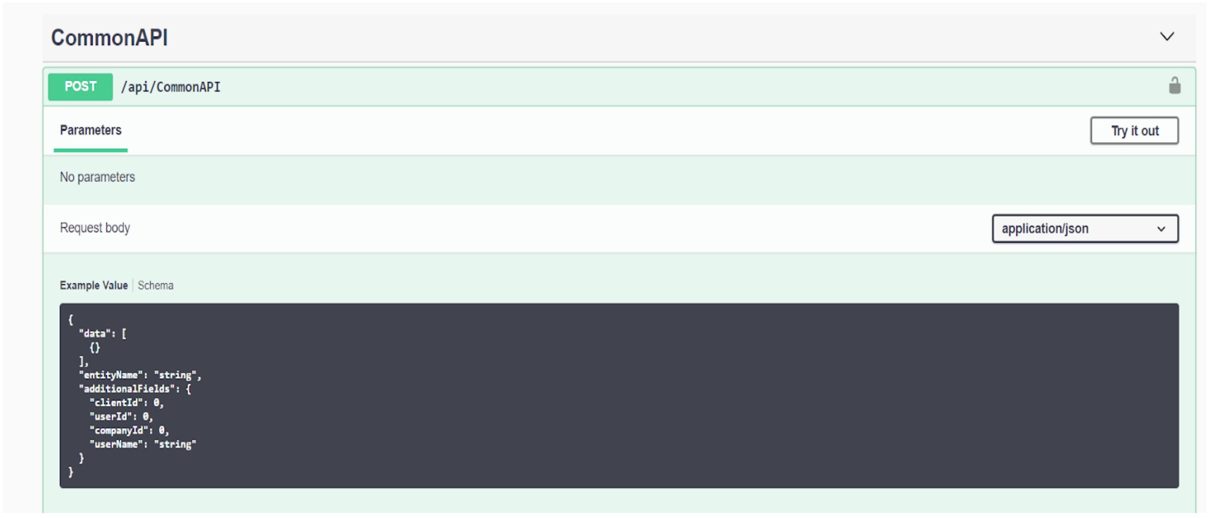
Swagger

Swagger is an open-source software framework backed by a large ecosystem of tools that helps developers design, build, document and consume RESTful web services. While most users identify Swagger by the Swagger UI tool, the Swagger toolset includes support for automated documentation, code generation, and test-case generation.

What are the things we are not doing in swagger?

1. we only use defaults of Swagger API.
2. There is no other Benefit of Using Swagger if we are not providing proper documentation
   - What is the purpose of this API.
   - What type of data we need to send in the Request for calling this API.
   - What type of response to we get from API.
3. Not providing sample data for testing.

Ex :-



```
{
  "data": [
    {}
  ],
  "entityName": "string",
  "additionalFields": {
    "clientId": 0,
    "userId": 0,
    "companyId": 0,
    "userName": "string"
  }
}
```

We can improve swagger documentation with the help of this document.

https://github.com/domaindrivendev/Swashbuckle.AspNetCore

There two way specify in the link for adding description in the swagger.

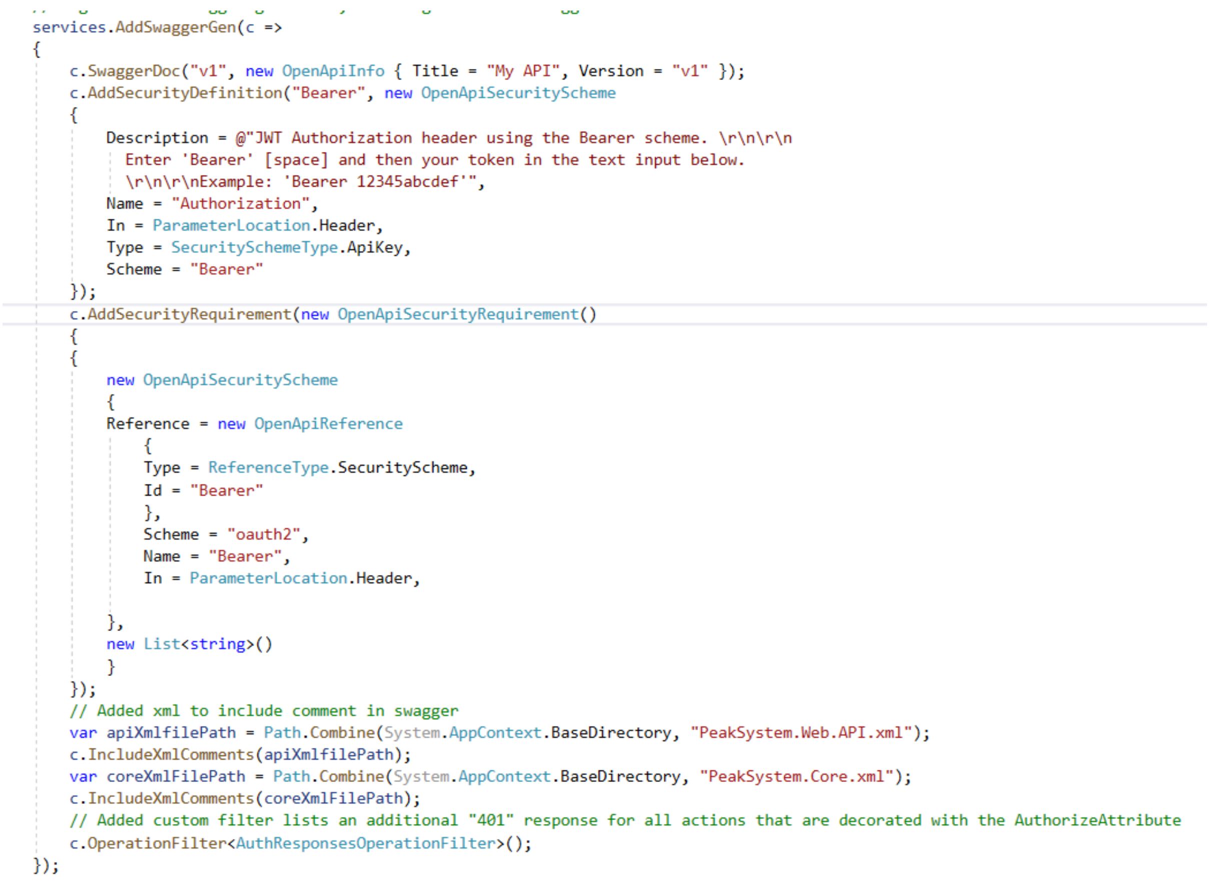
1. Include Descriptions from XML Comments.
2. Using Swashbuckle.AspNetCore.Annotations package.

I have done using xml approach.

1. Open the Properties dialog for your project, click the "Build" tab and ensure that "XML documentation file" is checked. This will produce a file containing all XML comments at build-time.

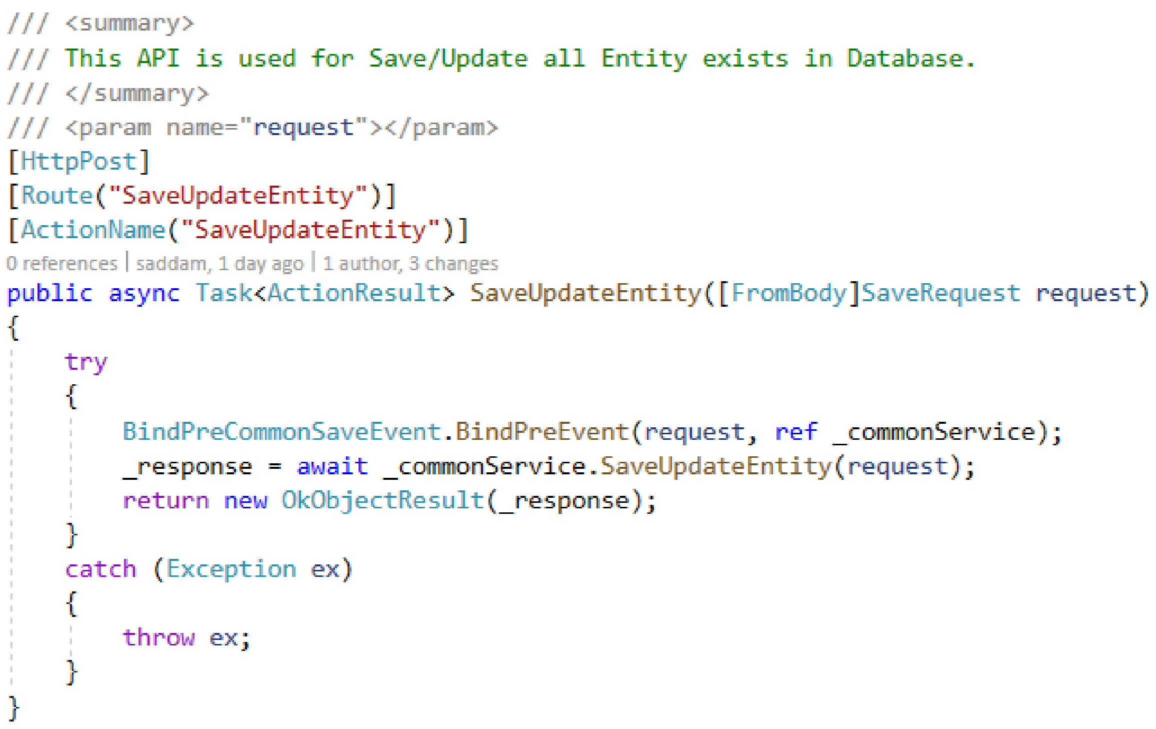
   At this point, any classes or methods that are NOT annotated with XML comments will trigger a build warning. To suppress this, enter the warning code "1591" into the "Suppress warnings" field in the properties dialog.

2. Configure Swashbuckle to incorporate the XML comments on file into the generated Swagger JSON:



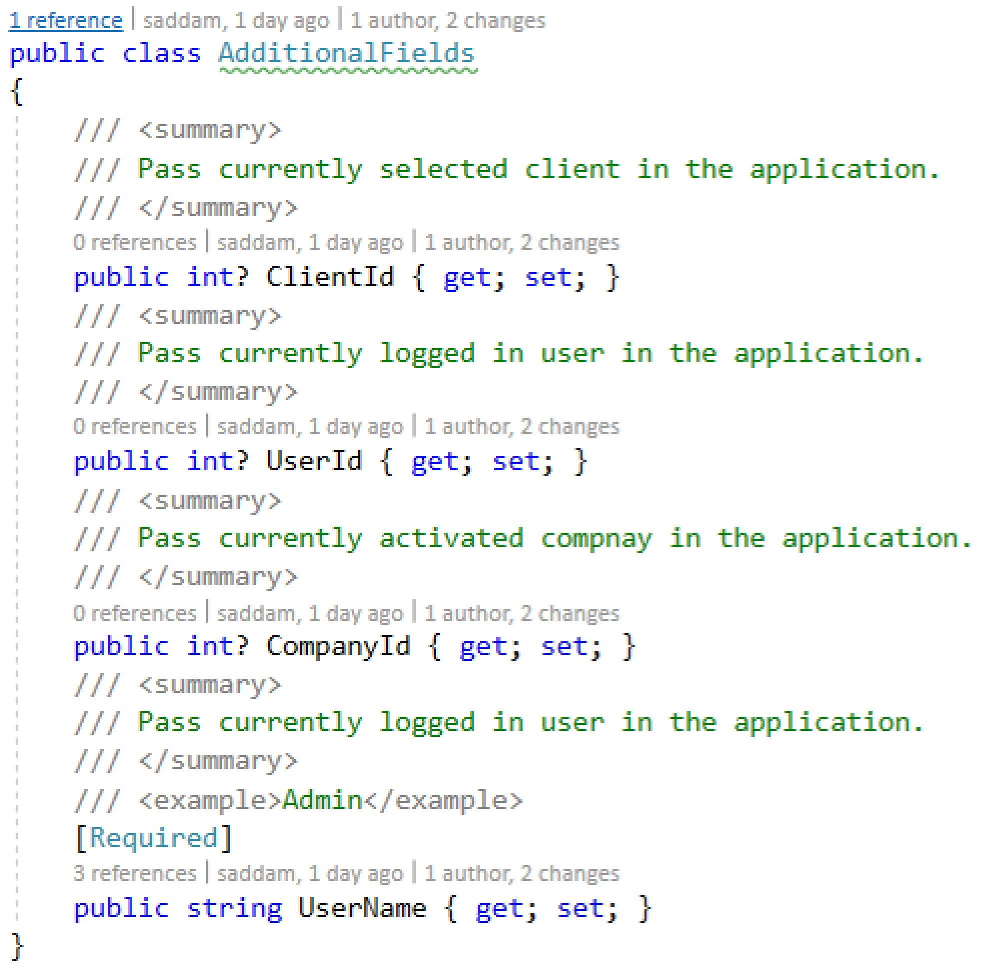
```
services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "My API", Version = "v1" });
    c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
    {
        Description = @"JWT Authorization header using the Bearer scheme. \r\n\r\n
          Enter 'Bearer' [space] and then your token in the text input below.
          \r\n\r\nExample: 'Bearer 12345abcdef'",
        Name = "Authorization",
        In = ParameterLocation.Header,
        Type = SecuritySchemeType.ApiKey,
        Scheme = "Bearer"
    });
    c.AddSecurityRequirement(new OpenApiSecurityRequirement()
    {
    {
        new OpenApiSecurityScheme
        {
        Reference = new OpenApiReference
            {
            Type = ReferenceType.SecurityScheme,
            Id = "Bearer"
            },
            Scheme = "oauth2",
            Name = "Bearer",
            In = ParameterLocation.Header,

        },
        new List<string>()
        }
    });
    // Added xml to include comment in swagger
    var apiXmlfilePath = Path.Combine(System.AppContext.BaseDirectory, "PeakSystem.Web.API.xml");
    c.IncludeXmlComments(apiXmlfilePath);
    var coreXmlFilePath = Path.Combine(System.AppContext.BaseDirectory, "PeakSystem.Core.xml");
    c.IncludeXmlComments(coreXmlFilePath);
    // Added custom filter lists an additional "401" response for all actions that are decorated with the AuthorizeAttribute
    c.OperationFilter<AuthResponsesOperationFilter>();
});
```

3. Annotate your actions with summary, remarks and response tags:

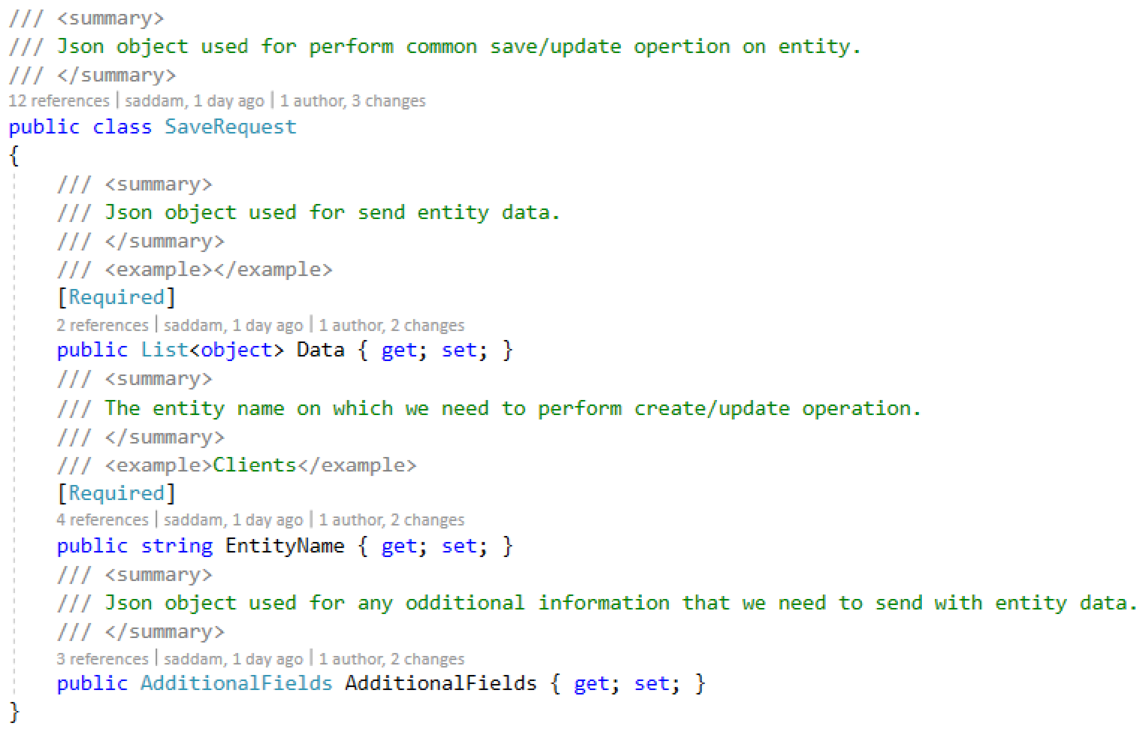


```
/// <summary>
/// This API is used for Save/Update all Entity exists in Database.
/// </summary>
/// <param name="request"></param>
[HttpPost]
[Route("SaveUpdateEntity")]
[ActionName("SaveUpdateEntity")]
0 references | saddam, 1 day ago | 1 author, 3 changes
public async Task<ActionResult> SaveUpdateEntity([FromBody]SaveRequest request)
{
    try
    {
        BindPreCommonSaveEvent.BindPreEvent(request, ref _commonService);
        _response = await _commonService.SaveUpdateEntity(request);
        return new OkObjectResult(_response);
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

4. You can also annotate types with summary and example tags:

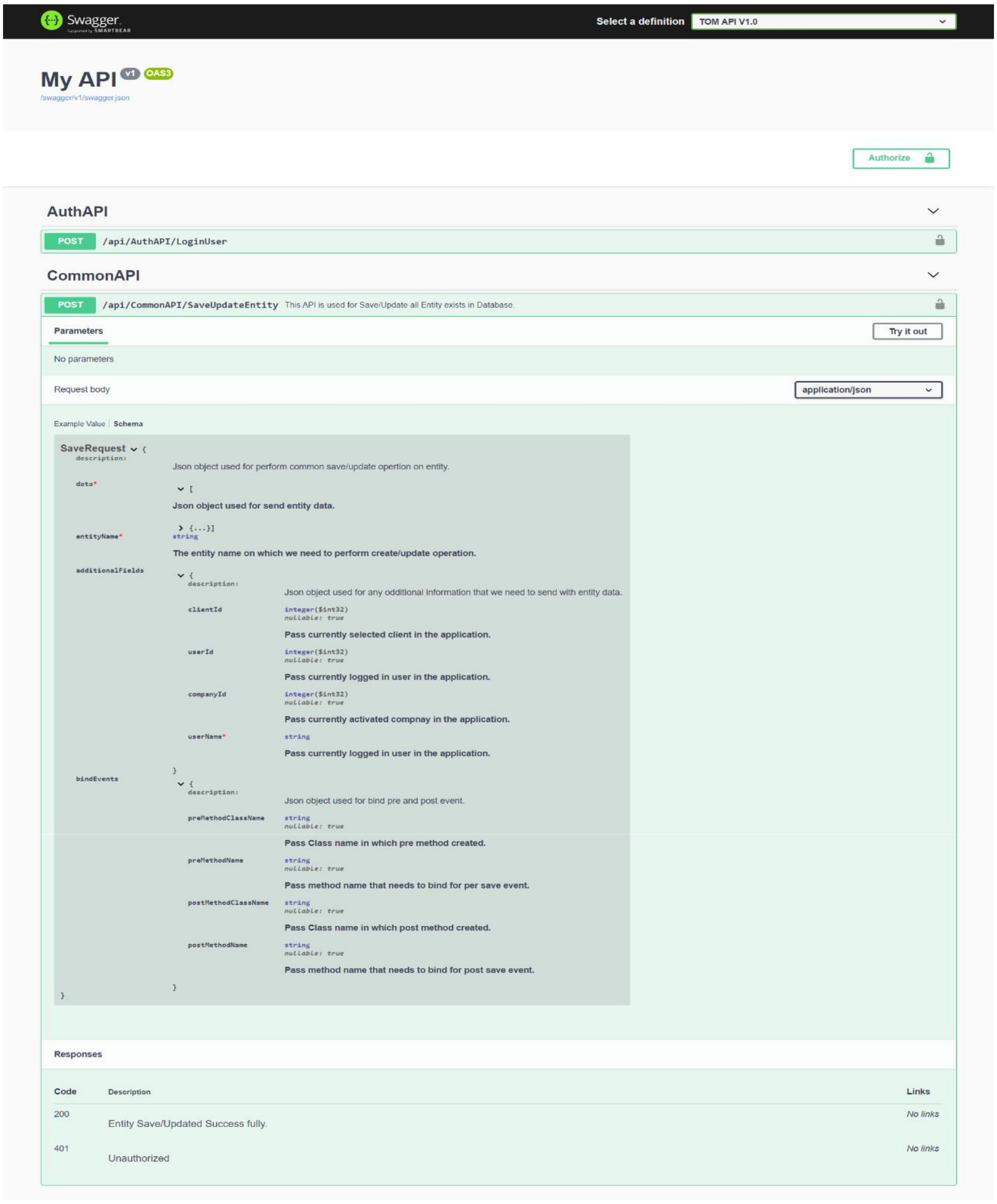


```
1 reference | saddam, 1 day ago | 1 author, 2 changes
public class AdditionalFields
{
    /// <summary>
    /// Pass currently selected client in the application.
    /// </summary>
    0 references | saddam, 1 day ago | 1 author, 2 changes
    public int? ClientId { get; set; }
    /// <summary>
    /// Pass currently logged in user in the application.
    /// </summary>
    0 references | saddam, 1 day ago | 1 author, 2 changes
    public int? UserId { get; set; }
    /// <summary>
    /// Pass currently activated compnay in the application.
    /// </summary>
    0 references | saddam, 1 day ago | 1 author, 2 changes
    public int? CompanyId { get; set; }
    /// <summary>
    /// Pass currently logged in user in the application.
    /// </summary>
    /// <example>Admin</example>
    [Required]
    3 references | saddam, 1 day ago | 1 author, 2 changes
    public string UserName { get; set; }
}
```



```
/// <summary>
/// Json object used for perform common save/update opertion on entity.
/// </summary>
12 references | saddam, 1 day ago | 1 author, 3 changes
public class SaveRequest
{
    /// <summary>
    /// Json object used for send entity data.
    /// </summary>
    /// <example></example>
    [Required]
    2 references | saddam, 1 day ago | 1 author, 2 changes
    public List<object> Data { get; set; }
    /// <summary>
    /// The entity name on which we need to perform create/update operation.
    /// </summary>
    /// <example>Clients</example>
    [Required]
    4 references | saddam, 1 day ago | 1 author, 2 changes
    public string EntityName { get; set; }
    /// <summary>
    /// Json object used for any oddtional information that we need to send with entity data.
    /// </summary>
    3 references | saddam, 1 day ago | 1 author, 2 changes
    public AdditionalFields AdditionalFields { get; set; }
}
```

5. Rebuild your project to update the XML Comments file and navigate to the Swagger JSON endpoint. Note how the descriptions are mapped onto corresponding Swagger fields.



```
SaveRequest ⌄ {
    description:            Json object used for perform common save/update opertion on entity.
    data*                   ⌄ [
                            Json object used for send entity data.
                            > {...}]
    entityName*             string
                            The entity name on which we need to perform create/update operation.
    additionalFields        ⌄ {
                              description:          Json object used for any odditional information that we need to send with entity data.
                              clientId              integer($int32)
                                                    nullable: true
                                                    Pass currently selected client in the application.
                              userId                integer($int32)
                                                    nullable: true
                                                    Pass currently logged in user in the application.
                              companyId             integer($int32)
                                                    nullable: true
                                                    Pass currently activated compnay in the application.
                              userName*             string
                                                    Pass currently logged in user in the application.
                            }
    bindEvents              ⌄ {
                              description:          Json object used for bind pre and post event.
                              preMethodClassName    string
                                                    nullable: true
                                                    Pass Class name in which pre method created.
                              preMethodName         string
                                                    nullable: true
                                                    Pass method name that needs to bind for per save event.
                              postMethodClassName   string
                                                    nullable: true
                                                    Pass Class name in which post method created.
                              postMethodName        string
                                                    nullable: true
                                                    Pass method name that needs to bind for post save event.
                            }
}
```

Responses

Code	Description	Links
200	Entity Save/Updated Success fully.	*No links*
401	Unauthorized	*No links*