2020-01-28: DFDS A/S Tech Hub @ Harbour House II

CrmWebApiUtil +LINQ Provider

F# dotnet + C# dotnet = Cloud Ready (Docker)


```
var accounts = new List<ModPAX.Account>();

try
{
    using var crm = new CrmClient(settings);

    var ctx = new WebAPI.Context<ModPAX.Account>(crm, RelPAX.Lookups);

    var query =
        from a in ctx
        where
            accountids.Contains(a.AccountId)
        select
            new ModPAX.Account
            { AccountId = a.AccountId
            , AccountNumber = a.AccountNumber
            /*                */
            , Name = a.Name
            , CustomerTypeCode = a.CustomerTypeCode
            ,
            , NumberOfEmployees = a.NumberOfEmployees
            ,
            ,
            , OwningUser =
                new ModPAX.SystemUser
                    { SystemUserId = a.OwningUser.SystemUserId
                    , FullName     = a.OwningUser.FullName
                    }
            , OwningTeam =
                new ModPAX.Team
                    { TeamId = a.OwningTeam.TeamId
                    , Name   = a.OwningTeam.Name
                    }
            ,
            ,
            /*            */
            ,
            ,
            , StageId = a.StageId
            ,
            , PrimaryContactId =
                new ModPAX.Contact
                    { ContactId = a.PrimaryContactId.ContactId
                    }
            };

    accounts.AddRange(query);
}
catch (Exception ex)
{
    throw new ErrorHandling.Services.CRM.Account
        ( ex.Message
        , ex.InnerException
        );
}

return await Tasks.Task.Run(() => accounts);
```

DFDS

Overview

- Background
- CrmWebApiUtil (F# .NET Core)
- LINQ Provider (C# .NET Core)
- Cloud Ready (Docker)
- Summary

DFDS

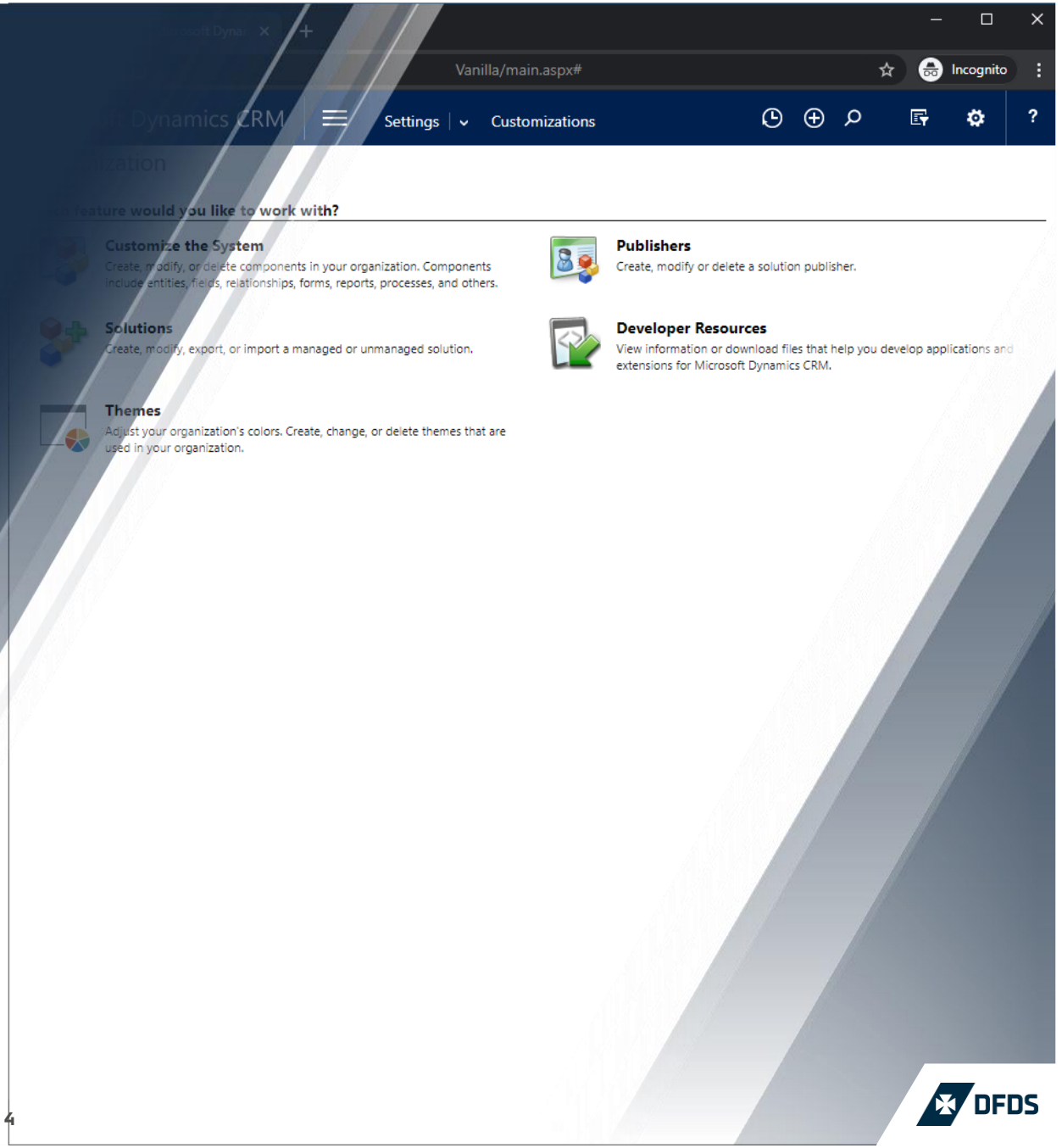
Matching of expectations

- A short **introduction** to the platform: **Microsoft CRM** / Dynamics 365 Customer Engagement
- What **problem** are we **solving** and why is it a problem in the first place
- Hopefully we will be able to convince you that the **chosen approach gives sense** and you might find **inspiration** to do something similar with your own platform

Note: Please **save your questions** to the **QA** at the end of the talk unless I say something that is really **incorrect** ☺

DFDS

Background


Vanilla/main.aspx#
Incognito
Settings
Customizations
feature would you like to work with?
Customize the System
Create, modify, or delete components in your organization. Components include entities, fields, relationships, forms, reports, processes, and others.
Solutions
Create, modify, export, or import a managed or unmanaged solution.
Themes
Adjust your organization's colors. Create, change, or delete themes that are used in your organization.
Publishers
Create, modify or delete a solution publisher.
Developer Resources
View information or download files that help you develop applications and extensions for Microsoft Dynamics CRM.

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Overview > Customer Relationship Management

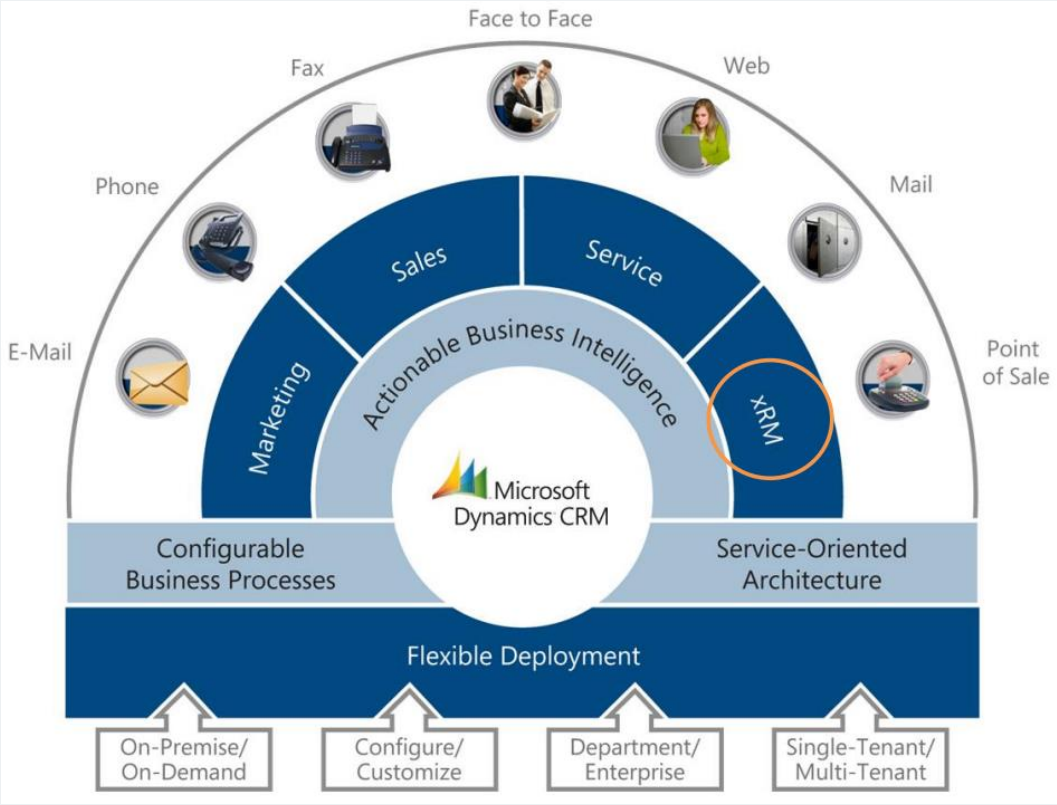


Image source: http://blog.stermon.com/assets/talks/2013-01-31_intro_rsm_and_crm_technical.pdf

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Overview > Integration Hub

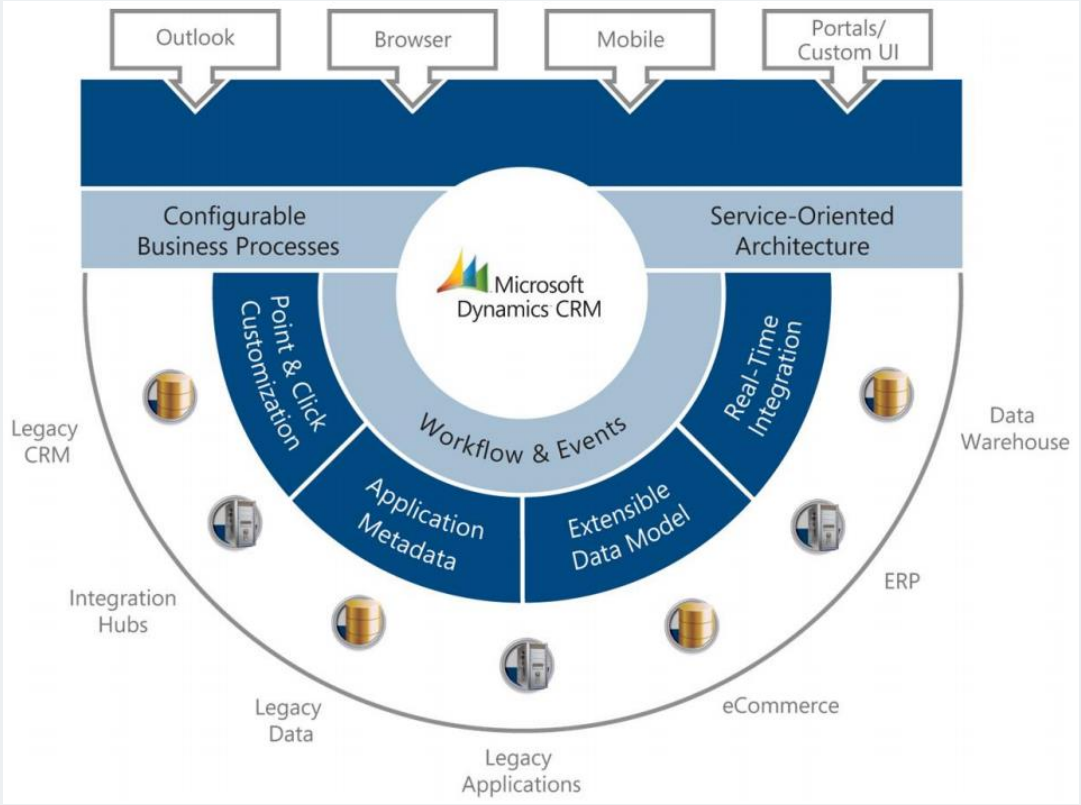


Image source: http://blog.stermon.com/assets/talks/2013-01-31_intro_rsm_and_crm_technical.pdf

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > Loose (SharePoint) vs Very-tight and formal (CRM)

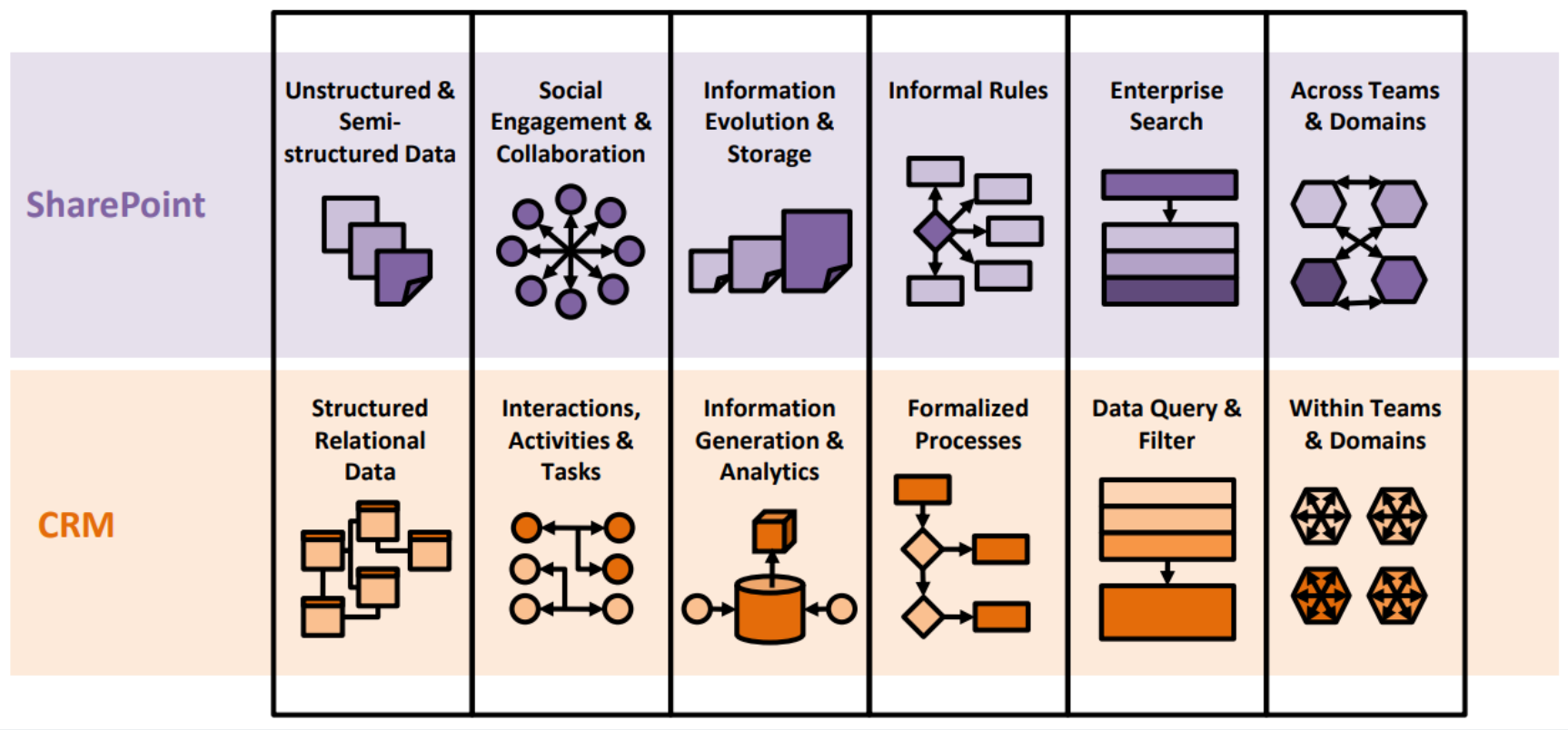


Image source: http://blog.stermon.com/assets/talks/2013-01-31_intro_rsm_and_crm_technical.pdf

DFDS

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > MVC pattern (state changes go through WS Controller and Views can read directly)

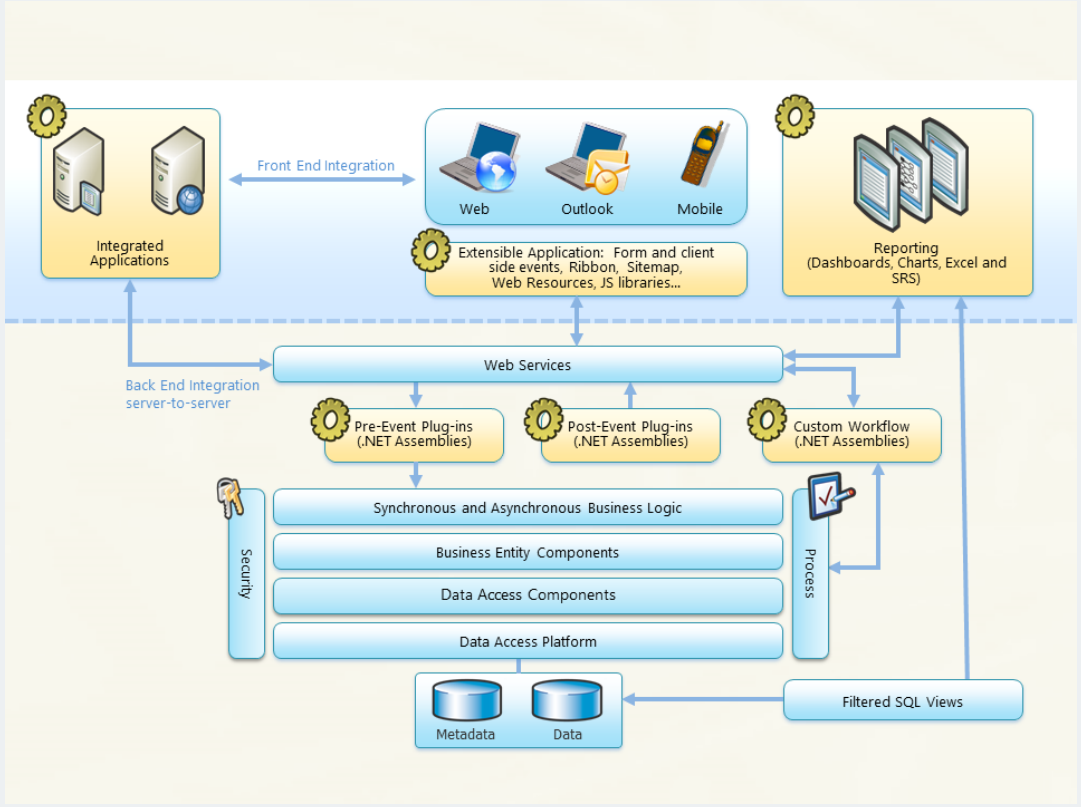


Image source: https://docs.microsoft.com/en-us/dynamics365/customerengagement/on-premises/developer/extend-customer-engagement

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > SOAP (XML) and REST / WebAPI (JSON)

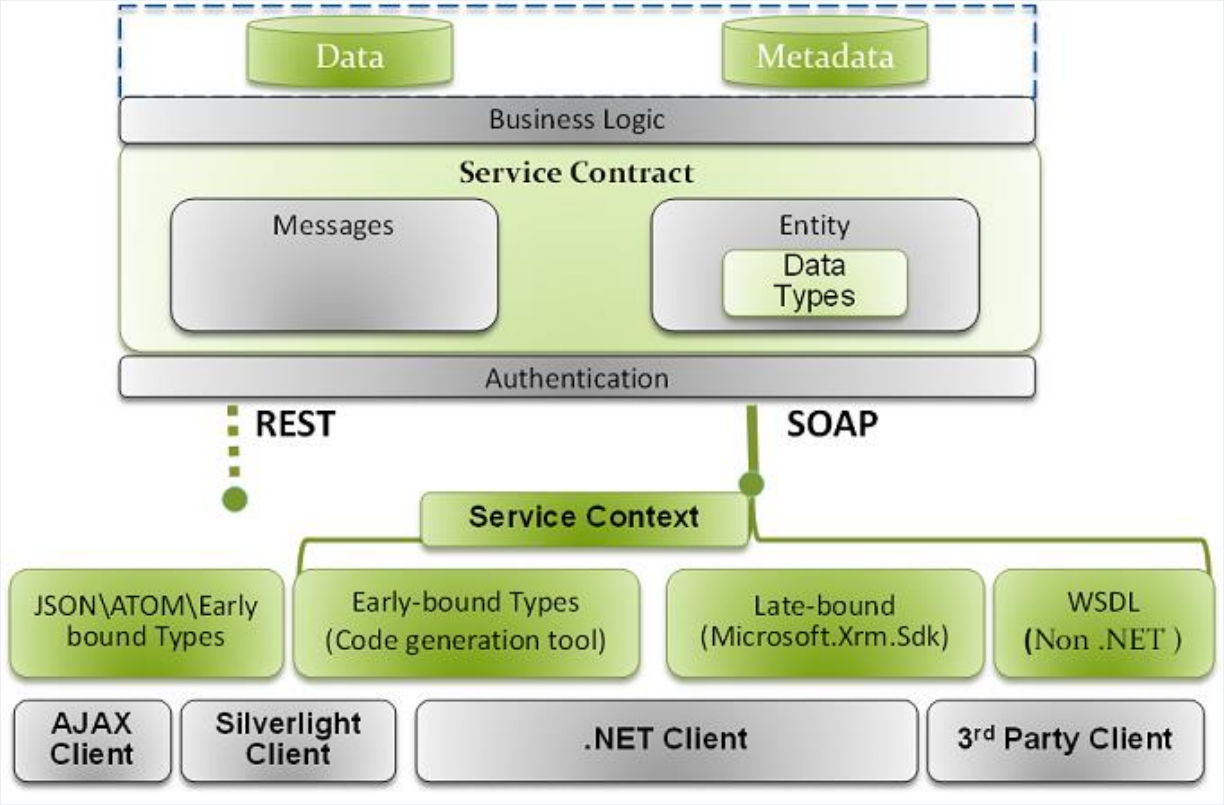


Image source: https://community.dynamics.com/crm/b/microsoftdynamicscrmxrm/posts/programming-models-for-ms-dynamics-crm-2011

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > SOAP (XML) > Early-bound (Code generation tool)

- Create early bound entity classes with the code generation tool (**CrmSvcUtil**):
  - https://docs.microsoft.com/en-us/dynamics365/customerengagement/on-premises/developer/org-service/create-early-bound-entity-classes-code-generation-tool
- The tool will create **type-safe C# classes** based on the data model from CRM application instances
- This will allow to write code with **LINQ** (subset of Entity Framework) in order to receive and send data to CRM
- Maintained by **Microsoft** and accessible through **NuGet**

DFDS

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > SOAP (XML) > Early-bound (Code generation tool)

- Alternatively use **XrmContext**, as it's very similar to **CrmSvcUtil**, but it has several new features that help you code better and more reliably, and generates files that takes up less space:
  - https://github.com/delegateas/XrmContext/wiki
- Maintained by **Delegate A/S** and accessible through **NuGet** as well

Image source: http://delegateas.github.io/

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > SOAP (XML) vs REST / WebAPI (JSON)

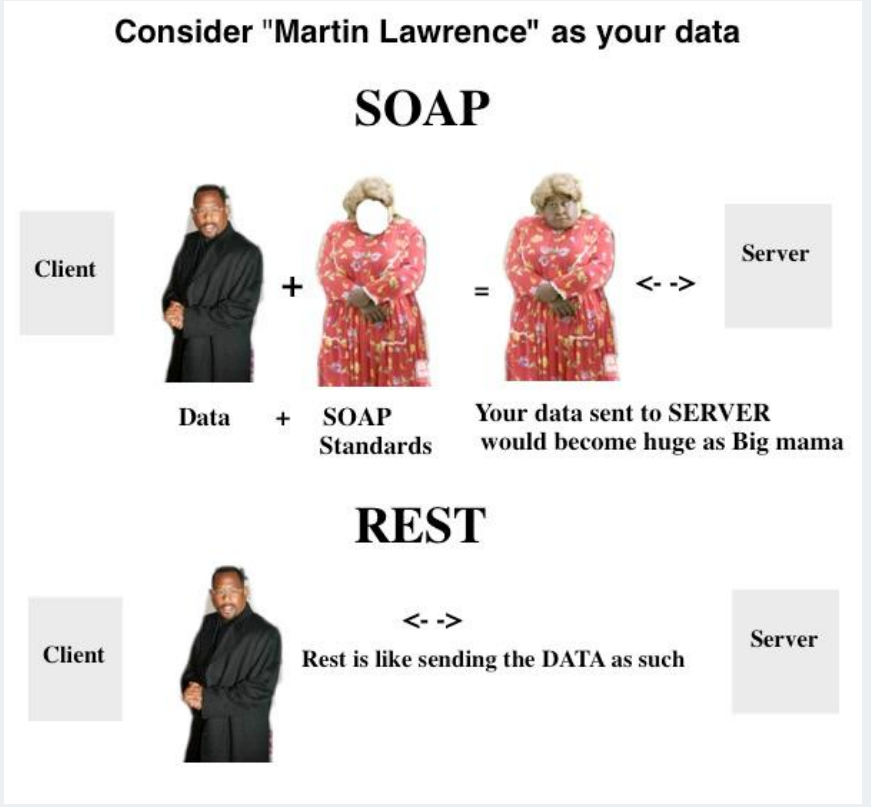


Image source: https://stackoverflow.com/a/8983122

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > MSDN > Send > Untyped, well JSON (Newtonsoft) typed !

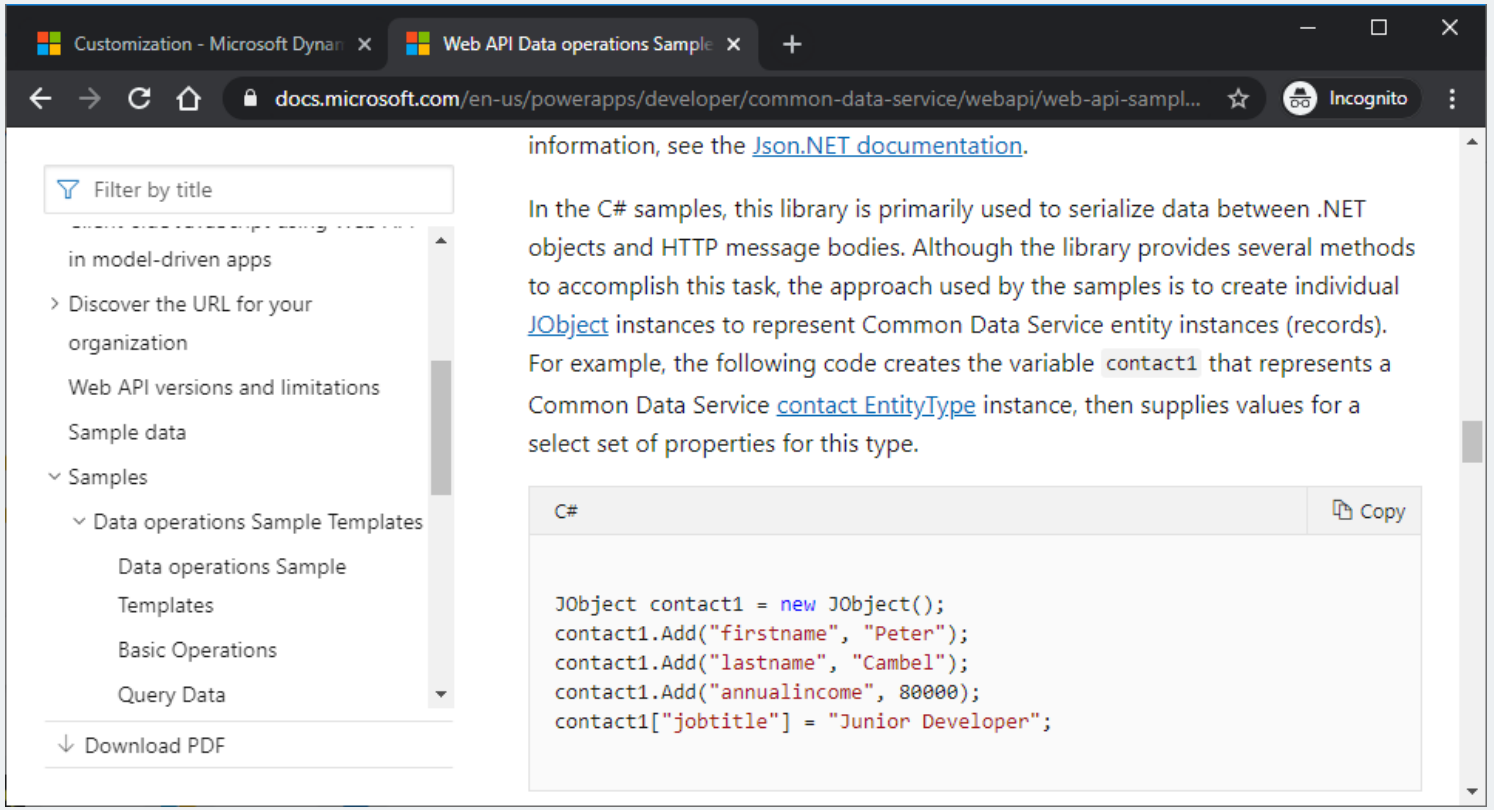


Image source: https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/webapi/web-api-samples-csharp

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > MSDN > Send > Untyped, well JSON (Newtonsoft) typed !

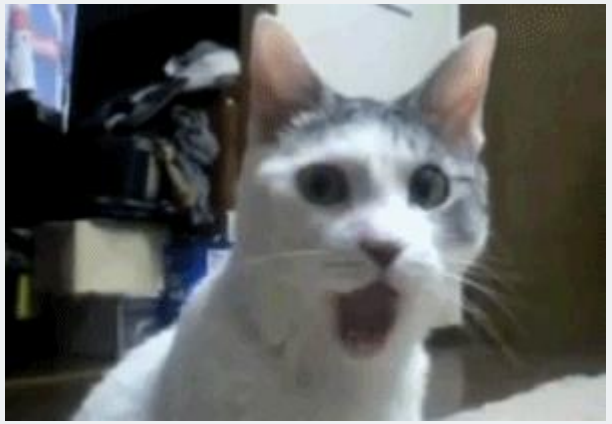


Image source: https://knowyourmeme.com/memes/omg-cat

DFDS

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > MSDN > Receive > Query String (not sanitized text)

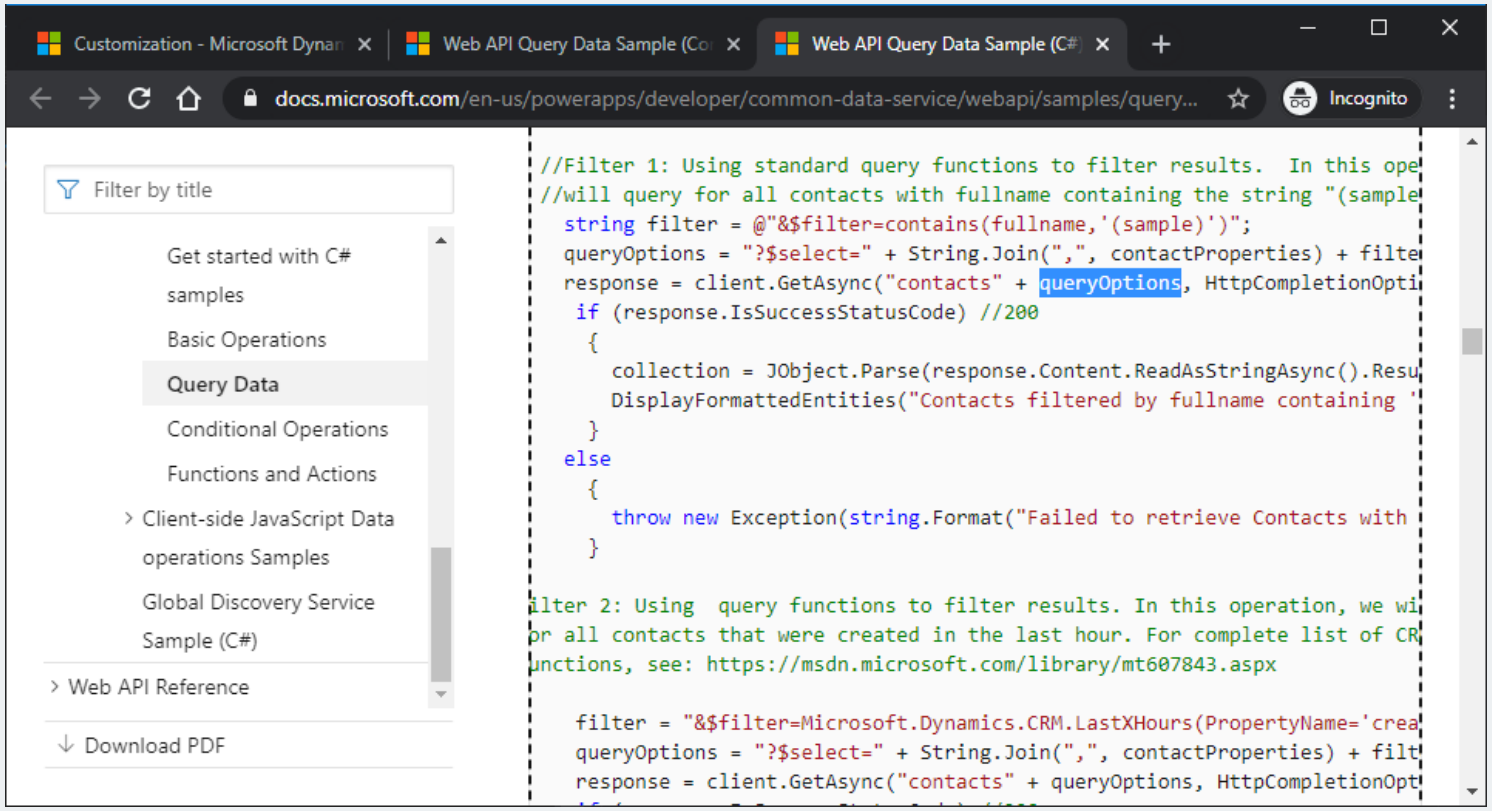


Image source: https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/webapi/samples/query-data-csharp

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > MSDN > Receive > Query String (not sanitized text)

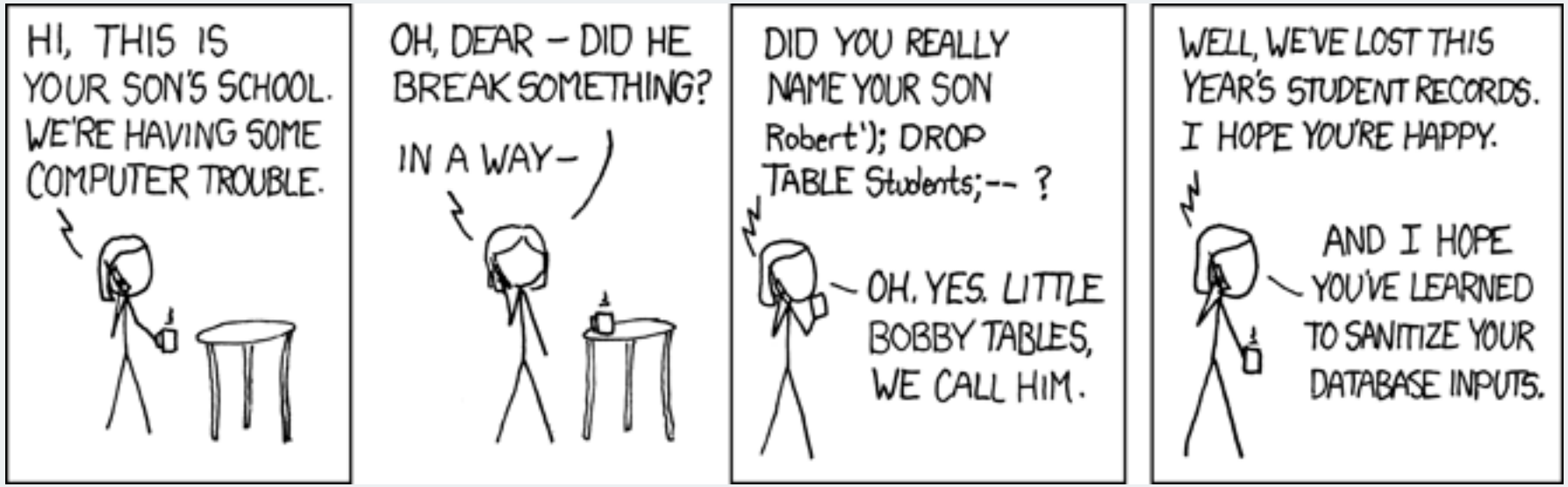


Image source: https://xkcd.com/327/

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > MSDN > Approach (Send + Receive)

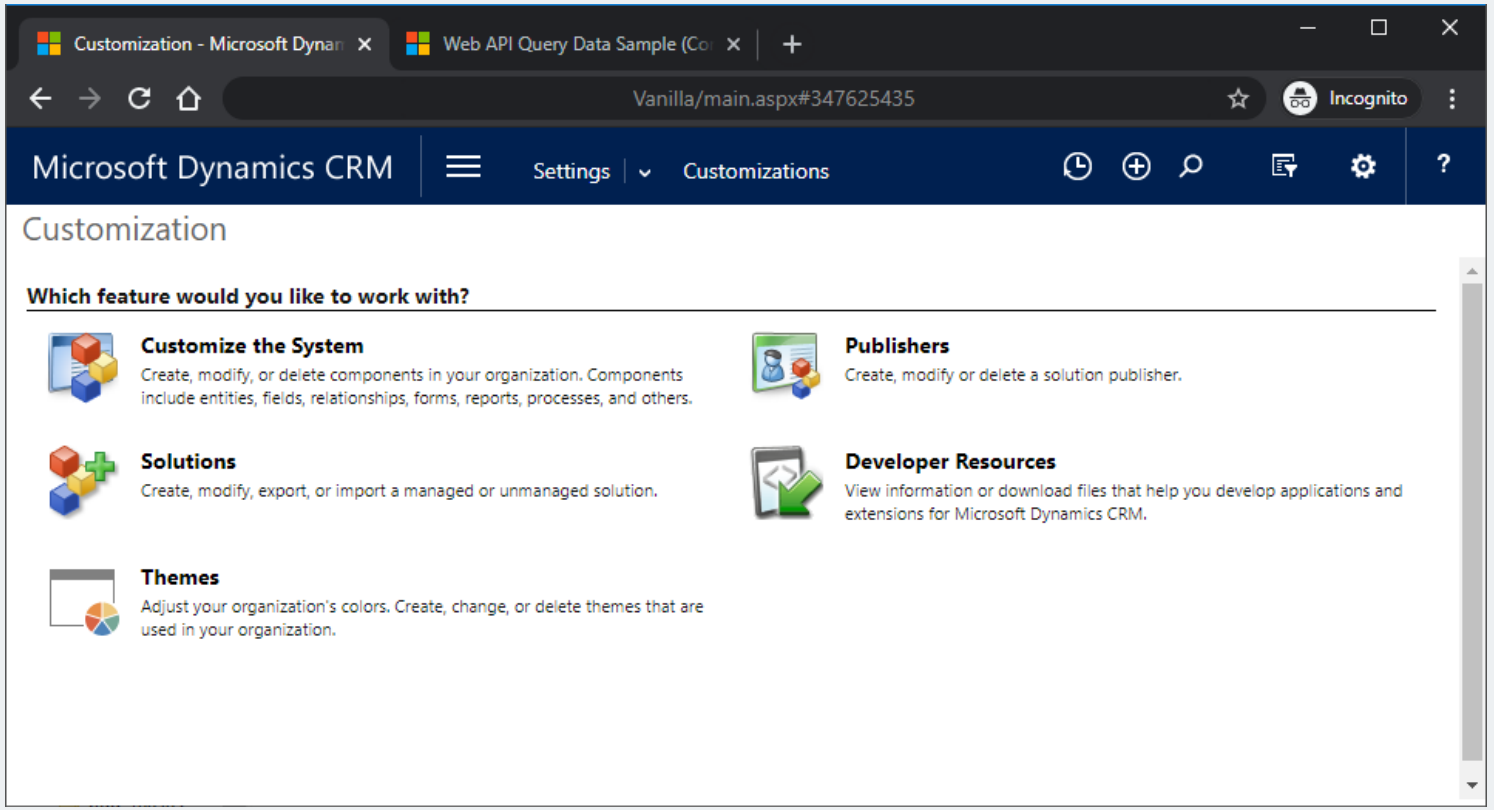
- Pros:
  - **Lightweight**
  - **No dependencies** to **any assembly**
  - **No dependencies** to **any platform**. Could easily run on a *nix box (iff, Mono or .NET Core are ported)

    Note: Microsoft recommends to use Json.NET - Newtonsoft library
- Cons:
  - **Not** really **type safe**
  - **Not sanitized** commands nor queries (CQRS)
  - Very **tedious to work with** …

DFDS

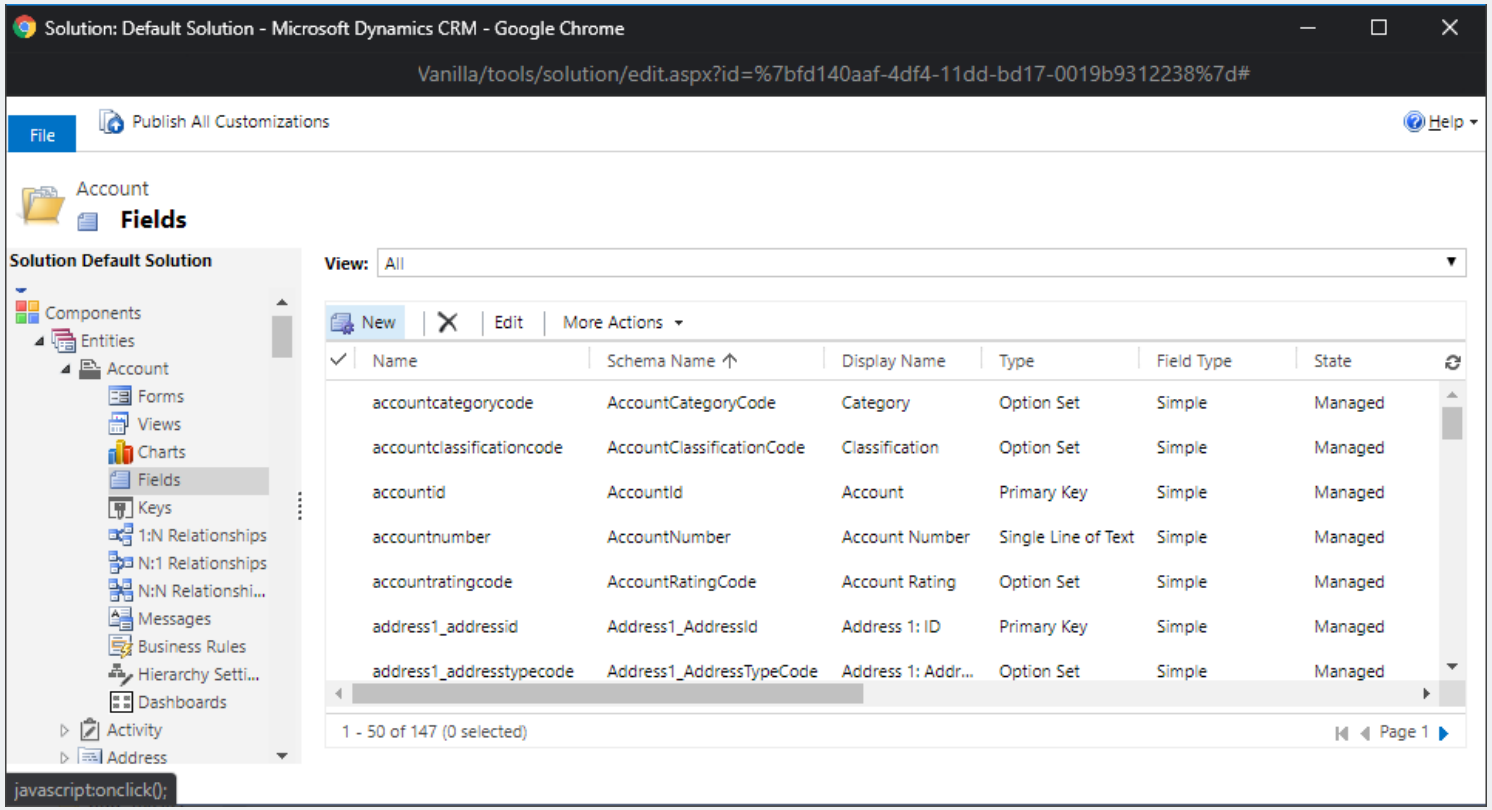
Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > MSDN > Approach > Tedious to work with > Add Lookup



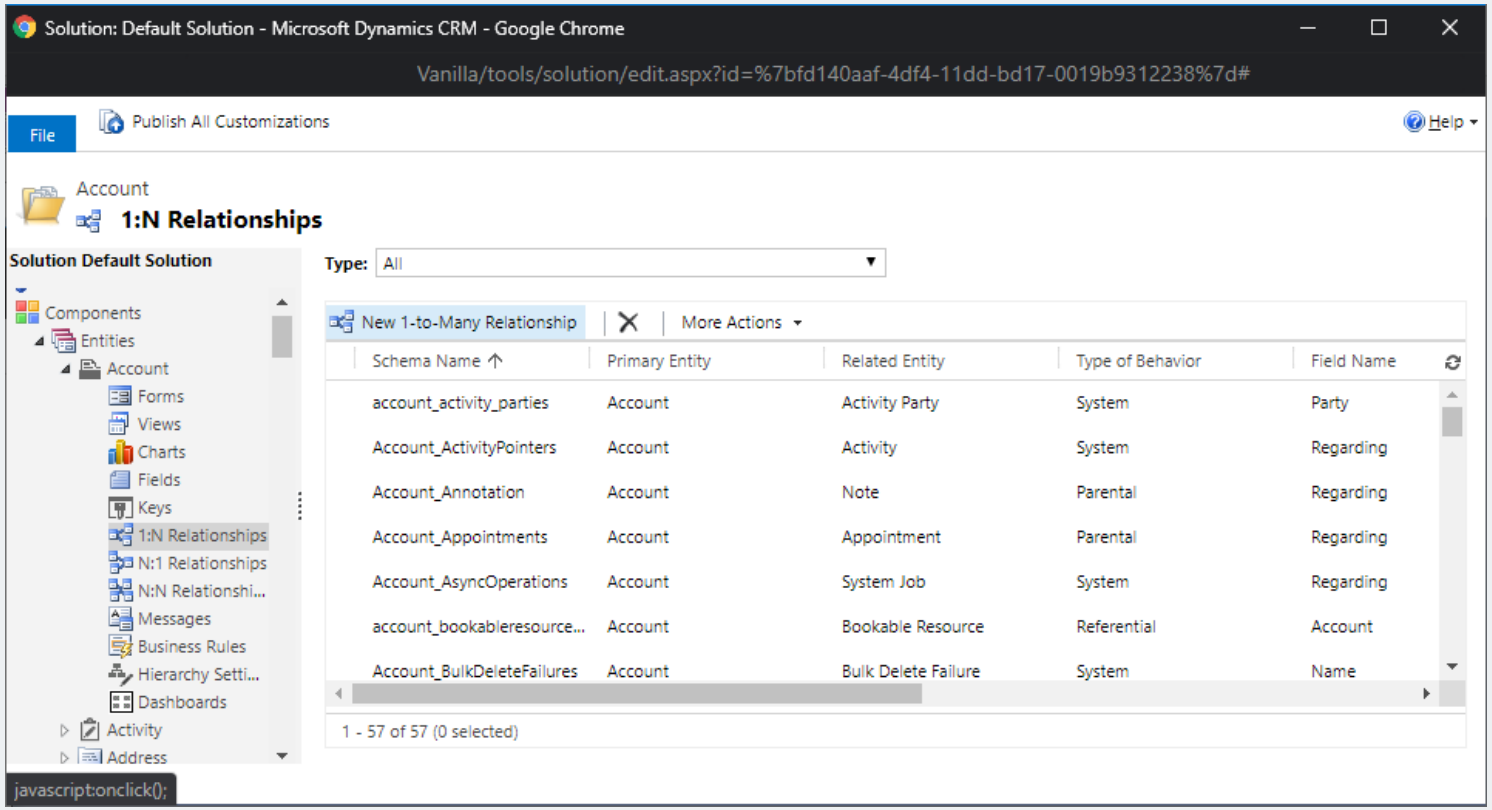
Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > MSDN > Approach > Tedious to work with > Add Lookup



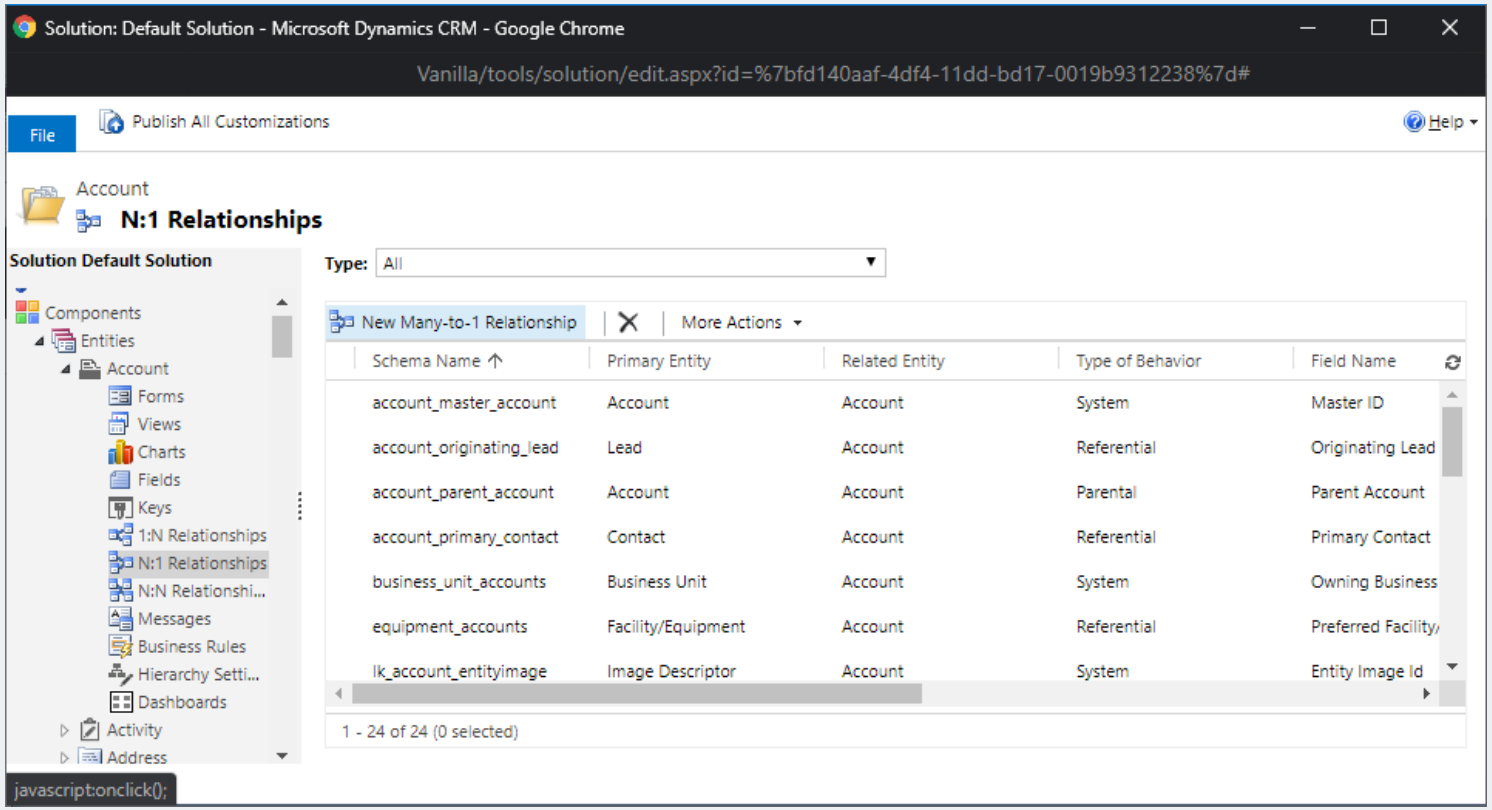
Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > MSDN > Approach > Tedious to work with > Add Lookup



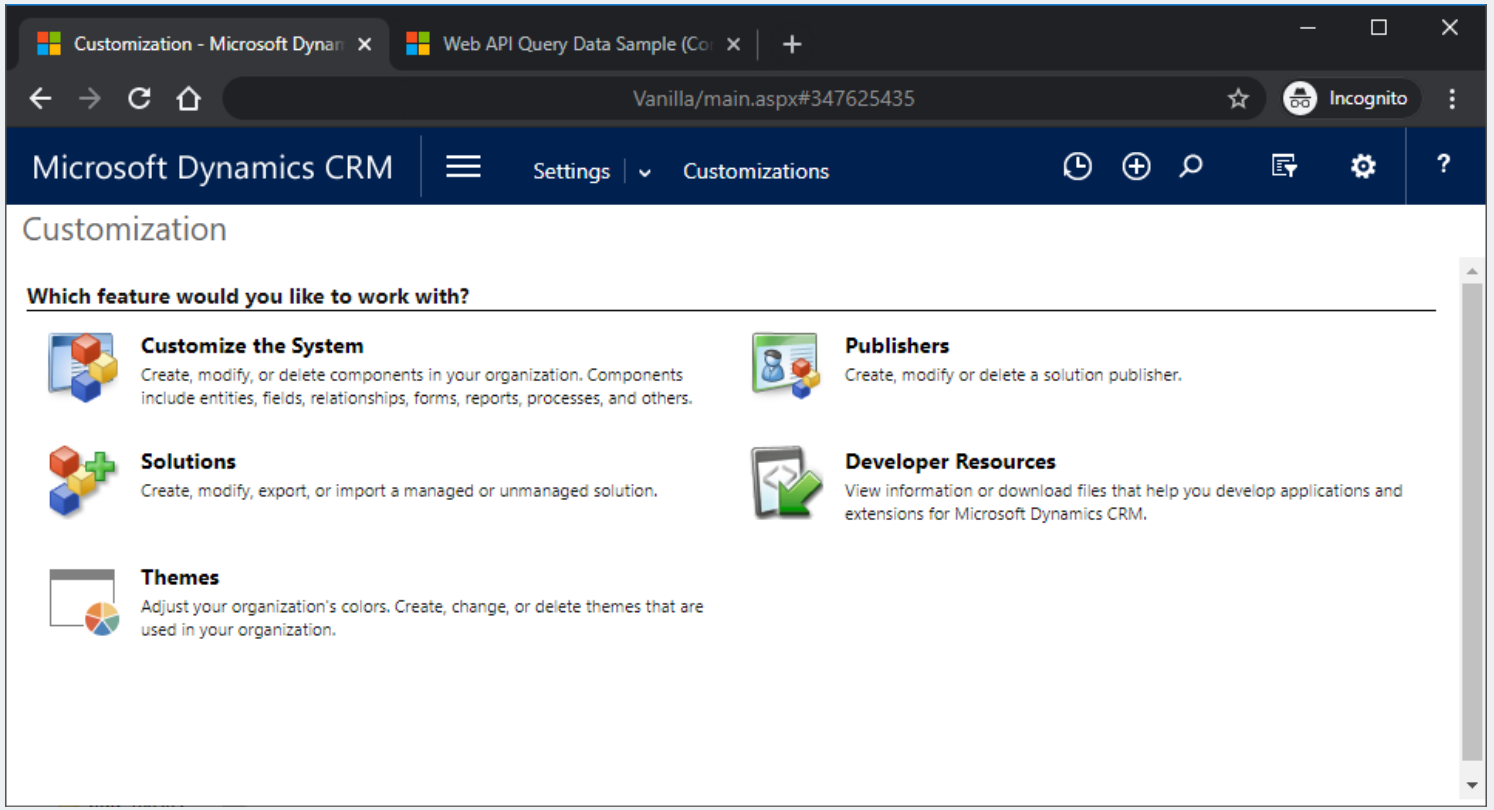
Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > MSDN > Approach > Tedious to work with > Add Lookup



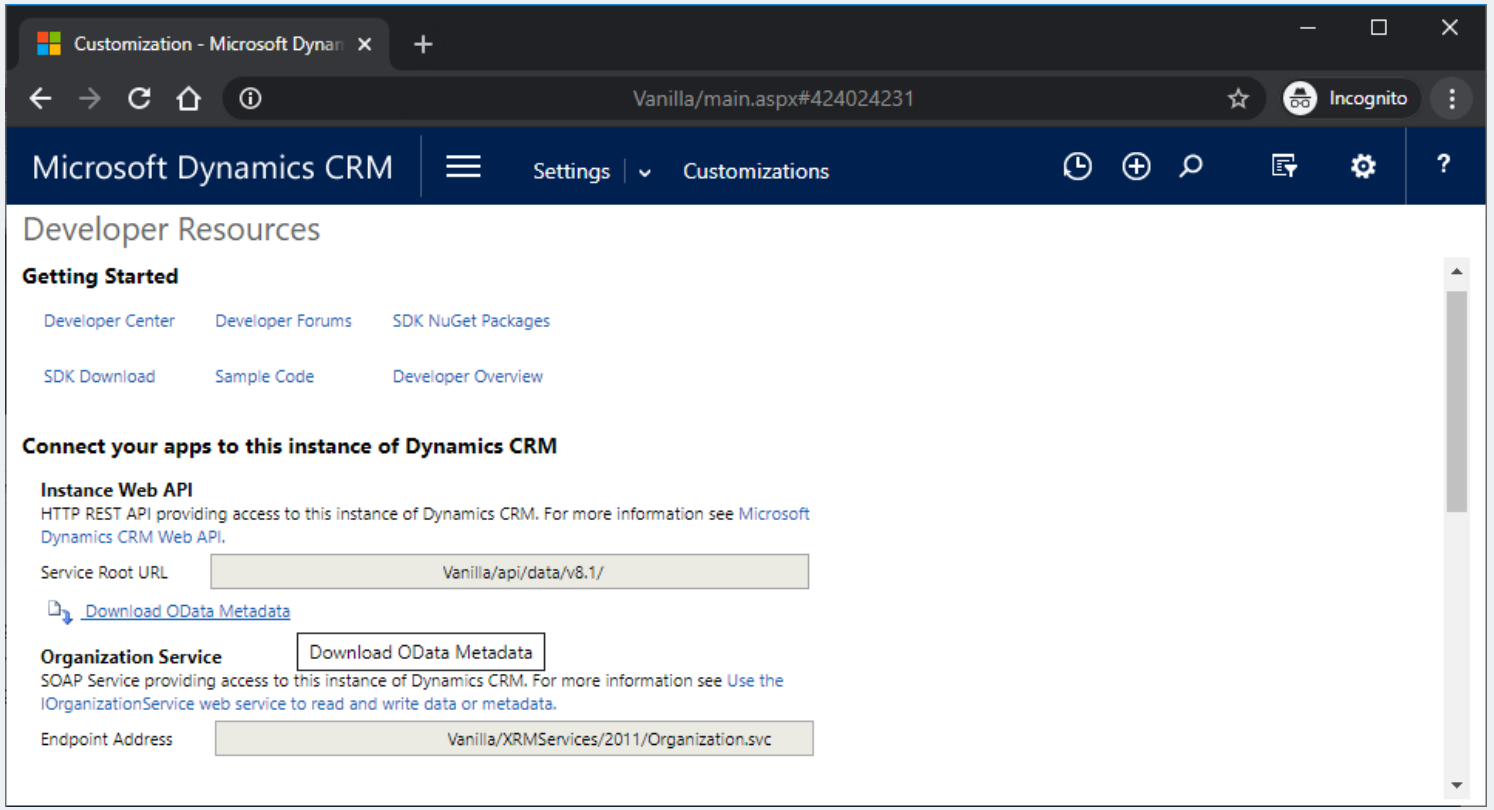
Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > MSDN > Approach > Tedious to work with > Add Lookup



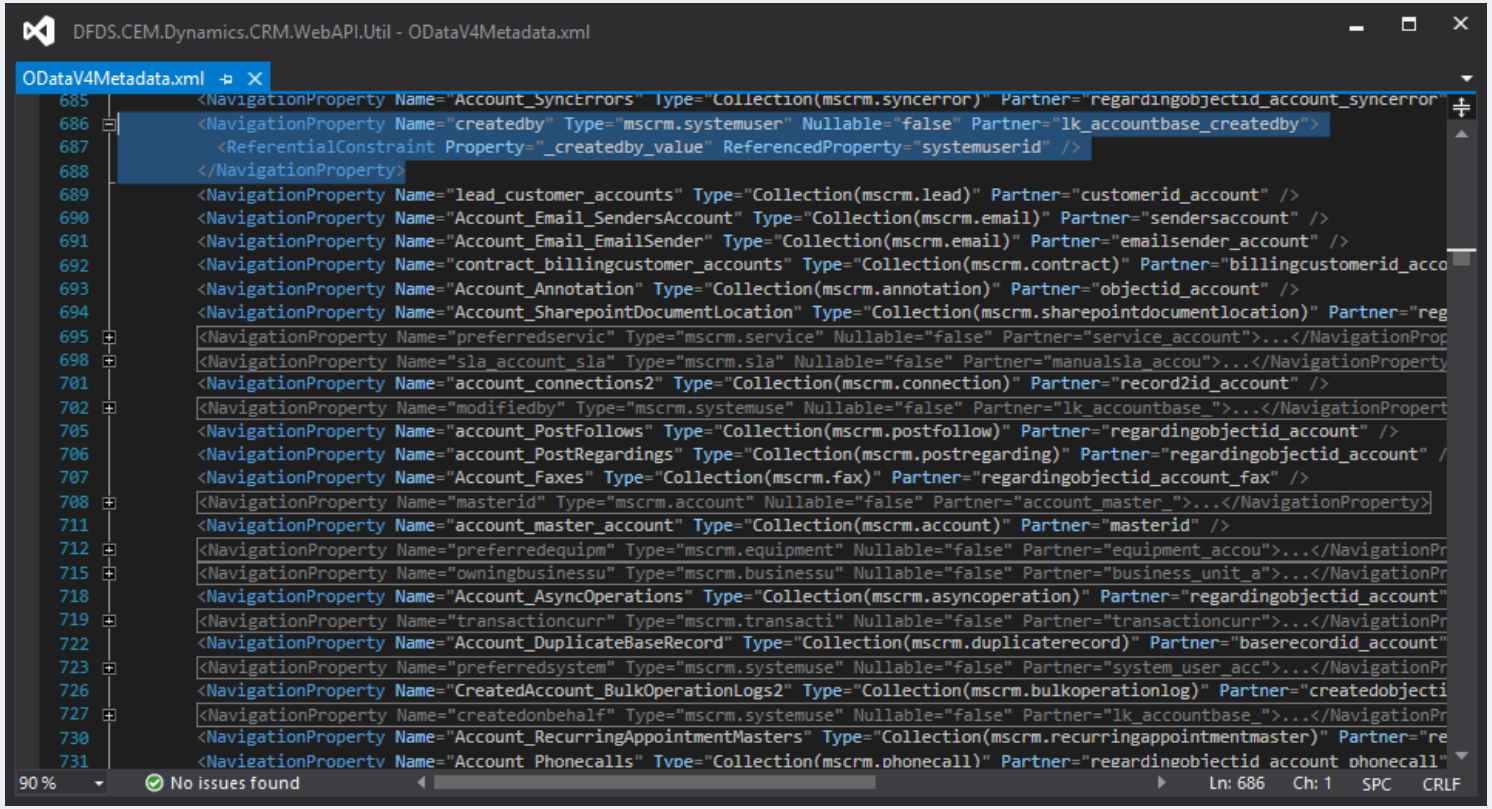
Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > MSDN > Approach > Tedious to work with > Add Lookup



Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > MSDN > Approach > Tedious to work with > Add Lookup



Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > MSDN > Approach > Tedious to work with > Add Lookup

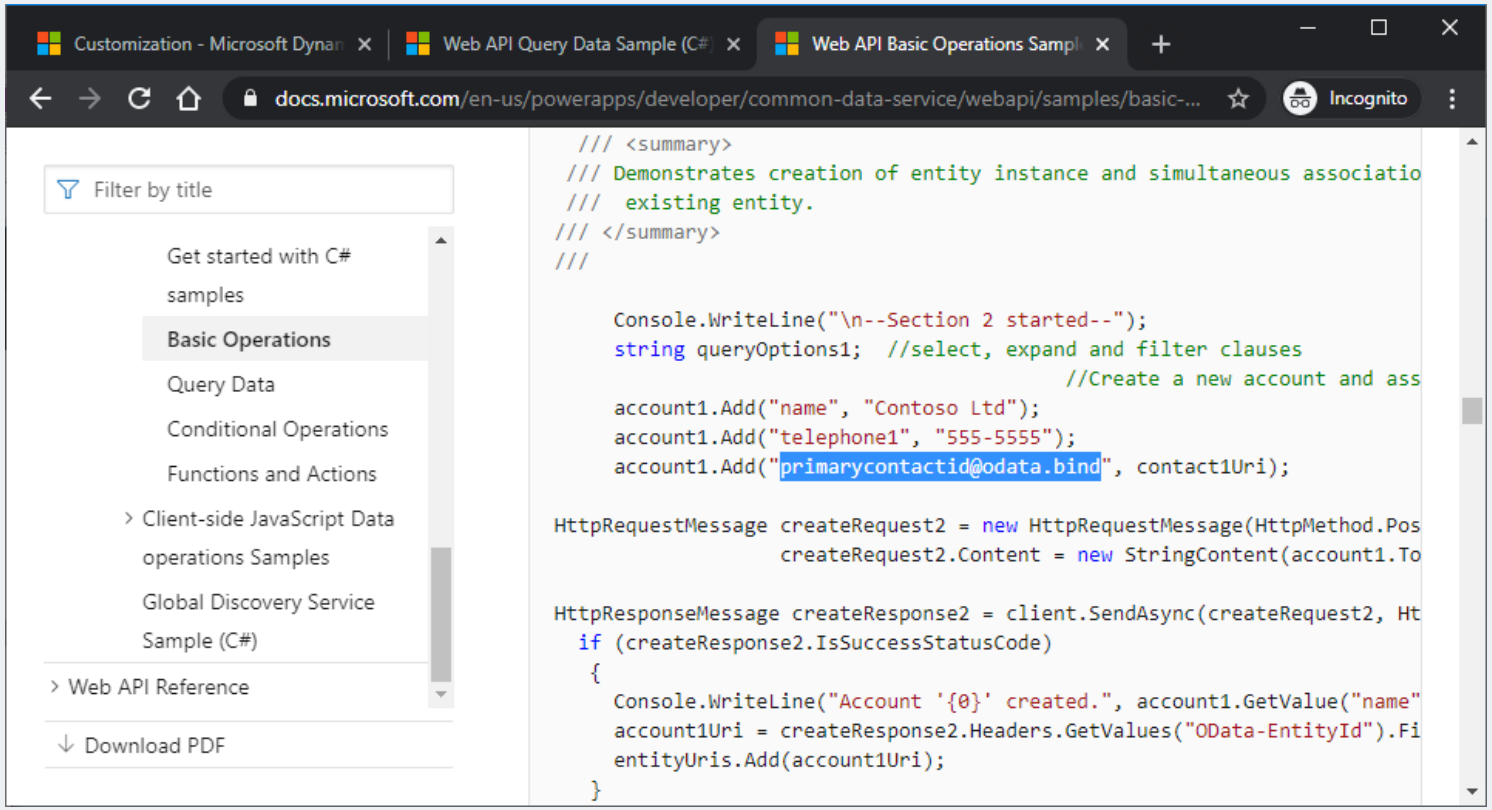


Image source: https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/webapi/samples/basic-operations-csharp

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > MSDN > Approach > Tedious to work with > Add Lookup

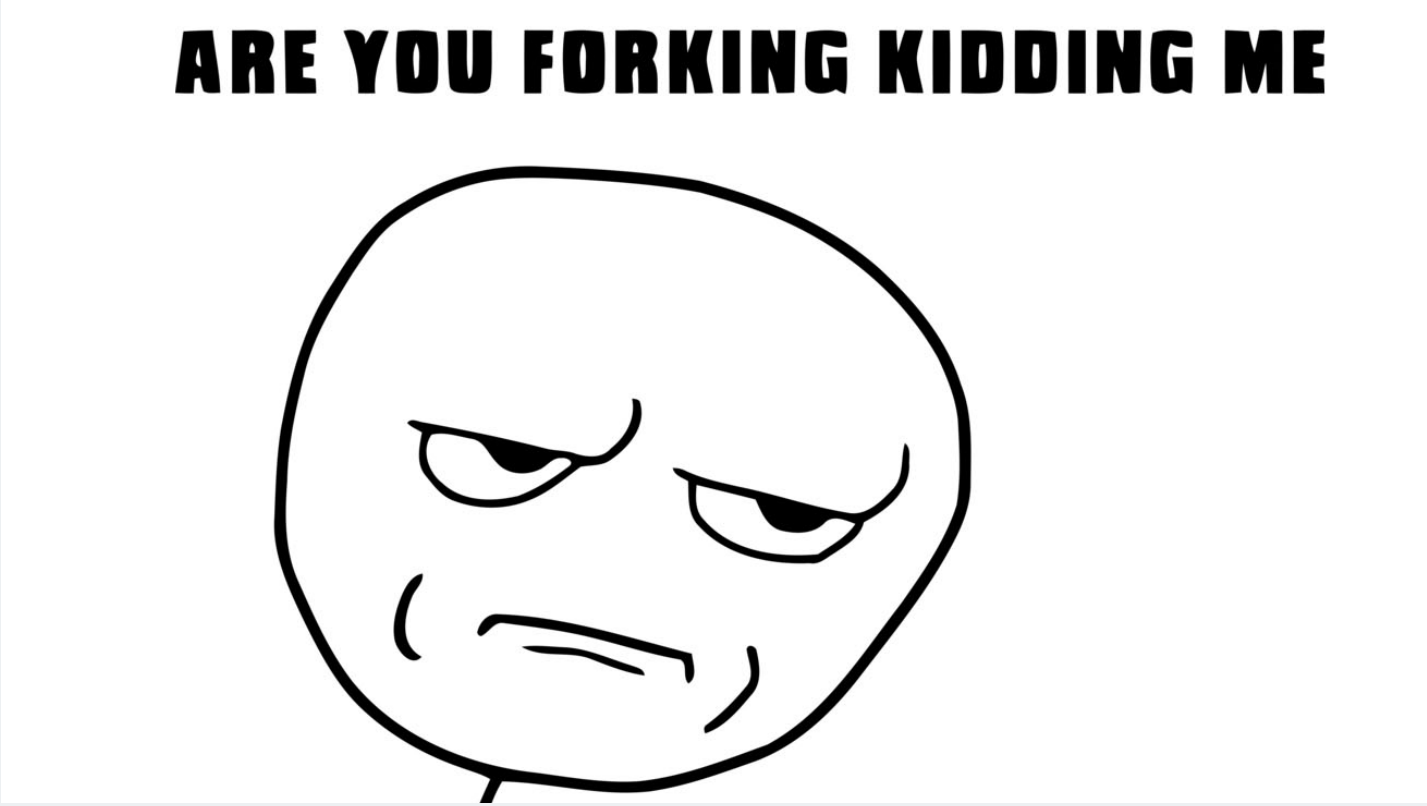


Image source (modded): https://knowyourmeme.com/memes/are-you-forking-kidding-me

DFDS

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > Delegate A/S > XrmDefinitelyTyped

- **XrmDefinitelyTyped** generates **TypeScript** declaration files based on your Dynamics 365/CRM/xRM solution:
  - https://github.com/delegateas/XrmDefinitelyTyped/wiki
- It is the **TypeScript** equivalent of **CrmSvcUtil**, but instead of generating early-bound .NET classes for server-side code, it generates TypeScript interfaces for all your client-side coding.
- Maintained by **Delegate A/S** and accessible through **NuGet**

Tool Source: https://github.com/delegateas/XrmDefinitelyTyped

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > Delegate A/S > XrmDefinitelyTyped

- Pros:
  - **Lightweight**, as generated declaration files will never be part of the final output
  - **Generation of types** for custom xRM data entities created in the CRM application instances
  - **Ideal** to use with **CRM front-end development** instead of JavaScript
- Cons:
  - **Slow**, as TypeScript is transpiled to JavaScript which means that we only have access to single thread:
    - Welcome back to 2001 (Power4 was the first mainstream computer processor with more than one core on a single die)
    - Or welcome to 2020, as the OCaml community would say ☺ (shots fired)
  - **Runtime** infrastructure: **Dependency** to **NPM** to execute outputted JavaScript
  - **Development** setup: **Dependency** to **MS CRM SDK** assemblies (not .NET Core friendly)
    - https://github.com/delegateas/XrmDefinitelyTyped/blob/gh_master/paket.dependencies#L8-L10
  - **Buggy** as the WebAPI is pretty complicated to understand and it might be an error to implement all of it in a naive way

Tool Source: https://github.com/delegateas/XrmDefinitelyTyped

DFDS

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > Ideally what would we like to have?

- Generate **type-safe C# classes** for xRM data entities of **any CRM application instance**
- Support for **LINQ code**
- Data **transformation from and to JSON** should be done **automatically** (sanitization)
- **Lightweight** size of generated types
- **Concurrency** / **parallelism**
- Support **.NET Core** v.3.1 (LTS)
  - Runtime infrastructure
  - Development setup
- **Few dependencies**
  - At most, Json.NET - Newtonsoft library, as Microsoft recommends to use it
  - None to the xRM SDKs as they don't support .NET Core
- **No SDK 2011 endpoint**
  - Only **/api/data/v8.1/** and above must be used

DFDS

Microsoft CRM / Dynamics 365 Customer Engagement

Background > Platform > Architecture > REST / WebAPI (JSON) > Ideally what would we like to have? (Missed since 2016)

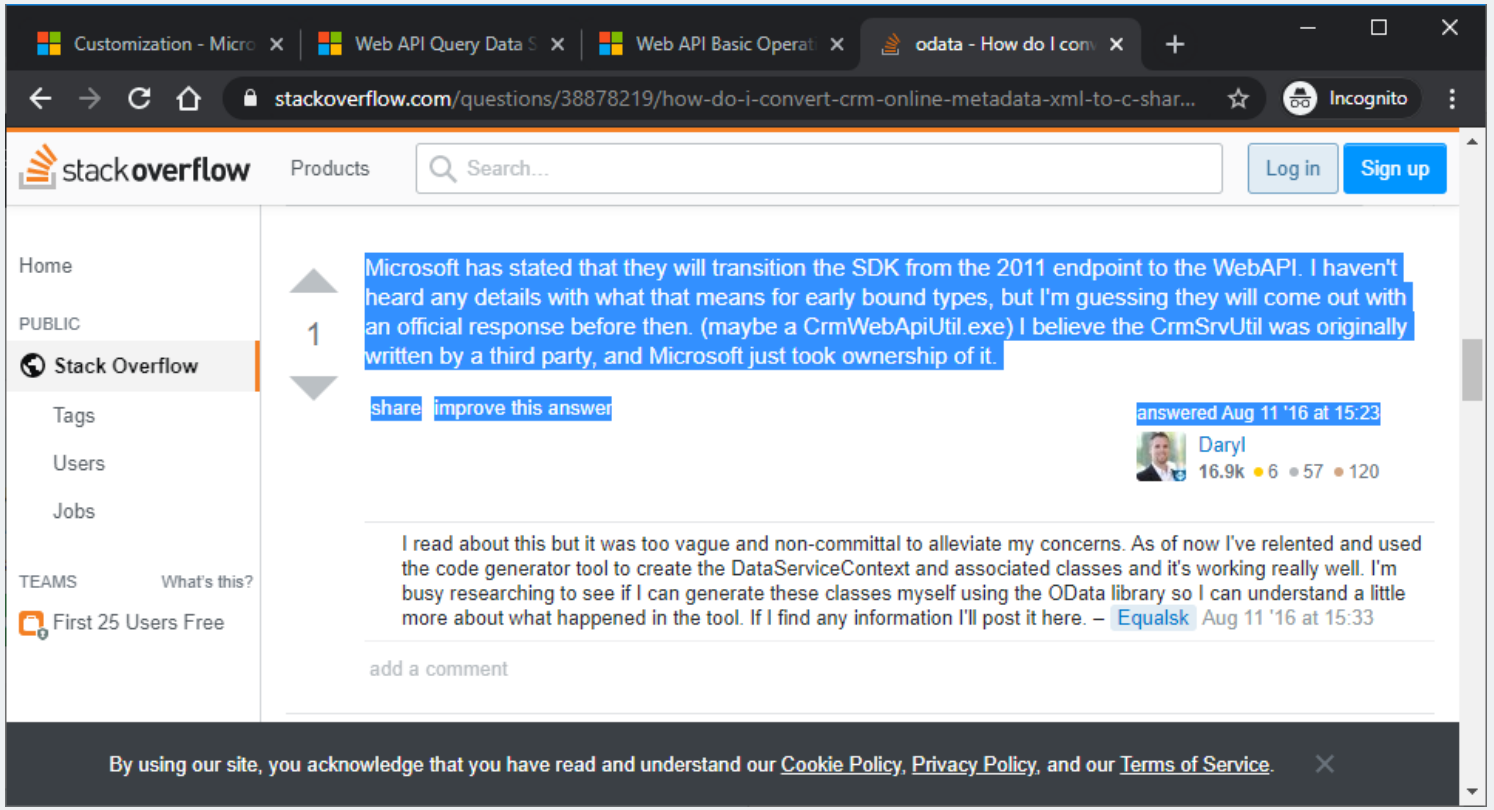
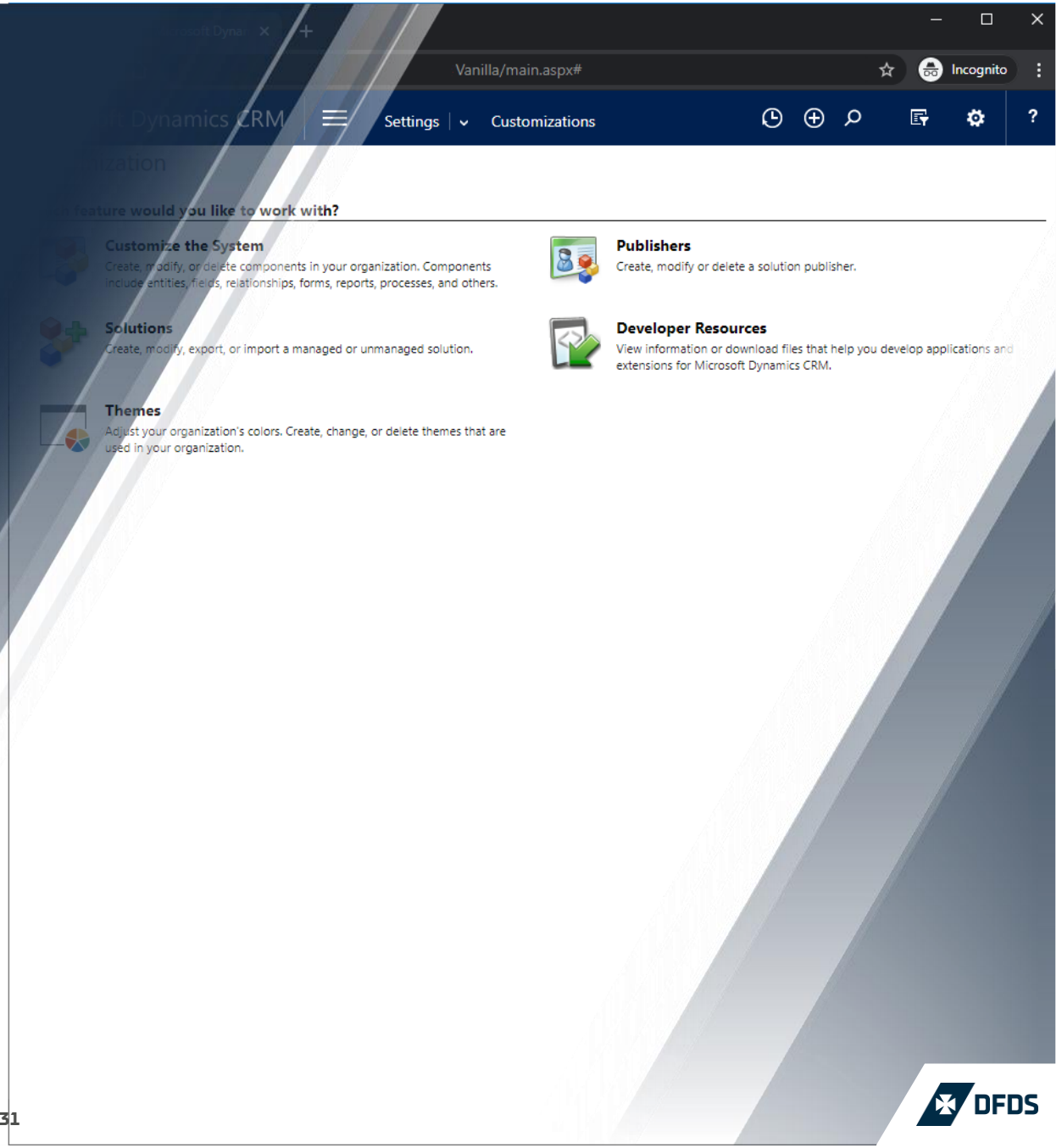


Image Source: https://stackoverflow.com/a/38900408

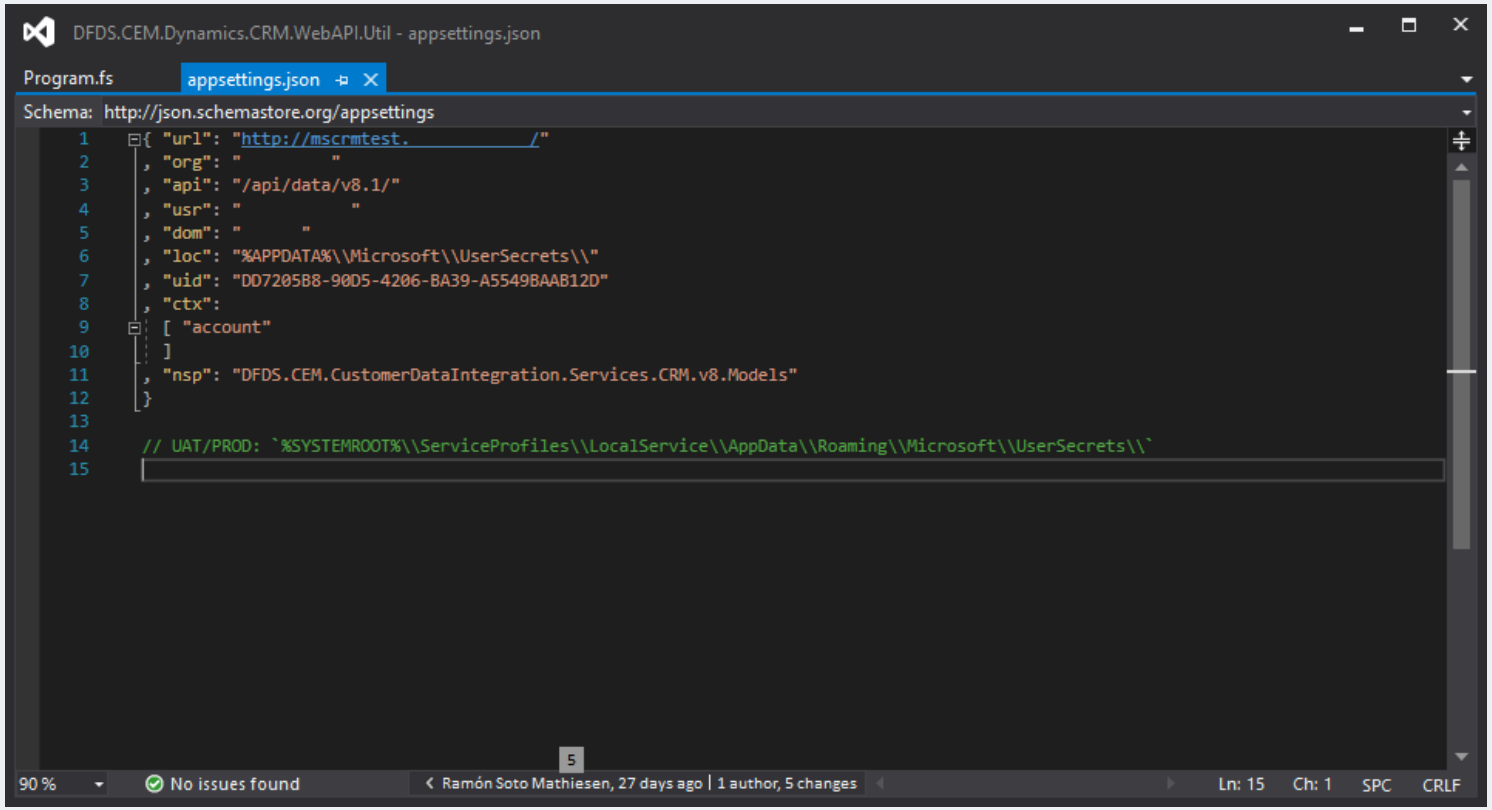
CRM WebAPI Util



DFDS

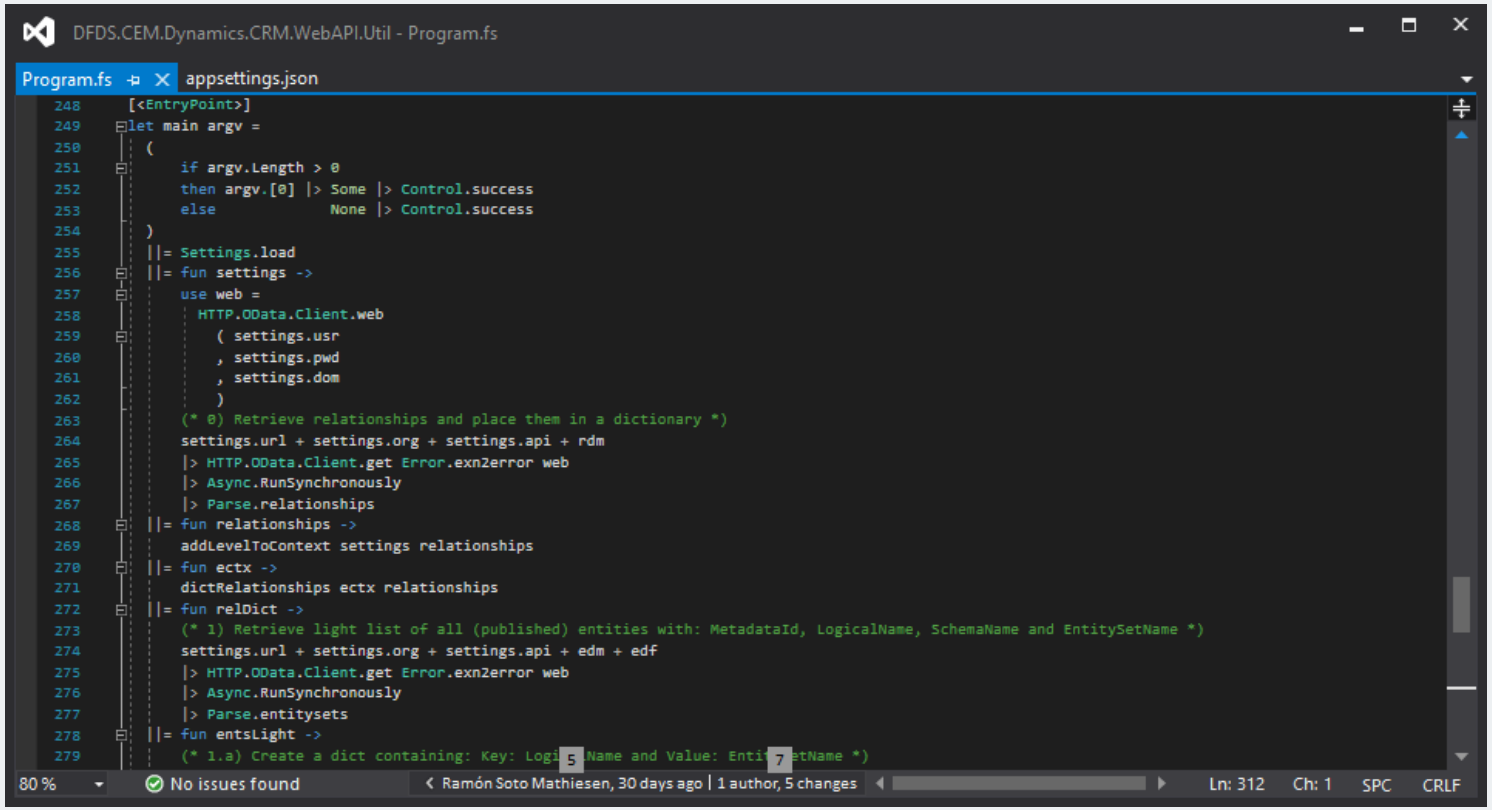
CRM WebAPI Util > DFDS.CEM.Dynamics.CRM.WebAPI.Util (bin)

CRM WebAPI Util > appsettings.json



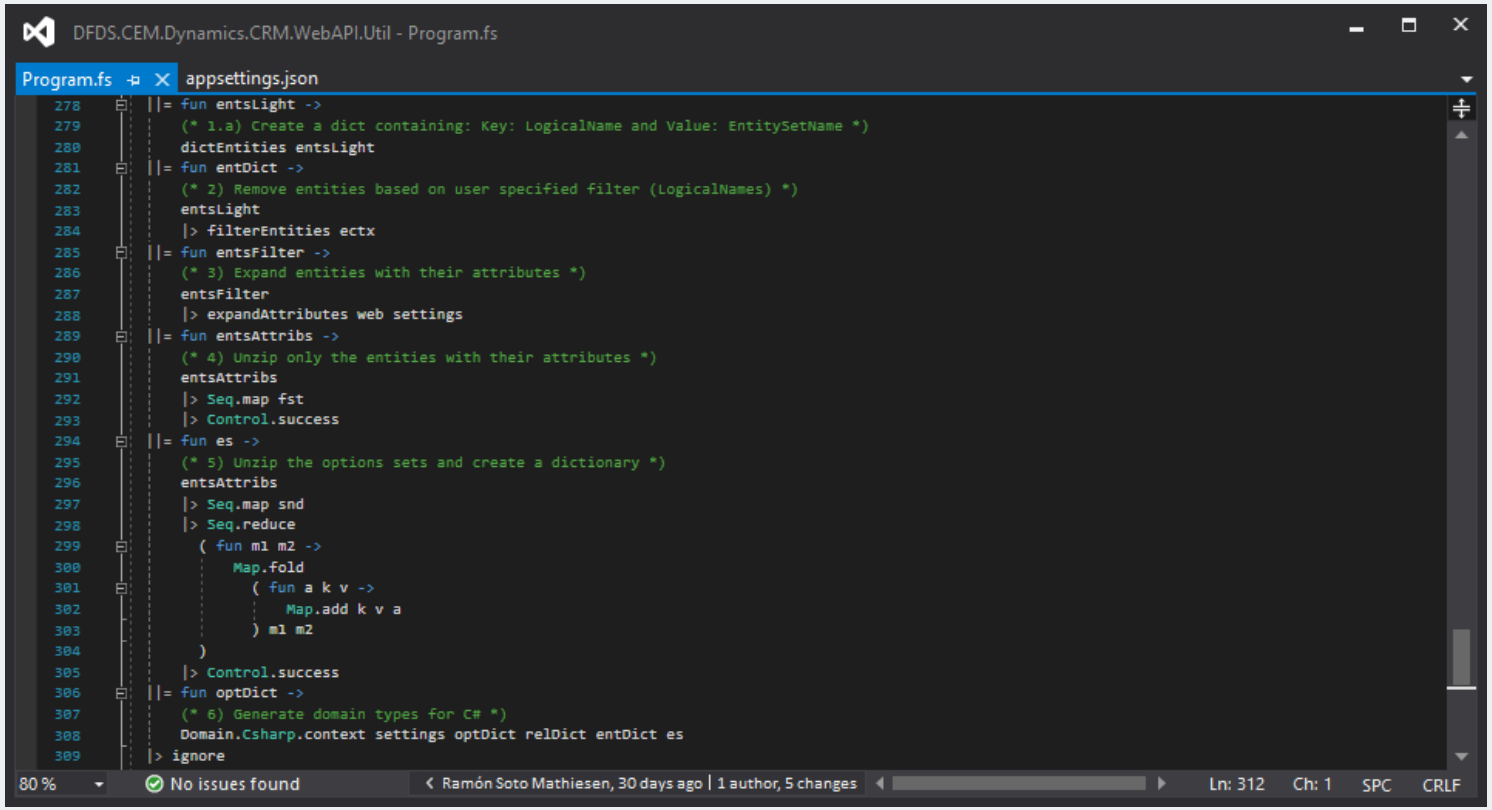
CRM WebAPI Util > DFDS.CEM.Dynamics.CRM.WebAPI.Util (bin)

CRM WebAPI Util > Main functionality (1/2)



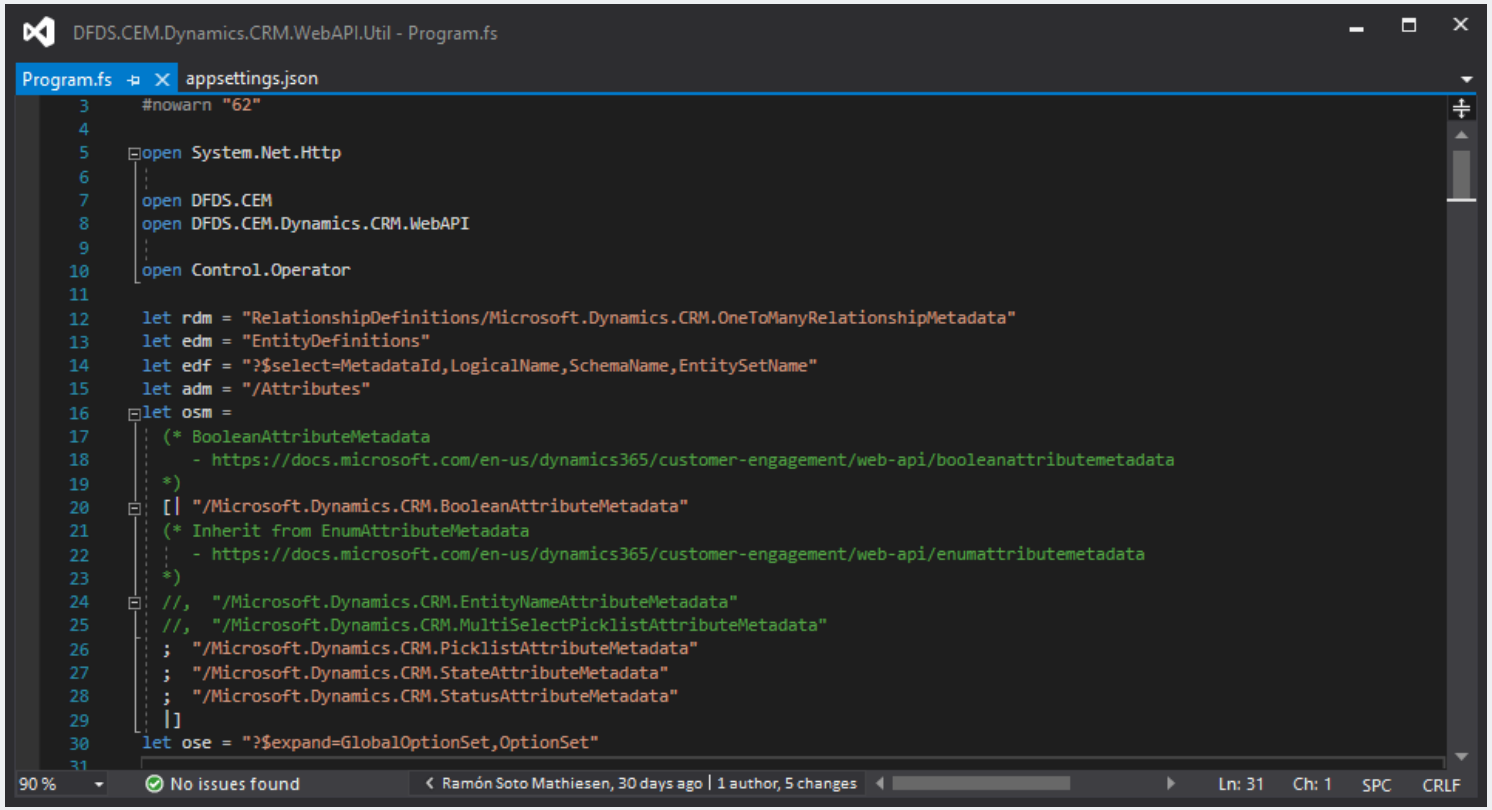
CRM WebAPI Util > DFDS.CEM.Dynamics.CRM.WebAPI.Util (bin)

CRM WebAPI Util > Main functionality (2/2)



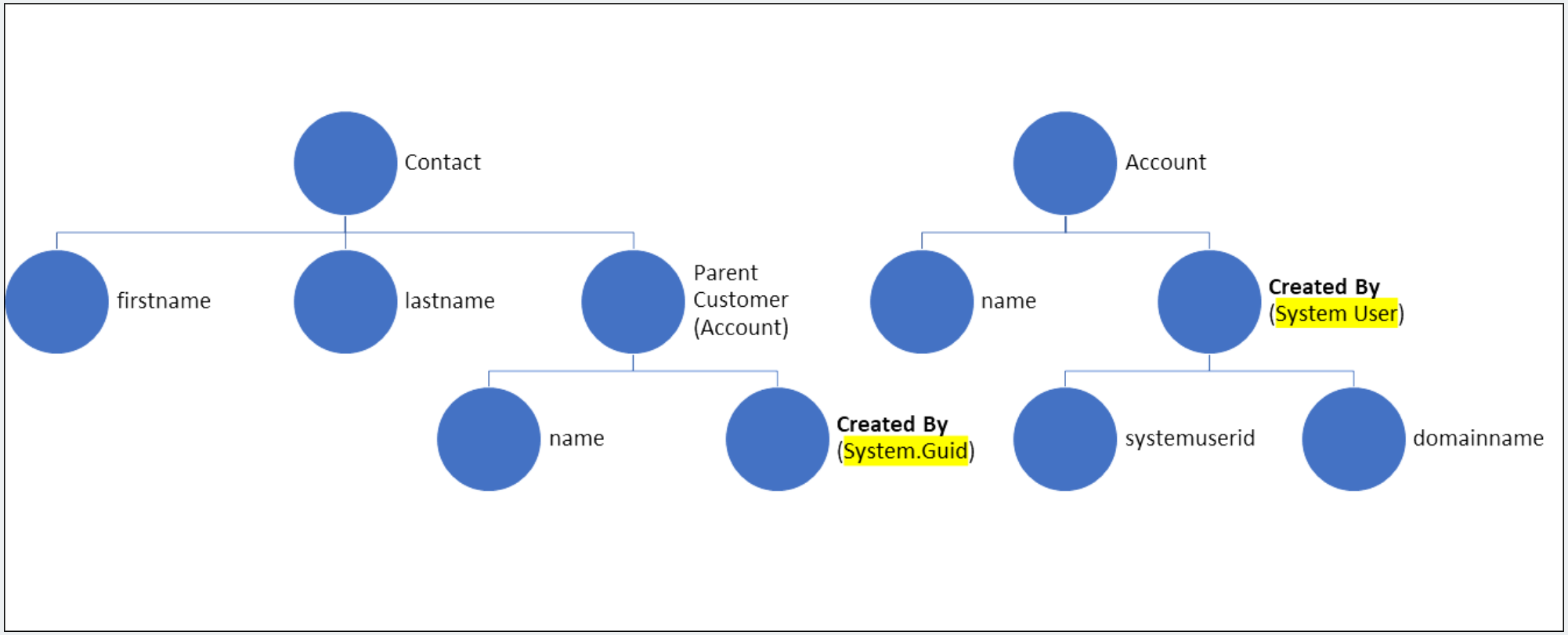
CRM WebAPI Util > DFDS.CEM.Dynamics.CRM.WebAPI.Util (bin)

CRM WebAPI Util > Metadata filters



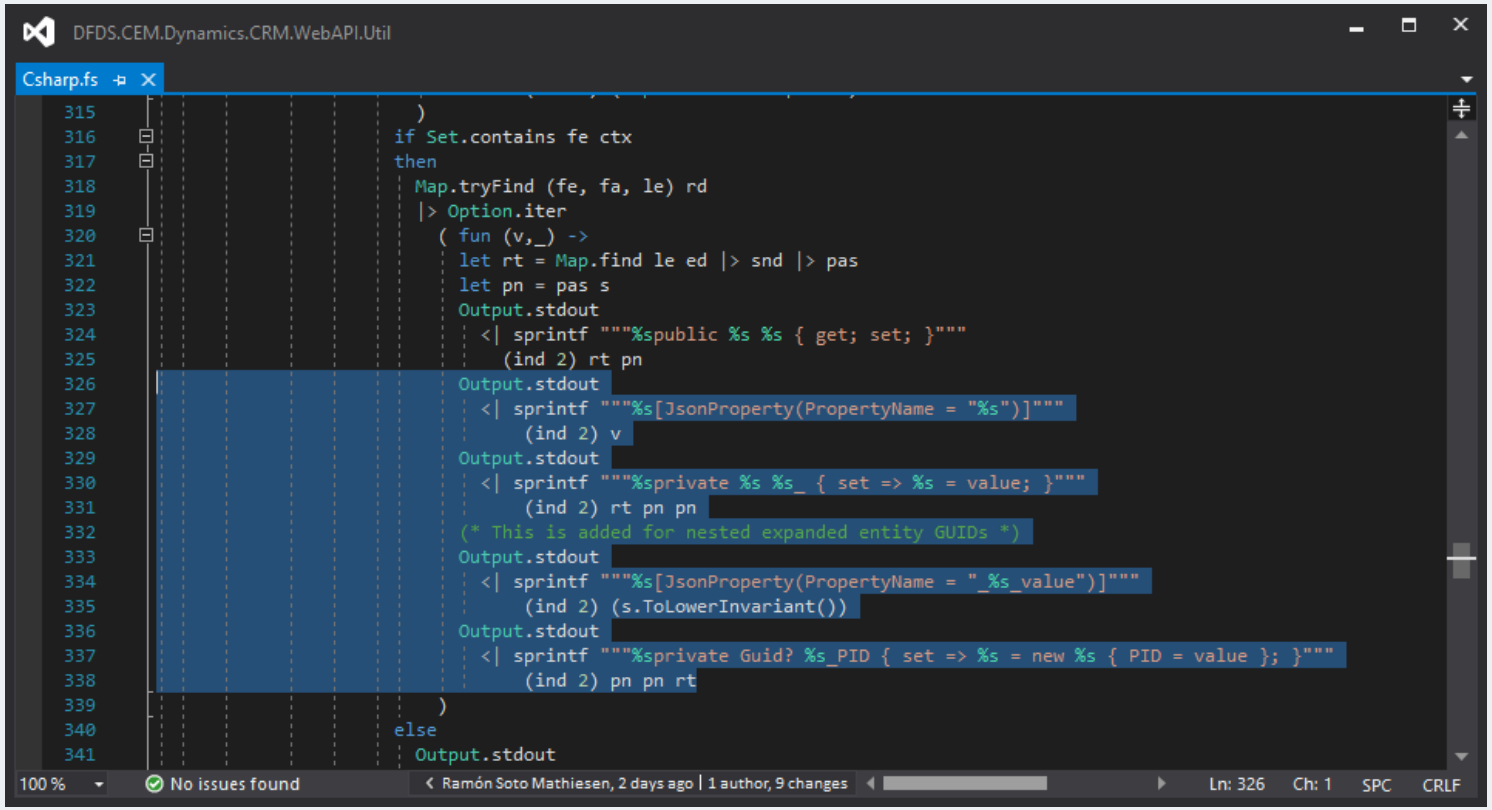
CRM WebAPI Util > DFDS.CEM.Dynamics.CRM.WebAPI.Util (bin)

CRM WebAPI Util > WebAPI Sum types (1/3)



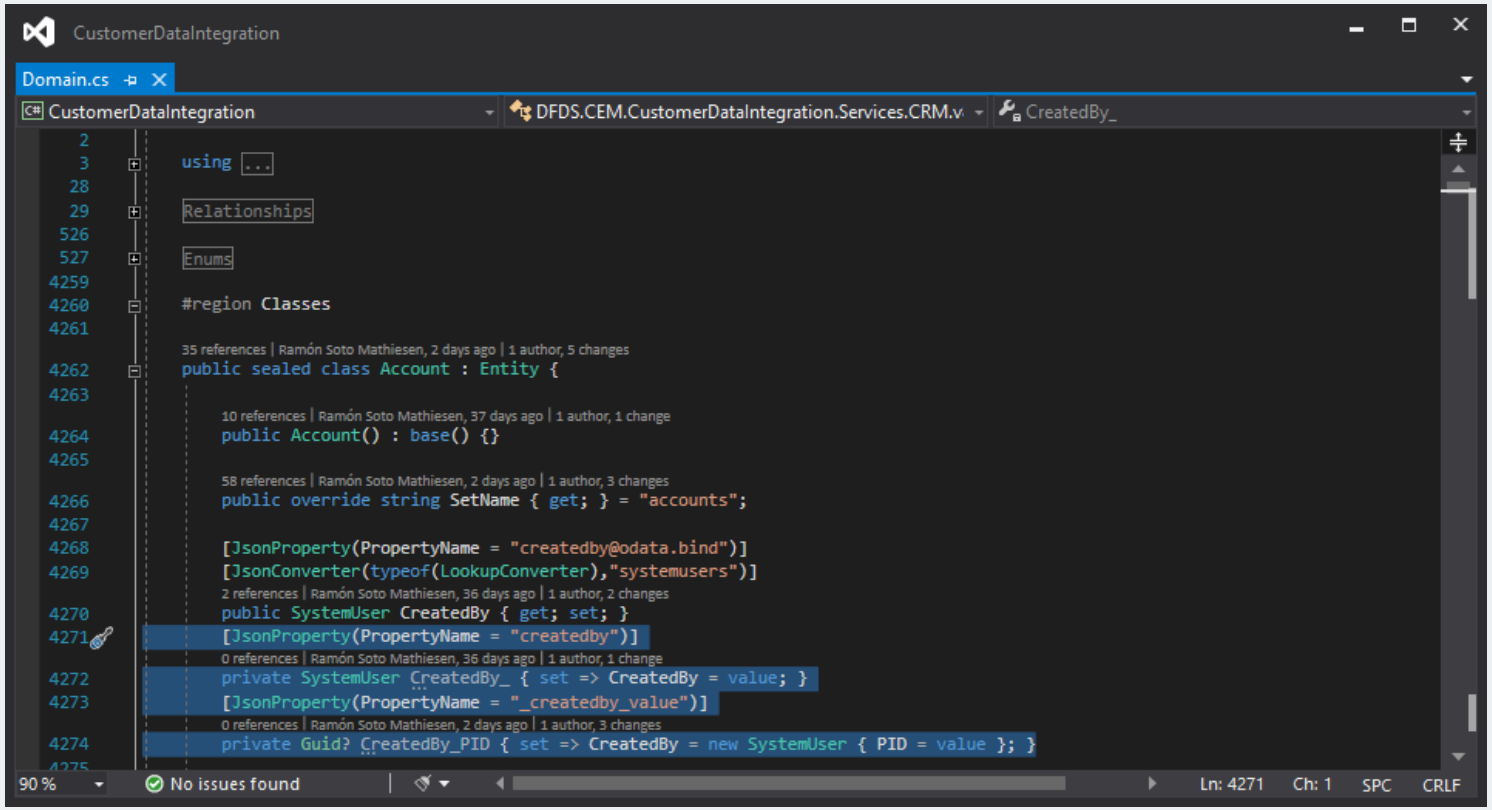
CRM WebAPI Util > DFDS.CEM.Dynamics.CRM.WebAPI.Util (bin)

CRM WebAPI Util > WebAPI Sum types (2/3) (Json.NET - Newtonsoft and OOPD encapsulation to the rescue)



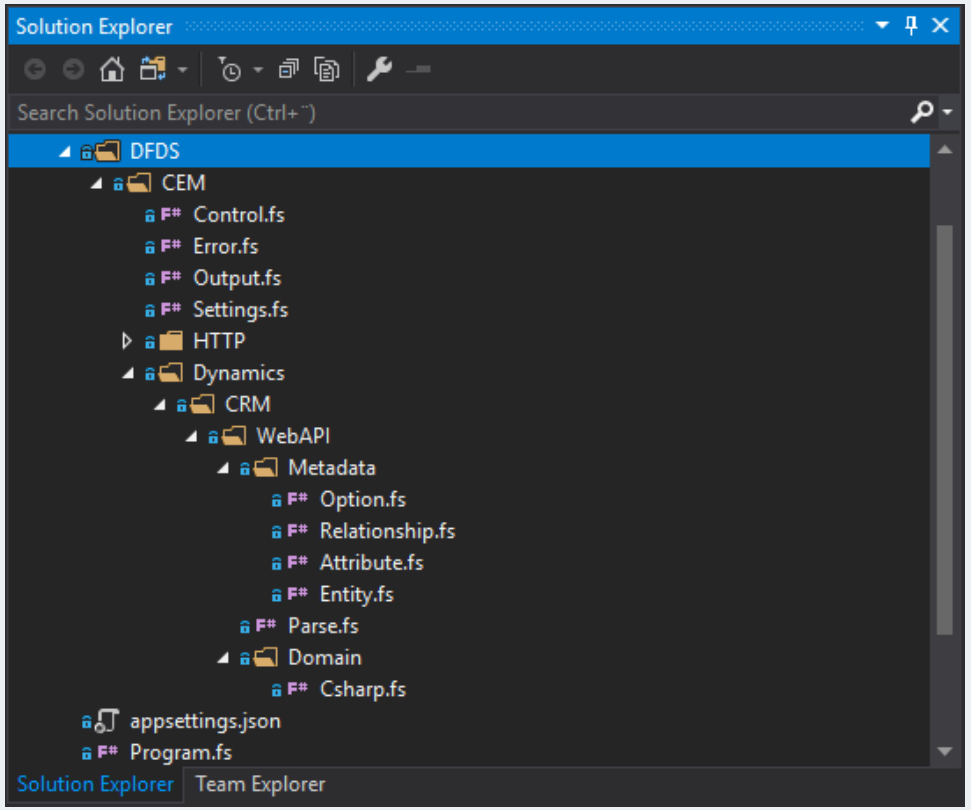
CRM WebAPI Util > DFDS.CEM.Dynamics.CRM.WebAPI.Util (bin)

CRM WebAPI Util > WebAPI Sum types (3/3) (Json.NET - Newtonsoft and OOPD encapsulation to the rescue)



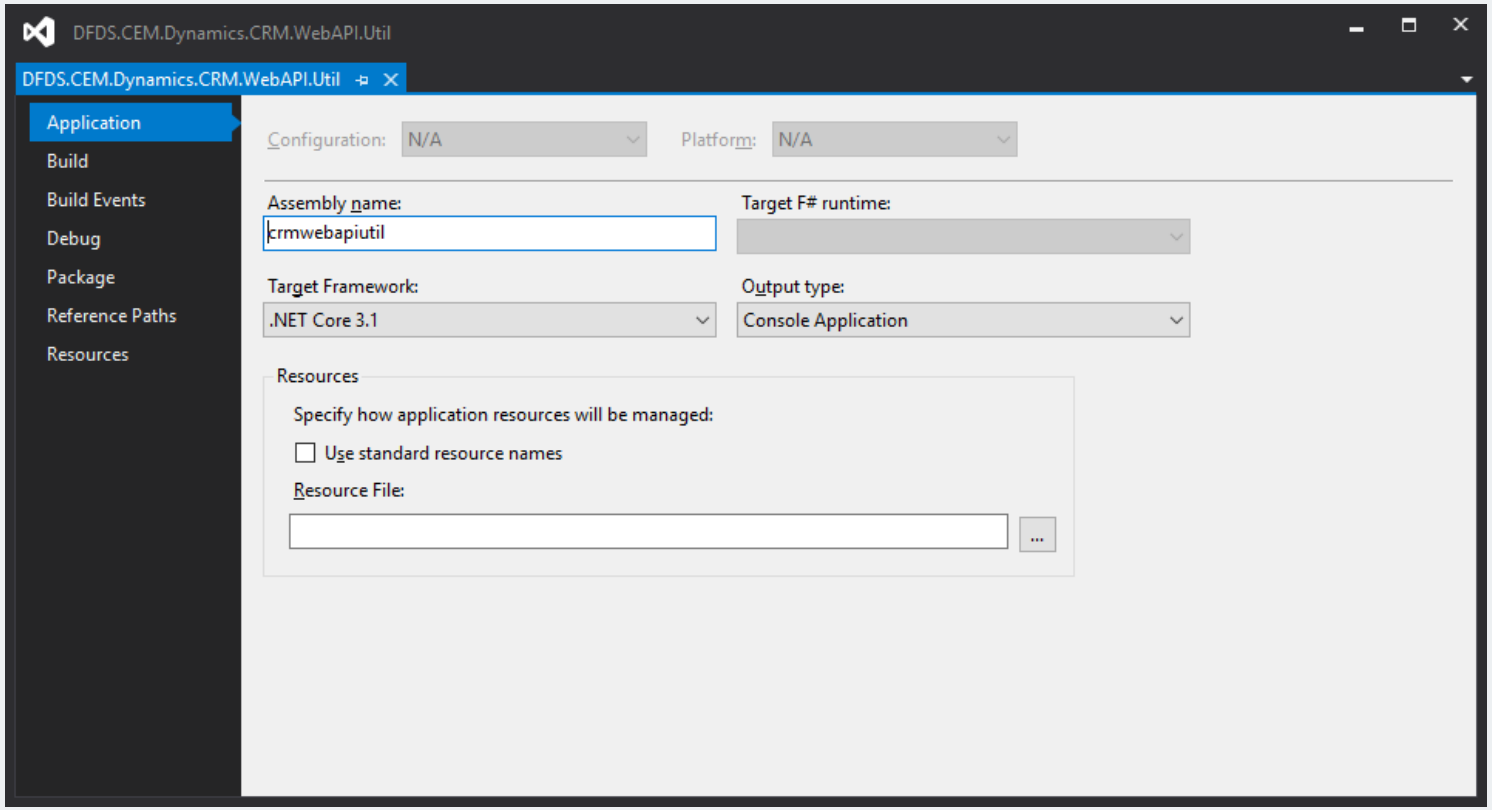
CRM WebAPI Util > DFDS.CEM.Dynamics.CRM.WebAPI.Util (bin)

CRM WebAPI Util > (simple) Visual Studio (F#) Project



CRM WebAPI Util > DFDS.CEM.Dynamics.CRM.WebAPI.Util (bin)

CRM WebAPI Util > (simple) Visual Studio (F#) Project > .NET Core v.3.1 (LTS)



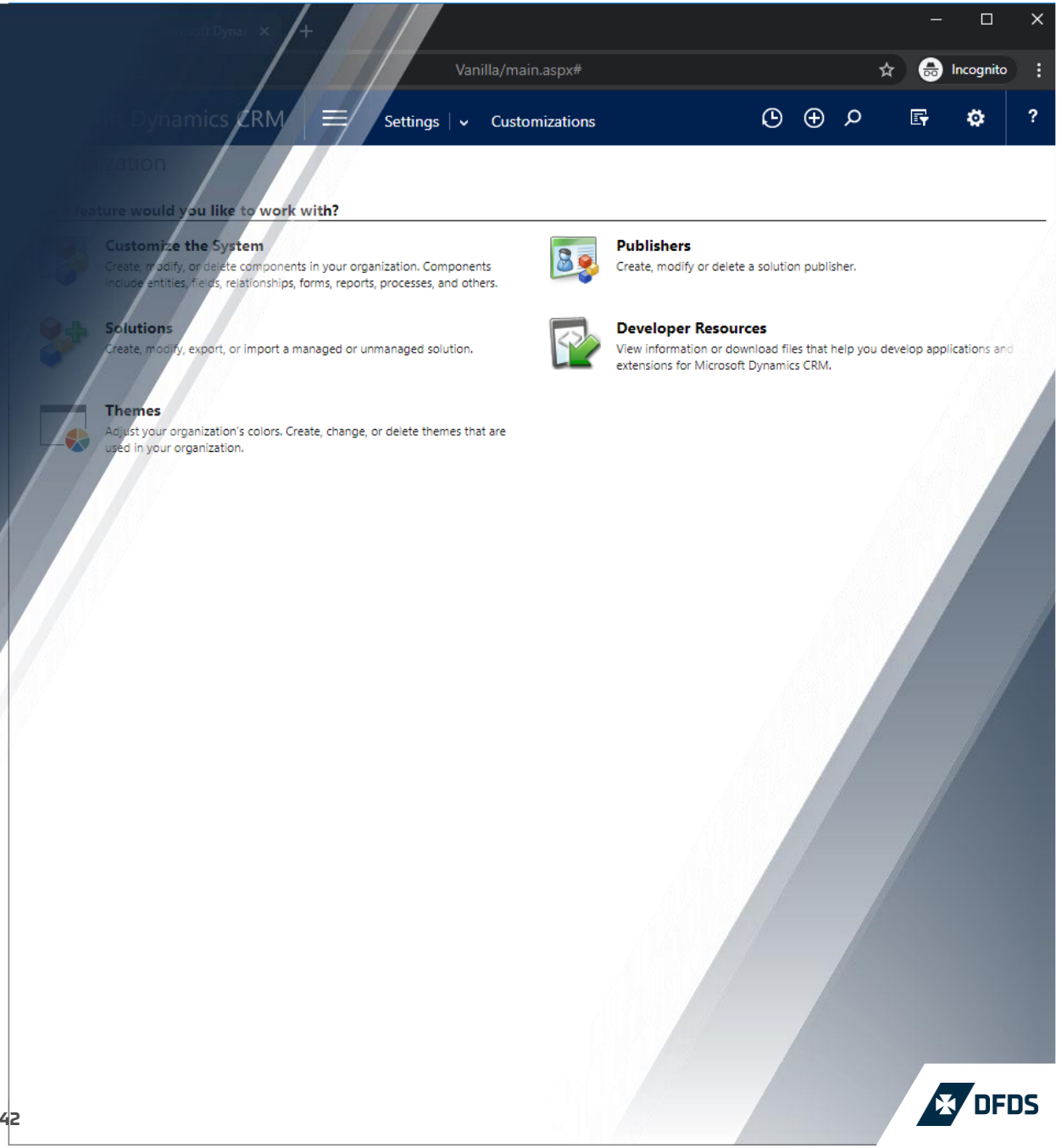
CRM WebAPI Util > DFDS.CEM.Dynamics.CRM.WebAPI.Util (bin)

Background > Platform > Architecture > REST / WebAPI (JSON) > Ideally what would we like to have?

- ✓ Generate **type-safe C# classes** for xRM data entities of **any CRM application instance**
- Support for **LINQ code**
- ✓ Data **transformation from and to JSON** should be done **automatically** (sanitization)
- ✓ **Lightweight** size of generated types
- **Concurrency** / **parallelism**
- ✓ Support **.NET Core** v.3.1 (LTS)
  - Runtime infrastructure
  - Development setup
- ✓ **Few dependencies**
  - At most, Json.NET - Newtonsoft library, as Microsoft recommends to use it
  - None to the xRM SDKs as they don't support .NET Core
- ✓ **No SDK 2011 endpoint**
  - Only **/api/data/v8.1/** and above must be used

DFDS

LINQ Provider



DFDS

LINQ Provider > DFDS.CEM.Dynamics.CRM.WebAPI.LINQ (lib)

LINQ Provider > Based on Charlie Calvert's TerraServer Sample

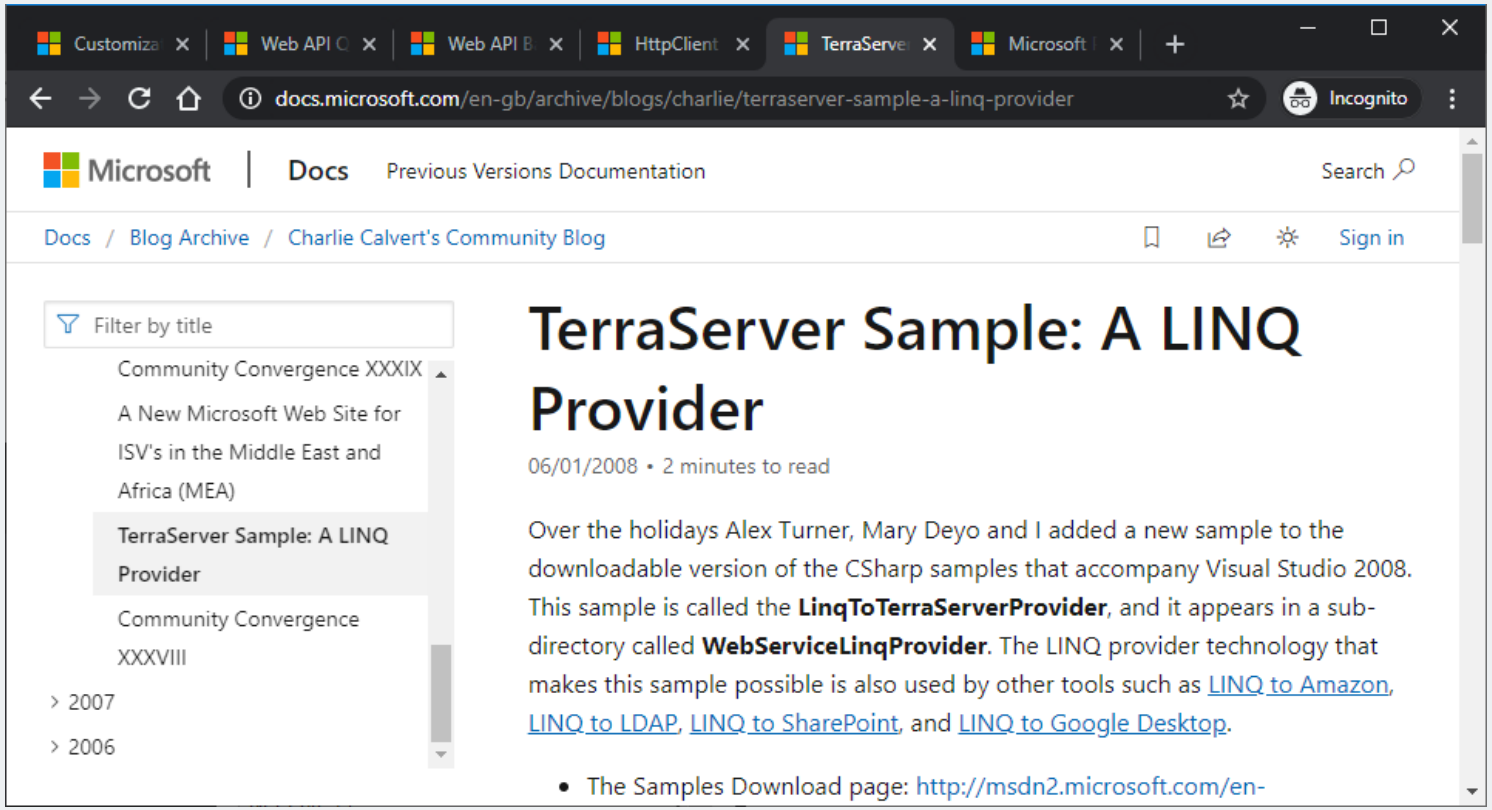


Image source: https://docs.microsoft.com/en-gb/archive/blogs/charlie/terraserver-sample-a-linq-provider

LINQ Provider > DFDS.CEM.Dynamics.CRM.WebAPI.LINQ (lib)

LINQ Provider > Based on Charlie Calvert's TerraServer Sample (Open Source)

Image source: https://docs.microsoft.com/en-us/previous-versions/bb894665(v=msdn.10)?redirectedfrom=MSDN

LINQ Provider > DFDS.CEM.Dynamics.CRM.WebAPI.LINQ (lib)

LINQ Provider > Queries > Supported operations

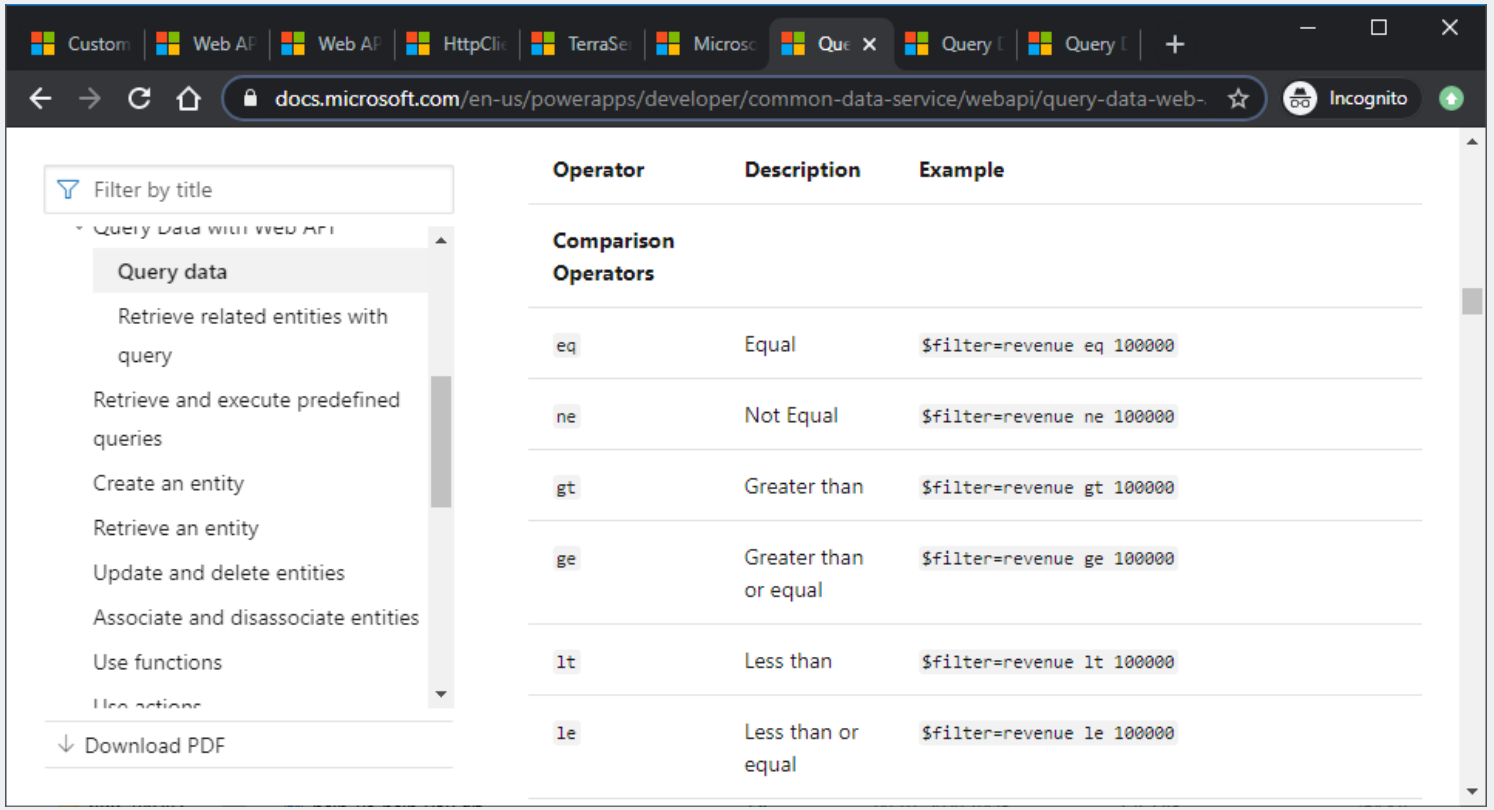


Image source: https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/webapi/query-data-web-api#standard-filter-operators

LINQ Provider > DFDS.CEM.Dynamics.CRM.WebAPI.LINQ (lib)

LINQ Provider > Queries > Supported operations

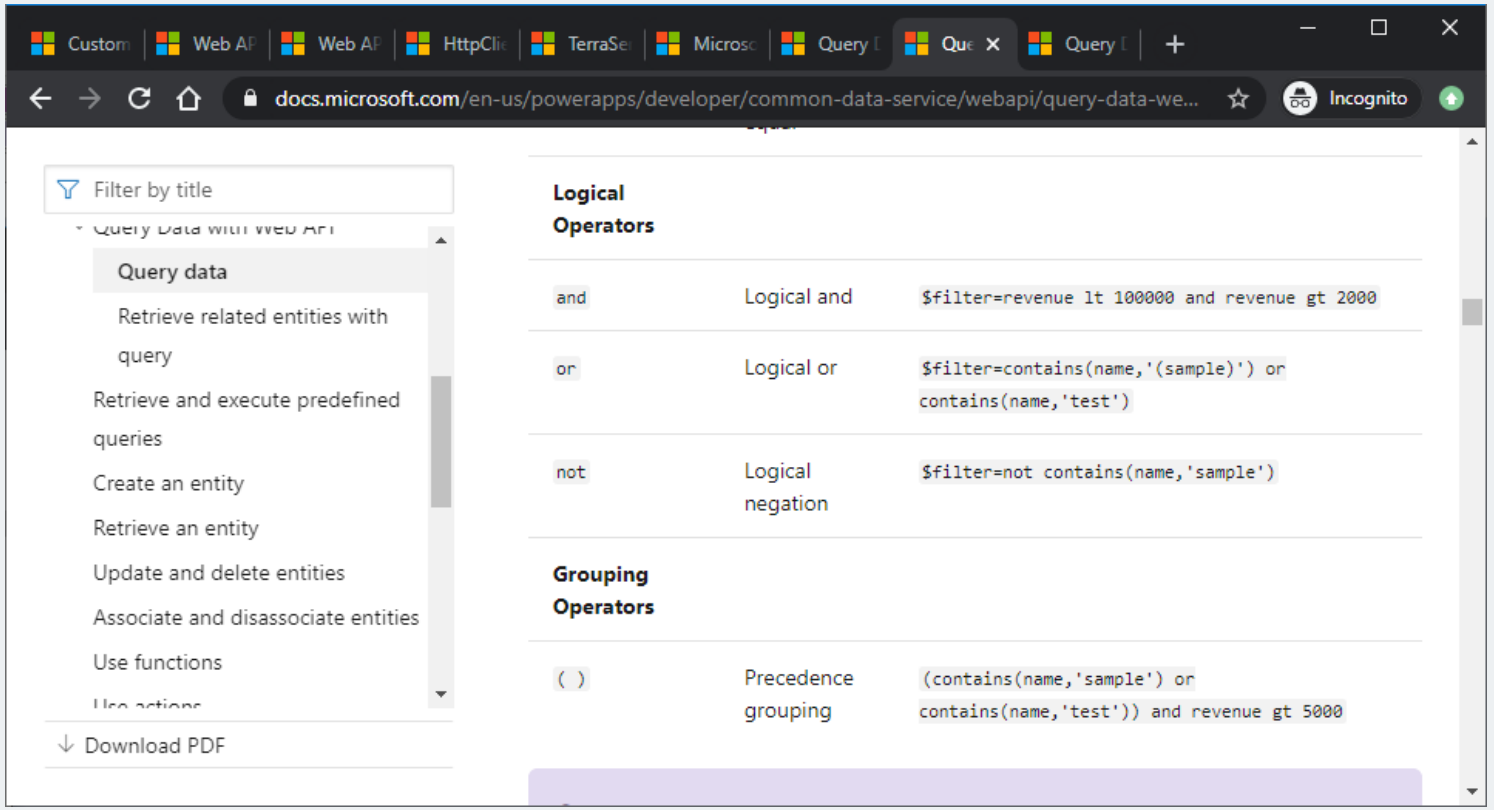


Image source: https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/webapi/query-data-web-api#standard-filter-operators

LINQ Provider > DFDS.CEM.Dynamics.CRM.WebAPI.LINQ (lib)

LINQ Provider > Queries > Supported operations

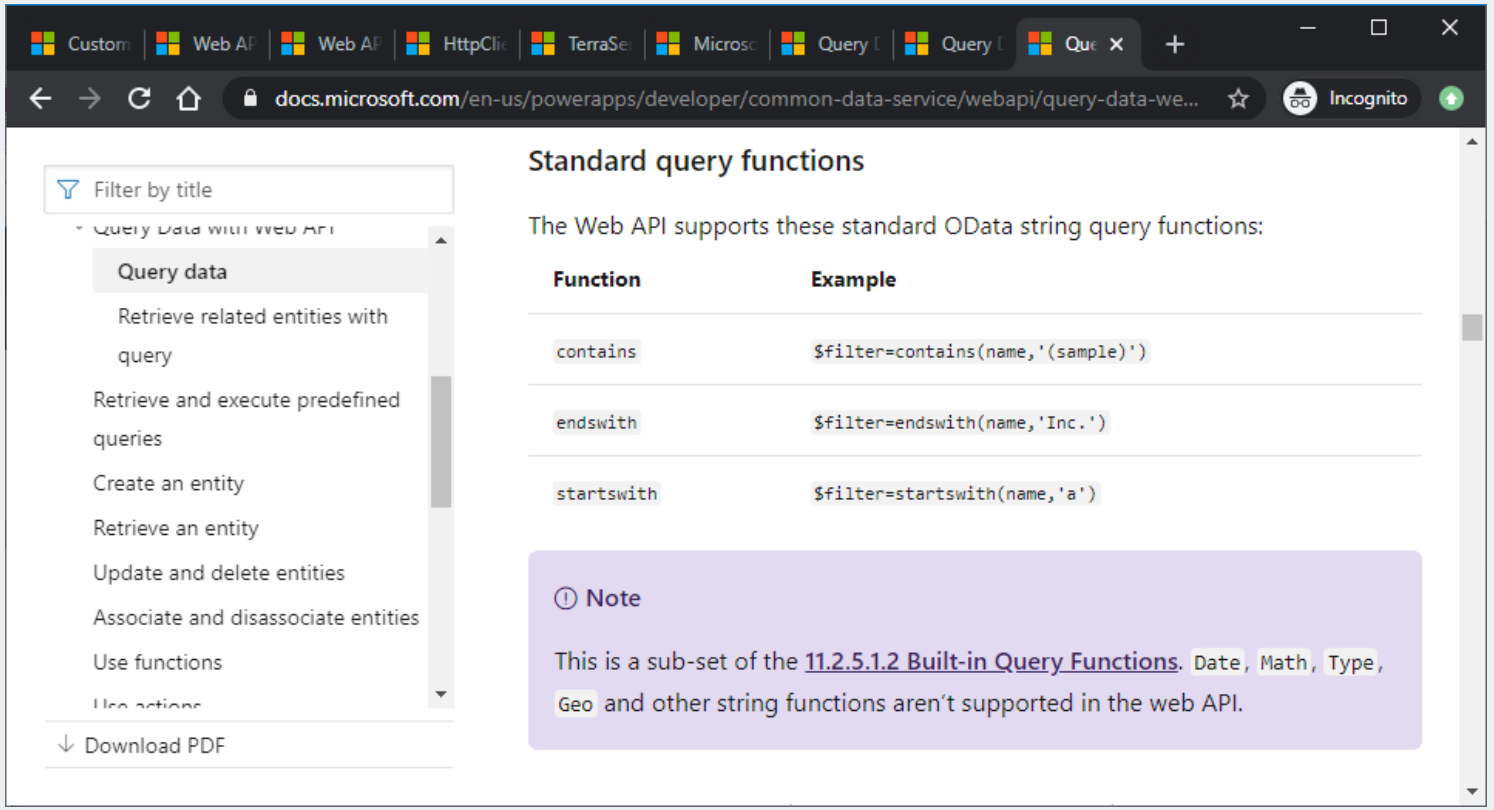
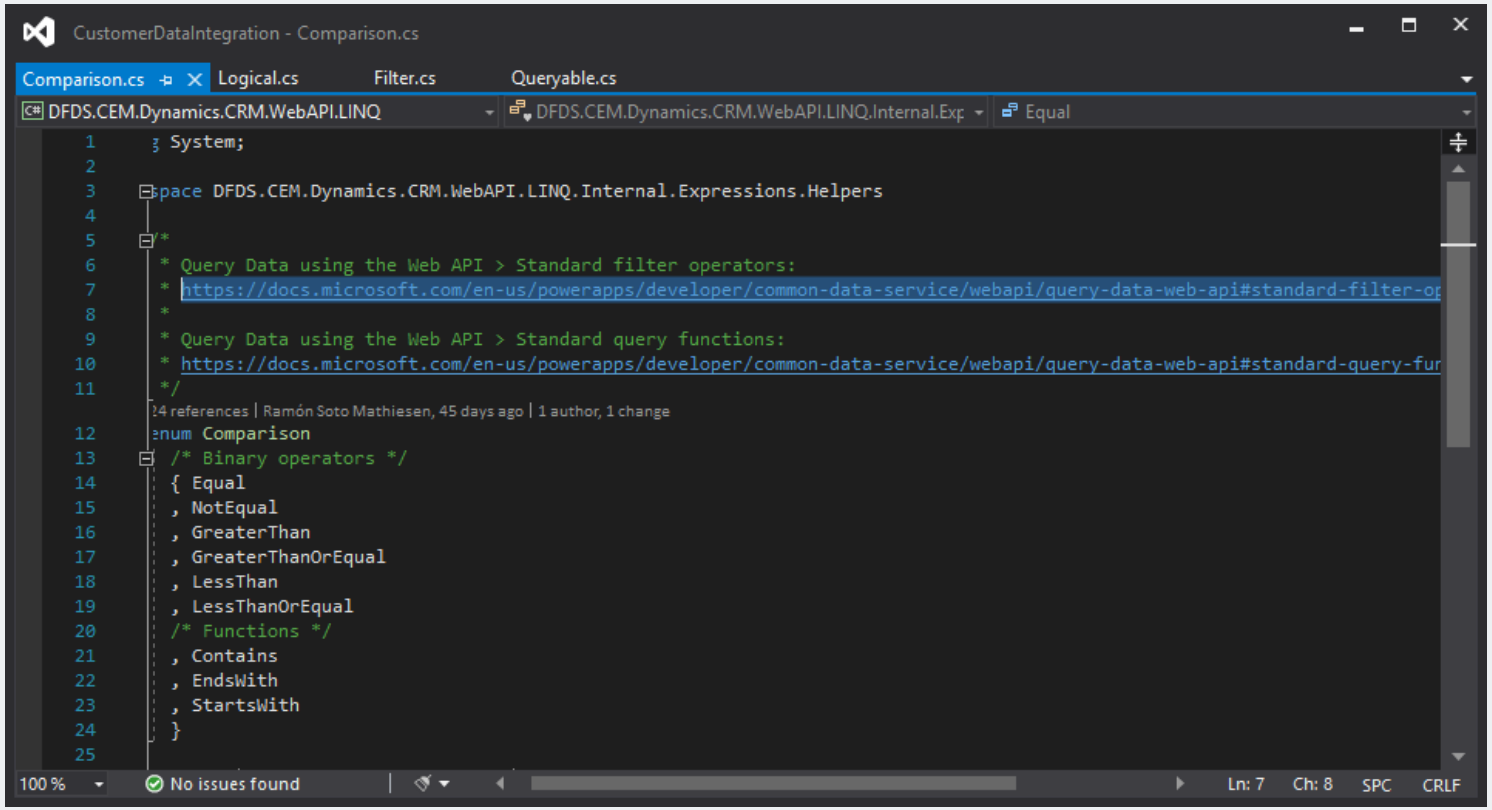


Image source: https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/webapi/query-data-web-api#standard-filter-operators

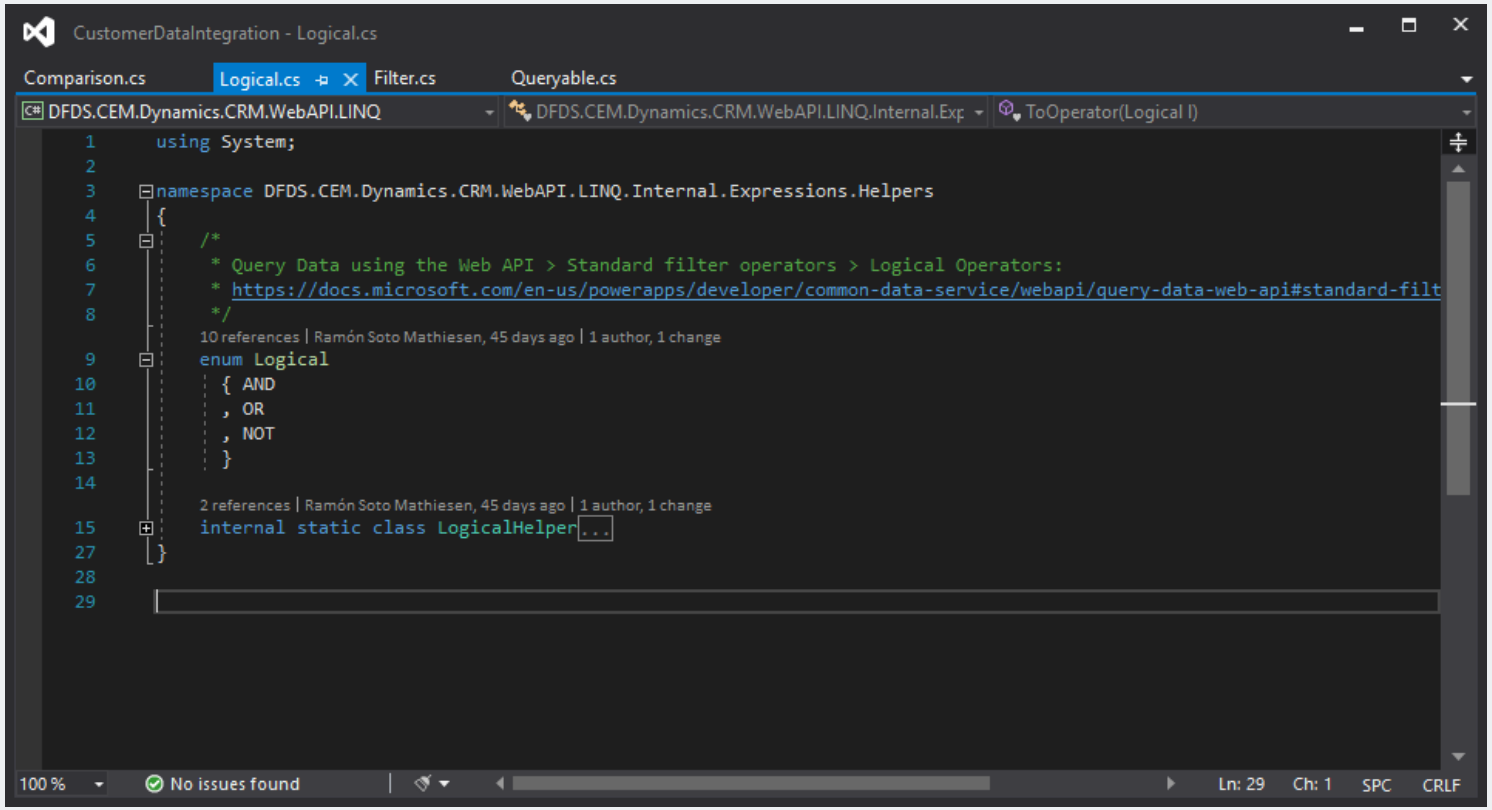
LINQ Provider > DFDS.CEM.Dynamics.CRM.WebAPI.LINQ (lib)

LINQ Provider > Queries > Supported operations



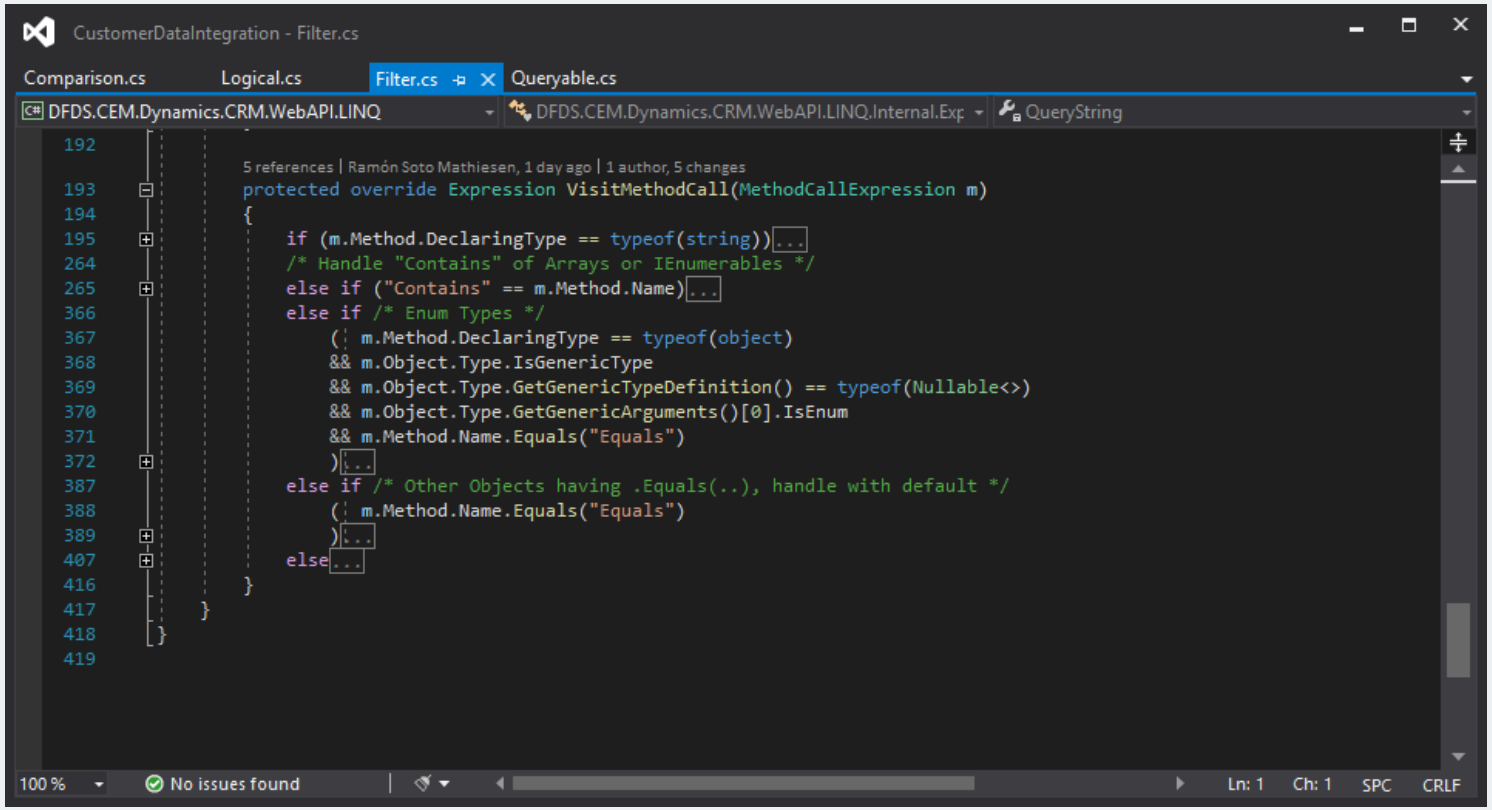
LINQ Provider > DFDS.CEM.Dynamics.CRM.WebAPI.LINQ (lib)

LINQ Provider > Queries > Supported operations



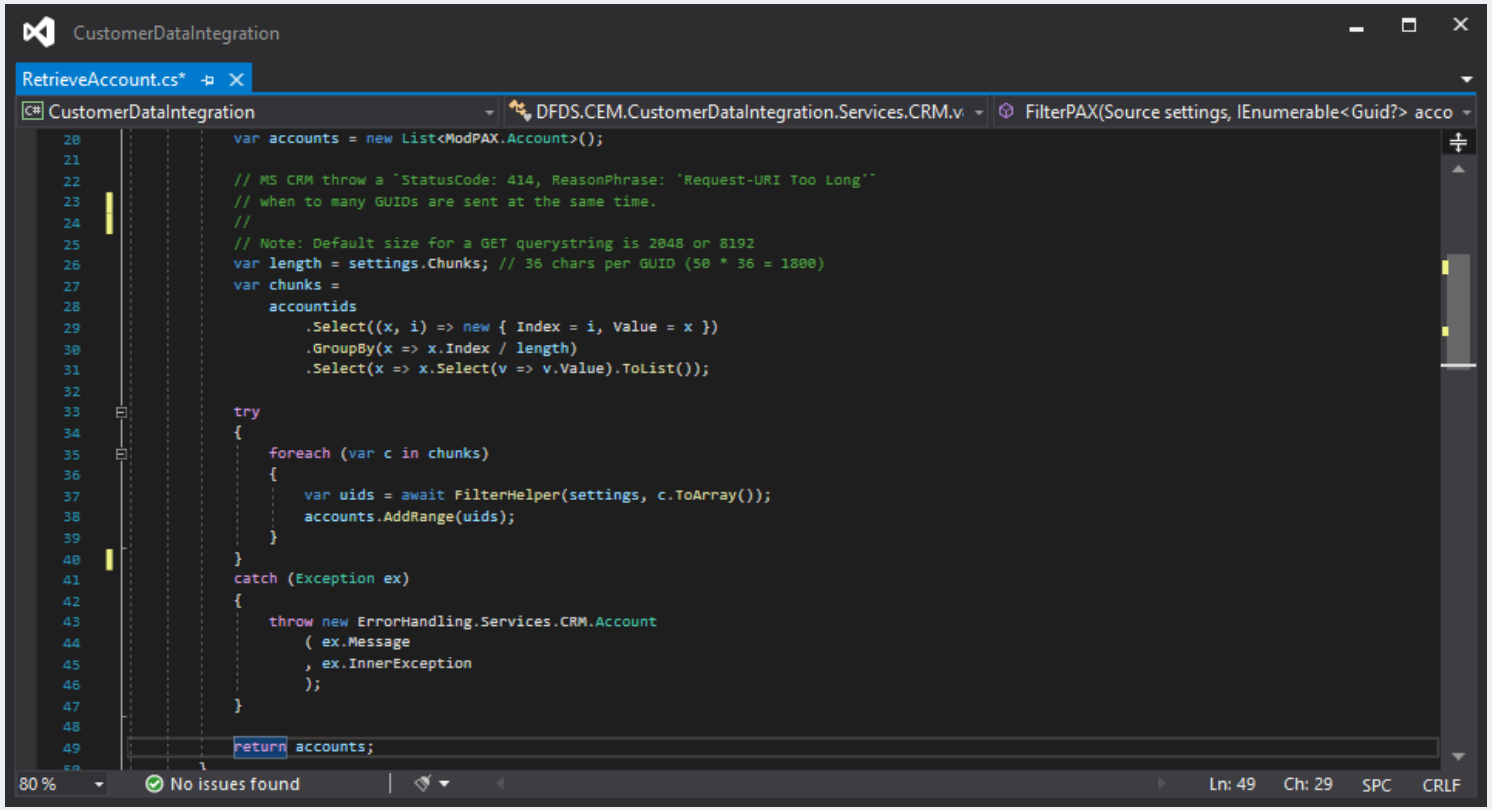
LINQ Provider > DFDS.CEM.Dynamics.CRM.WebAPI.LINQ (lib)

LINQ Provider > Queries > Supported operations (+ Contains for containers data types)



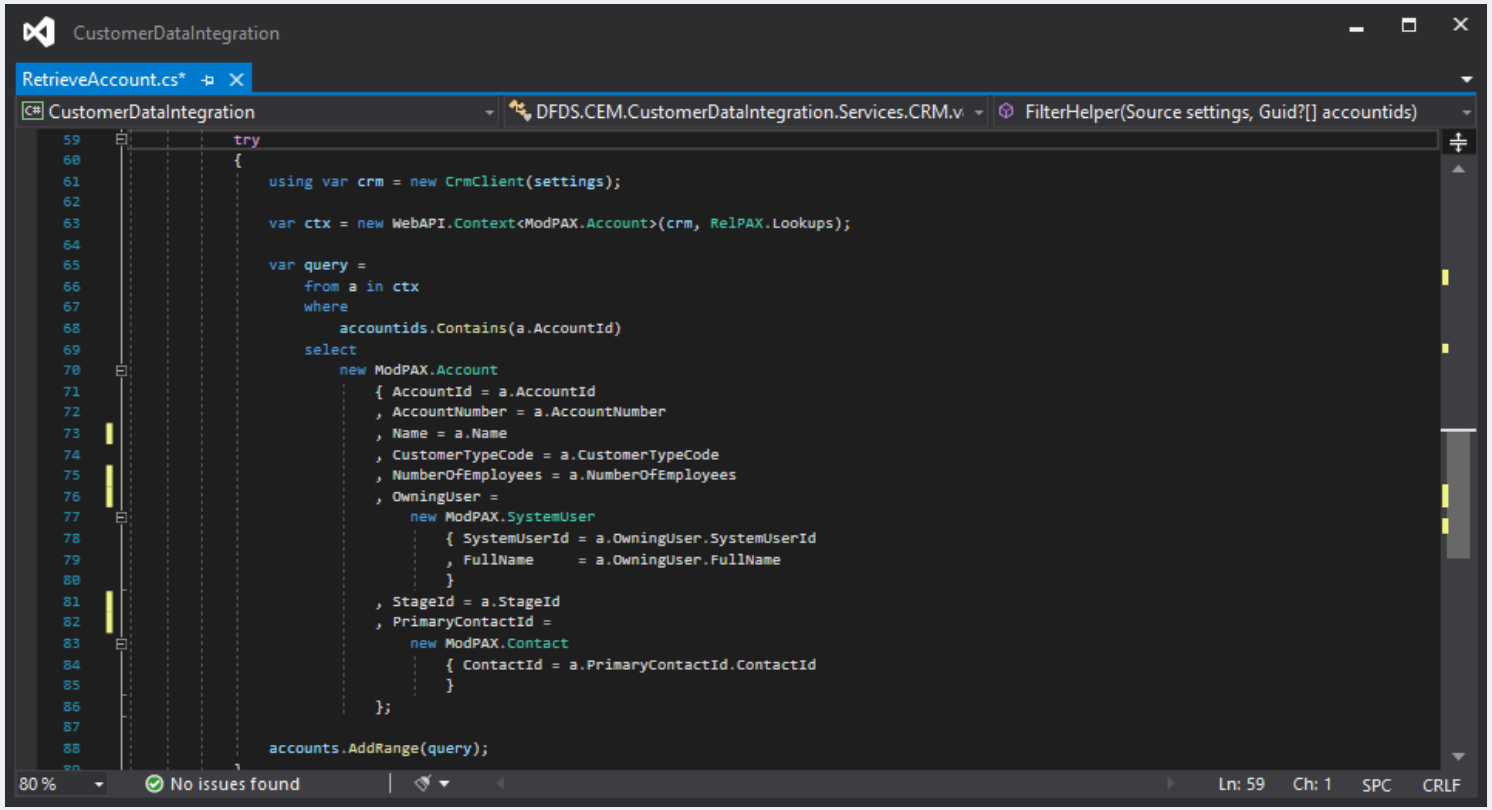
LINQ Provider > DFDS.CEM.Dynamics.CRM.WebAPI.LINQ (lib)

LINQ Provider > Queries > Chunks as translated output will be a Query String (limitation WebAPI)



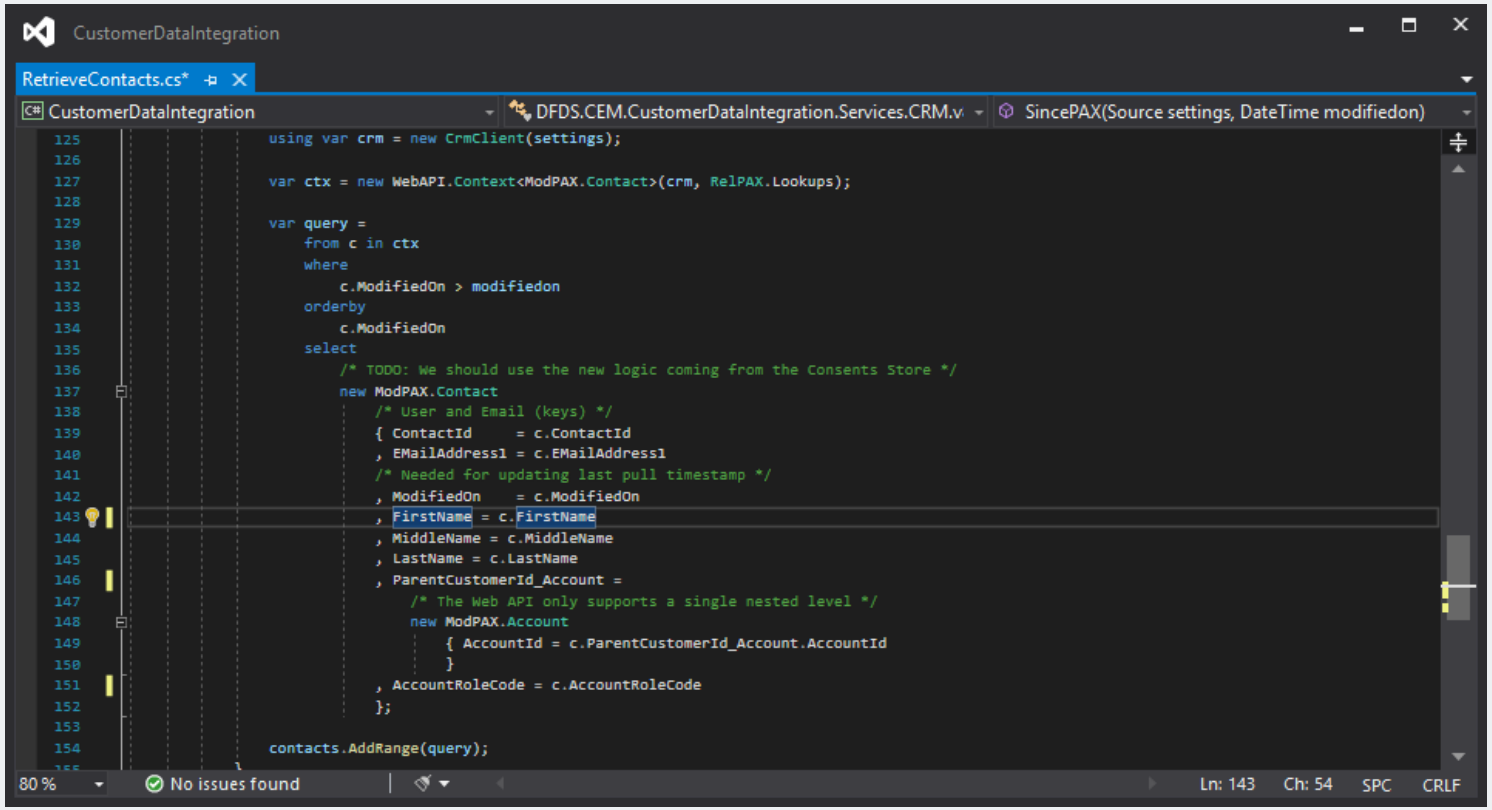
LINQ Provider > DFDS.CEM.Dynamics.CRM.WebAPI.LINQ (lib)

LINQ Provider > Queries > Chunks as translated output will be a Query String (limitation WebAPI)



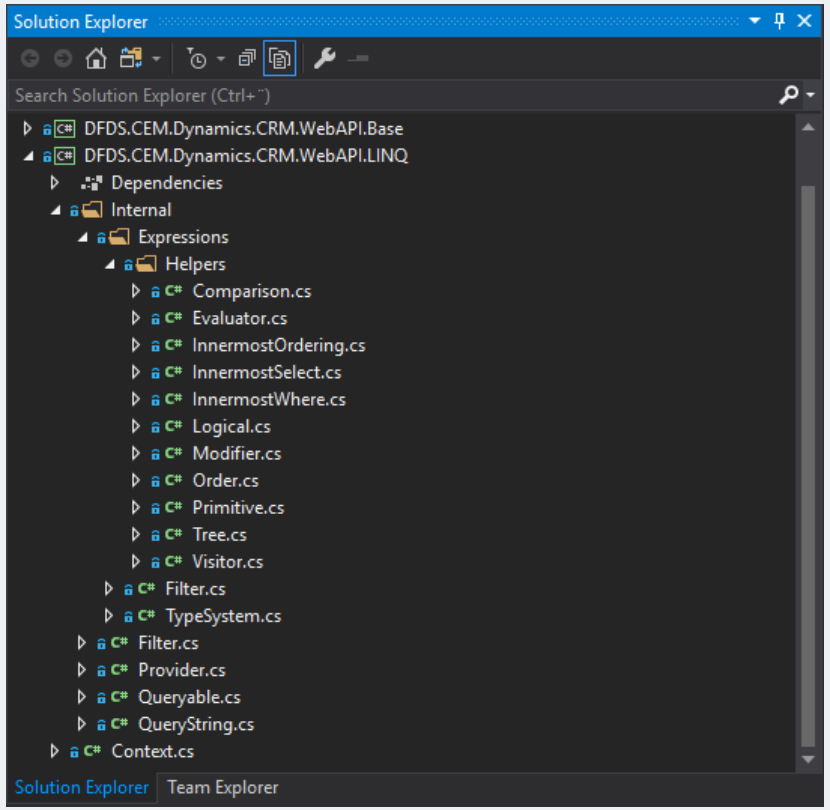
LINQ Provider > DFDS.CEM.Dynamics.CRM.WebAPI.LINQ (lib)

LINQ Provider > Queries > Only one level of nested entities is permitted (sadly we can only catch this runtime)



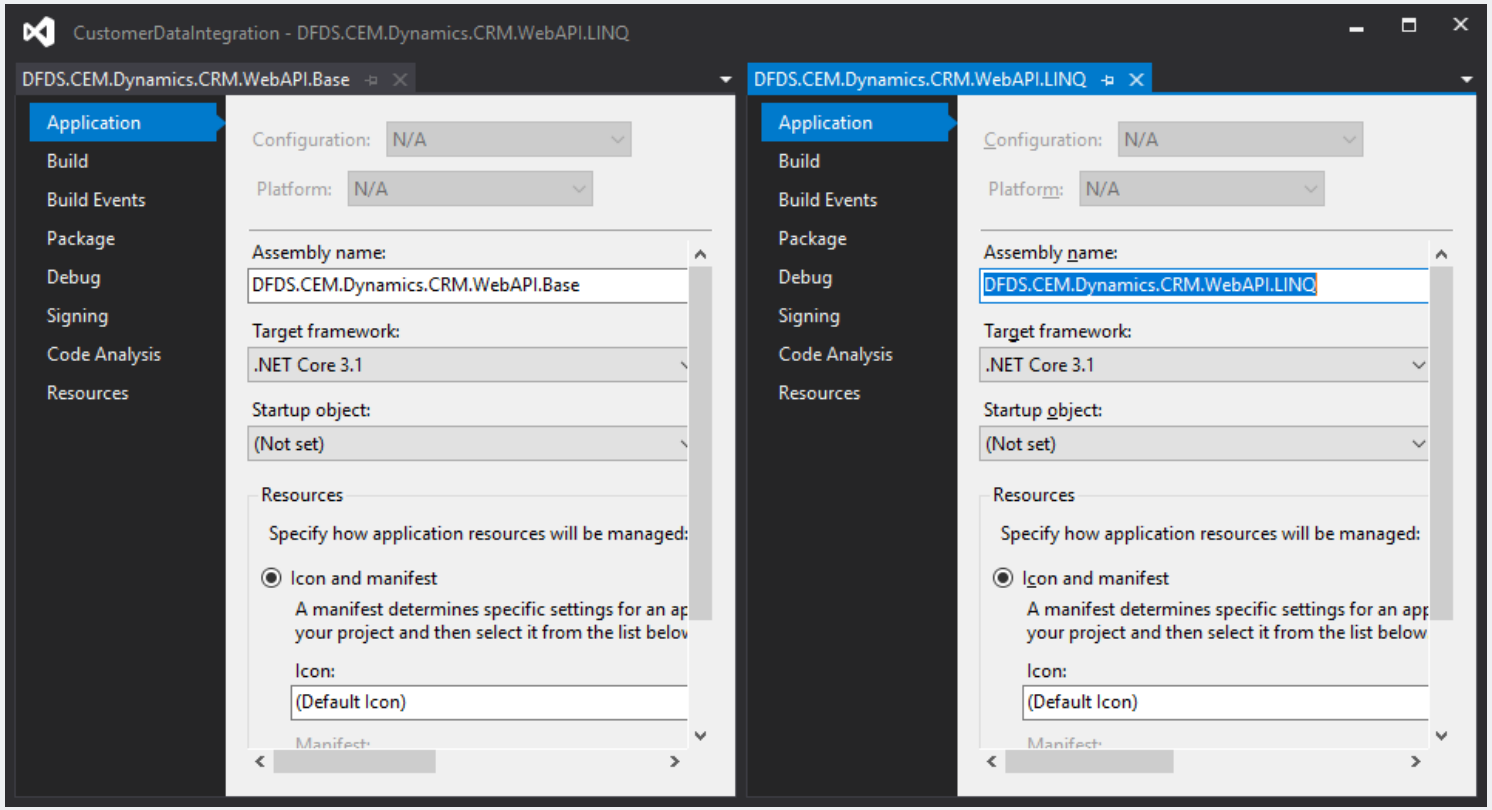
LINQ Provider > DFDS.CEM.Dynamics.CRM.WebAPI.LINQ (lib)

LINQ Provider > (simple) Visual Studio (C#) Project



LINQ Provider > DFDS.CEM.Dynamics.CRM.WebAPI.LINQ (lib)

LINQ Provider > (simple) Visual Studio (C#) Project > .NET Core v.3.1 (LTS)



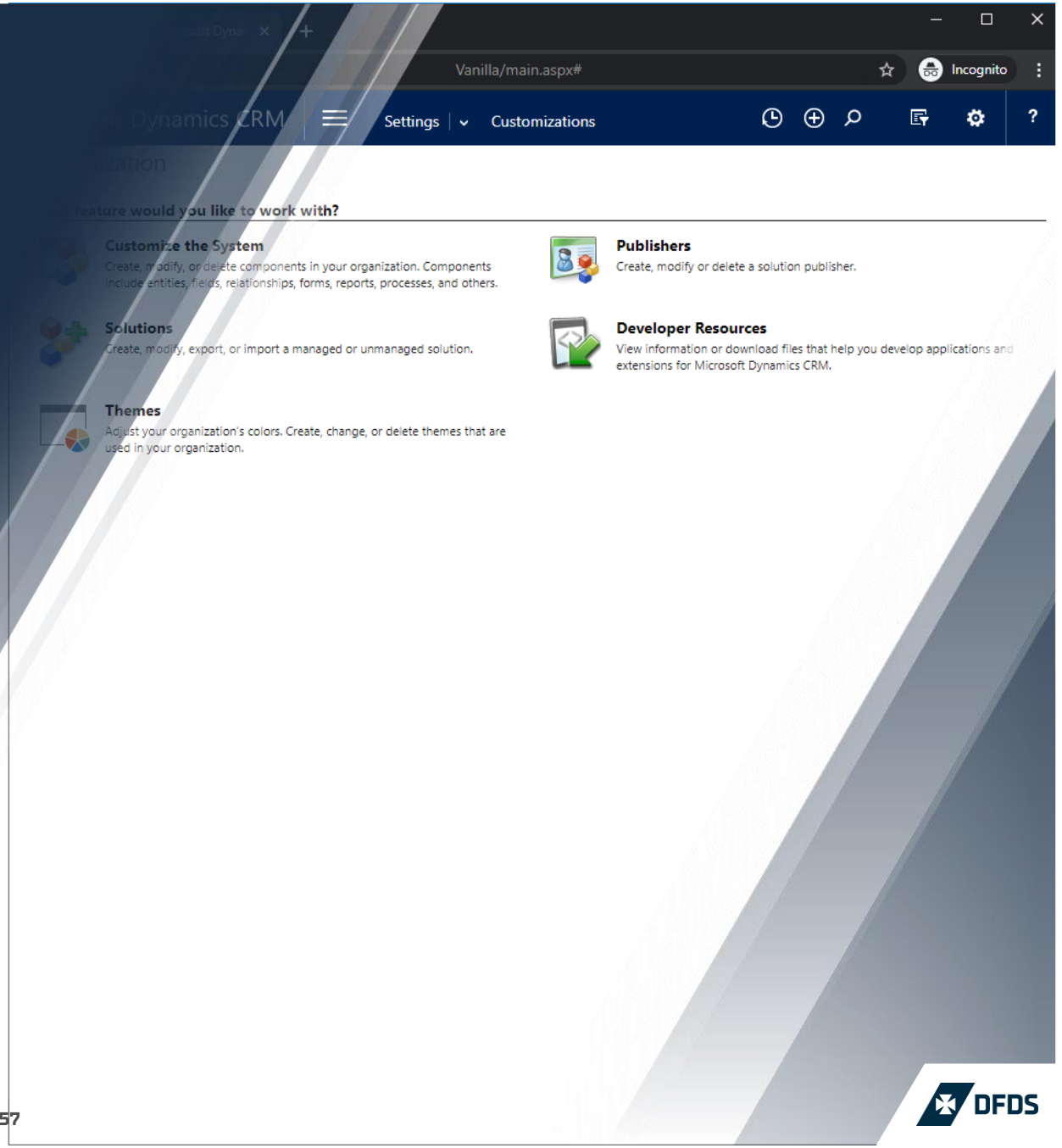
LINQ Provider > DFDS.CEM.Dynamics.CRM.WebAPI.LINQ (lib)

Background > Platform > Architecture > REST / WebAPI (JSON) > Ideally what would we like to have?

- Generate **type-safe C# classes** for xRM data entities of **any CRM application instance**
- ✓ Support for **LINQ code**
- ✓ Data **transformation from and to JSON** should be done **automatically** (sanitization)
- **Lightweight** size of generated types
- ✓ **Concurrency** / **parallelism**
- ✓ Support **.NET Core** v.3.1 (LTS)
  - Runtime infrastructure
  - Development setup
- ✓ **Few dependencies**
  - At most, Json.NET - Newtonsoft library, as Microsoft recommends to use it
  - None to the xRM SDKs as they don't support .NET Core
- **No SDK 2011 endpoint**
  - Only **/api/data/v8.1/** and above must be used

DFDS

Cloud Ready


Vanilla/main.aspx#
Incognito
Settings
Customizations
feature would you like to work with?
Customize the System
Create, modify, or delete components in your organization. Components include entities, fields, relationships, forms, reports, processes, and others.
Solutions
Create, modify, export, or import a managed or unmanaged solution.
Themes
Adjust your organization's colors. Create, change, or delete themes that are used in your organization.
Publishers
Create, modify or delete a solution publisher.
Developer Resources
View information or download files that help you develop applications and extensions for Microsoft Dynamics CRM.

Cloud Ready

Cloud Ready > Design and develop CRM/xRM micro-apps

- **CQRS** + **DDD** (Strongly typed to catch data inconsistencies at build time)

- **Micro-applications**, read as in micro-services
  - **Stateless** applications with **low CPU** and **memory footprint**

- 100% **.NET Core** 3.1 (LTS) = Docker ready

- **Scalable** as all exposed methods are **concurrent**:

```
public static async Task<..> MethodLogic
```

- And therefore can be **used** with the (**thread-safe**) **HttpClient** to achieve **parallelism** …

DFDS

Cloud Ready

Cloud Ready > Design and develop CRM/xRM micro-apps > Scalable > Concurrency and parallelism

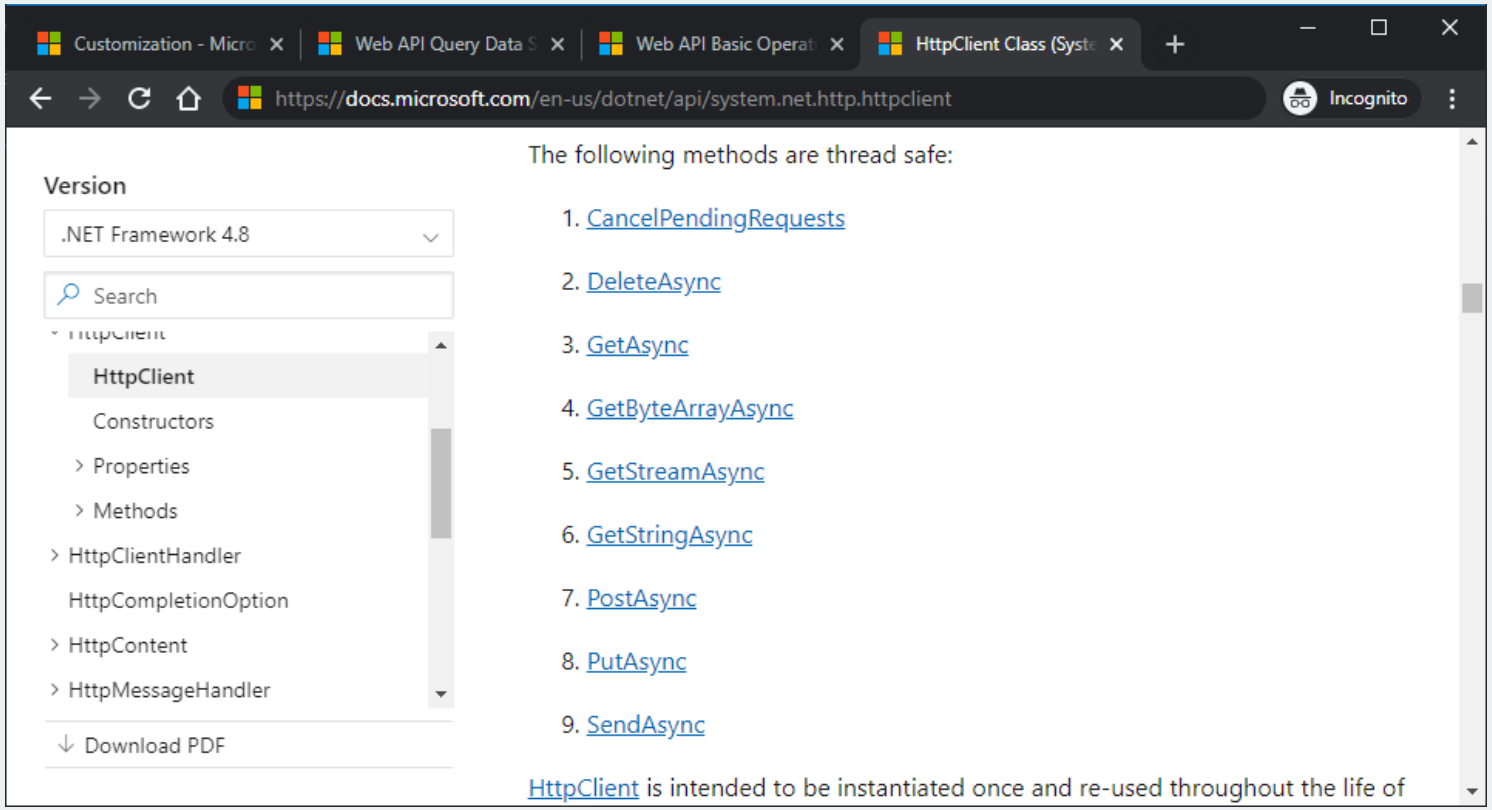
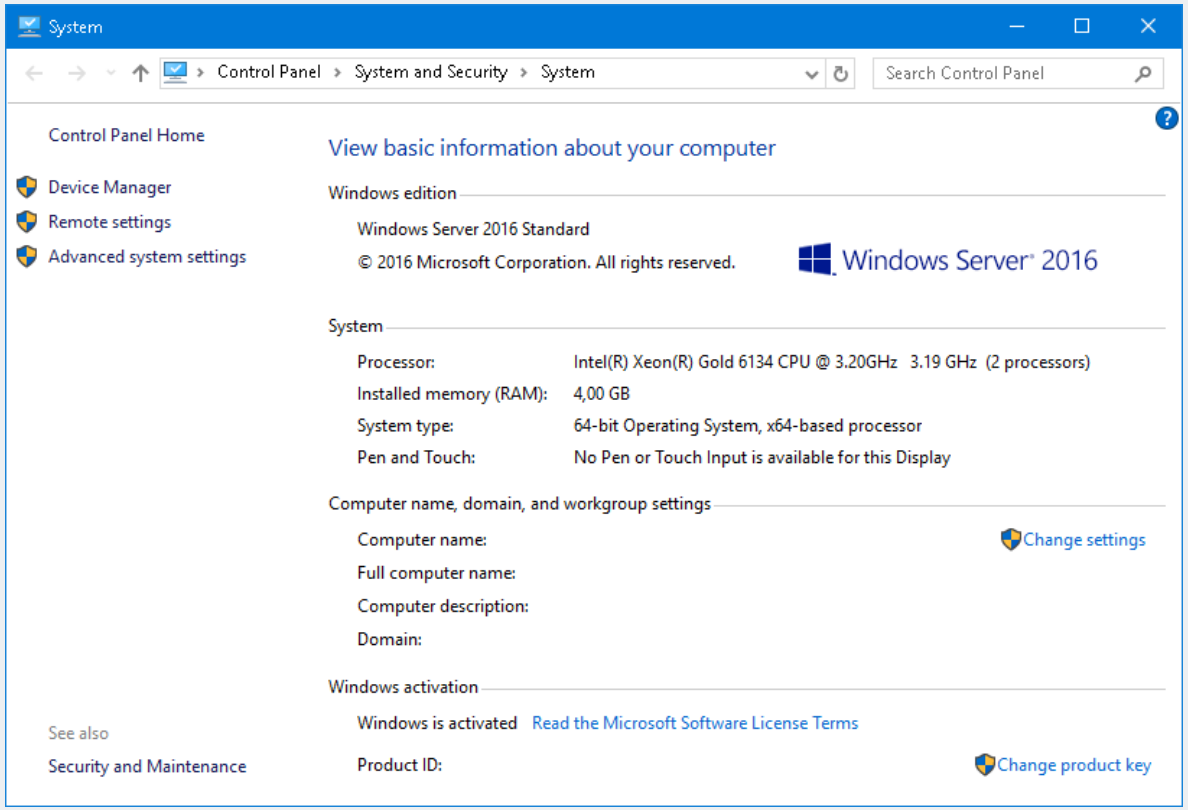


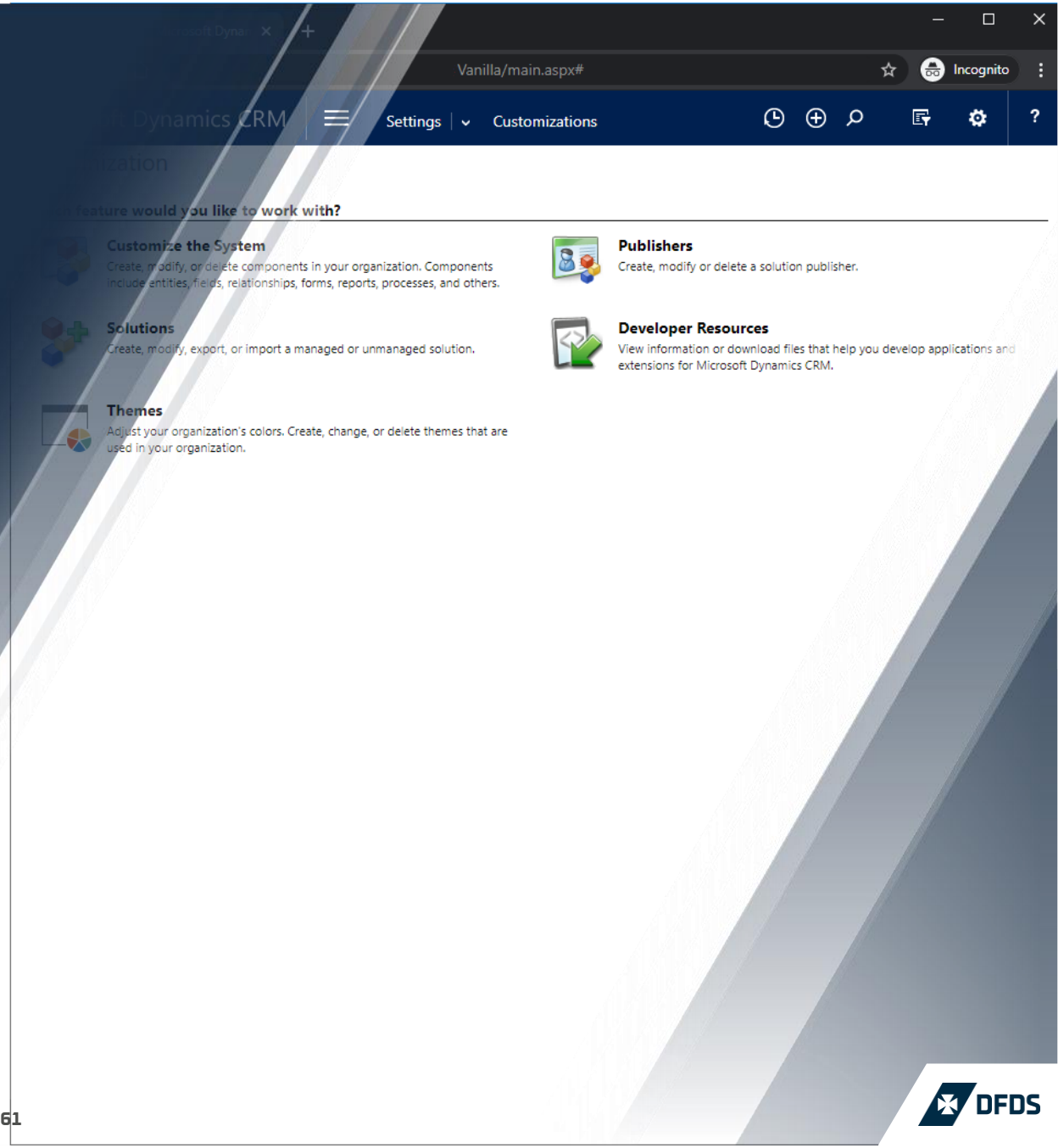
Image source: https://docs.microsoft.com/en-us/dotnet/api/system.net.http.httpclient

Cloud Ready

Cloud Ready > Design and develop CRM/xRM micro-apps > Robust and Stable (≈ 10% CPU and ≈ 50 MB when peaking)



Summary


Vanilla/main.aspx#
Incognito
Settings
Customizations
feature would you like to work with?
Customize the System
Create, modify, or delete components in your organization. Components include entities, fields, relationships, forms, reports, processes, and others.
Solutions
Create, modify, export, or import a managed or unmanaged solution.
Themes
Adjust your organization's colors. Create, change, or delete themes that are used in your organization.
Publishers
Create, modify or delete a solution publisher.
Developer Resources
View information or download files that help you develop applications and extensions for Microsoft Dynamics CRM.

CRM WebAPI Util + LINQ Provider = Cloud Ready

Background > Platform > Architecture > REST / WebAPI (JSON) > Ideally what would we like to have?

- ✓ Generate **type-safe C# classes** for xRM data entities of **any CRM application instance**
- ✓ Support for **LINQ code**
- ✓ Data **transformation from and to JSON** should be done **automatically** (sanitization)
- ✓ **Lightweight** size of generated types
- ✓ **Concurrency** / **parallelism**
- ✓ Support **.NET Core** v.3.1 (LTS)
  - Runtime infrastructure
  - Development setup
- ✓ **Few dependencies**
  - At most, Json.NET - Newtonsoft library, as Microsoft recommends to use it
  - None to the xRM SDKs as they don't support .NET Core
- ✓ **No SDK 2011 endpoint**
  - Only **/api/data/v8.1/** and above must be used

DFDS

Summary

Summary > Open Source? It's up to DFDS A/S (https://github.com/dfds)

Image source: https://opensource.org/logo-usage-guidelines

Summary

Summary > Help us, help you > In the age of collaboration/cooperation, ensure your platform is as easy-to-use as possible

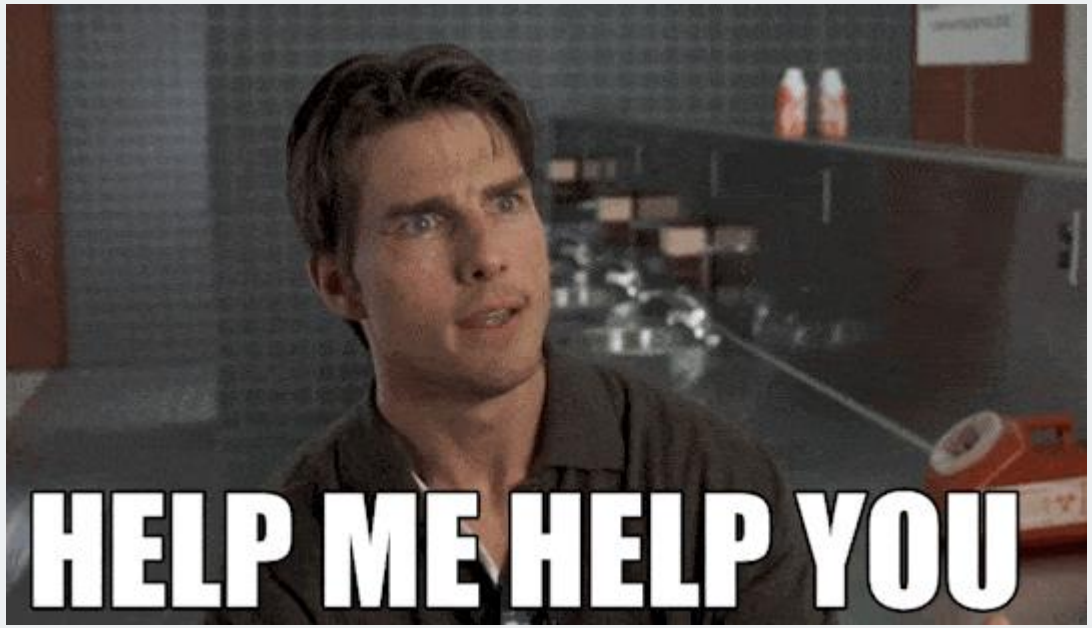


Image source: unknown

Q & A

DFDS