Manual Extensibilidade

PRIMAVERA Fiscal Reporting
v8.0

Versão 1.0

Março 2016

(última atualização em 23.03.2016)

Índice

Introdução

A extensibilidade do PRIMAVERA Fiscal Reporting (PFR) permite elaborar fórmulas personalizadas, de acordo com as necessidades de cada utilizador e/ou organização. Esta funcionalidade introduz a possibilidade de criar novas fórmulas, dando assim ao técnico integrador maior liberdade e agilidade em desenvolvimentos personalizados.

Permite igualmente realizar consultas à base de dados da empresa em edição no PRIMAVERA Fiscal Reporting, realizar a integração com qualquer tipo de fonte de dados, com o motor ERP PRIMAVERA ou até mesmo com uma web API.

Com esta funcionalidade é ainda possível a sobreposição de fórmulas de sistema, evitando assim a modificação o cenário Standard em fórmulas já existentes no modelo.

Este manual explica a arquitetura montada na funcionalidade de extensibilidade e fornece um conjunto de exemplos de utilização.

A funcionalidade de extensibilidade está apenas disponível com licenciamento **Manager**.

Requisitos mínimos de utilização

Para poder usufruir da extensibilidade disponibilizada é necessário ter uma instalação com:

- PRIMAVERA ERP v9.0 (ou superiores);
- PRIMAVERA Fiscal Reporting v8.0 (plataforma);
- PRIMAVERA Connectors v8.0;
- PRIMAVERA Fiscal Reporting Extensibility (disponível na Visual Studio Gallery);
- Visual Studio 2012/2013/2015.

Antes de inicializar a criação de novas fórmulas deverá garantir-se que todo o ambiente está operacional, nomeadamente que o "Conector ERP v9.0" se encontra corretamente configurado e ativado (em **PRIMAVERA Fiscal Reporting** | **Administrador** | **Conectores**).

Licenciamento

Esta funcionalidade está apenas disponível para utilizadores com licença **Manager**.

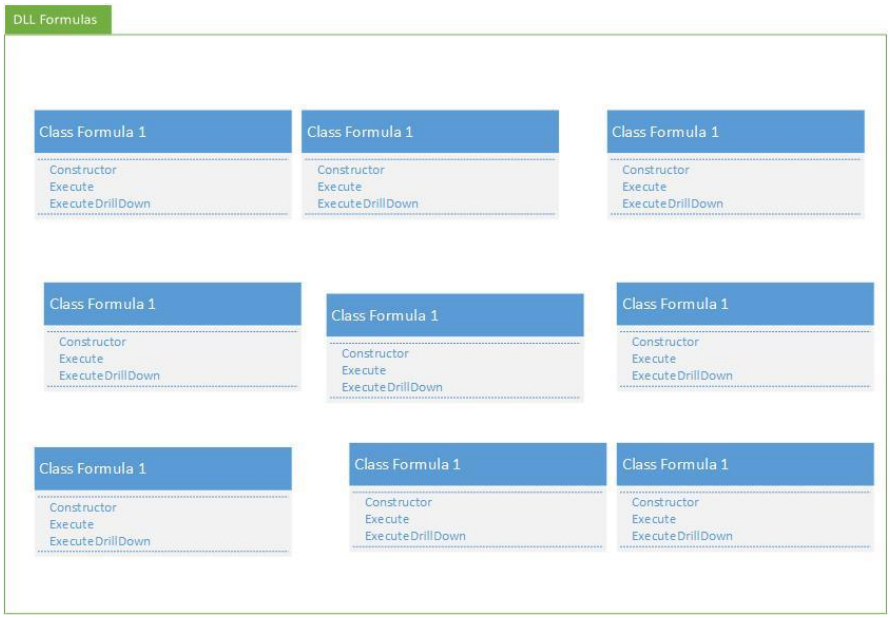
Os utilizadores sem licenciamento Manager do PRIMAVERA Fiscal Reporting não terão acesso a eventuais fórmulas de extensibilidade existentes.

No caso de algum cenário conter uma fórmula de extensibilidade, para um utilizador sem licença Manager associada, poderão observar-se os seguintes comportamentos:

- Se não existir nenhuma fórmula de sistema com nome igual, é obtido o seguinte erro ao processar o modelo e/ou validar a fórmula: "A função <nome função> não é válida.";
- Se existir uma fórmula de sistema com o mesmo nome da fórmula implementada, é executada a fórmula de sistema.

Arquitetura de Extensibilidade

A imagem seguinte exemplifica o modo de disponibilização de novas fórmulas na plataforma PFR, sendo representada a estrutura de uma *Assembly* com várias classes de fórmulas:

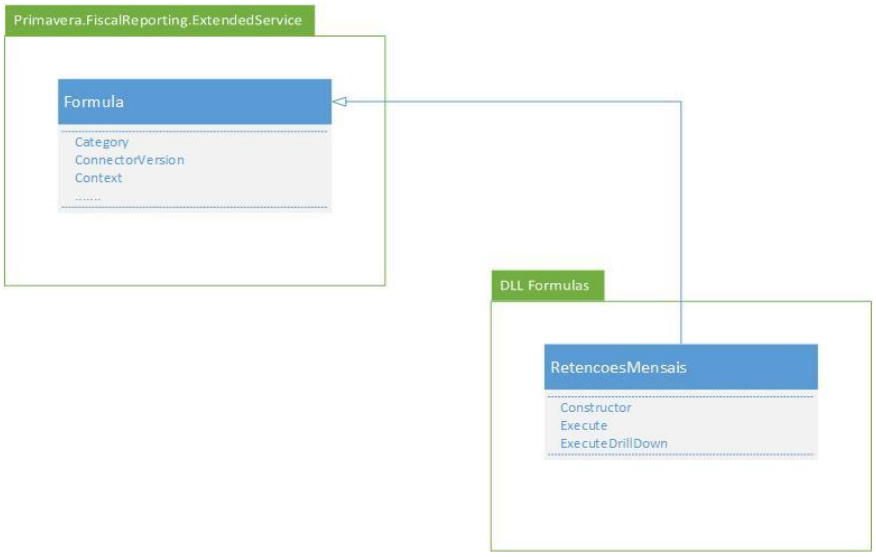


A criação de novas *Assemblies* de fórmulas será feita de acordo com a arquitetura interna demonstrada, sendo que no contexto geral uma *assembly* poderá conter várias fórmulas e deverá, obrigatoriamente, referenciar a *assembly Primavera.FiscalReports.ExtendedService*, que se encontra na pasta de instalação do produto:

Sistemas operativos x64 - C:\Program Files\PRIMAVERA\PFR800\Apl

Sistemas operativos x86 - C:\Program Files (x86)\PRIMAVERA\PFR800\Apl

Cada classe fórmula deverá herdar a *abstract* classe *Formula* que está na *assembly Primavera.FiscalReports.ExtendedService*. Deste modo, a arquitetura geral será idêntica à imagem seguinte:



Classe *Formula*

Antes de iniciar a implementação de uma nova fórmula deverá ter-se em conta a estrutura de propriedades apresentada de seguida e que devem ser definidas no construtor da fórmula. A execução efetiva da fórmula principal é assegurada pela implementação (*override)* do método *Execute*, enquanto que a execução do *drill down* da fórmula, se aplicável, deverá ser assegurada pela implementação (*override)* do método *ExecuteDrillDown.*

Propriedades

Cada nova fórmula deverá implementar a herança da classe *Formula* presente na *assembly* de extensibilidade (*Primavera.FiscalReports.ExtendedService*) sendo que as propriedades disponibilizadas para a classe base são as seguintes:

Nome	**Descrição**	**Tipo**
Category	Corresponde à categoria à qual a fórmula ficará associada. Encontra-se no separador "Assistente de Campos" quando o modelo é editado. Caso não seja definido o nome da categoria, é assumida por omissão a categoria "Extensibilidade".	String
Description	Descrição da fórmula. É uma informação que se encontra no separador "Assistente de Campos".	String
Name	Nome da fórmula. Este deve ser único no conjunto de todas as fórmulas e não deverão existir, no conjunto de *assemblies* de extensibilidade, fórmulas distintas com o mesmo nome. Os caracteres especiais não são permitidos neste campo. Exemplo: FormulaTeste	String
Example	Exemplo de utilização da fórmula com dados demonstrativos. No caso do tratamento de texto do C# ter em atenção que caso sejam inseridos parâmetros alfanuméricos deverão ser inseridos na forma do exemplo. Exemplo: FormulaTeste(\"parametroTexto\")	String
Syntax	Texto com a sintaxe de utilização genérica da fórmula com os parâmetros de entrada. É importante que o nome dado nos parâmetros seja igual ao definido na propriedade "Parameters", para que a construção da fórmula seja feita corretamente no "Assistente de Campos". Exemplo: FormulaTeste(parametroTexto)	String
DrillDown	Indica se o *dril ldown* da fórmula está disponível ou não. Esta operação apenas está disponível no licenciamento Manager.	Boolean
FormulaType	Tipo de fórmula para tratamento interno da aplicação: 0 – Field; 1 – Drill Down; 2 – Table.	FormulaType
ConnectorVersion	Versão do conector na qual a fórmula será executada.	ConnectorVersion

Context	Informações relevantes sobre o contexto em que a fórmula foi executada. Este contexto é apenas de leitura (*read-only*).	ExtendedFormulaContext
Parameters	Lista onde podem ser definidos os parâmetros da fórmula.	List<FormulaParameter>
DrillDownKeys	Devem ser indicados quais os campos que são passados como chaves no método *ExecuteDrillDown*, sendo por norma utilizados no *drill down* de linha nas grelhas.	List<FormulaDrillDownKeys>

Classes derivadas

ExtendedFormulaContext

Informações de contexto relacionadas com a execução da fórmula.

Propriedade	**Descrição**	**Tipo**
ExerciseYear	Ano de exercício no processamento da fórmula.	Integer
FieldID	Identificador do campo onde a fórmula foi executada.	String
ModelInfo	Informações de contexto do modelo onde serão executadas as fórmulas.	ModelInfo
EnterpriseInfo	Informação sobre a empresa em edição e sobre a qual serão executadas as fórmulas.	EnterpriseInfo

ModelInfo

Informação relativa ao modelo.

Propriedade	**Descrição**	**Tipo**
Culture	Cultura utilizada no modelo processado. Especialmente útil para a formatação de valores numéricos e datas.	CultureInfo

EnterpriseInfo

Informação relativa à empresa.

Propriedade	**Descrição**	**Tipo**
Name	Nome da empresa.	String
DatabaseName	Nome da base de dados da empresa no ERP PRIMAVERA.	String
BasePostingType	Tipo de lançamento por omissão definido na empresa.	String

Currency	A moeda da empresa.	String

FormulaParameter

Descrição do parâmetro de entrada na fórmula.

Propriedade	**Descrição**	**Tipo**
Name	Nome do parâmetro.	String
Type	Tipo de dados.	ParameterDataType
InputValue	Valor de entrada na fórmula após a sua execução.	String

FormulaDrillDownKey

Descrição das chaves de linha para o *drill down* de grelhas.

Propriedade	**Descrição**	**Tipo**
ColumnIndex	Índice da coluna à qual será retornada a chave (de 1 a N).	Integer
InputValue	Valor da chave de linha após a execução do *drill down* de linha.	String

Enumeradores

FormulaType

Define qual o tipo de fórmula.

Nome	**Descrição**
Field	Fórmula do tipo campo.
Table	Fórmula do tipo tabela.

ConnectorVersion

Versão do conector sobre onde a fórmula foi executada.

Nome	**Descrição**
Pricon900	PRIMAVERA ERP V9 – PRICON900

ParameterDataType

Tipo de dados que podem ser utilizados nos parâmetros de entradas nas fórmulas.

Nome	**Descrição**
Boolean	Campo booleano de valor "true" ou "false".
String	Campo Alfanumérico.
Integer	Campo de valor inteiro.

Métodos

Método	**Descrição**
Execute	Método para cálculo do valor pretendido para a fórmula.
ExecuteDrillDown	Método para cálculo do *drill down* de uma fórmula.

Eventos

Evento	**Descrição**
DataTableRowFillEventHandler	Neste evento é possível alterar o respetivo valor dos registos antes de estes serem serializados para a aplicação. Evento é invocado por cada linha no preenchimento de uma DataTable.

Operações SQL

A classe *SQLOperations* contida na *Primavera.FiscalReports.ExtendedService*, permite realizar operações sobre a base de dados da empresa do ERP PRIMAVERA, associada ao cenário/modelo em edição.

Nesta classe, por motivos de segurança, existe uma série de restrições em que não é permitido efetuar as seguintes operações:

- DELETE ;
- UPDATE;
- INSERT;
- ALTER;
- DROP;
- EXEC / EXECUTE;
- SELECT INTO.

Nas operações anteriores aplicam-se as seguintes exceções:

- Tabelas temporárias;

- Variáveis do tipo tabela.

Na criação de *Stored Procedures* deve obrigatoriamente ser adicionado o prefixo "**PFREXT_**", tanto no nome da *Stored Procedure* como no conteúdo que é passado para a criação da mesma.

Este prefixo evita que sejam sobrepostas indevidamente *Stored Procedures* de sistema.

Métodos

CreateProcedure

Este método permite a criação de uma *Stored Procedure*. A invocação do método pressupõe a realização de uma operação interna *DROP* que assegura a sobreposição de uma versão anterior da *Stored Procedure* com o mesmo nome.

Parâmetro	**Descrição**	**Opcional?**
ProcedureName	Nome da *Stored Procedure* a criar.	Não
ProcedureContent	Conteúdo da *Stored Procedure* a criar.	Não

ExecuteSQLFormula

Este método permite executar um comando SQL na base dados da empresa em edição, retornando apenas um valor único. Deve ser aplicado em campos de texto ou numéricos.

Parâmetro	**Descrição**	**Opcional?**
QueryString	Query SQL a ser efetuada.	Não

ExecuteSQLTable

Este método permite executar um comando SQL na base dados da empresa em edição para obter um conjunto de dados para uma tabela, sendo que retorna um objeto do tipo *DataTable.*

Parâmetro	**Descrição**	**Opcional?**
QueryString	Query SQL a ser efetuada.	Não

ExecuteSQLProcedureFormula

Este método permite executar uma *Stored Procedure*, previamente criada pelo técnico integrador através do método *CreateProcedure*, retornando neste caso apenas um valor único.
Deve ser aplicado em campos de texto ou numéricos.

Parâmetro	**Descrição**	**Opcional?**
Name	Nome da *Procedure* a ser executada.	Não
Parameters	Parâmetros de entrada.	Não (contudo pode ser inserido um *array* vazio)
DropProcedureAfterExecute	Remove a *Procedure* após a sua execução.	Sim

ExecuteSQLProcedureTable

Este método permite executar uma *Stored Procedure*, previamente criada pelo técnico integrador através do método *CreateProcedure*, para obter um conjunto de dados para uma tabela, sendo que retorna um objeto do tipo *DataTable.*

Parâmetro	**Descrição**	**Opcional?**
Name	Nome da *Procedure* a ser executada.	Não
Parameters	Parâmetros de entrada.	Não (contudo pode ser inserido um *Array* vazio)
DropProcedureAfterExecute	Remove a *Procedure* após a sua execução.	Sim

Exemplos de demonstração

Configuração do ambiente de desenvolvimento

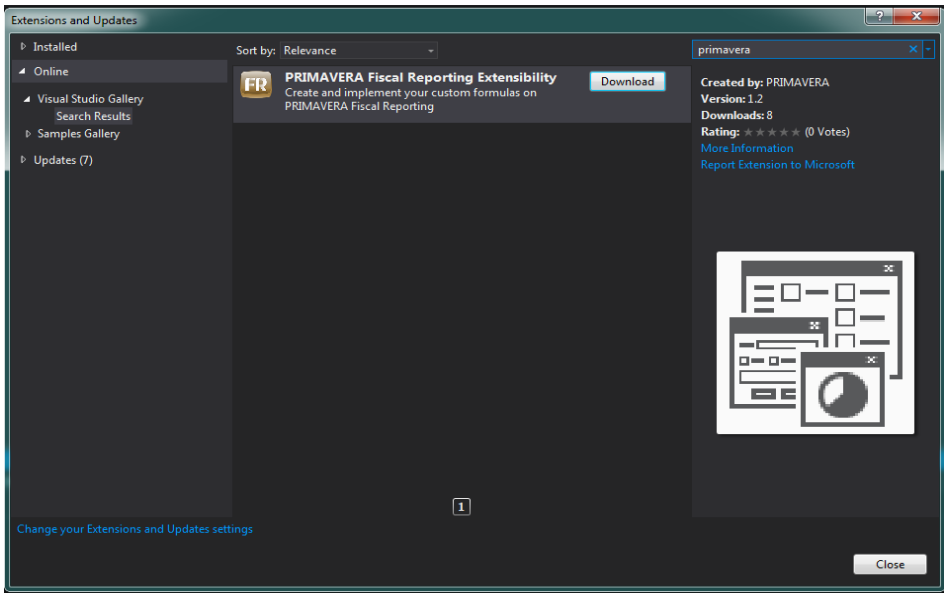
Associado à funcionalidade de extensibilidade, e para facilitar o trabalho do técnico integrador, está disponível para *download* na *Visual Studio Gallery* um utilitário que auxilia no processo de criação de fórmulas.

Este utilitário permite a criação de um projeto funcional com três fórmulas de exemplo sendo que, caso tenha instalado o PFR nas pastas por defeito, a referência obrigatória para a *assembly* *Primavera.FiscalReports.ExtendedService* é automaticamente adicionada, bem como o caminho para onde será compilada (build) *a assembly* para a pasta de fórmulas da instância de utilizador: *<Pasta_Instalação PFR>*\Conectors\Formulas\Extensibility\Default.

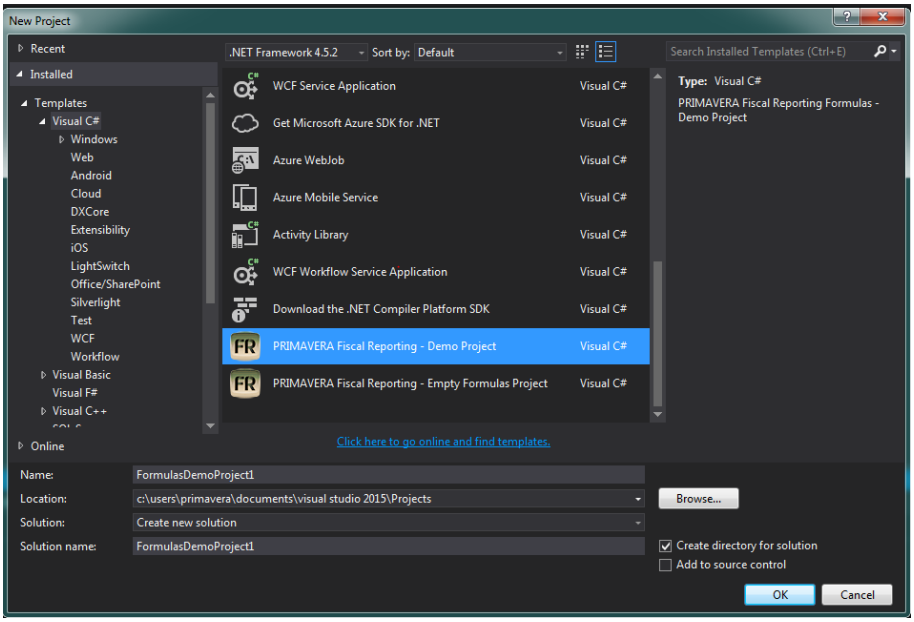
É também possível adicionar novas classes de fórmulas.

De seguida são exemplificados os passos de configuração do ambiente de desenvolvimento:

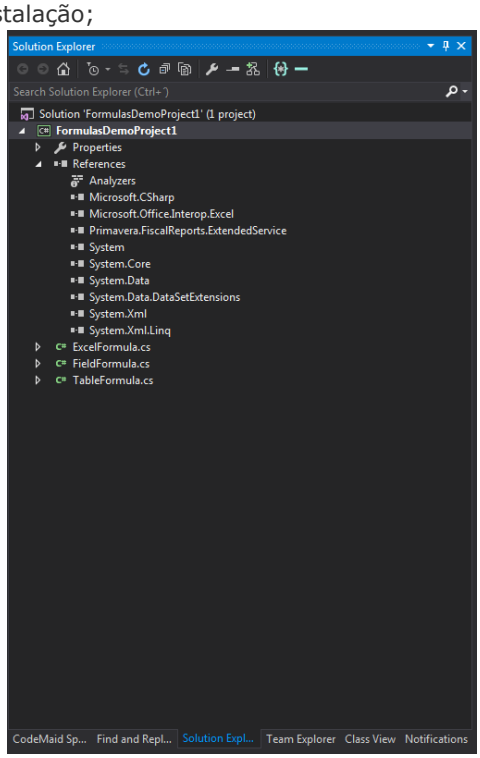
1. Após garantidos os requisitos mínimos, iniciar a instalação do *add-on* "PRIMAVERA Fiscal Reporting Extensibility" existente na Visual Studio Gallery;



2. Após a instalação do VSIX anterior, iniciar a criação de um projeto de demonstração através dos seguintes passos: File | New | Project;



3. Após criar um projeto de demonstração é verificado se o sistema tem o PFR instalado, sendo a referência à *assembly Primavera.FiscalReporting.ExtendedService* direcionada para a pasta de instalação;



4. Após a criação do projeto demo, compilar (build) a *solution* criada. Será disponibilizada a *assembly* na pasta de instalação do PFR no caminho: *<pasta_instalação_PFR>*\Connectors/Formulas/Extensibility/Default.

   A pasta "Default" significa que as fórmulas de extensibilidade estão disponíveis para a instância Default. Caso o nome da instância seja diferente, basta alterar o caminho utilizado e executar novamente a compilação (build).

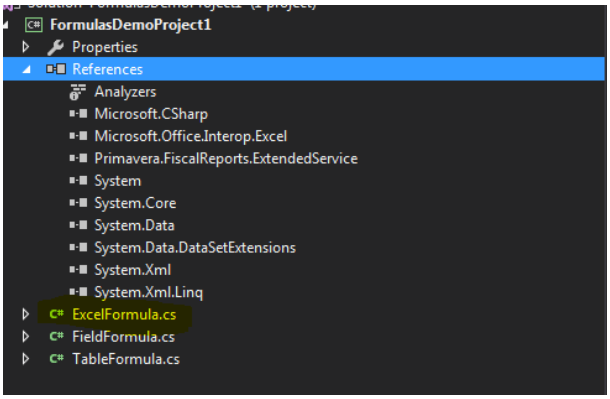
   Para definir outro caminho para onde se pretenda compilar (*build*) o projeto, basta alterar o seguinte campo na barra superior do Visual Studio, em Project | Properties | Build e modificar o campo *Output* para o caminho pretendido.

Exemplo fórmula de Excel

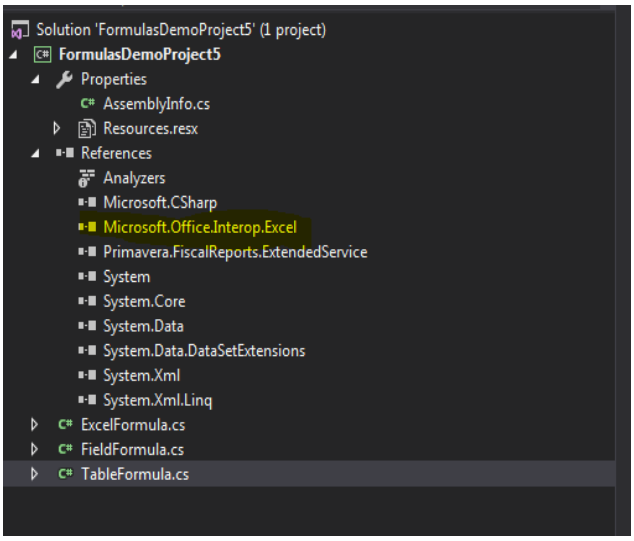
Neste exemplo o objetivo é criar uma fórmula que permita a ligação com o Microsoft Excel. Esta fórmula será utilizada para ligar ao relatório distribuído pela aplicação PRIMAVERA Office Extensions e recolher os dados.

Para criar a fórmula referida, realizar os seguintes passos:

1. No *Visual Studio* abrir a solução que foi criada na configuração do ambiente de desenvolvimento (ver secção anterior);
2. No *Solution Explorer* abrir a classe *ExcelFormula.cs*;



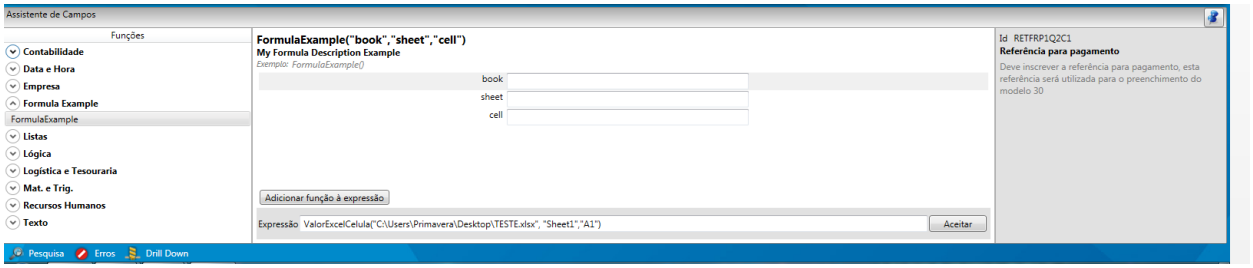
3. Verificar que a *assembly* do Excel Interop foi corretamente carregada (é necessária a instalação do Microsoft Excel no sistema);



4. Após abrir a classe *ExcelFormula.cs*, verificar que todo o código da fórmula se encontra gerado.
   No construtor desta classe encontram-se definidos os parâmetros de fórmula necessários ao sistema, sendo esta composta por 3 parâmetros de entrada:

   - *book* - o caminho de sistema para a ficheiro Excel;
   - *sheet* - a página de Excel pretendida;
   - cell - é referente ao código da célula.

5. Compilar para a pasta de extensibilidade do PFR. Abrir o PFR e verificar no assistente de fórmulas a nova fórmula criada.

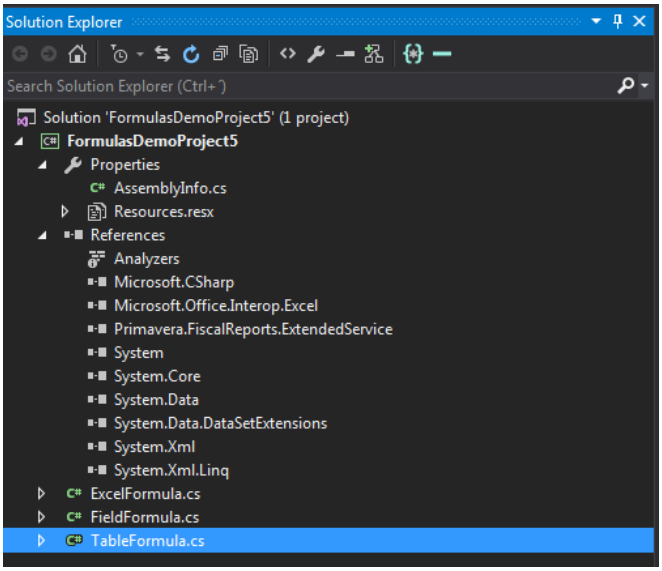


Exemplo fórmula de Tabela

A fórmula de tabela deverá ser criada para uma tabela específica e deverá respeitar toda a estrutura (colunas e tipos de dados).

Para criar a fórmula referida, realizar os seguintes passos:

1. No *Visual Studio* abrir a solução que foi criada na configuração do ambiente de desenvolvimento;
2. No *Solution Explorer* abrir a classe *TableFormula.cs*;



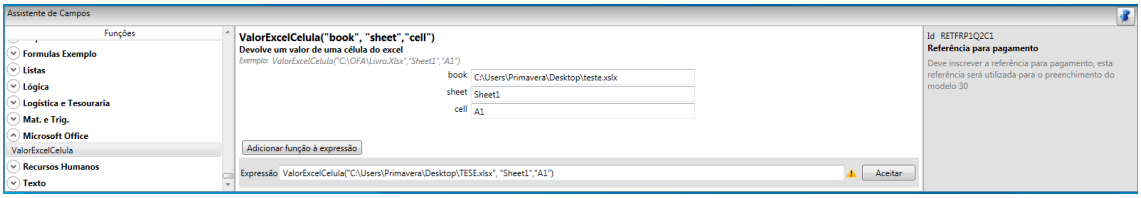
3. No construtor da classe são especificados todos os parâmetros necessários para a sua implementação. De realçar que, sendo uma grelha, pode utilizar-se o *drill down* de linha ou grelha. No caso do *drill down* de linha é necessário ativar as chaves de linha, neste caso relativas à coluna 1 e 2;

4. Na implementação deste tipo de fórmula no método *Execute* é necessário que seja específica para uma determinada tabela devido ao número de colunas e tipo de campos. Deste modo, pode utilizar-se uma *Stored Procedure* ou *Select* que devolva exatamente o número de colunas pretendidas. Neste caso específico, será implementada uma fórmula para o Modelo Retenções na Fonte (Portugal), na grelha Retenções (RETFRP1Q1T1);

5. Após retornar à plataforma PFR a *DataTable* com o conteúdo da *Stored Procedure* poderá ser necessário efetuar um tratamento de dados. No decorrer do processo de inserção na grelha pode utilizar-se o evento *DataTableRowFillEvent*, que será ativado sempre que for processada uma linha. Neste caso específico, vai-se subtrair um valor à coluna 3;

6. Na implementação do *drill down* de grelha aplicado na função *ExecuteDrillDown*, é necessária a existência de uma verificação que permita identificar se o *drill down* executado se trata de uma linha ou da grelha completa. Para tal, pode verificar-se se os parâmetros chave (DrillDownKeys) contêm algum valor no *Input*. Se existir, então é obrigatoriamente um *drill down* de linha, significando que existe uma chave que identifica univocamente a linha sobre a qual se pretende executar a operação;

7. Compilar para a pasta de extensibilidade do PFR. No PRIMAVERA Fiscal Reporting abrir o modelo e aceder ao Assistente de Campos onde se pode verificar no menu esquerdo (Categorias) a nova fórmula criada.

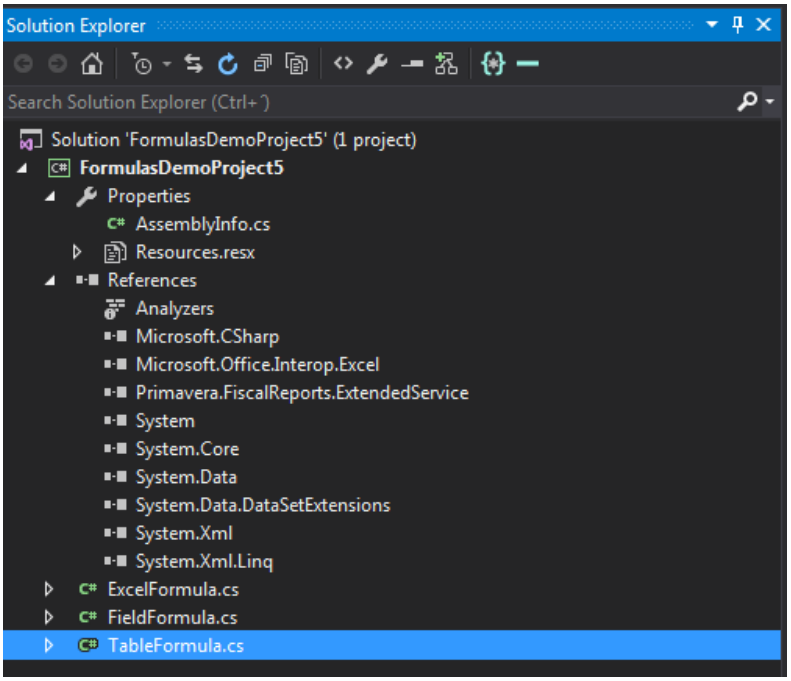


Exemplo fórmula de Campo

A fórmula de campo permite retornar conteúdo para um campo seja ele texto ou numérico.

Para criar a fórmula referida, realizar os seguintes passos:

1. No *Visual Studio* abrir a solução que foi criada na configuração do ambiente de desenvolvimento;
2. No Solution Explorer abrir a classe *FieldFormula.cs*;



3. No construtor da classe são especificados todos os parâmetros necessários para a sua implementação, sendo bastante semelhante ao exemplo anterior;

4. Na execução da fórmula (método *Execute*) será efetuada uma consulta à tabela *Funcionarios* do ERP PRIMAVERA e retornado o número de funcionários;

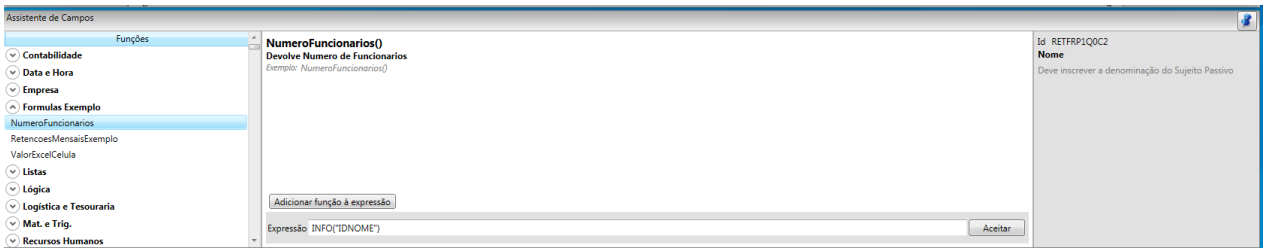
5. Para a execução do drill down, será implementada uma consulta à mesma tabela:

   SQLOperations.ExecuteSQLFormula("SELECT Count(*) FROM Funcionarios")

   Contudo, será devolvida a listagem dos nomes dos funcionários utilizando o seguinte exemplo:

   SQLOperations.ExecuteSQLTable("SELECT Nome AS 'Nome dos Funcionarios' FROM Funcionarios")

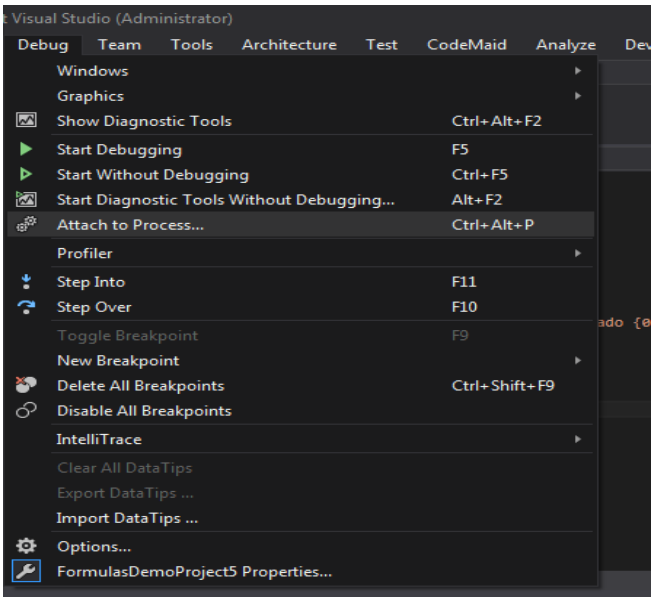
6. Compilar para a pasta de extensibilidade do PFR. Abrir o PRIMAVERA Fiscal Reporting, abrir o modelo e aceder ao Assistente de Campos onde se pode verificar no menu esquerdo (Categorias) a nova fórmula criada.



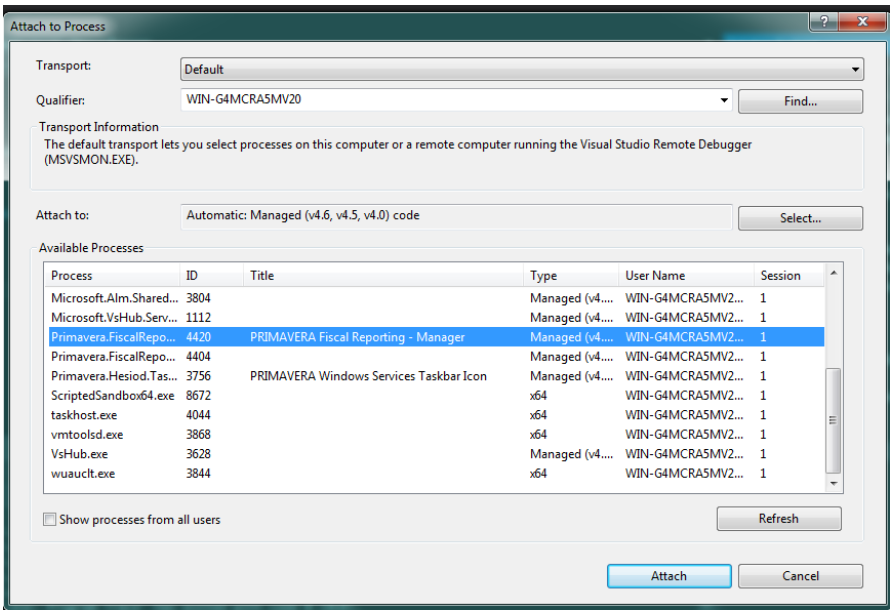
Como fazer *Debug* ao exemplo?

No desenvolvimento das fórmulas anteriores poderá ser necessário executar os seguintes exemplos em *debug*. Os seguintes passos mostram como colocar a *assembly* criada em modo *debug*:

1. Abrir a *Solution* com o projeto das fórmulas desenvolvidas;
2. Efetuar compilação da solução em modo *debug* (verificar que a *assembly* na pasta de extensibilidade do PFR foi atualizada e contém um ficheiro com o mesmo nome da *assembly* e com extensão ".pdb");
3. Abrir o PFR e efetuar o login;
4. No *Visual Studio* ir ao menu *Debug* e selecionar *Attach to Process* (ctrl+alt+p);



5. Selecionar o processo a fazer Attach que será o PRIMAVERA Fiscal Reporting:

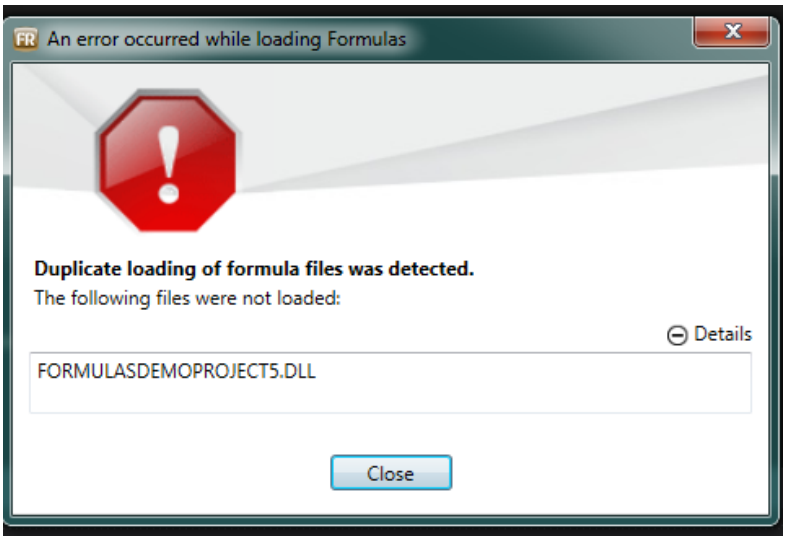


6. A partir deste momento poderá realizar-se qualquer tipo de *breakpoint* no código, sendo que o modo de *debug* está ativado.

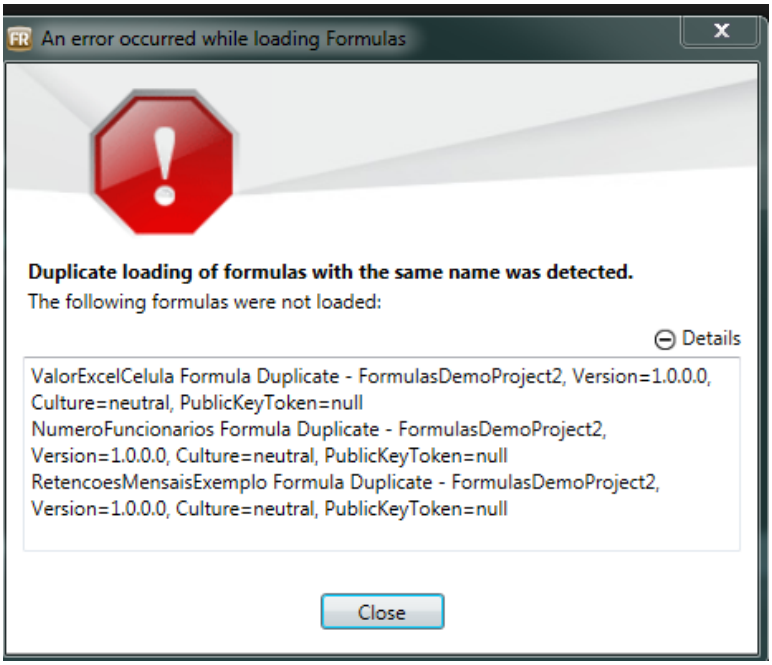
Regras de carregamento de fórmulas

De modo a evitar incompatibilidades ou sobreposições indesejadas, o sistema de carregamento de fórmulas gere-se pelas seguintes regras:

1. Não poderão existir *Assemblies* com o mesmo nome sendo que, caso isso aconteça, aparecerá a seguinte mensagem na abertura da aplicação:



2. Não poderão existir fórmulas de extensibilidade com o mesmo nome. Caso isso aconteça aparecerá a seguinte mensagem na abertura da aplicação:



FAQ

Dúvida	**Respostas / Ações**
Efetuei alterações no meu projeto de fórmulas, mas o PFR não atualizou as minhas fórmulas.	• Verificar que a *assembly* compilada se encontra atualizada; • Reiniciar o PFR.
Numa fórmula que executa código SQL é-me retornado o erro "operação não permitida".	• Consultar a secção SQLOperations e verificar se existe alguma operação não permitida no código.
Não consigo efetuar *breakpoints* na classe da fórmula.	• Verificar se realizou um *Rebuild* da solução; • Verificar se a *assembly* foi atualizada na pasta; • Verificar se a opção "*Attach to Process*" se encontra ativada sobre o processo do PFR; • Consultar a secção "Como fazer debug ao exemplo?".
Ao iniciar o PFR aparece-me um erro de *assemblies* em duplicado.	• Verificar se tem *assemblies* iguais na pasta de Extensibilidade. • Verificar se o nome atribuido à *assembly* não é igual a outra assembly em alguma pasta da extensbilidade (incluindo a pasta Connectors\Formulas\System).
Posso utilizar a pasta System para colocar as minhas fórmulas?	• Não, a pasta *System* encontra-se exclusivamente reservada para fórmulas de sistema, sendo que **nunca** deverá ser alterada.
Criei uma nova fórmula que ficou disponível no assistente de campos, mas ao executar aparece-me a mensagem de erro "A função XYZ não é válida".	• Verificar na classe da fórmula se todos os parâmetros do construtor estão corretamente definidos; • Verificar as fórmulas do projeto exemplo.

Termos de Utilização

Ao utilizar esta extensão ou a qualquer conteúdo que nela se encontre, o utilizador aceita os termos e condições que se apresentam em baixo.

1. A extensão é propriedade da PRIMAVERA. Todos os direitos estão reservados.
2. A PRIMAVERA declina qualquer responsabilidade pelo seu uso indevido. Em qualquer momento reserva-se o direito de modificar, sem aviso prévio, o conteúdo e forma desta extensão.
3. Declina igualmente qualquer responsabilidade por eventuais danos ou problemas que possam surgir na sequência da utilização desta extensão ou partes de *software* que dela advém.
4. Declina ainda qualquer responsabilidade sobre o suporte, manutenção e evolução da mesma.