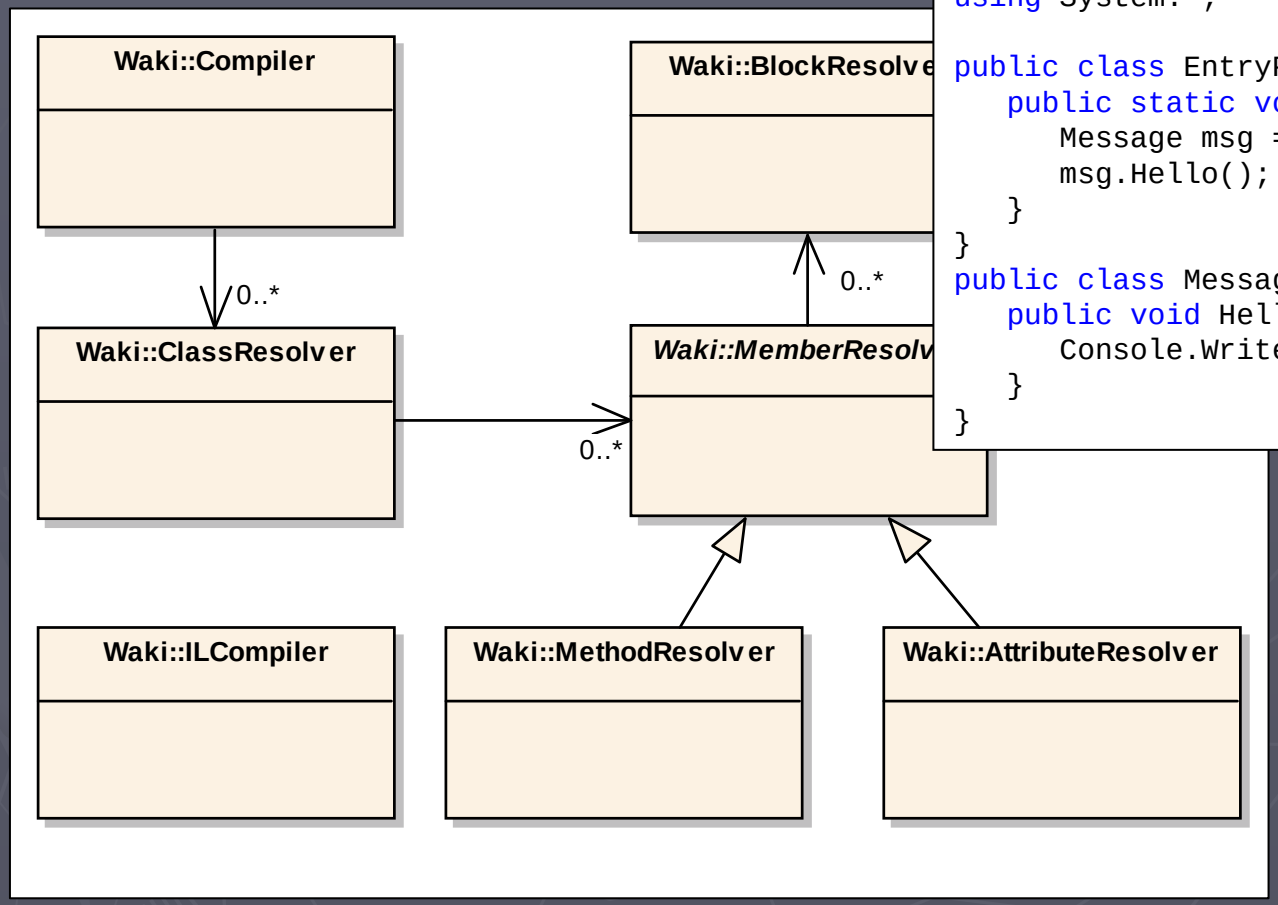


# Desenvolvimento do compilador

- JavaCCCS
  - léxico e sintático
  - gramática LL
  - EBNF

```
CommandNode whileCmd() : {
  WhileCmd ret = new WhileCmd();
} {
  "while" "(" ret.Cond = expression() ")"
  ret.Block = blockCmd()
  {return ret;}
}
```

# Desenvolvimento do compilador




```
namespace org.furb.tcc;
using System.*;

public class EntryPoint {
   public static void main(String[] args) {
      Message msg = new Message();
      msg.Hello();
   }
}
public class Message {
   public void Hello() {
      Console.WriteLine("Hello World!");
   }
}
```

# Desenvolvimento do compilador

- gerador de código

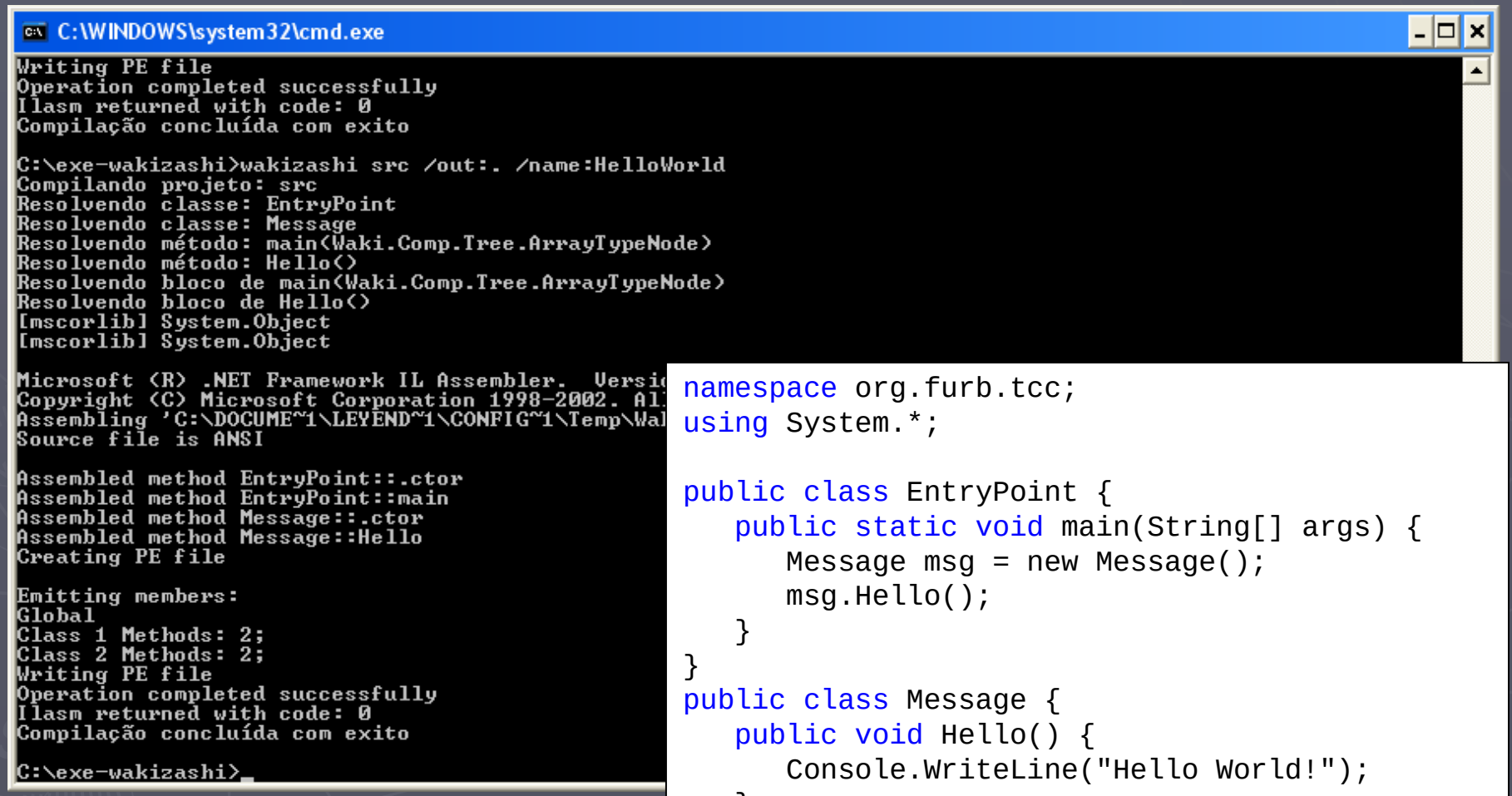
```
namespace org.furb.tcc;
using System.*;

public class EntryPoint {
   public static void main(String[] args) {
      Message msg = new Message();
      msg.Hello();
   }
}
public class Message {
   public void Hello() {
      Console.WriteLine("Hello World!");
   }
}
```

**MSIL**

```
.namespace org.furb.tcc {
   .class public ansi auto EntryPoint extends [mscorlib] Sy
      .method public void .ctor() cil managed {
         ret
      }
      .method public static hidebysig void main(string[] ar
         .entrypoint
         .locals init(class org.furb.tcc.Message)
         newobj instance void org.furb.tcc.Message::.ctor()
         stloc.s 0
         ldloc.s 0
         callvirt instance void org.furb.tcc.Message::Hello()
         ret
   } }
   .class public ansi auto Message extends [mscorlib] System.Object {
      .method public void .ctor() cil managed {
         ret
      }
      .method public final hidebysig void Hello() cil managed {
         ldstr "Hello World!"
         call void [mscorlib] System.Console::WriteLine(string)
         ret
} } }
```

# Desenvolvimento do compilador



# Resultados e conclusão

- recursos básicos de OO
- não implementa *interfaces*, *enums* e *structs*
- ganho de produtividade (*like* e *datetime*)

# Resultados e conclusão

- características desejadas

| | |
|---|---|
| legibilidade | bom |
| redigibilidade | bom |
| confiabilidade | regular |
| eficiência | independe da linguagem |
| aprendizado | regular |
| ortogonalidade | bom |
| reusabilidade | bom |
| modificabilidade | bom |
| portabilidade | independe da linguagem |

# Resultados e conclusão

- C# x Wakizashi

| | |
|---|---|
| Rápida curva de aprendizagem | Produtividade p/ camada de negócio |
| Recuperação de erros semânticos | Um erro por compilação |
| Expressões primitivas | *datetime*, *timespan*, *like* |
| *Interfaces*, *enums* e *structs* | Apenas classes |
| Suporta totalmente CTS | Suporta parcialmente CTS |
| 3 anos no mercado | - |

# Extensões

- linguagem
  - suportar herança múltipla
  - implementar instruções para controle de concorrência
  - implementar tratamento de exceção
- compilador
  - implementar recuperação de erros
  - criar IDE
  - permitir depuração

# Apresentação do protótipo