COMO ESTENDER A WEB API

A Web API da versão 10 do ERP PRIMAVERA é uma peça tecnológica inovadora que permite às empresas integrar o ERP com outros sistemas.

Ainda que Web API apresente, em forma de serviços REST, a maior parte dos métodos nativos da API do ERP, existem outros que não foram considerados. A possibilidade de estender esta Web API permite acrescentar novas funcionalidades não cobertas nativamente, assim como outras funcionalidades que o parceiro integrador necessite. Estes novos métodos irão usar o mesmo *end point*.

Para demonstrar a capacidade de extensão, neste artigo vamos usar como exemplo a impressão de um documento de venda em formato PDF pronto a ser apresentado.

Pré-requisitos

- Primavera ERP com Web API instalada e licenciada
- Visual Studio & .NET Framework 4.7.1

Passo 1 - Criar o projeto

Primeiro, é necessário criar um projecto com o **Visual Studio** de forma a criar uma classe que adicione mais funcionalidades. Este projeto deve ser do tipo **Class Library (.NET Framework)**.

Passo 2 - Packages

Os seguintes *packages* devem ser adicionados ao projeto, tendo em atenção as versões dos mesmos, uma vez que são essas as versões usadas nas assemblies, tanto no ERP como na Web API:

A classe que foi adicionada ao projeto deve obedecer a dois requisitos de forma a ser reconhecida pela Web API:

1. O primeiro requisito relaciona-se com nome da mesma. Este deve ter, obrigatoriamente, o sufixo *Controller*.
2. Esta classe também deve derivar da classe **Assembly System.Web.Http.ApiControler**

```
public class ClassExtendedController: ApiController
```

Neste exemplo, intitulamos a classe de "**ClassExtendedController**".

Passo 3 - RoutePrefix

Como como qualquer classe da Web API, também esta necessita da anotação *RoutePrefix* que terá que ser igual ao próprio nome da classe, mas sem o sufixo *Controller*.

```
[RoutePrefix("ClassExtended")]
public class ClassExtendedController: ApiController
```

Passo 4 - Codificação

Tal como explicado, o exemplo demonstrado será o de imprimir um documento de venda em formato PDF. Será um pedido *HttpGet* com a seguinte rota:

```
[Authorize]
[Route("PrintSalesDocToPDF/{TipoDoc}/{Serie}/{Numdoc}/{Filial}/{Numvias}/{NomeReport}/{Segu
[HttpGet]
```

```
[HttpGet]
public HttpResponseMessage PrintSalesDocToPDF(string TipoDoc, string Serie, int Numdoc, string
{
    try
    {
        HttpResponseMessage response = Request.CreateResponse(HttpStatusCode.BadRequest);

        // Garantir que a extensão está definida
        if (string.IsNullOrWhiteSpace(Path.GetExtension(NomePDF)))
        {
            NomePDF += ".pdf";
        }

        // Garantir um local fisico para guardar o pdf gerado
        if (string.IsNullOrWhiteSpace(Path.GetDirectoryName(NomePDF)))
        {
            NomePDF = Path.Combine(Path.GetTempPath(), NomePDF);
        }

        // Imprime o documento para pdf via API
        bool imprimedocumento = ProductContext.MotorLE.Vendas.Documentos.ImprimeDocumen

        // Prepara a resposta
        if (imprimedocumento)
        {
            var dataBytes = File.ReadAllBytes(NomePDF);
            var dataStream = new MemoryStream(dataBytes);

            byte[] buffer = new byte[0];
            buffer = dataStream.ToArray();

            var contentLength = buffer.Length;

            var statuscode = HttpStatusCode.OK;
            response = Request.CreateResponse(statuscode);
            response.Content = new StreamContent(new MemoryStream(buffer));
            response.Content.Headers.ContentType = new MediaTypeHeaderValue("application/p
            response.Content.Headers.ContentLength = contentLength;
```

```
            return response;
        }
        catch (Exception ex)
        {
            throw new HttpResponseException(Request.CreateResponse(HttpStatusCode.BadRequest,
        }
    }
```

Este serviço, que imprime um documento de venda, usa o método **ImprimeDocumento** que se encontra disponível na API do módulo de vendas. O acesso ao contexto do ERP está acessível na própria Web API, sem que seja necessário implementar qualquer outro mecanismo. Todas as questões de segurança estão também, desde já, garantidas.

O acesso ao contexto do ERP é dado pela classe **ProductContext** que se encontra na *assembly* **Primavera.WebAPI.Integration**, sendo necessário referenciar a mesma no projeto.

Passo 5 - Compilação e Deployment

Tendo tudo montado como explicado anteriormente, basta apenas compilar o projeto.

No output do projeto será gerada a *assembly* que terá de ser copiada para a pasta da Web API, tipicamente para a pasta "**PRIMAVERASG100AplWebApibin**".

Passo 6 - Teste

Para testar este código pode ser usado o Postman.

Depois de pedido o token, pode ser feito o pedido para a impressão de um documento através da seguinte rota:

docGT1.pdf.

```
{{apiUrl}}ClassExtended/PrintSalesDocToPDF/GT/C/1/000/1/GCPVLS01/false/docGTC1/1
```

Como o Postman não tem um preview de documentos em formato PDF, na resposta podemos gravar o mesmo para um documento e validar se tudo aconteceu como esperado.

Conclusão

Criar extensões para a Web API é a melhor forma de tirar partido de todo o potencial que esta tecnologia possibilita, sem necessidade de criar novos web services. Todos os serviços não disponibilizados nativamente podem, desta forma, ser implementados, bem como qualquer outro serviço decorrente de um processo de implementação.

ARTIGOS RELACIONADOS

Começar a Usar

Características das entidades e serviços

Conceito de Integração

ÚLTIMOS ARTIGOS VISTOS

Como registar projetos de extensibilidade

Como abrir listas em aplicações externas.

Utilização do Padrão Observer na V10.

Integração de documentos certificados e assinados por aplicações terceiras.

Começar a Usar

PRIMAVERA

APOIO

SITE

Siga-nos