



Vlaanderen
is landbouw & visserij

TECHNICAL DOCUMENTATION

Larvae & Egg Web API

smartdots

ILVO

Institute for Agricultural
and Fisheries Research

www.ilvo.vlaanderen.be

<i>Date</i>	<i>Version</i>
2023/02	Version 1.0

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	11
4. Methods.....	12
4.1. Screen 1: Authentication.....	12
4.2. Screen 2: Analyses	14
4.3. Screen 3: Larvae.....	16
5. Swagger.....	20

1. Introduction

SmartDots is a software application for performing age determination and maturity analysis of fish. Recently a module has been added to perform analysis on fish larvae and eggs.

To analyse larvae and eggs end-users study images on which they have to decide and draw a few parameters. The user has some tools available to make measurements on the image. In addition an annotation can have a quality indication and some comment.

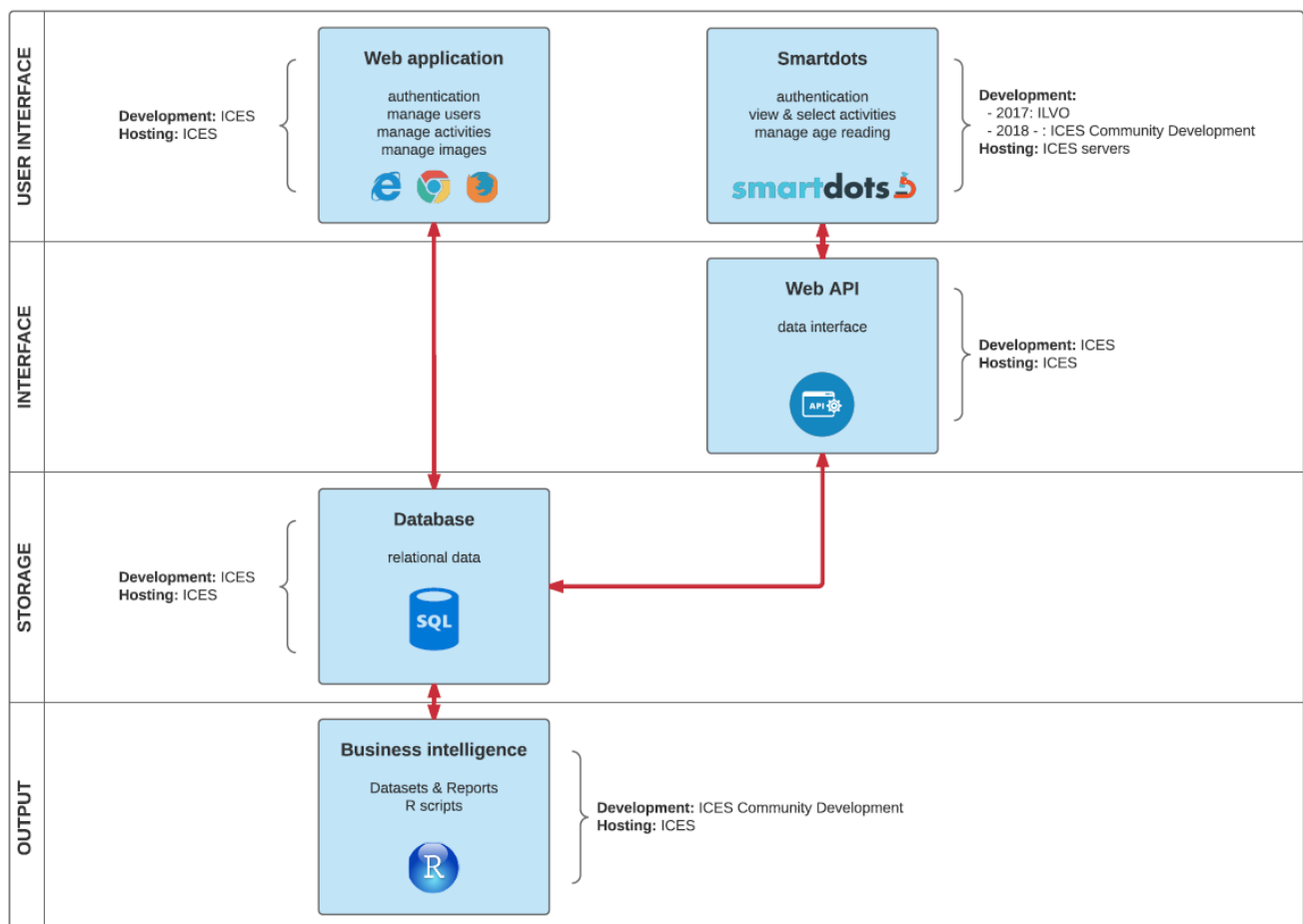
The SmartDots user interface exists out of 3 main screens:

- 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, Larvae or Egg

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_SmartDotsContext())
        {
            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 string LarvaeEggAPI { 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 is SmartDots does not rely on other software.
- **EventAlias:** The name of the objects loaded in the 2nd 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 2nd 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.
- **LarvaeEggAPI:** 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 vocabulary data. E.g LarvaeQuality or LarvaePresence
public class DtoLookupItem
{
    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 LarvaeEggAnalysis

```
// Class contains larvae or egg analysis data
public class DtoAnalysis
{
    public Guid Id { get; set; }
    public string HeaderInfo { get; set; } // Used to display in the title-bar
    public string Type { get; set; } // Either "Larvae" or "Egg"
    public string AllowSetScale { get; set; } // If the logged in user can set a scale
    public List<DtoLarvaeEggSample> LarvaeEggSamples { get; set; } // The list of samples
    public List<DtoLarvaeEggParameter> LarvaeEggParameters { get; set; } // The list of parameters
}
```

3.1.7. DTO LarvaeEggSample

```
// Class containing larvae or egg sample data and child entities
public class DtoLarvaeEggSample
{
    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
    properties
    public List<DtoLarvaeEggFile> Files { get; set; } // The list of image files
    public List<DtoLarvaeEggAnnotation> Annotations { get; set; } // The list of annotations
}
```

3.1.8. DTO LarvaeEggFile

```
// Class contains file data, including related annotation and sample data
public class DtoLarvaeEggFile
{
    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 string Name { get; set; } // The name of the image
    public bool IsReadOnly { get; set; }
    public decimal? Scale { get; set; } // Scale in pixels/mm
}
```

3.1.9. DTO LarvaeEggAnnotation

```
// Class contains annotation data, including related parameter result data
public class DtoLarvaeEggAnnotation
{
    public Guid Id { get; set; }
    public Guid LarvaeEggSampleId { get; set; }
    public Guid UserId { get; set; }
    public string User { get; set; }
    public DateTime Date { get; set; }
    public Guid? SpeciesId { get; set; }
    public Guid? QualityId { get; set; }
    public Guid? AnalFinPresenceId { get; set; } // Larvae only
    public Guid? PelvicFinPresenceId { get; set; } // Larvae only
    public Guid? DorsalFinPresenceId { get; set; } // Larvae only
    public Guid? DevelopmentStageId { get; set; }
    public Guid? EmbryoPresenceId { get; set; } // Egg only
}
```



```

    public Guid? EmbryoSizeId { get; set; } // Egg only
    public Guid? YolkSegmentationId { get; set; } // Egg only
    public Guid? OilGlobulePresenceId { get; set; } // Egg only
    public bool IsApproved { get; set; }
    public string Comments { get; set; }
    public List<DtoLarvaeEggAnnotationParameterResult> AnnotationParameterResult { get; set; } //
parameters that need to be drawn on the images
}

```

3.1.10. DTO LarvaeEggAnnotationParameterResult

```

// Class contains annotation parameter result data
public class DtoLarvaeEggAnnotationParameterResult
{
    public Guid Id { get; set; }
    public Guid AnnotationId { get; set; }
    public Guid ParameterId { get; set; }
    public Guid FileId { get; set; }
    public int Result { get; set; } // Can either be a length in pixels, diameter in pixels or
number of dots
    public List<DtoLarvaeEggDot> Dots { get; set; } // Larvae only
    public List<DtoLarvaeEggDot> Lines { get; set; } // Larvae only
    public DtoLarvaeEggCircle Circle { get; set; } // Egg only
}

```

3.1.11. DTO LarvaeEggParameter

```

// Class contains annotation parameter data
public class DtoLarvaeEggParameter
{
    public Guid Id { get; set; }
    public string Name { get; set; }
    public string Color { get; set; } // Hexcode
    public string ShapeType { get; set; } // "Circle" for egg, "Line" or "Dot" for larvae
}

```

9

3.1.12. DTO LarvaeEggDot

```

// Class containing dot data
public class DtoLarvaeEggDot
{
    public Guid Id { get; set; }
    public Guid AnnotationParameterResultId { get; set; }
    public int X { get; set; } // horizontal pixel location of the dot
    public int Y { get; set; } // vertical pixel location of the dot
    public int Width { get; set; } // size of the dot
}

```

3.1.13. DTO LarvaeEggLine

```

// Class containing line data
public class DtoLarvaeEggLine
{
    public Guid Id { get; set; }
    public Guid AnnotationParameterResultId { get; set; }
    public int X1 { get; set; } // horizontal pixel location of the start point
    public int Y1 { get; set; } // vertical pixel location of the start point
    public int X2 { get; set; } // horizontal pixel location of the end point
    public int Y2 { get; set; } // vertical pixel location of the end point
    public int Width { get; set; } // width of the line
}

```

```
    public int? LineIndex { get; set; } // order of the line segment  
}
```

3.1.14. DTO LarvaeEggCircle

```
// Class containing circle data  
public class DtoLarvaeEggCircle  
{  
    public Guid Id { get; set; }  
    public Guid AnnotationParameterResultId { get; set; }  
    public int X1 { get; set; } // horizontal pixel location of the start point  
    public int Y1 { get; set; } // vertical pixel location of the start point  
    public int X2 { get; set; } // horizontal pixel location of the end point  
    public int Y2 { get; set; } // vertical pixel location of the end point  
    public int Width { get; set; } // width of the border  
}
```

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 larvae and egg events, the only requirement is that these objects have an ID and an `EventType` (= "Larvae" or "Egg") property. For all other kinds of events, the `EventType` should also be set. EventTypes are ("AgeReading", "Maturity", "Larvae", "Egg"). Please also 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.

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 = "Larvae";

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.



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	DtoUserAuthentication 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.

13

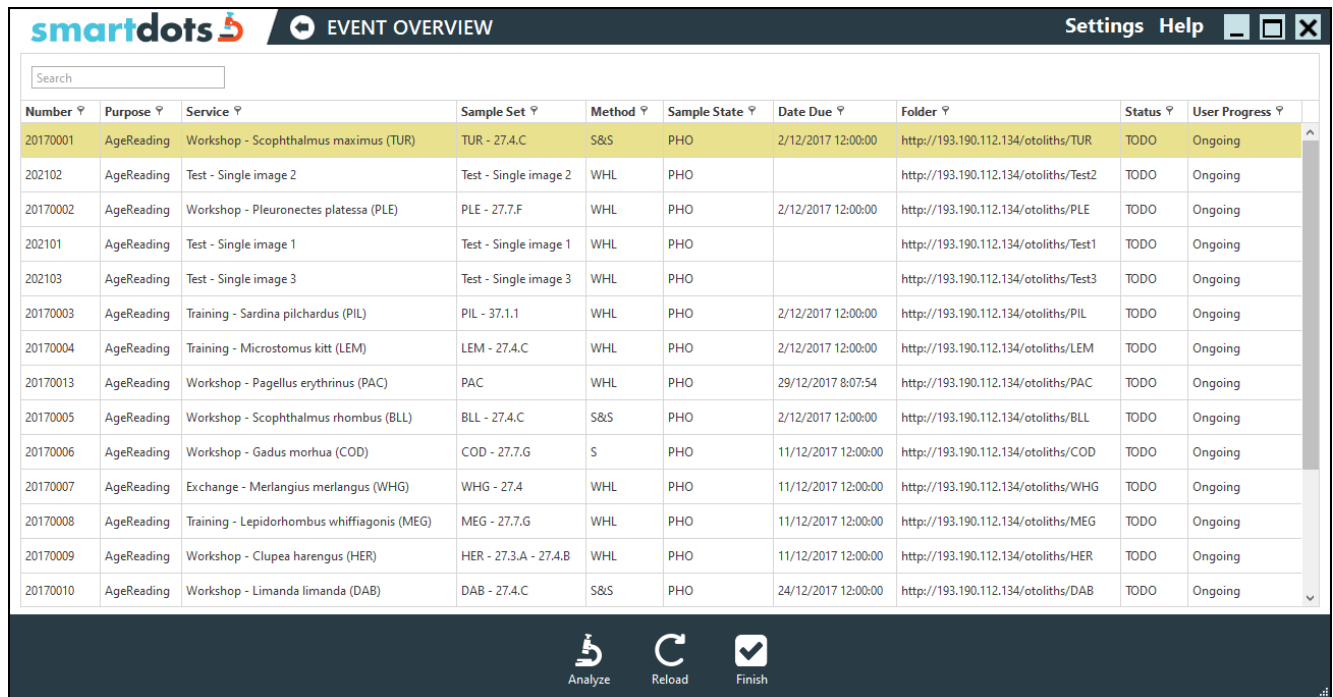
URL	/getsettings
Method	GET
URL Params	string token
Post Params	
Response	(JSON) WebApiResponse with a DtoSmartDotsSettings-object as the Result property

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.



The screenshot shows the 'smartdots' application interface with a dark header bar containing 'EVENT OVERVIEW', 'Settings', and 'Help'. Below the header is a search bar and a table with 10 columns: Number, Purpose, Service, Sample Set, Method, Sample State, Date Due, Folder, Status, and User Progress. The table lists 16 analysis entries, each with a unique ID and various details. At the bottom of the screen are three icons: 'Analyze' (microscope), 'Reload' (refresh), and 'Finish' (checkmark).

Number	Purpose	Service	Sample Set	Method	Sample State	Date Due	Folder	Status	User Progress
20170001	AgeReading	Workshop - Scophthalmus maximus (TUR)	TUR - 27.4.C	S&S	PHO	2/12/2017 12:00:00	http://193.190.112.134/otoliths/TUR	TODO	Ongoing
202102	AgeReading	Test - Single image 2	Test - Single image 2	WHL	PHO		http://193.190.112.134/otoliths/Test2	TODO	Ongoing
20170002	AgeReading	Workshop - Pleuronectes platessa (PLE)	PLE - 27.7.F	WHL	PHO	2/12/2017 12:00:00	http://193.190.112.134/otoliths/PLE	TODO	Ongoing
202101	AgeReading	Test - Single image 1	Test - Single image 1	WHL	PHO		http://193.190.112.134/otoliths/Test1	TODO	Ongoing
202103	AgeReading	Test - Single image 3	Test - Single image 3	WHL	PHO		http://193.190.112.134/otoliths/Test3	TODO	Ongoing
20170003	AgeReading	Training - Sardina pilchardus (PIL)	PIL - 37.1.1	WHL	PHO	2/12/2017 12:00:00	http://193.190.112.134/otoliths/PIL	TODO	Ongoing
20170004	AgeReading	Training - Microstomus kitt (LEM)	LEM - 27.4.C	WHL	PHO	2/12/2017 12:00:00	http://193.190.112.134/otoliths/LEM	TODO	Ongoing
20170013	AgeReading	Workshop - Pagellus erythrinus (PAC)	PAC	WHL	PHO	29/12/2017 8:07:54	http://193.190.112.134/otoliths/PAC	TODO	Ongoing
20170005	AgeReading	Workshop - Scophthalmus rhombus (BLL)	BLL - 27.4.C	S&S	PHO	2/12/2017 12:00:00	http://193.190.112.134/otoliths/BLL	TODO	Ongoing
20170006	AgeReading	Workshop - Gadus morhua (COD)	COD - 27.7.G	S	PHO	11/12/2017 12:00:00	http://193.190.112.134/otoliths/COD	TODO	Ongoing
20170007	AgeReading	Exchange - Merlangius merlangus (WHG)	WHG - 27.4	WHL	PHO	11/12/2017 12:00:00	http://193.190.112.134/otoliths/WHG	TODO	Ongoing
20170008	AgeReading	Training - Lepidorhombus whiffiagonis (MEG)	MEG - 27.7.G	WHL	PHO	11/12/2017 12:00:00	http://193.190.112.134/otoliths/MEG	TODO	Ongoing
20170009	AgeReading	Workshop - Clupea harengus (HER)	HER - 27.3.A - 27.4.B	WHL	PHO	11/12/2017 12:00:00	http://193.190.112.134/otoliths/HER	TODO	Ongoing
20170010	AgeReading	Workshop - Limanda limanda (DAB)	DAB - 27.4.C	S&S	PHO	24/12/2017 12:00:00	http://193.190.112.134/otoliths/DAB	TODO	Ongoing

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. ToggleLarvaeEggAnalysisUserProgress

4.2.2.1. Definition

Toggles the user's progress for the event/analysis

URL	/togglelarvaeegganalysisuserprogress
Method	POST
URL Params	string token, string type (= "Larvae" or "Egg")
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("togglelarvaeegganalysisuserprogress")]
public ActionResult ToggleAnalysisUserProgress(string token, string type, [FromBody] Guid analysisid)
{
    //Implementation logic
}
```

4.3. Screen 3: Larvae or Egg

4.3.1. GetLarvaeEggAnalysis

4.3.1.1. Definition

Returns a Single DtoLarvaeEggAnalysis. The properties of this object are used in the Larvae or egg screen.

URL	/getlarvaeegganalysis
Method	GET
URL Params	string token, string type (= "Larvae" or "Egg"), Guid id
Post Params	
Response	(JSON) WebApiResponse with a DtoLarvaeEggAnalysis as the Result property

4.3.1.2. ASP.NET implementation example

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

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

4.3.2. GetLarvaeEggSample

4.3.2.1. Definition

Returns a DtoLarvaeEggSample.

URL	/getlarvaeeggssample
Method	GET
URL Params	string token, string type (= "Larvae" or "Egg"), Guid id
Post Params	
Response	(JSON) WebApiResponse with a DtoLarvaeEggSample as the Result property

4.3.2.2. ASP.NET implementation example

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

Note:

- All properties should have values.

4.3.3. GetVocab

4.3.3.1. Definition

Returns a List<DtoLookupItem> which contains the various lookups containing larvae and egg parameter values.

URL	/getvocab
Method	GET
URL Params	string token, Guid analysisid, 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, Guid analysisid, string codeType)
{
    var webApiResponse = new WebApiResponse();
    try
    {
        using (var ctx = new D1_SmartDotsContext())
        {
            List<DtoLookupItem> larvaeLookup = new LookupItem();

            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 "laq": // quality codes for larvae events
                    larvaeLookup.Add(new DtoLookupItem()
                    {
                        ID = Guid.Parse("0088a031-32df-4372-841e-0211209ce503"),
                        Code = "AQ1",
                        Description = "Absolutely sure"
                    });
                    larvaeLookup.Add(new DtoLookupItem()
                    {
                        ID = Guid.Parse("0188a031-32df-4372-841e-0211209ce503"),
                        Code = "AQ2",
                        Description = "Not sure"
                    });
                    larvaeLookup.Add(new DtoLookupItem()
                    {
                        ID = Guid.Parse("0288a031-32df-4372-841e-0211209ce503"),
                        Code = "AQ3",
                        Description = "Unreadable"
                    });
                    break;
                default:
                    break;
            }

            webApiResponse.Result = larvaeLookup;
        }
    }
}
```

```

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

```

Note:

- Supported codetypes need to be:
 - “laq” (Larvae Quality codes)
 - “eaq” (Egg Quality codes)
 - “lpr” (Larvae Presence codes)
 - “epr” (Egg Presence codes)
 - “lds” (Larvae Development Stage codes)
 - “eds” (Egg Development Stage codes)
 - “lsp” (Larvae Species codes)
 - “esp” (Egg Species codes)
 - “eys” (Egg Yolk Segmentation codes)
 - “ees” (Egg Embryo Size codes)

4.3.4. UpdateLarvaeEggFile

4.3.4.1. Definition

Updates a file, only used for saving the scale

URL	/updatelarvaeeggfile
Method	POST
URL Params	string token, string type (= “Larvae” or “Egg”)
Post Params	DtoLarvaeEggFile file
Response	(JSON) WebApiResult with a Boolean if succeeded as the Result property

4.3.4.2. ASP.NET implementation example

```

[HttpPost]
[Route("updatelarvaeeggfile")]
//Updates a file, returns true if succeeded
public ActionResult UpdateLarvaeEggFile(string token, [FromBody] DtoLarvaeEggFile file)
{
    var webApiResult = new WebApiResult();
    try
    {
        using (var ctx = new D1_SmartDotsContext())
        {
            var dbfile = ctx.LarvaeEggFile.FirstOrDefault(x => x.Id == file.ID);
            dbfile.Scale = file.Scale;
            ctx.SaveChanges();

            webApiResult.Result = true;
            return Json(webApiResult);
        }
    }
    catch (Exception e)
    {
    }
}

```

```

    {
        webApiResponse.ErrorMessage = e.Message;
        return Json(webApiResponse);
    }
}

```

Note:

- This is only used to update the scale.

4.3.5. SaveLarvaeEggAnnotation

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	/savelarvaeeggannotation
Method	POST
URL Params	string token, string type (= "Larvae" or "Egg")
Post Params	DtoLarvaeEggAnnotation annotation
Response	(JSON) WebApiResponse with a DtoLarvaeEggSample as the Result property

4.3.5.2. ASP.NET implementation example

```

[HttpPost]
[Route("savelarvaeeggannotation")]
public ActionResult SaveLarvaeEggAnnotation(string token, string type, [FromBody]
DtoLarvaeEggAnnotation annotation)
{
    //Implementation logic
}

```

Note:

- Please return a DtoLarvaeEggSample 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. 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.