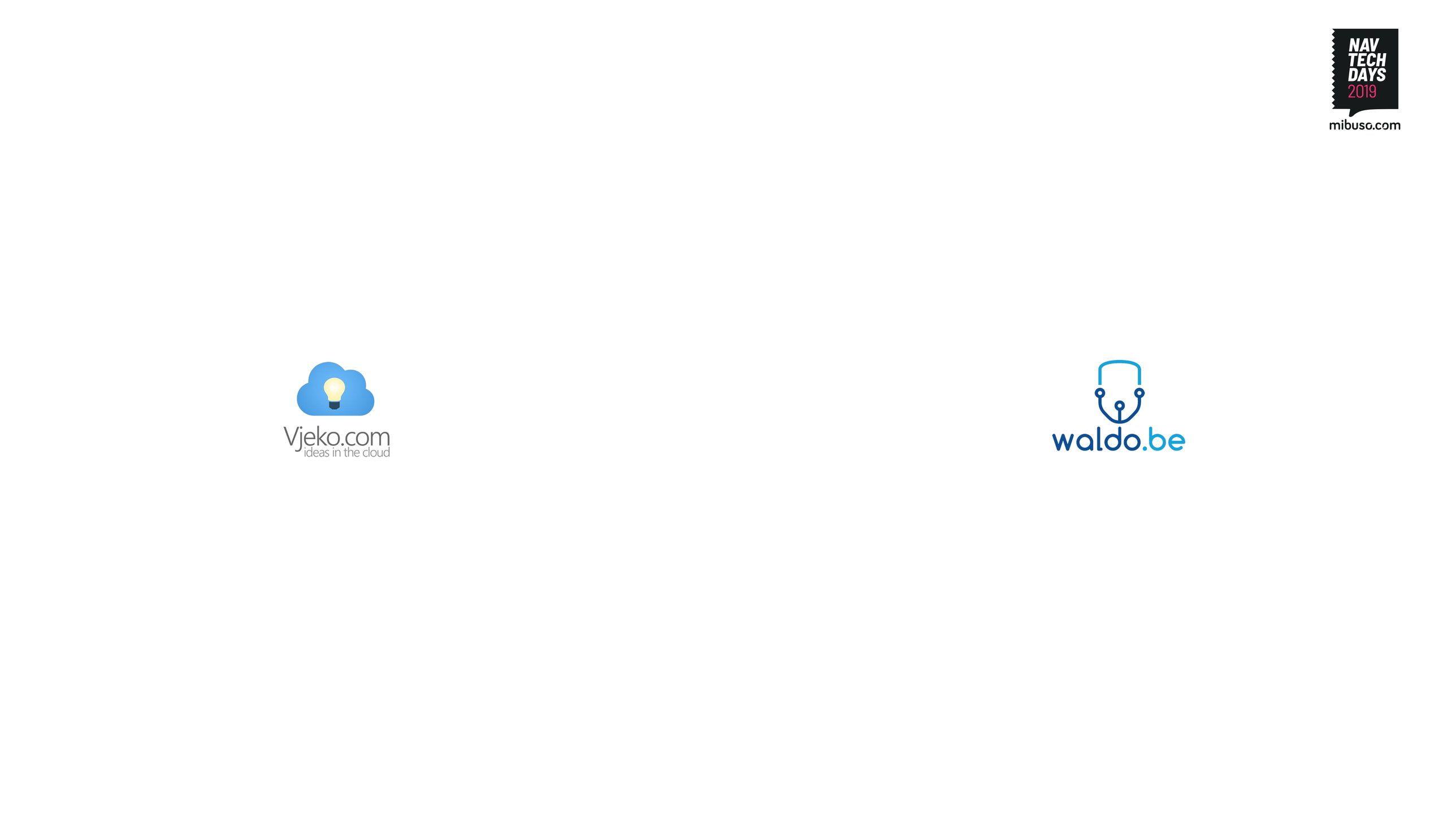
{Connect app}²

Eric Wauters, Vjekoslav Babic

When you are passionate about
Microsoft Dynamics NAV/365 Business Central


NAV
TECH
DAYS
2019
mibuso.com
Vjeko.com
ideas in the cloud
waldo.be

Before we even begin

Vjeko.com
ideas in the cloud

waldo.be

You are our demo!

Before we even begin

NAV TECH DAYS 2019
mibuso.com

Vjeko.com
ideas in the cloud

Install the Expo client app…

…from your app store

waldo.be

GET IT ON Google Play

Download on the App Store

{Connect app}²

Connect app

NAV TECH DAYS 2019
mibuso.com

What

Technology

Development

Connect app

NAV TECH DAYS 2019
mibuso.com

What	Technology	Development
• Any 3rd party app that uses data from Business Central	• REST • Web hooks	• Your choice


TIME SH
Time In
9:00 AM
9:00 A
9:0
ondy
OLL
come Tax Return
Last name
BOOOOOOOOOOORING


9:41
X1234
killer

UID: ntd2019

PWD: Antwerp.19

What are Custom APIs?

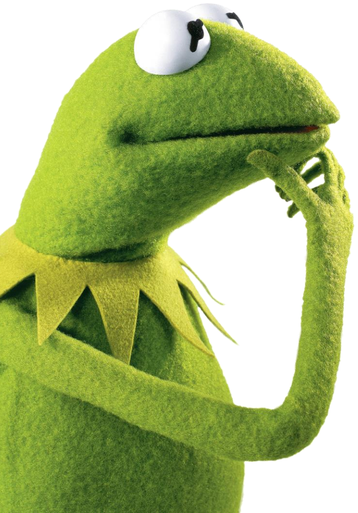
What are Custom APIs?

- Can be developed by any Partner
- Included in Add-on apps
- Can be accessed using specific endpoints
- Easy to create

Custom API considerations

Microsoft makes sure all their APIs are consistent
Consistency of your APIs is **your** responsibility

- Versioning principle
- Naming and schema conventions
- Consistent behavior



Custom API considerations

NAV TECH DAYS 2019
mibuso.com

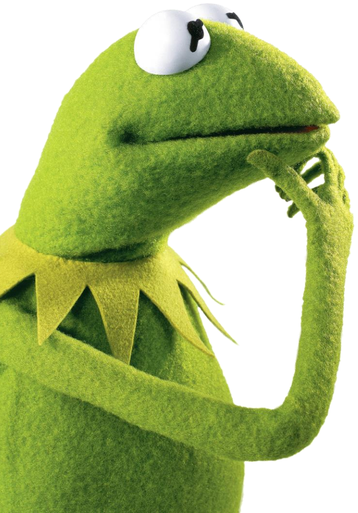
Microsoft makes sure all their APIs are consistent
Consistency of your APIs is <u>your</u> responsibility

Versioning principle

- You must **not break** existing versions
- Any breaking change requires **up-versioning**
- API layer does **no versioning validations** for you

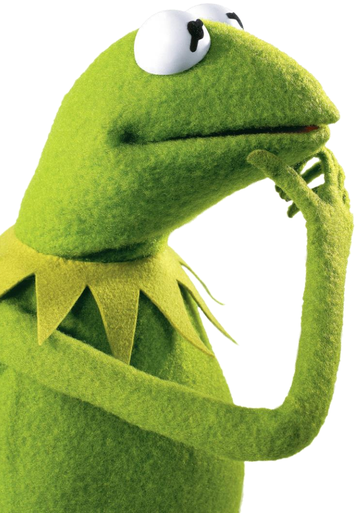
Naming and schema conventions

Consistent behavior



Custom API considerations

Microsoft makes sure all their APIs are consistent
Consistency of your APIs is <u>your</u> responsibility



Versioning principle

- You must **not break** existing versions
- Any breaking change requires **up-versioning**
- API layer does **no versioning validations** for you

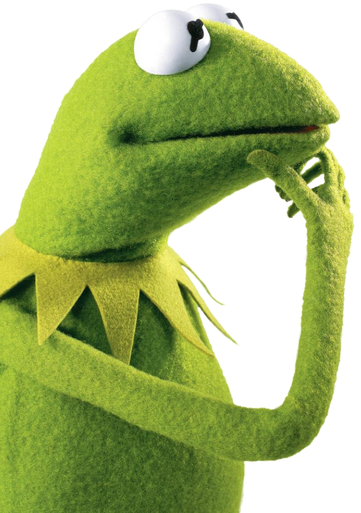
Naming and schema conventions

- Include **SystemId**
- **lastModifiedDateTime**
- Always use **camelCase**

Consistent behavior

Custom API considerations

Microsoft makes sure all their APIs are consistent
Consistency of your APIs is <u>your</u> responsibility



Versioning principle

- You must **not break** existing versions
- Any breaking change requires **up-versioning**
- API layer does **no versioning validations** for you

Naming and schema conventions

- Include **SystemId**
- **lastModifiedDateTime**
- Always use **camelCase**

Consistent behavior

- Table relationships
- Complex types
- Insertion / modification behavior

Developing Custom APIs

The basics

APIs are OData

OData web services can be built as:
- Pages
- Queries

APIs can be developed as:
- Pages
- Queries

```
page 78912 "Lego Item Entity KND"
{
    PageType = API;
    Caption = 'lego Item Entity', locked = true
    APIPublisher = 'waldo';
    APIGroup = 'kinder';
    APIVersion = 'v1.0';
    EntityName = 'legoItem';
    EntitySetName = 'legoItems';
    SourceTable = "LegoItem KND";
    ODataKeyFields = SystemId;
    DelayedInsert = true;

    You, 6 hours ago | 2 authors (Eric Wauters and others)
    layout
    {
        0 references | You, 6 hours ago | 2 authors (Eric Wau
        area(Content)        Eric Wauters, 8 day
        {
            0 references
            repeater(LegoItems)
            {
                0 references
                field(systemId; SystemId)
                {
                    ApplicationArea = All;
                    Caption = 'systemId', Locke
                    Editable = false;
                }
                0 references
                field(number; "No.")
```

```
query 78910 "Lego Item Entity KND"
{
    QueryType = API;
    Caption = 'lego Item Entity2', locked = true;
    APIPublisher = 'waldo';
    APIGroup = 'kinder';
    APIVersion = 'v1.0';
    EntityName = 'legoItem2';
    EntitySetName = 'legoItems2';

    elements
    {
        0 references
        dataitem(legoItem; "LegoItem KND")
        {
            0 references
            column(systemId; SystemId) { }
            0 references
            column(number; "No.") { }
            0 references
            column(name; Name) { }
            0 references
            column(year; Year) { }
            0 references
            column(setUrl; SetUrl) { }
        }
    }
}
```

API definition properties

NAV TECH DAYS 2019
mibuso.com

Namespace properties define the endpoint hierarchy:

- Become a part of the endpoint URL
- These are case-insensitive!
- Should use camelCase

```
APIPublisher = ‘waldo';
APIGroup = ‘killerApp';
APIVersion = ‘v1.0';
```

API definition properties

Namespace properties define the endpoint hierarchy:

- Become a part of the endpoint URL
- These are case-insensitive!
- Should use camelCase

```
APIPublisher = ‘waldo';
APIGroup = ‘killerApp';
APIVersion = ‘v1.0';
```

`https://api.businesscentral.dynamics.com/v2.0/<tenant_id>/<sandboxname>/api/waldo/killerApp/v1.0`

API definition properties

Namespace properties define the endpoint hierarchy:

- Become a part of the endpoint URL
- These are case-insensitive!
- Should use camelCase

```
APIPublisher = ‘waldo';
APIGroup = ‘killerApp';
APIVersion = ‘v1.0';
```

Entity properties define the EDM (Entity Data Model)

- These are case-sensitive!
- Should use camelCase

```
EntityName = ‘crime’;
EntitySetName = ‘crimes';
```

`https://api.businesscentral.dynamics.com/v2.0/<tenant_id>/<sandboxname>/api/waldo/killerApp/v1.0`

API definition properties

Namespace properties define the endpoint hierarchy:

- Become a part of the endpoint URL
- These are case-insensitive!
- Should use camelCase

```
APIPublisher = ‘waldo';
APIGroup = ‘killerApp';
APIVersion = ‘v1.0';
```

Entity properties define the EDM (Entity Data Model)

- These are case-sensitive!
- Should use camelCase

```
EntityName = ‘crime’;
EntitySetName = ‘crimes';
```

https://api.businesscentral.dynamics.com/v2.0/<tenant_id>/<sandboxname>/api/waldo/killerApp/v1.0/companies(<company_id>)/crimes

Table considerations

Every entity table must provide required API fields and functionality

- SystemId field
- Last Modified DateTime field
- Primary key considerations
- Field considerations

Table considerations

NAV TECH DAYS 2019
mibuso.com

Every entity table must provide required API fields and functionality

SystemId field

- A GUID field
- Virtual field – you don't have to create or maintain it yourself – but you have to put it on the API page

Last Modified DateTime field

Primary key considerations

Field considerations

Table considerations

Every entity table must provide required API fields and functionality

SystemId field

- A GUID field
- Virtual field – you don't have to create or maintain it yourself – but you have to put it on the API page

Last Modified DateTime field

- A DateTime field
- Must be modified on every record modification.

Primary key considerations

Field considerations

Table considerations

Every entity table must provide required API fields and functionality

SystemId field

- A GUID field
- Virtual field – you don't have to create or maintain it yourself – but you have to put it on the API page

Last Modified DateTime field

- A DateTime field
- Must be modified on every record modification.

Primary key considerations

- The SystemId can't be the primary key
- The SystemId is more durable than the primary key – it survives the rename operation.

Field considerations

Table considerations

NAV TECH DAYS 2019
mibuso.com

Every entity table must provide required API fields and functionality

SystemId field

- A GUID field
- Virtual field – you don't have to create or maintain it yourself – but you have to put it on the API page

Last Modified DateTime field

- A DateTime field
- Must be modified on every record modification.

Primary key considerations

- The SystemId can't be the primary key
- The SystemId is more durable than the primary key – it survives the rename operation.

Field considerations

- Fields should follow Business Central naming conventions, and not use camelCase or PascalCase. Caption Case must be used instead.

Page considerations

Every page must provide correct mapping of the entity table to REST format and behavior

- Field names considerations
- Modify behavior

Page considerations

Every page must provide correct mapping of the entity table to REST format and behavior

Field names considerations

- All field names must be set to **camelCase** explicitly.
- Don’t set **ApplicationArea.**
- It’s the **Name** property of a page field that defines how the field is referenced in REST requests and responses.

Modify behavior

Page considerations

NAV TECH DAYS 2019
mibuso.com

Every page must provide correct mapping of the entity table to REST format and behavior

Field names considerations

- All field names must be set to **camelCase** explicitly.
- Don’t set **ApplicationArea.**
- It’s the **Name** property of a page field that defines how the field is referenced in REST requests and responses.

Modify behavior

- SystemId value must not be changed.
- Primary key values can be changed during a modification call, and this needs to propagate as a Rename call in AL.

Useful resources

NAV TECH DAYS 2019
mibuso.com

https://docs.microsoft.com/en-us/dynamics-nav/fin-graph/
https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-connect-apps-tips

NAV TECH DAYS 2019

mibusco.com

Useful Tools

Rest Client

NAV TECH DAYS 2019
mibuso.com

A VSCode Extension

Alows you to send HTTP request and view the response

You can make the files part of your app


```
@endpoint = https://westeurope.api.cognitive.microsoft.com/
2 references
@projectId = e116de8c-224a-4853-b569-2cec5932fbac

@Key = 3d089c169e944c848644194b246d6559

### GetTags
2 references
@url = {{endpoint}}/customvision/v3.0/training/projects/{{projectId}}/tags
Send Request
GET {{url}}
Training-Key: {{Key}}
Content-Type: application/json
      Eric Wauters, 3 days ago • Almost Creating Tags
### CreateTags
1 reference
@name = 'waldo5'
2 references
@url = {{endpoint}}/customvision/v3.0/training/projects/{{projectId}}/tags?name={{name}}
Send Request
POST {{url}}
Training-Key: {{Key}}
```

PowerShell

Resetting the environment
Upload data
Quickly and efficiently Test APIs
Create a (manual) release pipeline to easily “reset” multiple environments

```
$BCLegoItem = Invoke-RestMethod -Method Post `
    -Uri "$($url)/companies($($Company.id))/legoItems" -ContentType "application/json" `
    -Headers @{ "Authorization" = $AADToken.token_type + " " + $AADToken.access_token } `
    -Body (@{ `
            "number"     = $LegoSet.set_num
        "name"           = $LegoSetName;
        "numberOfPieces" = $LegoSet.num_parts
        "setUrl"         = $LegoSet.set_url
        "year"           = $LegoSet.year
        "themeId"        = $LegoSet.theme_id
    } |
        ConvertTo-Json -Depth 100)
```

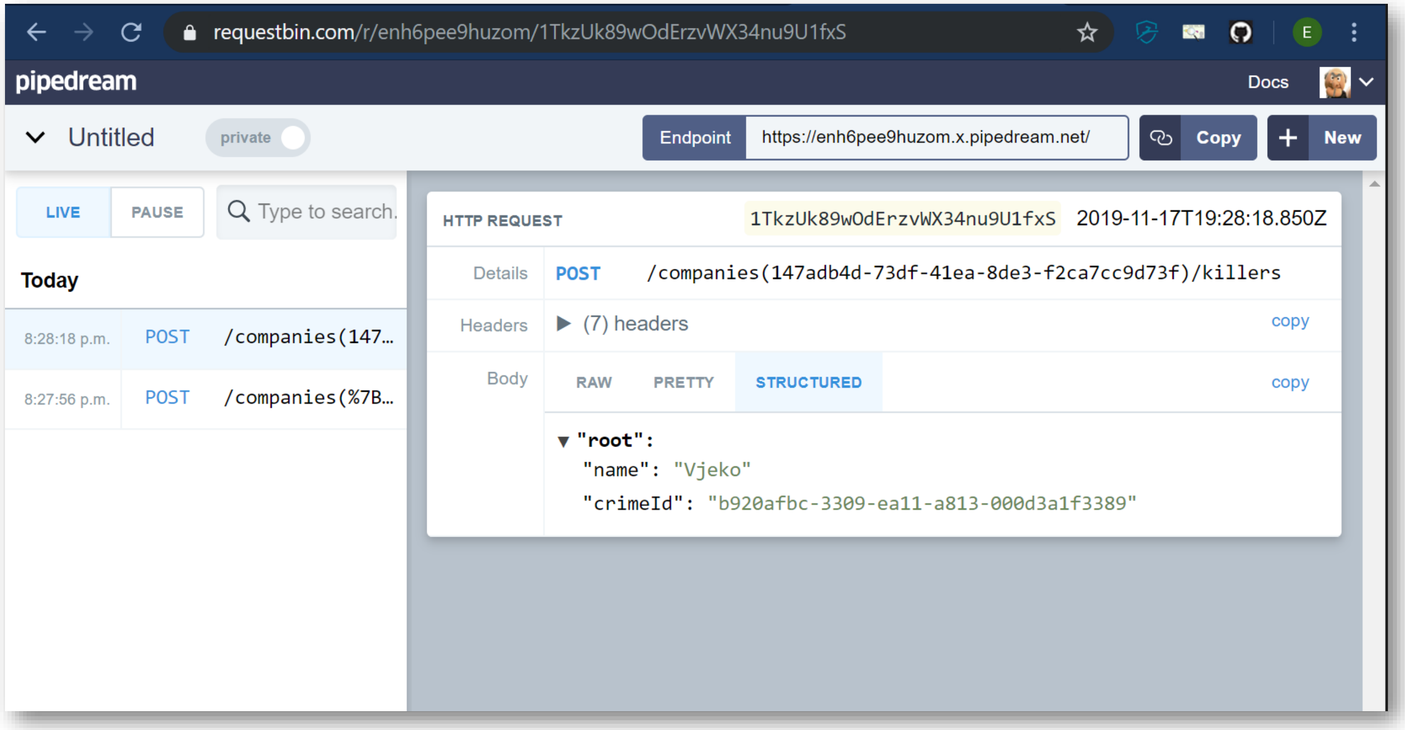
RequestBin

NAV TECH DAYS 2019
mibuso.com

https://requestbin.com/

Online tool that creates a subdomain which you can use to test an API call.

Requests sent to any path on the subdomain are forwarded to the browser in real time.



OAuth

Two types of authentication

NAV TECH DAYS 2019
mibuso.com

Dynamics 365 Business Central APIs support two types of authentication

Base Authentication

OAuth 2.0 Authentication

Two types of authentication

NAV TECH DAYS 2019
mibuso.com

Dynamics 365 Business Central APIs support two types of authentication

Base Authentication

- Client uses the credentials directly
- Performs actual authentication
- Lower security
- Potentially exposes user's secrets

- Works in on-prem, and cloud sandbox and development environments.

- Does not work in cloud production environments.

OAuth 2.0 Authentication

Two types of authentication

NAV TECH DAYS 2019
mibuso.com

Dynamics 365 Business Central APIs support two types of authentication

Base Authentication	OAuth 2.0 Authentication
• Client uses the credentials directly • Performs actual authentication • Lower security • Potentially exposes user's secrets	• Client uses a series of REST calls • Performs authentication delegation • Higher security • Does not expose user's secrets
• Works in on-prem, and cloud sandbox and development environments.	• Works in all environments when Azure AD authentication is used.
• Does not work in cloud production environments.	• Mandatory for all cloud production environments

What is OAuth 2.0

NAV TECH DAYS 2019
mibuso.com

An open standard for access delegation, that allows users to grant applications access to user information in other applications, without exposing user credentials.

- Delegation
- Token-based access
- Web-based protocol

What is OAuth 2.0

NAV TECH DAYS 2019
mibuso.com

An open standard for access delegation, that allows users to grant applications access to user information in other applications, without exposing user credentials.

Delegation

- User access is delegated to an application
- No user impersonation takes place
- User credentials are exchanged only between the user and the application that “owns” the credentials (in case of Business Central, it’s Azure Active Directory)
- Connecting application never has access to user’s credentials

Token-based access

Web-based protocol

What is OAuth 2.0

NAV TECH DAYS 2019
mibuso.com

An open standard for access delegation, that allows users to grant applications access to user information in other applications, without exposing user credentials.

Delegation

- User access is delegated to an application
- No user impersonation takes place
- User credentials are exchanged only between the user and the application that “owns” the credentials (in case of Business Central, it’s Azure Active Directory)
- Connecting application never has access to user’s credentials

Token-based access

- Connecting application only has access to an access token
- Access tokens are short-lived, issued to a specific client, and cannot be reused
- Tokens are issued by an authorization server (Azure Active Directory) with the approval of the resource owner (Business Central)

Web-based protocol

What is OAuth 2.0

NAV TECH DAYS 2019
mibuso.com

An open standard for access delegation, that allows users to grant applications access to user information in other applications, without exposing user credentials.

Delegation

- User access is delegated to an application
- No user impersonation takes place
- User credentials are exchanged only between the user and the application that “owns” the credentials (in case of Business Central, it’s Azure Active Directory)
- Connecting application never has access to user’s credentials

Token-based access

- Connecting application only has access to an access token
- Access tokens are short-lived, issued to a specific client, and cannot be reused
- Tokens are issued by an authorization server (Azure Active Directory) with the approval of the resource owner (Business Central)

Web-based protocol

- OAuth is REST-based
- Not applicable to non-HTTP scenarios
- All communication happens through HTTP REST endpoints

OAuth 2.0 is not a single thing

NAV TECH DAYS 2019
mibuso.com

- There are different flows in OAuth:

Authorization Code Grant	Implicit Grant	On-Behalf-Of
Client Credentials Grant	Device Authorization Grant	Resource Owner Password Credentials Grant

OAuth 2.0 is not a single thing

- There are different flows in OAuth:

Authorization Code Grant

Third-party web and native apps

Implicit Grant

On-Behalf-Of

Client Credentials Grant

Device Authorization Grant

Resource Owner Password Credentials Grant

OAuth 2.0 is not a single thing

- There are different flows in OAuth:

Authorization Code Grant Third-party web and native apps	Implicit Grant Single-page browser apps	On-Behalf-Of
Client Credentials Grant	Device Authorization Grant	Resource Owner Password Credentials Grant

OAuth 2.0 is not a single thing

- There are different flows in OAuth:

Authorization Code Grant Third-party web and native apps	Implicit Grant Single-page browser apps	On-Behalf-Of Pass-through multi-layer scenarios
Client Credentials Grant	Device Authorization Grant	Resource Owner Password Credentials Grant

OAuth 2.0 is not a single thing

NAV TECH DAYS 2019
mibuso.com

- There are different flows in OAuth:

Authorization Code Grant Third-party web and native apps	Implicit Grant Single-page browser apps	On-Behalf-Of Pass-through multi-layer scenarios
Client Credentials Grant Service-to-service	Device Authorization Grant	Resource Owner Password Credentials Grant

OAuth 2.0 is not a single thing

- There are different flows in OAuth:

Authorization Code Grant Third-party web and native apps	**Implicit Grant** Single-page browser apps	**On-Behalf-Of** Pass-through multi-layer scenarios
Client Credentials Grant Service-to-service	**Device Authorization Grant** Smart devices and IoT	**Resource Owner Password Credentials Grant**

OAuth 2.0 is not a single thing

NAV TECH DAYS 2019
mibuso.com

- There are different flows in OAuth:

Authorization Code Grant Third-party web and native apps	**Implicit Grant** Single-page browser apps	**On-Behalf-Of** Pass-through multi-layer scenarios
Client Credentials Grant Service-to-service	**Device Authorization Grant** Smart devices and IoT	**Resource Owner Password Credentials Grant** First-party apps with high level of trust

Which flows are supported by Business Central

NAV TECH DAYS 2019
mibuso.com

- OAuth is a feature of Azure Active Directory, not Business Central
- You can obtain tokens using all flows (tokens come from Azure)

- However (and a big one at that):
  - Business Central rejects tokens when actual user cannot be identified
  - This eliminates service-to-service calls
  - Reason: license compliance

- Officially, according to Microsoft, these are supported:
  - Authorization Code Grant
  - Implicit Grant

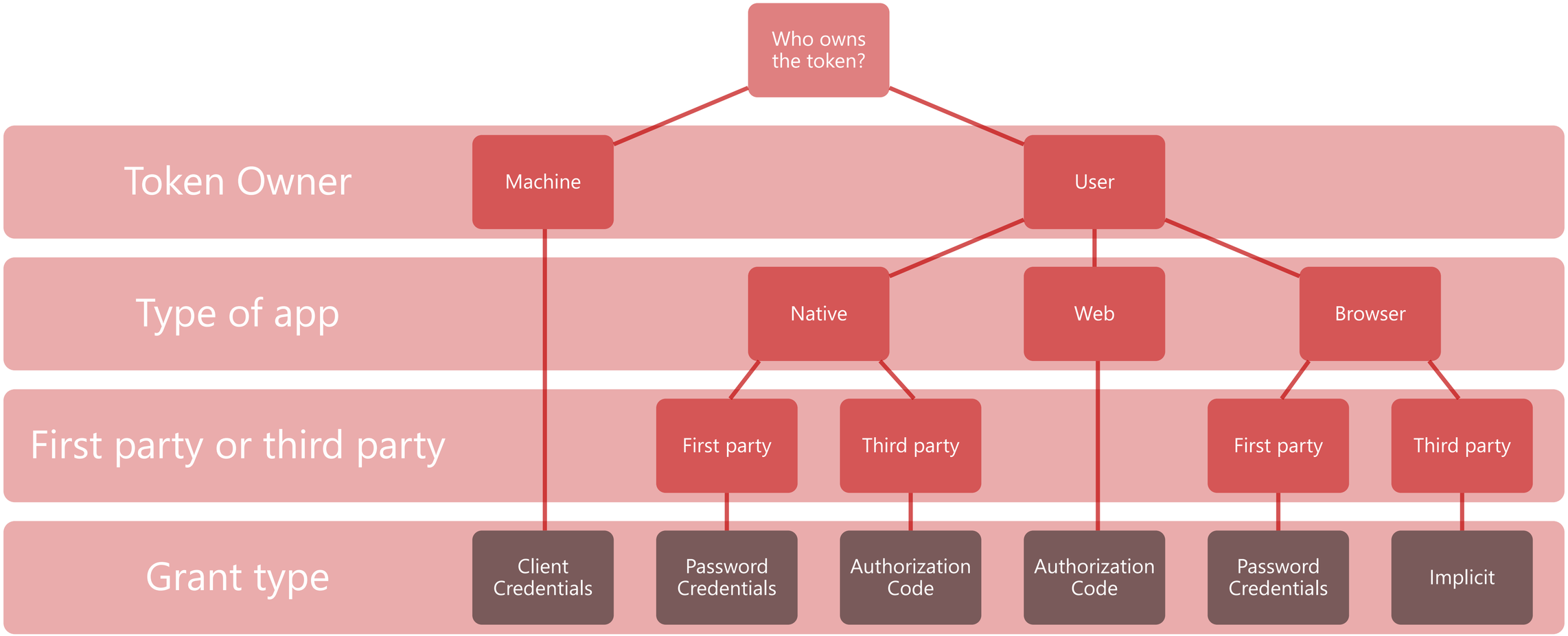
Psssst, don't tell anyone

NAV TECH DAYS 2019
mibuso.com

Resource Owner Password Credential Grant works just fine.

Use this one for service-to-service communication.

Selecting the right grant type



Authorization Code Grant Flow

NAV TECH DAYS 2019
mibuso.com

- The most typical grant type
- The most comprehensive grant type
- Other grant types perform a subset of operations of Authorization Code grant

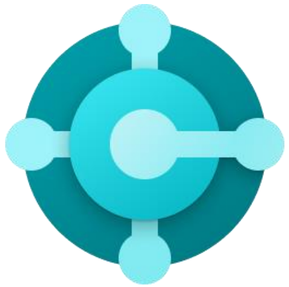
- The grant type most familiar to users

Parties in OAuth 2.0 authorization code flow

OAuth 2.0 process has three parties

Resource owner

The application that owns the resources the user has access to, and that the user wants to delegate access to.



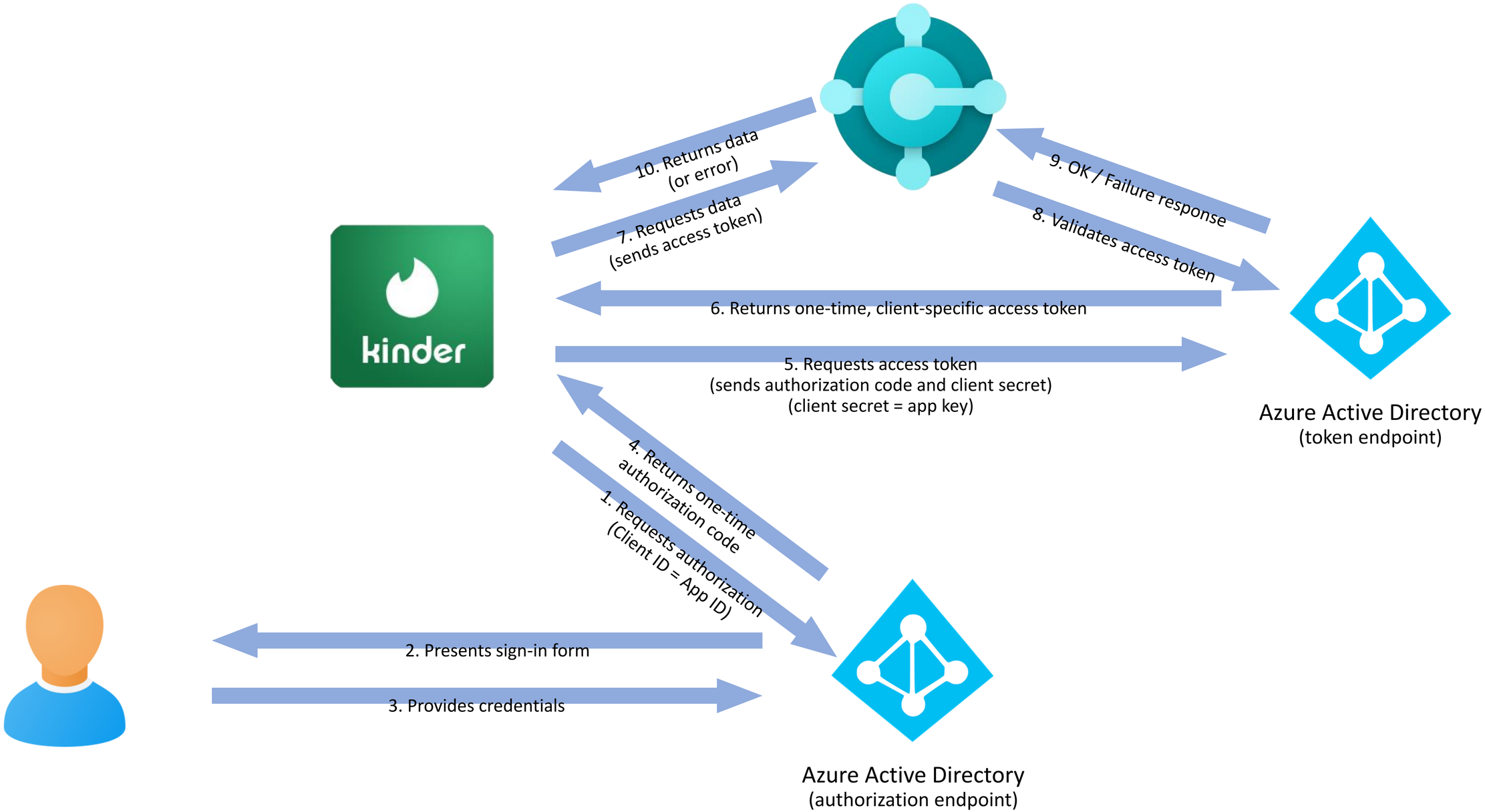
Authorization server

An authority trusted by the resource owner, that will issue access tokens on behalf of the resource owner.

End-user application

Any application that requires access to the resources.

How does OAuth 2.0 work in Business Central



Implementing OAuth 2.0 authentication

Implementing the client side of OAuth 2.0 protocol is your responsibility
Your OAuth 2.0 client must provide the described functionality

Redirect to /oauth2/authorize endpoint

When user requests a Business Central resource, redirect to the known endpoint.

Provide a callback URL that receives authorization code.

POST to /oauth2/token endpoint

Use HTTP POST method to send the authorization code to the known endpoint

This is a regular REST invocation.

Response contains the access token.

Use access token with Business Central

Every REST request against Business Central API must include the token in the authorization headers.

When token expires, request refresh token.

UID: ntd2019
PWD: Antwerp.19

Get ready for the demo

NAV TECH DAYS 2019
mibuso.com

Vjeko.com
ideas in the cloud

Install the Expo client app…

…from your app store

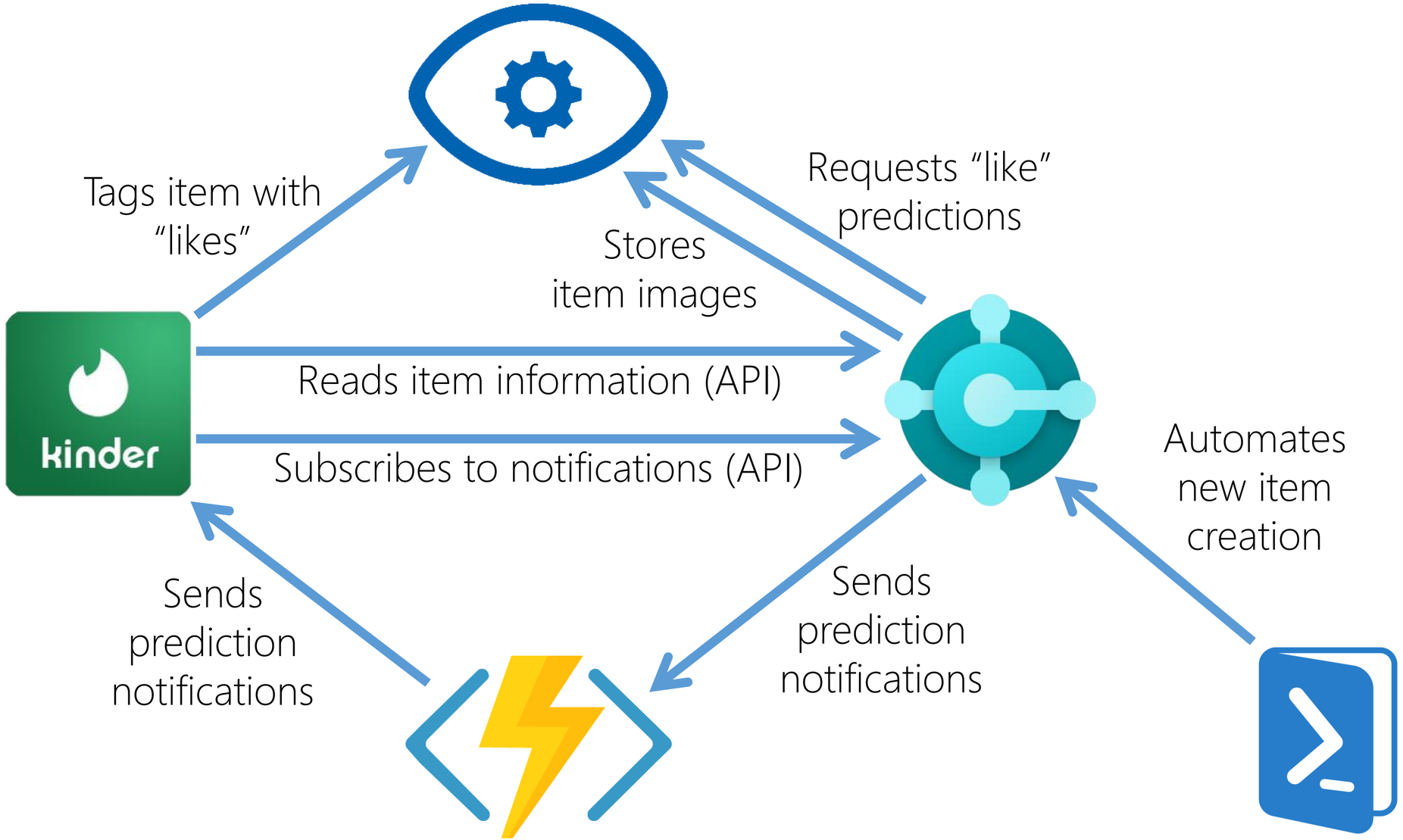
waldo.be

GET IT ON Google Play

UID: ntd2019
PWD: Antwerp.19

Download on the App Store

Architectural overview

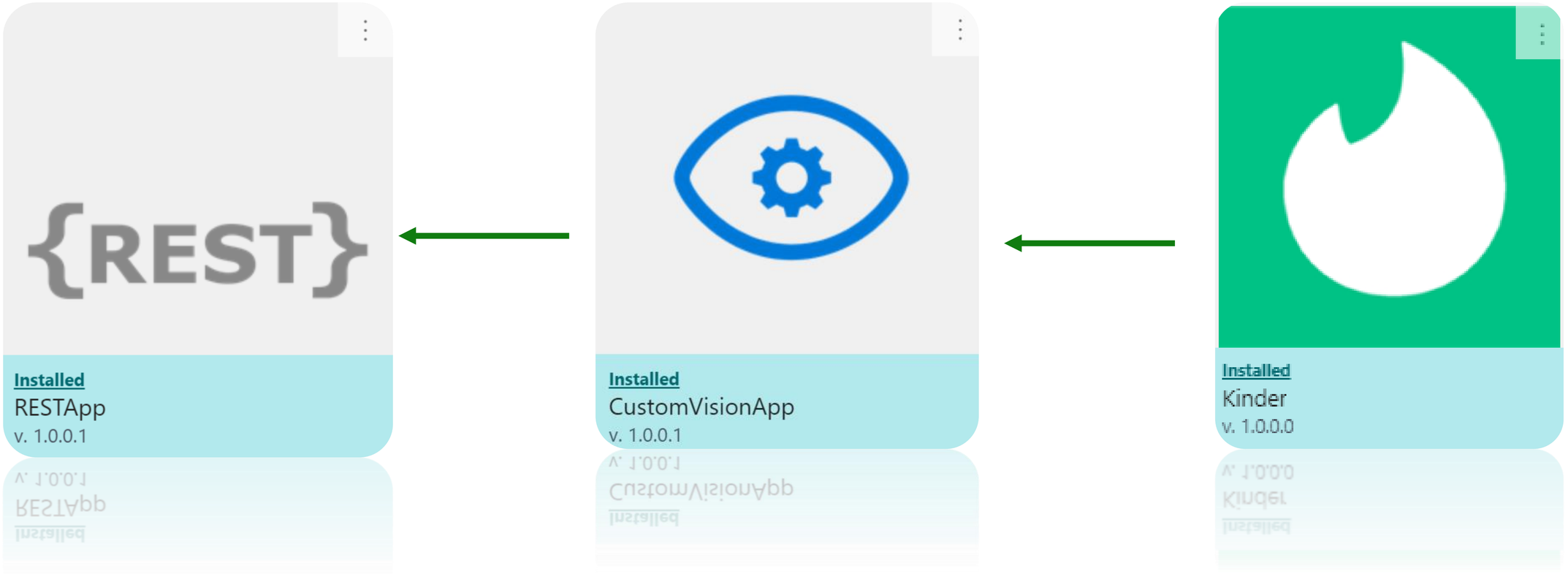


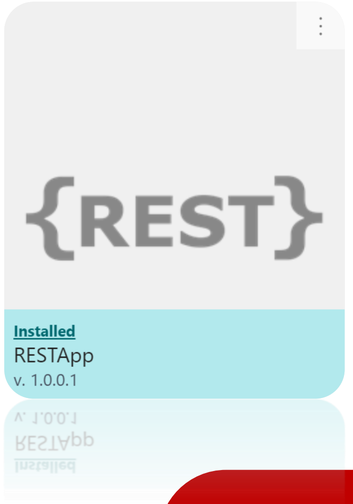
NAV TECH DAYS 2019

mibuso.com

Architectural Considerations in AL

Dependencies





Only responsible for REST calls

Helpers for JSON and HTTP objects

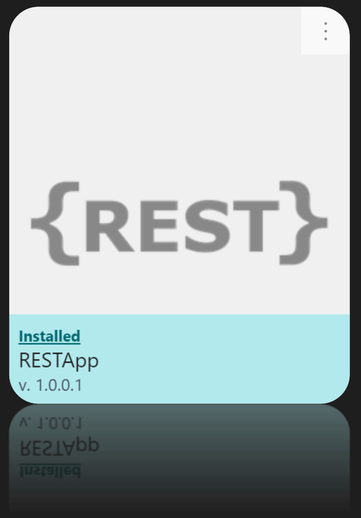
Introduce logging

- Request/Response
- Duration
- Timings
- …

Events

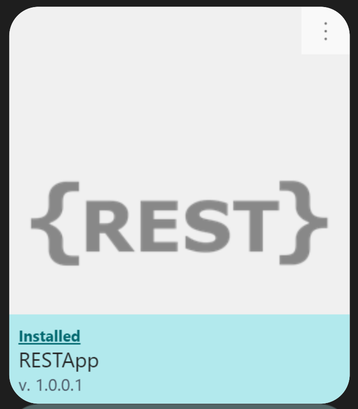
- OnBeforeSend
- OnAfterSend
- …

Helper to call all http-methods in the right order

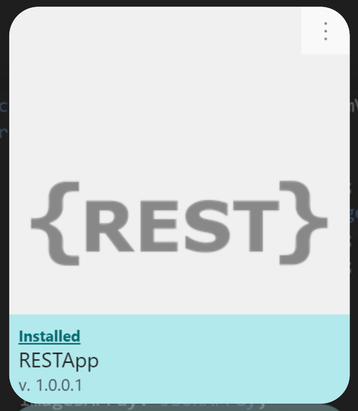


NAV TECH DAYS 2019

mibuso.com



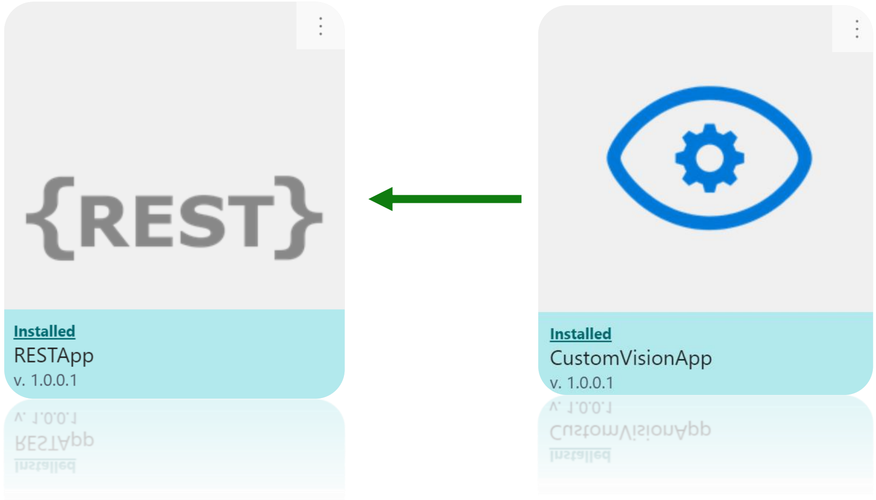

```
local procedure CreateImageInCustomVision(var CustomVisionImageKND: Codeunit "CustomVision Image KND")
var
    WebClient: HttpClient;
    WebRequest: HttpRequestMessage;
    WebResponse: HttpResponseMessage;
    WebRequestHeaders: HttpHeaders;
    WebContentHeaders: HttpHeaders;
    WebContent: HttpContent;
    SendSuccess: Boolean;
    Instr: InStream;
    ResponseContentText: Text;
    ImagesArray: JsonArray;
    ImagesObject: JsonObject;
    ImagesArrayText: Text;
    TempBlob: REcord TempBlob;
begin
    WebRequest.Method := 'Post';
    WebRequest.SetRequestUri('https://westeurope.api.cognitive.microsoft.com/customvision/v3.0/training/p

    WebRequest.GetHeaders(WebRequestHeaders);
    WebRequestHeaders.Add('Training-Key', 'ef073ddb45034159a72a35c1dc4b5f1d');


    ImagesObject.Add('Images', ImagesArray);
    ImagesArray.Add(CustomVisionImageKND.GetImageObject());
    ImagesObject.WriteTo(ImagesArrayText);
    WebContent.WriteFrom(ImagesArrayText);
    WebRequest.Content(WebContent);

    webcontent.GetHeaders(WebContentHeaders);
    if WebContentHeaders.Contains('Content-Type') then
        WebContentHeaders.Remove('Content-Type');
    WebContentHeaders.Add('Content-Type', 'application/json');

    SendSuccess := WebClient.Send(WebRequest, WebResponse);

    if WebResponse.IsSuccessStatusCode() then
        message('Success')
    else
        message('Error: ' + format(WebResponse.ReasonPhrase()));

    TempBlob.Blob.CreateInStream(Instr);
    WebResponse.Content().ReadAs(ResponseContentText);

    message(ResponseContentText);

end;
```




```
loc                                    Vision(var CustomVisionImageKND: Codeunit "CustomVision Image KND")
var
                                       e;
    ImagesObject: JsonObject;
    ImagesArrayText: Text;
    TempBlob: REcord TempBlob;
begin
    WebRequest.Method := 'Post';
    WebRequest.SetRequestUri('https://westeurope.api.cognitive.microsoft.com/customvis

    WebRequest.GetHeaders(WebRequestHeaders);
    WebRequestHeaders.Add('Training-Key', 'ef073ddb45034159a72a35c1dc4b5f1d');


    ImagesObject.Add('Images', ImagesArray);
    ImagesArray.Add(CustomVisionImageKND.GetImageObject());
    ImagesObject.WriteTo(ImagesArrayText);
    WebContent.WriteFrom(ImagesArrayText);
    WebRequest.Content(WebContent);

    webcontent.GetHeaders(WebContentHeaders);
    if WebContentHeaders.Contains('Content-Type') then
        WebContentHeaders.Remove('Content-Type');
    WebContentHeaders.Add('Content-Type', 'application/json');

    SendSuccess := WebClient.Send(WebRequest, WebResponse);

    if WebResponse.IsSuccessStatusCode() then
        message('Success')
    else
        message('Error: ' + format(WebResponse.ReasonPhrase()));

    TempBlob.Blob.CreateInStream(Instr);
    WebResponse.Content().ReadAs(ResponseContentText);

    message(ResponseContentText);

end;
```

```
procedure SendImage(var Reponse: codeunit "CustomVision Images Resp WLD")
var
    CVSetup: Record "CustomVision Setup WLD";
    RESTHelperWLD: Codeunit "REST Helper WLD";
    UrlLbl: Label '%1/customvision/v3.0/training/projects/%2/images/files', Locked = true;
    ImagesArrayText: Text;
begin
    CVSetup.Get();

    RESTHelperWLD.Initialize('Post', StrSubstNo(UrlLbl, CVSetup.EndPoint, CVSetup.ProjectId));

    RESTHelperWLD.AddRequestHeader('Training-Key', CVSetup."API Key");

    GetMainObject().WriteTo(ImagesArrayText);

    RESTHelperWLD.AddBody(ImagesArrayText);

    RESTHelperWLD.SetContentType('application/json');

    if not RESTHelperWLD.Send() then
        message('Error: '
            + format(RESTHelperWLD.GetHttpStatusCode()) + ' - '
            + RESTHelperWLD.GetResponseContentAsText());

    Reponse.initialize(RESTHelperWLD.GetResponseContentAsText());
end;
```

NAV TECH DAYS 2019



NAV TECH DAYS 2019
mibuso.com

Data

- Setup
- Tags

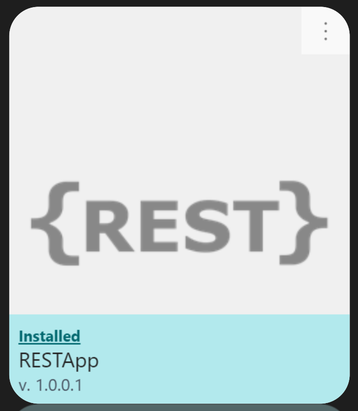
Methods

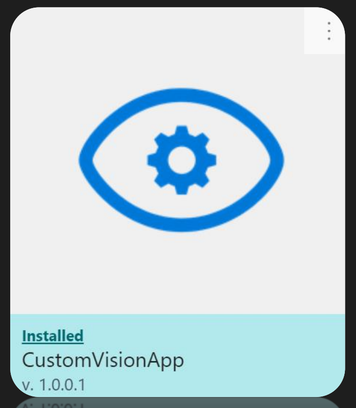
- Get TagsCreate TagCreate Image
- Create Image
- Create Tag Create Image
- Create Image
- Create Image
- Predict
- Train
- …

Objects

- Image
- ImageReponse
- Tag

Uses the REST App

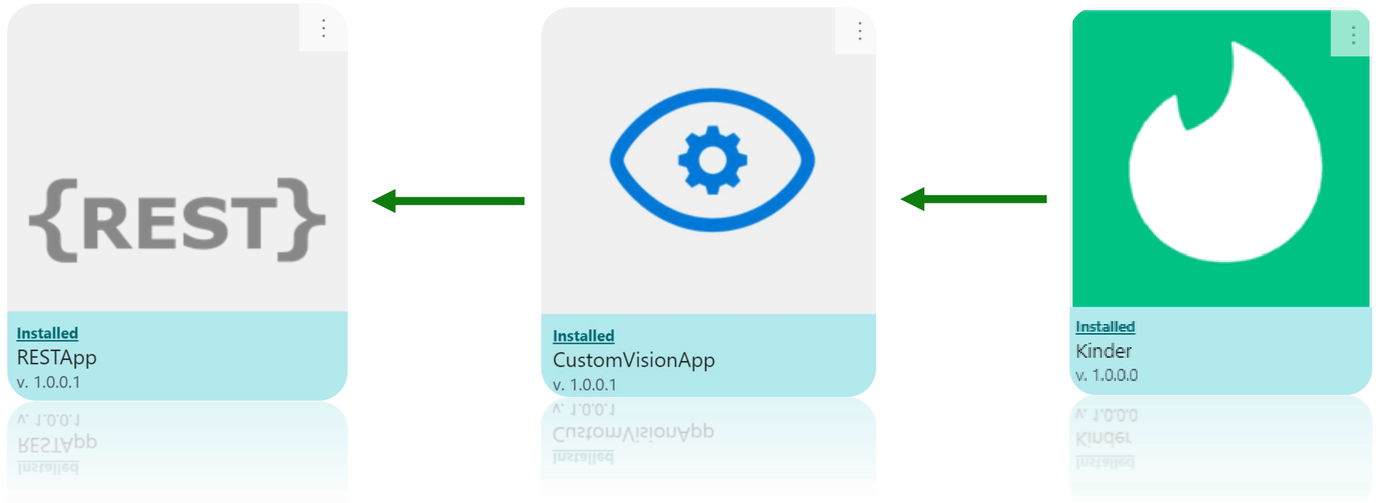




NAV TECH DAYS 2019

mibuso.com

```
local procedure GetTagsFromAPI(): Text
var
    CVSetup: Record "CustomVision Setup WLD";
    RESTHelper: Codeunit "REST Helper WLD";
    UrlLbl: label '%1/customvision/v3.0/training/projects/%2/tags', Locked = true;
begin
    CVSetup.Get();

    RESTHelper.Initialize('Get', StrSubstNo(UrlLbl, CVSetup.EndPoint, CVSetup.ProjectId));

    RESTHelper.AddRequestHeader('Training-Key', CVSetup."API Key");

    if not RESTHelper.Send() then
        message('Error: '
            + format(RESTHelper.GetHttpStatusCode()) + ' - '
            + RESTHelper.GetResponseContentAsText());

    exit(RESTHelper.GetResponseContentAsText());
end;
```

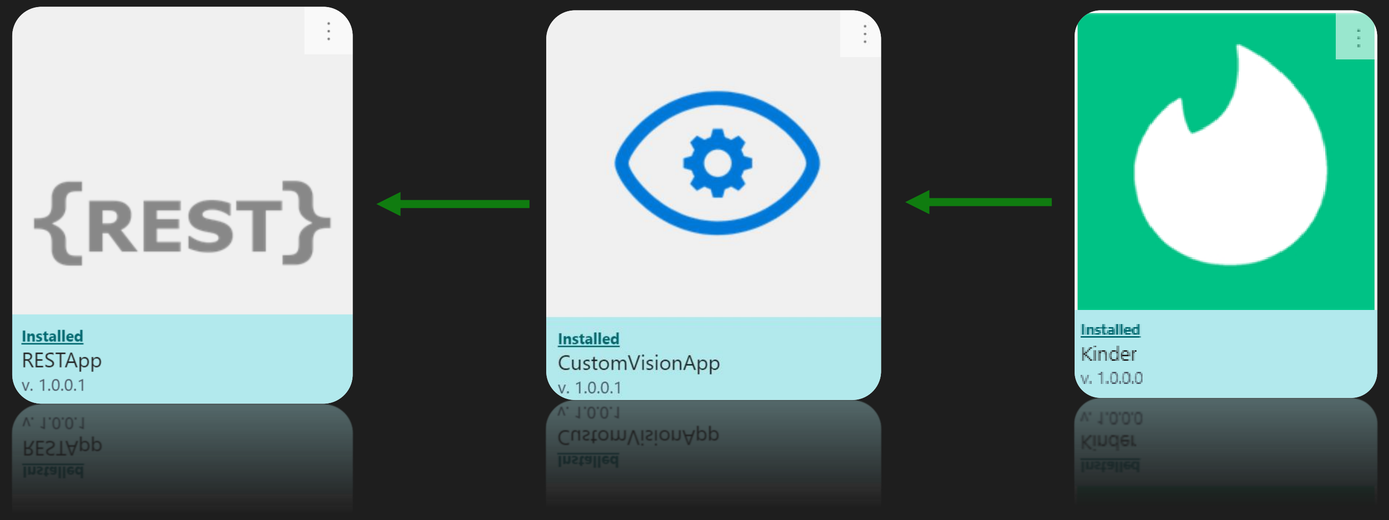


NAV TECH DAYS 2019
mibuso.com

The actual functionality

- New business logic
- New data model
- API

Uses CustomVision



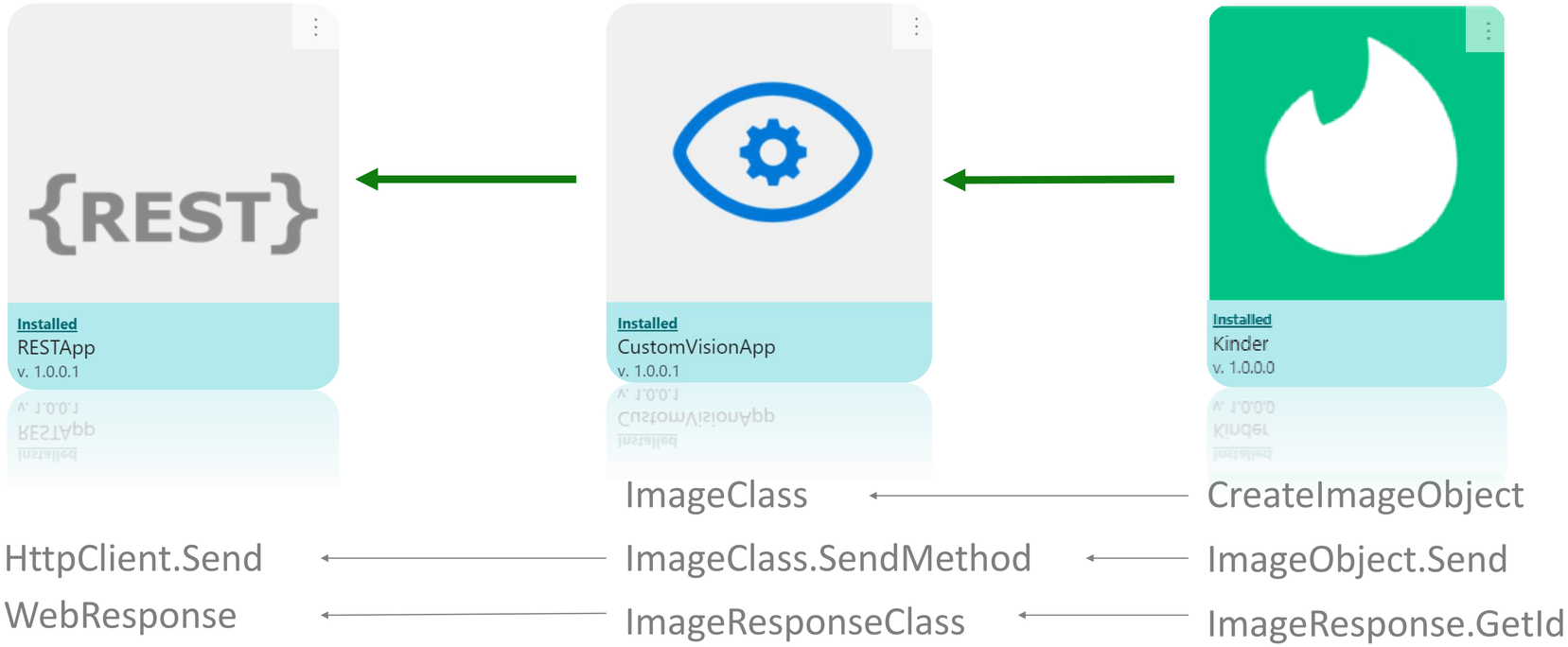
NAV TECH DAYS 2019

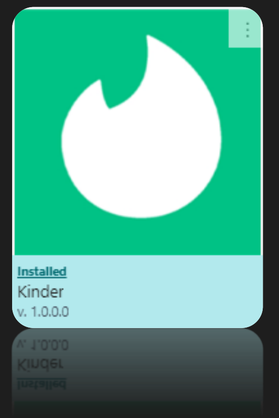
mibuso.com

```
local procedure SendImageToCustomVision(var LegoItem: Record "LegoItem KND"; var CustomVisionImagesKND: Codeunit "CustomVision Images WLD";
var
    Base64Convert: Codeunit "Base64 Convert";
    CustomVisionImageWLD: Codeunit "CustomVision Image WLD";
    Instr: Instream;
    Base64Text: Text;
begin
    LegoItem.CalcFields(ImageContent);
    LegoItem.ImageContent.CreateInStream(Instr);
    Base64Text := Base64Convert.ToBase64(Instr);

    CustomVisionImageWLD.SetName(LegoItem.Name);
    CustomVisionImageWLD.SetContents(Base64Text);

    CustomVisionImagesKND.AddImage(CustomVisionImageWLD);

    CustomVisionImagesKND.SendImage(CustomVisionImagesRespKND);
end;
```

“Objectification”

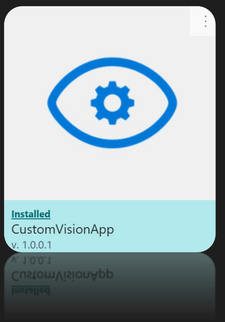
Try to make it easy to work with the objects that needs to be sent to an API call, and the objects you get back.





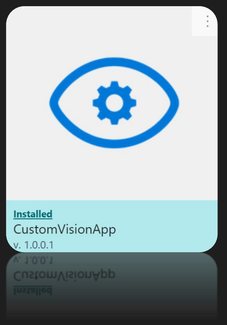

```
local procedure SendImageToCustomVision(var LegoItem: Record "LegoItem KND"; var CustomVisionImages: Codeunit "CustomVision Images WLD");
var
    Base64Convert: Codeunit "Base64 Convert";
    CustomVisionImage: Codeunit "CustomVision Image WLD";
    Instr: Instream;
    Base64Text: Text;
begin
    LegoItem.CalcFields(ImageContent);
    LegoItem.ImageContent.CreateInStream(Instr);
    Base64Text := Base64Convert.ToBase64(Instr);

    CustomVisionImage.SetName(LegoItem.Name);
    CustomVisionImage.SetContents(Base64Text);

    CustomVisionImages.AddImage(CustomVisionImage);

    CustomVisionImages.SendImage(CustomVisionImagesRespKND);
end;
```

```
codeunit 79932 "CustomVision Image WLD"
{
    Eric Wauters, 2 days ago | 1 author (Eric Wauters)
    var
        //Images: JsonArray;
        9 references
        MainObject: JsonObject;
        2 references
        ImageTags: JsonArray;

    1 reference
    procedure GetImageObject(): JsonObject
    begin
        exit(MainObject);
    end;

    0 references
    procedure SetName(name: Text)
    begin
        if MainObject.Contains('name') then
            MainObject.Replace('name', name)
        else
            MainObject.Add('name', name);
    end;

    0 references
    procedure SetContents(contents: Text)
    begin
        if MainObject.Contains('contents') then
            MainObject.Replace('contents', contents)
        else      Eric Wauters, 2 days ago • CostumVision App
            MainObject.Add('contents', contents);
    end;

    0 references
    procedure AddImageTag(ImageTag: Text)
    var
        CustomVisionTagWLD: Codeunit "CustomVision Tag WLD";
    begin
        if (not MainObject.Contains('tagIds')) then
            MainObject.Add('tagIds', ImageTags);

        CustomVisionTagWLD.Initialize(ImageTag);

        ImageTags.Add(CustomVisionTagWLD.GetId());
    end;
```

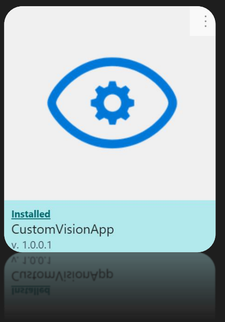



```
local procedure SendImageToCustomVision(var LegoItem: Record "LegoItem KND"; var Cus
var
    Base64Convert: Codeunit "Base64 Convert";
    CustomVisionImage: Codeunit "CustomVision Image WLD";
    Instr: Instream;
    Base64Text: Text;
begin
    LegoItem.CalcFields(ImageContent);
    LegoItem.ImageContent.CreateInStream(Instr);
    Base64Text := Base64Convert.ToBase64(Instr);

    CustomVisionImage.SetName(LegoItem.Name);
    CustomVisionImage.SetContents(Base64Text);

    CustomVisionImages.AddImage(CustomVisionImage);

    CustomVisionImages.SendImage(CustomVisionImagesRespKND);
end;
```

```
codeunit 79933 "CustomVision Images WLD"
{
    Eric Wauters, 2 days ago | 1 author (Eric Wauters)
    var
        5 references
        MainObject: JsonObject;
        2 references
        images: JsonArray;
        2 references
        tagIds: JsonArray;

    1 reference
    procedure GetMainObject(): JsonObject
    begin
        exit(MainObject);
    end;       Eric Wauters, 2 days ago • CostumVision App

    0 references
    procedure AddImage(Image: Codeunit "CustomVision Image WLD")
    begin
        if (not MainObject.Contains('images')) then
            MainObject.Add('images', images);

        images.Add(Image.GetImageObject());
    end;

    0 references
    procedure AddImageTag(ImageTag: Text)
    begin
        if (not MainObject.Contains('tagIds')) then
            MainObject.Add('tagIds', tagIds);

        tagIds.Add(ImageTag);
    end;

    0 references
    procedure SendImage(var Reponse: codeunit "CustomVision Images Resp WLD")
    var
        CVSetup: Record "CustomVision Setup WLD";
```




```
local procedure SendImageToCustomVision(var LegoItem: Record "LegoItem KND
var
    Base64Convert: Codeunit "Base64 Convert";
    CustomVisionImage: Codeunit "CustomVision Image WLD";
    Instr: Instream;
    Base64Text: Text;
begin
    LegoItem.CalcFields(ImageContent);
    LegoItem.ImageContent.CreateInStream(Instr);
    Base64Text := Base64Convert.ToBase64(Instr);

    CustomVisionImage.SetName(LegoItem.Name);
    CustomVisionImage.SetContents(Base64Text);

    CustomVisionImages.AddImage(CustomVisionImage);

    CustomVisionImages.SendImage(CustomVisionImagesRespKND);
end;
```




```
procedure SendImage(var Reponse: codeunit "CustomVision Images Resp WLD")
var
    CVSetup: Record "CustomVision Setup WLD";
    RESTHelperWLD: Codeunit "REST Helper WLD";
    UrlLbl: Label '%1/customvision/v3.0/training/projects/%2/images/files', Locked = true;
    ImagesArrayText: Text;
begin
    CVSetup.Get();

    RESTHelperWLD.Initialize('Post', StrSubstNo(UrlLbl, CVSetup.EndPoint, CVSetup.ProjectId));

    RESTHelperWLD.AddRequestHeader('Training-Key', CVSetup."API Key");

    GetMainObject().WriteTo(ImagesArrayText);

    RESTHelperWLD.AddBody(ImagesArrayText);

    RESTHelperWLD.SetContentType('application/json');

    if not RESTHelperWLD.Send() then
        message('Error: '
            + format(RESTHelperWLD.GetHttpStatusCode()) + ' - '
            + RESTHelperWLD.GetResponseContentAsText());

    Reponse.initialize(RESTHelperWLD.GetResponseContentAsText());
end;
```

```
local procedure ProcessResponse(var LegoItem: Record "LegoItem KND"; var CustomVisionImagesRespWLD: Codeunit "CustomVision Images Resp WLD")
begin
    LegoItem.CustomVisionId := CopyStr(CustomVisionImagesRespWLD.GetImageId(), 1, MaxStrLen(LegoItem.CustomVisionId));
    LegoItem.Uploaded := true;
    LegoItem.Modify(true);
end;
```

```
procedure SendImage(var Reponse: codeunit "CustomVision Images Resp WLD")
var
    CVSetup: Record "CustomVision Setup WLD";
    RESTHelperWLD: Codeunit "REST Helper WLD";
    UrlLbl: Label '%1/customvision/v3.0/training/projects/%2/images/files', Locked = true;
    ImagesArrayText: Text;
begin
    CVSetup.Get();

    RESTHelperWLD.Initialize('Post', StrSubstNo(UrlLbl, CVSetup.EndPoint, CVSetup.ProjectId));

    RESTHelperWLD.AddRequestHeader('Training-Key', CVSetup."API Key");

    GetMainObject().WriteTo(ImagesArrayText);

    RESTHelperWLD.AddBody(ImagesArrayText);

    RESTHelperWLD.SetContentType('application/json');

    if not RESTHelperWLD.Send() then
        message('Error: '
            + format(RESTHelperWLD.GetHttpStatusCode()) + ' - '
            + RESTHelperWLD.GetResponseContentAsText());

    Reponse.initialize(RESTHelperWLD.GetResponseContentAsText());
end;
```

NAV TECH DAYS 2019

mibuso.com

Notifications

Don’t call us, we will call you

Notifications

- Webhooks for Business Central APIs
- Interested parties can subscribe to data changes
- Business Central sends out notifications when changes occur

- Both standard and custom APIs expose the **`/subscriptions`** endpoint

```
https://api.businesscentral.dynamics.com/v2.0/<tenant_id>/<sandboxname>
/api/waldo/killerApp/v1.0
```

Notifications

- Webhooks for Business Central APIs
- Interested parties can subscribe to data changes
- Business Central sends out notifications when changes occur

- Both standard and custom APIs expose the **`/subscriptions`** endpoint

```
https://api.businesscentral.dynamics.com/v2.0/<tenant_id>/<sandboxname>
/api/waldo/killerApp/v1.0/subscriptions
```

Operations on the `/subscriptions` endpoint

`GET` `POST` `PATCH` `DELETE`

Operations on the `/subscriptions` endpoint

GET	POST	PATCH	DELETE
• Retrieves list of existing subscriptions			

Operations on the `/subscriptions` endpoint

GET	POST	PATCH	DELETE
• Retrieves list of existing subscriptions	• Creates a new subscription		

Operations on the `/subscriptions` endpoint

GET	POST	PATCH	DELETE
• Retrieves list of existing subscriptions	• Creates a new subscription	• Renews an existing subscription	

Operations on the `/subscriptions` endpoint

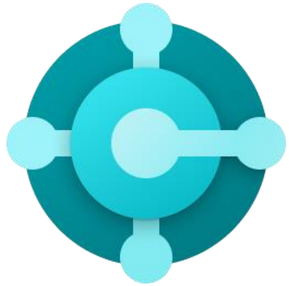
GET	POST	PATCH	DELETE
• Retrieves list of existing subscriptions	• Creates a new subscription	• Renews an existing subscription	• Deletes an existing subscription

Parties involved in subscription

NAV TECH DAYS 2019
mibuso.com

Business Central

Provides subscription endpoints.



End-user application

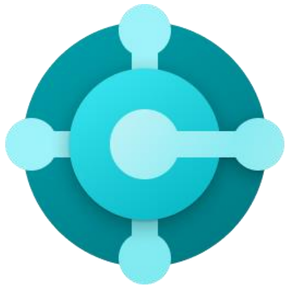
Application that requires notifications

Parties involved in subscription

NAV TECH DAYS 2019
mibuso.com

Business Central

Provides subscription endpoints.



Notification dispatcher

A middleware component needed in case the end-user application is a native application or a single-page browser application.

End-user application

Application that requires notifications

Create a new subscription

NAV TECH DAYS 2019
mibuso.com

- Send a POST request to the **`/subscribe`** endpoint

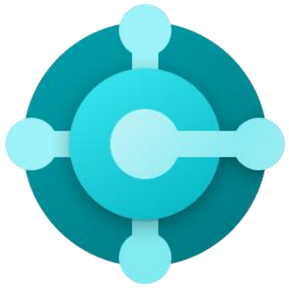
```
post {{baseurl}}/subscriptions
Authorization: Bearer {{oauth_token}}
Content-Type: application/json

{
    "clientState": "a_random_string",
    "notificationUrl": "https://demo-kinder.azurewebsites.net/api/subscription-notify",
    "resource": "{{baseurl}}/companies({{company}})/legoItems"
}
```

- **`clientState`** a random string known only to the subscribing party
- **`notificationUrl`** URL that Business Central calls to confirm the subscription
- **`resource`** API entity to subscribe to

Create a new subscription

```
post {{baseurl}}/subscriptions
Authorization: Bearer {{oauth_token}}
Content-Type: application/json

{
    "clientState": "a_random_string",
    "notificationUrl": "https://demo-kinder.azurewebsites.net/api/subscription-notify",
    "resource": "{{baseurl}}/companies({{company}})/legoItems"
}
```

- Handshake occurs



Sends a GET request to the notificationUrl

Calls the notificationUrl

Echoes back the clientState and validationToken

Passes back the validationToken

notificationUrl

Receiving notifications

- Business Central automatically sends notifications to `notificationUrl`

- Changes are accumulated in 30-second chunks
  - When the first change occurs, a timeout is created
  - Any change happening within the next 30 seconds is noted
  - After timeout elapses, all noted changes are sent as one message

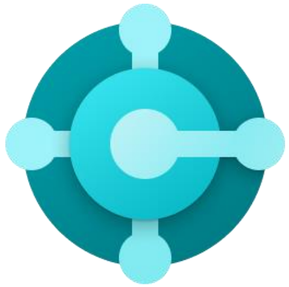
Receiving notifications

- Notification payload
  - Does not contain the actual changes that occurred
  - Contains the list of endpoints to invoke and the change type that occurred

```
{
    "subscriptionId": "53fe5e1f5ceb4b87bf2a6c82cac1243d",
    "clientState": "",
    "expirationDateTime": "2019-10-11T21:03:45Z",
    "resource": "https://api.businesscentral.dynamics.com/v2.0/
    ab174d1e-2020-46ce-9da2-d8fc6d03547f/DEV/api/waldo/kinder/v1.0/companies
    (147adb4d-73df-41ea-8de3-f2ca7cc9d73f)/legoItemPredictions
    (3adaed16-22ec-e911-a815-000d3a1f3389)",
    "changeType": "created",
    "lastModifiedDateTime": "2019-10-11T12:24:38.18Z"
}
```

Receiving notifications


```
{
    "subscriptionId": "53fe5e1f5ceb4b87bf2a6c82cac1243d",
    "clientState": "",
    "expirationDateTime": "2019-10-11T21:03:45Z",
    "resource": "https://api.businesscentral.dynamics.com/v2.0/
    ab174d1e-2020-46ce-9da2-d8fc6d03547f/DEV/api/waldo/kinder/v1.0/companies
    (147adb4d-73df-41ea-8de3-f2ca7cc9d73f)/legoItemPredictions
    (3adaed16-22ec-e911-a815-000d3a1f3389)",
    "changeType": "created",
    "lastModifiedDateTime": "2019-10-11T12:24:38.18Z"
}
```



Sends the notification payload

Notifies the client whichever way (push, pull, etc.)

Click to add title

- _vjeko & _waldo

- bros

- Give us feedback in the App

- TODO: make at least 2 jokes

Click to add title

- _vjeko & _waldo

- bros

- Give us feedback in the App

- TODO: make at least 2 jokes