



**Vlaanderen**  
is landbouw & visserij

## TECHNICAL DOCUMENTATION

### **Maturity Web API**

**smartdots**

**ILVO**

Institute for Agricultural  
and Fisheries Research

[www.ilvo.vlaanderen.be](http://www.ilvo.vlaanderen.be)

<i>Date</i>	<i>Version</i>
2022/10	Version 1.0
2023/02	Version 2.0

Contact:

**Wim Allegaert**, wim.allegaert@ilvo.vlaanderen.be

**Kevin De Coster**, kevin.decoster@ilvo.vlaanderen.be

Institute for Agricultural and Fisheries Research  
Animal Sciences Unit - Fisheries and Aquatic Production  
Jacobsenstraat 1, B-8400 Oostende, Belgium  
Tel + 32 59 34 22 50

<http://www.ilvo.vlaanderen.be>



# Table of contents

---

1. Introduction.....	4
2. Basic concepts .....	5
2.1. Requests .....	5
2.2. Responses .....	5
2.3. Example .....	5
3. Data Transfer Objects .....	6
3.1. DTO classes .....	6
3.2. Dynamic objects .....	9
4. Methods.....	10
4.1. Screen 1: Authentication.....	10
4.2. Screen 2: Analyses .....	11
4.3. Screen 3: Maturity .....	14
5. Swagger.....	19

# 1. Introduction

SmartDots is a software application for performing age determination and maturity analysis of fish.

To determine the maturity of the fish end-users study images on which they have to decide a few parameters. The user has some tools available to make temporary measurements on the image. In addition an annotation can a quality indication and some comment.

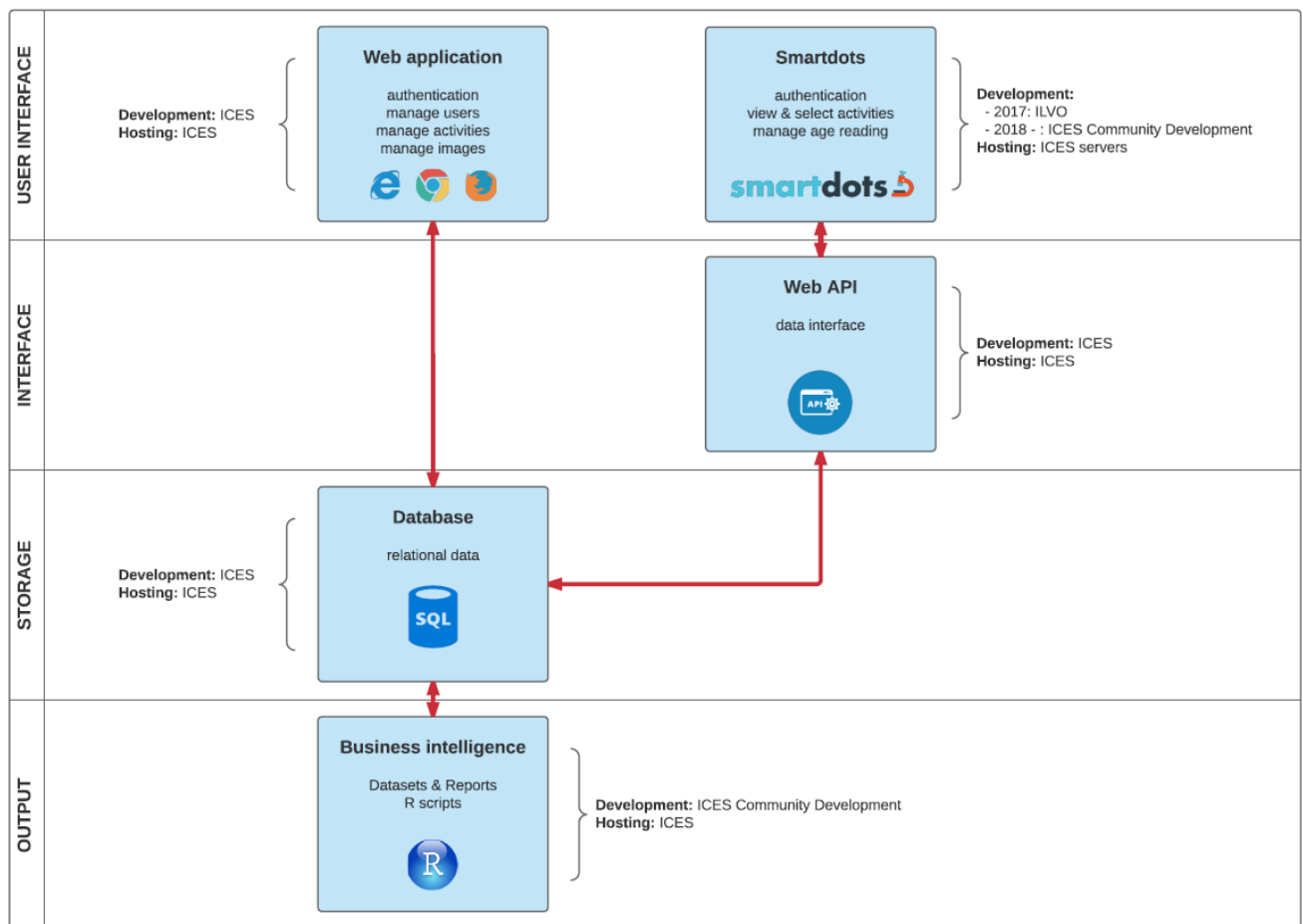
The SmartDots user interface exists out of 3 main screens:

- Screen 1: Authentication
- Screen 2: The Analyses/Event screen
- Screen 3: The Annotate screen, depending on the chosen event/analysis this can either be AgeReading, Maturity, or something else

SmartDots is developed as a client-server application. SmartDots (the client) is a Windows-based application (Windows Presentation Foundation). SmartDots exchange messages with a Web API (the server) in a request-response messaging pattern. The Web API accesses the database.

The Web API can be written in any technology. Communication between the Web API and the software happens through JSON-objects. The implementation in this manual shows how it is done in ASP.NET Core. There's also a swagger page for the latest version: <https://webapi.smartfisheries.be/swagger/>

Notice the different naming conventions for property-names between .NET and json (PascalCase and camelCase respectively). For sending objects to SmartDots, the case doesn't matter.



## 2. Basic concepts

---

### 2.1. Requests

Web API methods used in SmartDots are either POST or GET (PUT and DELETE are not used). POST-requests are used when non-primitive-type objects are send to the Web API. Examples of these objects are Lists and Complex types.

### 2.2. Responses

Every successful Web API response returns a custom 'WebApiResponse' object. The response is in JSON-format. This is the way the SmartDots application expects it.

The WebApiResponse Class looks like this:

```
public class WebApiResponse
{
    public dynamic Result { get; set; }
    public string ErrorMessage { get; set; }
}
```

Where the Result property contains the desired output. This is a dynamic property which means it can contain anything. If an error occurred, for example user permissions, or a code exception, the error can be put in the ErrorMessage property. If SmartDots receives a WebApiResponse, it knows how to handle it.

### 2.3. Example

```
[HttpGet]
[Route("getreadabilityqualities")]
//Returns a List<Quality> of AQ quality codes
public string GetReadabilityQualities(string token)
{
    var webApiResponse = new WebApiResponse();
    if (!HasPermission(token))
    {
        webApiResponse.ErrorMessage = "User has no permission to get Qualities!";
        return Json(webApiResponse);
    }
    try
    {
        using (var ctx = new D1_SmartLabContext())
        {
            var qualities = ctx.Quality.ToList();

            webApiResponse.Result = sdQualities;
            return Json(webApiResponse);
        }
    }
    catch (Exception e)
    {
        webApiResponse.ErrorMessage = e.Message;
        return Json(webApiResponse);
    }
}
```

## 3. Data Transfer Objects

---

The data between SmartDots and the Web API is exchanged via Data Transfer Objects (DTO). DTO objects are used to aggregate the data and reduce the number of requests. The DTO objects are transferred via the WebApiResponse's Result property.

### 3.1. DTO classes

#### 3.1.1. DTO UserAuthentication

```
// Class contains login data
public class DtoUserAuthentication
{
    public string Username { get; set; }
    public string Password { get; set; }
    public DtoAuthenticationMethod AuthenticationMethod { get; set; }
}
```

#### 3.1.2. DTO AuthenticationMethod

```
// Enum contains authentication method data
public enum DtoAuthenticationMethod { Windows, Basic, Token }
```

#### 3.1.3. DTO User

```
// Class contains user and security data
public class DtoUser
{
    public Guid Id { get; set; }
    public string AccountName { get; set; }
    public string Token { get; set; }
}
```

#### 3.1.4. DTO SmartDotsSettings

```
// Class contains SmartDots configuration settings
public class DtoSmartDotsSettings
{
    public bool CanAttachDetachSample { get; set; }
    public bool CanBrowseFolder { get; set; }
    public bool UseSampleStatus { get; set; }
    public bool CanApproveAnnotation { get; set; }
    public bool RequireAQ1ForApproval { get; set; }
    public bool RequireAQForApproval { get; set; }
    public bool RequireParamForApproval { get; set; }
    public bool AutoMeasureScale { get; set; }
    public bool ScanFolder { get; set; }
    public bool AnnotateWithoutSample { get; set; }
    public bool OpenSocket { get; set; }
    public string EventAlias { get; set; }
    public string SampleAlias { get; set; }
    public bool CanMarkEventAsCompleted { get; set; }
    public bool AllowMultipleApprovements { get; set; }
    public bool IgnoreMultiUserColor { get; set; }
    public string MaturityAPI { get; set; }
    public float MinRequiredVersion { get; set; };
}
```

Note:

- **CanAttachDetachSample:** Enable sample attach & detach button in file window. This allows the user to remove the SampleId from the DtoFile or add a new one (Used at ILVO)
- **CanBrowseFolder:** Enable browse button in file window. Only set this 'true' when you allow the user to select pictures from the file system.
- **UseSampleStatus:** Show linked sample status in file window. (First column, colored box)
- **CanApproveAnnotation:** Show property approve in annotation grid & edit window
- **RequireAQ1ForApproval:** Enable quality control: approval only possible for annotations with 'AQ1' quality code. Please make sure there is an 'AQ1' ReadabilityQuality quality code as the software searches for this string.
- **RequireAQForApproval:** Enable quality control: approval only possible for annotation with a AQ code.
- **RequireParamForApproval:** Enable quality control: approval only possible for annotations with a parameter.
- **AutoMeasureScale:** Automatically try to measure the scale as soon as a file with no scale is loaded. If set to false, the user will have to click the Measure scale button manually.
- **ScanFolder:** Searches all filenames in a folder to pass to the Web API. Set to false if you only want to retrieve filenames from the database.
- **AnnotateWithoutSample:** Lets the user make annotations on files with no linked sample.
- **OpenSocket:** Opens a socket that listens on 127.0.0.1 on port 11000 to incoming commands. Used at ILVO to communicate with other software. Set to false if SmartDots does not rely on other software.
- **EventAlias:** The name of the objects loaded in the 2<sup>nd</sup> screen with GetAnalysisDynamic Web API call. It will be displayed in the title bar.
- **SampleAlias:** An alternative name for 'Sample' will be show throughout the application. Leave empty to keep the default 'Sample'.
- **CanMarkEventAsCompleted:** When set to true, this will show the Finish/Reopen button in the 2<sup>nd</sup> screen.
- **AllowMultipleApprovements:** When set to true, this allows the user to approve multiple annotations in the Annotation panel.
- **IgnoreMultiUserColor:** When set to true, all dots will keep their original color when selecting multiple annotations.
- **MaturityAPI:** The API where all maturity related events are sent to.
- **MinRequiredVersion:** Used to check if the version is still compatible. Incompatible versions will prompt the user to update. Maturity events were released in version 3.0 of the software.

7

### 3.1.5. DTO LookupItem

```
// Class contains maturity related vocabulary. E.g Sex, Maturity, Readability
public class DtoMaturityLookupItem
{
    public Guid Id { get; set; }
    public string Code { get; set; }
    public string Description { get; set; }
}
```

Example data:

Id	Code	Description
0088a031-32df-4372-841e-0211209ce503	AQ1	Absolutely sure
0188a031-32df-4372-841e-0211209ce503	AQ2	Not sure
0288a031-32df-4372-841e-0211209ce503	AQ3	Unreadable

### 3.1.6. DTO MaturityAnalysis

```
// Class contains maturity analysis data
public class DtoAnalysis
{
    public Guid Id { get; set; }
    public string HeaderInfo { get; set; } // Used to display in the title-bar
    public List<DtoMaturitySample> MaturitySamples { get; set; } // The list of samples
}
```

### 3.1.7. DTO MaturitySample

```
// Class maturity sample data and child entities
public class DtoAnalysisParameter
{
    public Guid Id { get; set; }
    public string StatusCode { get; set; }
    public string StatusColor { get; set; }
    public int StatusRank { get; set; }
    public bool IsReadOnly { get; set; }
    public bool AllowApproveToggle { get; set; }
    public bool UserHasApproved { get; set; }

    public Dictionary<string,string> SampleProperties { get; set; } // dynamic sample properties
    public Dictionary<string,string> AnnotationProperties { get; set; } // dynamic annotation
properties
    public List<DtoMaturityFile> Files { get; set; } // The list of image files
    public List<DtoMaturityAnnotation> Annotations { get; set; } // The list of annotations
}
```

### 3.1.8. DTO MaturityFile

```
// Class contains file data, including related annotation and sample data
public class DtoFile
{
    public Guid Id { get; set; }
    public string Path { get; set; } // The full location of the image
    public string ThumbnailPath { get; set; } // The full location of the thumbnail image
    public bool IsReadOnly { get; set; }
    public decimal? Scale { get; set; } // Scale in pixels/mm
}
```

### 3.1.9. DTO MaturityAnnotation

```
// Class contains annotation data, including related line and dot data
public class DtoAnnotation
{
    public Guid Id { get; set; }
    public Guid MaturitySampleId { get; set; }
    public Guid UserId { get; set; }
    public string User { get; set; }
    public DateTime Date { get; set; }
    public Guid SexId { get; set; }
    public Guid MaturityId { get; set; }
    public Guid MaturityQualityId { get; set; }
    public bool IsApproved { get; set; }
    public string Comments { get; set; }
}
```



## 3.2. Dynamic objects

### 3.2.1. Analysis

In the Analysis Overview screen, a `List<dynamic>` is displayed. The institute can decide how these objects look. It should contain all relevant information about an analysis. It can be a mix of different kinds of events. For maturity events, the only requirement is that these objects have an ID and an EventType ( = "Maturity") property. For all other kinds of events, the EventType should also be set. EventTypes are ("AgeReading", "Maturity", ...). Please also make sure no property is a complex type as it's `ToString()` will not show any convenient information. The ID-property will not be displayed in the Grid.

#### 3.2.1.1. Example

```
var webApiResponse = new WebApiResponse();

var dtoAnalyses = new List<object>();

dynamic dtoAnalysis = new ExpandoObject();
dtoAnalysis.ID = Guid.NewGuid(); // Don't do it like this, instead get the real id of the
event/analysis from the database
dtoAnalysis.Number = 2;
dtoAnalysis.Name = "SRV01";
dtoAnalysis.EventType = "Maturity";

dtoAnalyses.Add(dtoAnalysis);


webApiResponse.Result = dtoAnalyses;

return Json(webApiResponse);
```

## 4. Methods

### 4.1. Screen 1: Authentication

In order to call Web API methods, the Web API url has to be set in the first screen. An example on a local machine looks like this: <http://localhost:63216/api/SmartDots/>. API urls can be the same for different kinds of events, or they could also be different.

The image shows a login interface for 'smartdots'. At the top is the 'smartdots' logo with a microscope icon. Below it is a 'Log in' header. There are two input fields: 'API URL' with the value 'http://srvsqldevd1:81/api/smartdots/' and 'Security' with the value 'Windows Authentication'. Both fields have dropdown arrows. At the bottom is a 'Connect' button.

This screen is used for Authentication only. A `DtoUserAuthentication` object is created with the user input and is then used to send to the Web API.

#### 4.1.1. GetGuestToken

##### 4.1.1.1. Definition

Returns token (string) for the user to login with

URL	/getguesttoken
Method	GET
URL Params	
Post Params	
Response	(JSON) WebApiResponse with a token (string) as the Result property

##### 4.1.1.2. ASP.NET implementation example

```
[HttpGet]
[Route("getguesttoken")]
public ActionResult GetGuestToken()
{
    //Implementation logic
}
```

## 4.1.2. Authenticate

### 4.1.2.1. Definition

Returns a DtoUser object if authentication is succeeded.

URL	/authenticate
Method	POST
URL Params	
Post Params	<a href="#">DtoUserAuthentication</a> userauthentication
Response	(JSON) WebApiResponse with a DtoUser-object as the Result property

### 4.1.2.2. ASP.NET implementation example

```
[HttpPost]
[Route("authenticate")]
public ActionResult Authenticate([FromBody] DtoUserAuthentication userauthentication)
{
    //Implementation logic
}
```

## 4.1.3. GetSettings

### 4.1.3.1. Definition

Returns a DtoSmartDotsSettings object which contains the institute depending settings. The DtoSmartDotsSettings objects is used in SmartDots to enable/disable certain features.

11

URL	/getsettings
Method	GET
URL Params	string token
Post Params	
Response	(JSON) WebApiResponse with a DtoSmartDotsSettings-object as the Result property

### 4.1.3.2. ASP.NET implementation example

```
[HttpGet]
[Route("getsettings")]
public ActionResult GetSettings(string token)
{
    //Implementation logic
}
```

## 4.2. Screen 2: Analyses

This screen displays a list of dynamic objects. It should contain all relevant information about an analysis. The institute can decide how these objects look. The only requirement is that these objects have an ID and a Folder

property. Please make sure no property is a complex type as its ToString() will not show any convenient information. The ID-property will not be displayed in the Grid.

smartdots Event overview							
Number	Service	Sample Set	Method	Sample State	Date Due	Folder	Status
20170001	Workshop - Scopthal...	TUR - 27.4.C	S&S	PHO	2/12/2017 12:00:00	http://193.190.112.136/...	TODO
20170002	Workshop - Pleuronec...	PLE - 27.7.F	WHL	PHO	2/12/2017 12:00:00	http://193.190.112.136/...	TODO
20170003	Training - Sardina pilc...	PIL - 37.1.1	WHL	PHO	2/12/2017 12:00:00	http://193.190.112.136/...	TODO
20170004	Training - Microstomu...	LEM - 27.4.C	WHL	PHO	2/12/2017 12:00:00	http://193.190.112.136/...	TODO
20170013	Workshop - Pagellus e...	PAC	WHL	PHO	29/12/2017 8:07:54	http://193.190.112.136/...	TODO
20170005	Workshop - Scopthal...	BLL - 27.4.C	S&S	PHO	2/12/2017 12:00:00	http://193.190.112.136/...	TODO
20170006	Workshop - Gadus mo...	COD - 27.7.G	S	PHO	11/12/2017 12:00:00	http://193.190.112.136/...	TODO
20170007	Exchange - Merlangius...	WHG - 27.4	WHL	PHO	11/12/2017 12:00:00	http://193.190.112.136/...	TODO
20170008	Training - Lepidorhom...	MEG - 27.7.G	WHL	PHO	11/12/2017 12:00:00	http://193.190.112.136/...	TODO
20170009	Workshop - Clupea ha...	HER - 27.3.A - 27.4.B	WHL	PHO	11/12/2017 12:00:00	http://193.190.112.136/...	TODO

## 4.2.1. GetAnalysesDynamic

### 4.2.1.1. Definition

Returns a List<dynamic>

URL	/getanalysesdynamic
Method	GET
URL Params	string token
Post Params	
Response	(JSON) WebApiResponse with a List<dynamic> as the Result property

### 4.2.1.2. ASP.NET implementation example:

```
[HttpGet]
[Route("getanalysesdynamic")]
public ActionResult GetAnalysesDynamic(string token)
{
    //Implementation logic
}
```

## 4.2.2. ToggleMaturityAnalysisUserProgress

### 4.2.2.1. Definition

Toggles the user's progress for the event/analysis

URL	/togglematurityanalysisuserprogress
Method	POST
URL Params	string token
Post Params	Guid analysisid

Response	(JSON) WebApiResponse with a Boolean if succeeded as the Result property
----------	--

#### 4.2.2.2. ASP.NET implementation example

```
[HttpPost]
[Route("togglematurityanalysisuserprogress")]
public ActionResult ToggleAnalysisUserProgress(string token, [FromBody] Guid analysisid)
{
    //Implementation logic
}
```

## 4.3. Screen 3: Maturity

### 4.3.1. GetMaturityAnalysis

#### 4.3.1.1. Definition

Returns a Single DtoMaturityAnalysis. The properties of this object are used in the Maturity screen.

URL	/getmaturityanalysis
Method	GET
URL Params	string token, Guid id
Post Params	
Response	(JSON) WebApiResponse with a DtoMaturityAnalysis as the Result property

#### 4.3.1.2. ASP.NET implementation example

```
[HttpGet]
[Route("getmaturityanalysis")]
public ActionResult GetAnalysis(string token, Guid id)
{
    //Implementation logic
}
```

**Note:** Please make sure the DtoMaturityAnalysis.MaturitySamples has all of these properties filled in: Id, StatusCode, StatusColor, StatusRank, UserHasApproved, SampleProperties. The other properties (IsReadOnly, AllowApproveToggle, AnnotationProperties, Files, Annotations) are only important in the *getmaturitysample* request.

### 4.3.2. GetMaturitySample

#### 4.3.2.1. Definition

Returns a DtoMaturitySample.

URL	/getmaturitysample
Method	GET
URL Params	string token, Guid id
Post Params	
Response	(JSON) WebApiResponse with a DtoMaturitySample as the Result property

#### 4.3.2.2. ASP.NET implementation example

```
[HttpGet]
[Route("getmaturitysample")]
public ActionResult GetMaturitySample(string token, Guid id)
{
    //Implementation logic
}
```

**Note:**

- All properties should have values.

### 4.3.3. GetVocab

#### 4.3.3.1. Definition

Returns a List<List<DtoLookupItem>> which contains the quality sex, maturity and quality lookups.

URL	/getvocab
Method	GET
URL Params	string token, string codeType
Post Params	
Response	(JSON) WebApiResponse with a List<DtoLookupItem> as the Result property

#### 4.3.3.2. ASP.NET implementation example

```
[HttpGet]
[Route("getvocab")]
public ActionResult GetVocab(string token, string codeType) // todo
{
    var webApiResponse = new WebApiResponse();
    try
    {
        using (var ctx = new D1_SmartDots_MaturityContext())
        {
            List<DtoLookupItem> maturityLookup = new List<DtoLookupItem>();

            var userid = Guid.Parse(token);

            // now fill the lookup with relevant data out of the database
            // for demo purposes I will use dummy objects

            switch (codeType)
            {
                case "sex":
                    maturityLookup.Add(new DtoMaturityLookupItem()
                    {
                        ID = Guid.Parse("0088a031-32df-4372-841e-0211209ce503"),
                        Code = "Male",
                        Description = "Male"
                    });
                    maturityLookup.Add(new DtoMaturityLookupItem()
                    {
                        ID = Guid.Parse("0188a031-32df-4372-841e-0211209ce503"),
                        Code = "Female",
                        Description = "Female"
                    });
                    maturityLookup.Add(new DtoMaturityLookupItem()
                    {
                        ID = Guid.Parse("0288a031-32df-4372-841e-0211209ce503"),
                        Code = "Undefined",
                        Description = "Undefined"
                    });
                    break;
                case "mat":
                    maturityLookup.Add(new DtoMaturityLookupItem()
                    {
                        ID = Guid.Parse("0088a031-32df-4372-841e-0211209ce503"),
                        Code = "Mature",
                        Description = "Mature"
                    });
            }
        }
    }
}
```

```

        maturityLookup.Add(new DtoMaturityLookupItem()
        {
            ID = Guid.Parse("0188a031-32df-4372-841e-0211209ce503"),
            Code = "Immature",
            Description = "Immature"
        });
        maturityLookup.Add(new DtoMaturityLookupItem()
        {
            ID = Guid.Parse("0288a031-32df-4372-841e-0211209ce503"),
            Code = "Juvenile",
            Description = "Juvenile"
        });
        break;
    case "aqm": // quality codes for maturity
        maturityLookup.Add(new DtoMaturityLookupItem()
        {
            ID = Guid.Parse("0088a031-32df-4372-841e-0211209ce503"),
            Code = "AQ1",
            Description = "Absolutely sure"
        });
        maturityLookup.Add(new DtoMaturityLookupItem()
        {
            ID = Guid.Parse("0188a031-32df-4372-841e-0211209ce503"),
            Code = "AQ2",
            Description = "Not sure"
        });
        maturityLookup.Add(new DtoMaturityLookupItem()
        {
            ID = Guid.Parse("0288a031-32df-4372-841e-0211209ce503"),
            Code = "AQ3",
            Description = "Unreadable"
        });
        break;
    default:
        break;
}

webApiResult.Result = maturityLookup;

return Json(webApiResult);
}
}
catch (Exception e)
{
    webApiResult.ErrorMessage = e.Message;
    return Json(webApiResult);
}
}

```

**Note:**

- Supported codetypes need to be: “sex”, “mat” (maturity vocab) and “aqm” (maturity qualities vocab).

## 4.3.4. UpdateMaturityFile

### 4.3.4.1. Definition

Updates a file, only used for saving the scale

URL	/updatematurityfile
-----	---------------------



Method	POST
URL Params	string token
Post Params	DtoFile file
Response	(JSON) WebApiResponse with a Boolean if succeeded as the Result property

#### 4.3.4.2. ASP.NET implementation example

```
[HttpPost]
[Route("updatematurityfile")]
//Updates a file, returns true if succeeded
public ActionResult UpdateMaturityFile(string token, [FromBody] DtoMaturityFile file)
{
    var webApiResponse = new WebApiResponse();
    try
    {
        using (var ctx = new D1_SmartDots_MaturityContext())
        {
            var dbfile = ctx.MaturityFile.FirstOrDefault(x => x.Id == file.ID);
            dbfile.Scale = file.Scale;
            ctx.SaveChanges();

            webApiResponse.Result = true;
            return Json(webApiResponse);
        }
    }
    catch (Exception e)
    {
        webApiResponse.ErrorMessage = e.Message;
        return Json(webApiResponse);
    }
}
```

17

#### Note:

- This is only used to update the scale.

### 4.3.5. SaveMaturityAnnotation

#### 4.3.5.1. Definition

Adds an annotation to the database if it does not exist. Otherwise updates the annotation. Returns true if succeeded.

URL	/savematurityannotation
Method	POST
URL Params	string token
Post Params	DtoMaturityAnnotation annotation
Response	(JSON) WebApiResponse with a DtoMaturitySample as the Result property

#### 4.3.5.2. ASP.NET implementation example

```
[HttpPost]
[Route("savematurityannotation")]
public ActionResult SaveMaturityAnnotation(string token, [FromBody] DtoMaturityAnnotation annotation)
```

```
{  
    //Implementation logic  
}
```

**Note:**

- Please return a DtoMaturitySample and fill in the following properties: StatusCode, StatusRank, StatusColor, IsReadOnly, AllowApproveToggle, UserHasApproved and Annotations.
- You can decide what annotations the user can see. Do you only return all annotations of the user is ready with the analysis/event or do you always show them?

## 5. Swagger

---

You can try out the demo API on <https://webapi.smartfisheries.be/swagger/>.

Upon using the API you will notice that some GUID's/UUID's are shared between users and the different vocabs. But they are unique within their own list. If you request the next url, you will notice the returning id's:

<https://webapi.smartfisheries.be/api/demo/getmaturitylookups?token=0188a031-32df-4372-841e-0211209ce503>

For example: the sex "Male" has the same id as the maturity "Mature". This is because it is just demo data and some recycling was the fastest way to implement this.

Using swagger, you can follow the app flow. It is best to start with the authenticate request. For the demo api, we use the UserCredentials "Basic" authentication method. The Enum value for "Basic" is 1.

Example POST-data:

```
{
  "username": "user1",
  "password": "pwd1",
  "dtoAuthenticationMethod": 1
}
```

The return data contains a token. Use this token for every request that requires it.