INNOVATIVE BLOCKCHAIN TRACEABILITY TECHNOLOGY AND STAKEHOLDERS' ENGAGEMENT STRATEGY FOR BOOSTING SUSTAINABLE SEAFOOD VISIBILITY, SOCIAL ACCEPTANCE AND CONSUMPTION IN EUROPE

DELIVERABLE 3.2 – Aquaculture production software integration report

Lead Partner Organization	SMARTWATER PLANET
Due date	31-Dec-24
Issue date	24-Dec-24

Co-funded by the European Union

Document information

Settings	Value
Deliverable Title	Aquaculture production cloud integrated with Sea2See Blockchain
Work Package Number & Title	WP3 - Traceability technologies development
Deliverable number	D3.2
Description	Report on the integration of the aquaculture cloud platform into the Sea2See blockchain structure, their deployment and analysis.
Lead Beneficiary	SmartWater
Lead Authors	Gonzalo Pérez Vizuete
Contributors	Gonzalo Pérez Vizuete, Carlos Mazorra de Quero
Submitted by	Carlos Mazorra

Review History

Version	Date	Reviewer	Short Description of Changes
1	3-Dec-24	Arnaud THEVENARD	Minor comments
2	3-Dec-24	Adeline CAFFIN	Minor comments

Document Approval

Name	Role	Action	Date
Carlos Mazorra	Project Coordinator	*Approved*	24-Dec-24

Nature of the deliverable

R	Document, report (excluding the periodic and final reports)	
DEM	Demonstrator, pilot, prototype, plan designs	☐
DEC	Websites, patents filing, press & media actions, videos, etc.	☐
DATA	Data sets, microdata, etc.	☐
DMP	Data management plan	☐
Ethics	Deliverables related to ethics issues.	☐
SECURITY	Deliverables related to security issues	☐
Other	Software, technical diagram, algorithms, models, etc.	☑

Dissemination level

PU	Public — fully open (automatically posted online on the Project Results platforms)	☑
SEN	Sensitive — limited under the conditions of the Grant Agreement	

ACKNOWLEDGEMENT

This report forms part of the deliverables from the project Sea2See which has received funding from the European Union's Horizon Europe Research and Innovation Programme under grant agreement No. 101060564.

Current seafood traceability tools and services have the potential to take advantage of novel blockchain technologies to obtain a wide range of data making sustainable seafood practices more visible to consumers. Sea2See project will fill in existing seafood traceability gaps through development and demonstration of an innovative end-to-end blockchain traceability model throughout the seafood value chain and professional and consumer applications to increase trust and social acceptance of sustainably fished and farmed seafood.

The project will provide technological solutions to answer the need of a valuable source of data collected throughout the whole seafood value chain, verified, and covering inputs from diverse stakeholders. For that purpose, a specific focus will be put on active commitment of stakeholders and real empowerment of consumers through the implementation of societal and sectoral strategies for co-creation, communication and awareness raising.

The project runs from July 2022 to June 2026. It involves 14 partners from 6 EU countries, and is coordinated by SMARTWATER PLANET SL, Spain.

More information about the project can be found at: http://www.sea2see.eu/

COPYRIGHT

EXECUTIVE SUMMARY

The current document describes the integration process of SMARTWATER CLOUD (Aquaculture production management software) with the SEA2SEE Traceability platform. This report includes descriptions of the system, implementation, deployment and a final analysis of the use of the new implementations.

ACRONYMS AND ABBREVIATIONS

ACRONYM	DEFINITION
S2S	Sea 2 See
API	Application Programming Interface
CLOUD	Abbreviation for SMARTWATER CLOUD
EPCIS	Electronic Product Code Information Services

PROJECT PARTNERS

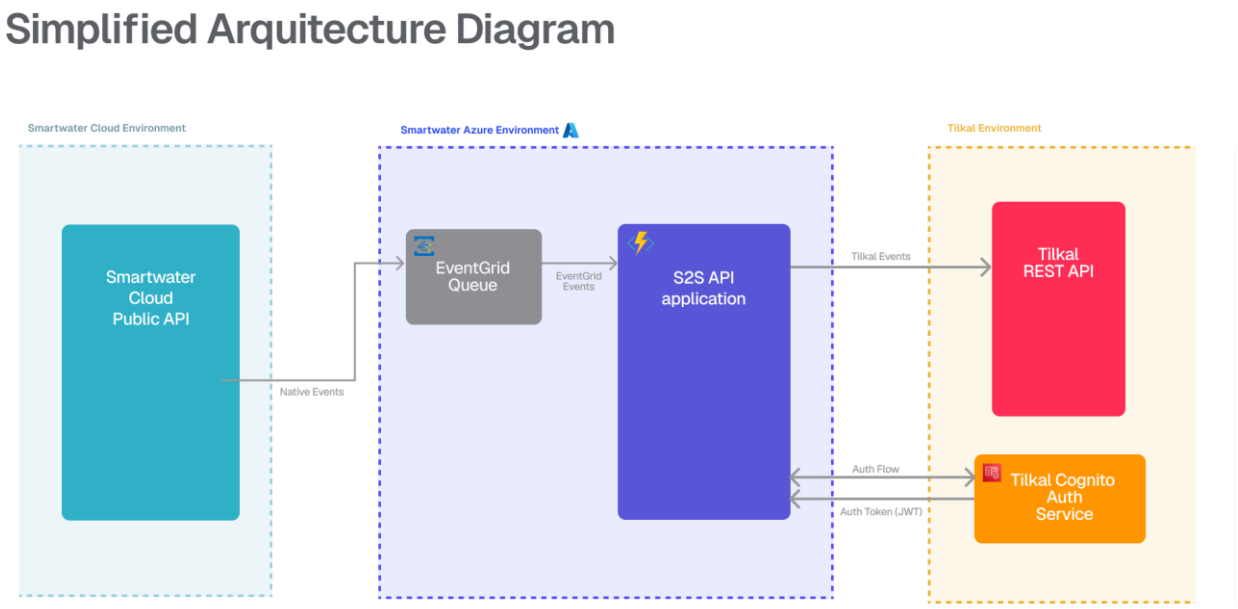
#	Partners full name	Short	Country	Website
1	SMARTWATER PLANET SL	SmartWater	ES	www.smartwaterplanet.com
2	TILKAL	Tilkal	FR	www.tilkal.com
3	PAGE UP	PAGE UP	FR	www.pageup.fr
4	SUBMON	SUBMON	ES	www.submon.org
5	CENTRO DE CIENCIAS DO MAR DO ALGARVE	CCMAR	PT	www.ccmar.ualg.pt
6	ASOCIACION NACIONAL DE FABRICANTES DE CONSERVAS DE PESCADOS Y MARISCOS-CENTRO TECNICO NACIONAL DE CONSERVACION DE PRODUCTOS DE LA PESCA	ANFACO	ES	www.anfaco.es
7	IOANNA N.ARGYROU SIMBOULOI EPICHEIR ISIAKIS ANAPTYXIS ETAIREIA PERIORISMENIS EYTHYNIS	NAYS	EL	www.nays.gr
8	SEAENTIA-FOOD, LDA	SEAentia	PT	www.seaentia.com
9	LANDLNG AQUACULTURE BV	LA	NL	www.landingaquaculture.com
10	UNIVERSIDADE DE AVEIRO	UAVR	PT	www.ua.pt
11	VITAGORA POLE	VITAGORA	FR	www.vitagora.com
12	ETHIC OCEAN	Ethic Ocean	FR	www.ethic-ocean.org
13	EVROPROJECT OOD	EP	BG	www.europroject.bg
14	ANP - ASSOCIACAO NATUREZA PORTUGAL	ANP	PT	www.natureza-portugal.org

TABLE OF CONTENTS

ACKNOWLEDGEMENT 3
COPYRIGHT 3
EXECUTIVE SUMMARY 4
ACRONYMS AND Abbreviations 4
Project partners 4
1. Description of the artifacts 6
1.1. Smartwater cloud 6
1.2. SEA2See API APPLICATION 6
Azure functions 7
EventGrid 7
1.3. Tilkal platform 8
2. Description of the implementation 8
2.1. Sending events from smartwater cloud 8
2.2. Event Ingestion from Smartwater Cloud 8
2.3. Message Transformation 8
Master data 8
Biz transaction 9
Event 9
Native Smartwater Cloud Event Types 10
Sending Messages to TILKAL platform 12
2.4. DEFINITION OF THE FUNCTION 13
cloud handler function 14
3. User interface 17
3.1. BATCH creation 17
3.2. Batch events 19
Average weight biometry 19
counting 20
STOCK loss (DEATH) 21
Classification and transfers 22
3.3. SALE of the batch 23

3.4. Visualizing the traceability in spotlight 24
3. Execution analytics 25

1. DESCRIPTION OF THE ARTIFACTS



Simplified overall architecture diagram

1.1. SMARTWATER CLOUD

SMARTWATER CLOUD is an aquaculture management software. This software has all the necessary tools and processes integrated for successfully managing all the use cases inside every aquaculture producer.

1.2. SEA2SEE API APPLICATION

The S2S API is a server application (backend) responsible for translating and routing events generated by the Smartwater Cloud application for consumption by the Tilkal platform, facilitating integration into the value chain. This functionality is implemented as a serverless application on Microsoft Azure using Azure Functions and Event Grid.

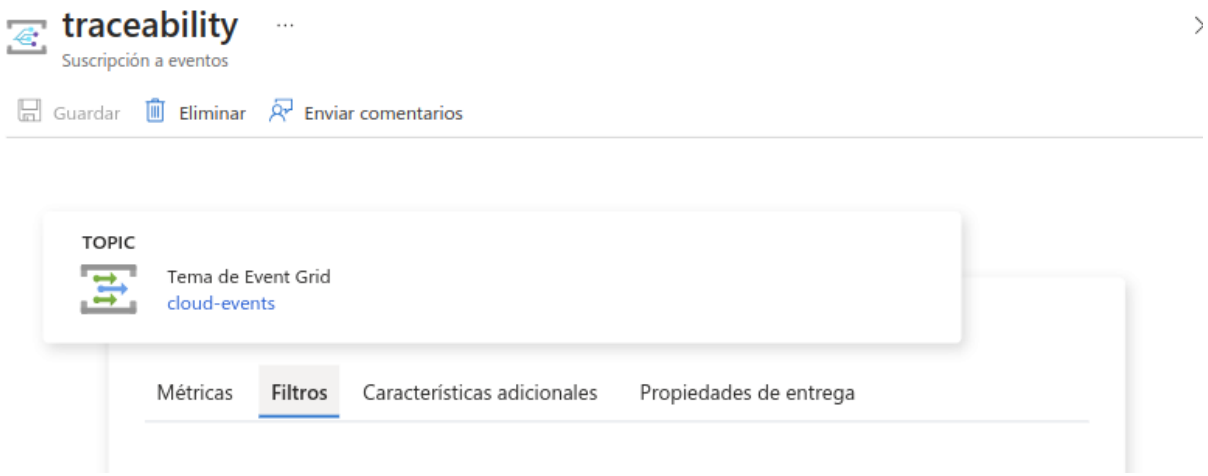
AZURE FUNCTIONS

Azure Functions enable developers to build and deploy scalable and event-driven solutions without the need to manage infrastructure. In the context of the S2S API, this means that the backend services are hosted in a cloud environment that automatically scales based on demand, providing a highly reliable and cost-effective solution. Azure Functions are used for executing code in response to various events. They provide a flexible way to integrate with other Azure services and external systems, allowing for quick response to incoming messages, which is crucial for handling events from Smartwater Cloud efficiently. With the event Sea2See Event API (API S2S) driven model, Azure Functions only execute when triggered, thus minimizing resource usage and costs.

EVENTGRID

Azure Event Grid is a fully managed event routing service that allows developers to build event-based architectures by connecting different services through event notifications. In this implementation, Event Grid acts as the messaging backbone that receives and routes events from Smartwater Cloud to the S2S API. Event Grid is designed for high availability and guarantees message delivery, making it an ideal choice for the real-time processing requirements of Sea2See's event ingestion system.

Event Grid supports various event handlers and sources, which makes it highly suitable for integrating complex workflows across cloud services. The use of Event Grid ensures that all relevant events from Smartwater Cloud are reliably delivered to the S2S API for further processing.



EventGrid subscription to cloud-event topic in Microsoft Azure

1.3. TILKAL PLATFORM

Tilkal platform is the backbone platform used to handle the traceability in blockchain. It exposes a well-formed and documented interface so we can handle traceability events using their API while tilkal abstracts all the blockchain management from 3rd parties that integrates with them.

2. DESCRIPTION OF THE IMPLEMENTATION

2.1. SENDING EVENTS FROM SMARTWATER CLOUD

A proprietary library has been deployed inside SMARTWATER CLOUD codebase for handling the events generated by the application and sending them to the EventGrid topic where the S2S API is listening to. This library contains a set of classes to manage each type of event that might be produced by the application and sent to the S2S API and all the necessary methods to push the events to EventGrid.

We have also included changes into SMARTWATER CLOUD's databases because we need to manage new information regarding traceability management. Those changes include new fields into existing tables.

2.2. EVENT INGESTION FROM SMARTWATER CLOUD

Whenever a new event is created in the Smartwater Cloud software for batches with Sea2See traceability enabled, it is sent to an Event Grid message queue hosted in Azure for consumption by the S2S API.

2.3. MESSAGE TRANSFORMATION

Once the messages are consumed from Event Grid by our application, they are translated from the native Smartwater Cloud format to the message type expected by the Tilkal platform, depending on their category. The Tilkal message types are categorized into three distinct types: MasterData, Biztransaction, and Event.

MASTER DATA

Represents master data related to locations or elements.

Properties:

- kind (str): Type of master data (e.g., "MasterData").

- vocabulary (str): Type of associated vocabulary
- id (str): Unique identifier.
- attributes (dict): Additional details in key-value format.
- locale (str): Regional settings (e.g., "en-US")
- bizLocation (list, optional): List of related locations.

BIZ TRANSACTION

Events representing business transactions.

Properties:

- kind (str): Type of event (e.g., "Biztransaction").
- eventTime (str): Date and time of the event.
- id (str): Unique transaction identifier.
- type (str): Type of transaction.
- bizLocation (str): Location of the related business.
- activityList (list): List of related activities.
- sourceList (list): Event sources.
- destinationList (list): Event destinations
- quantityList (list): Details of quantities involved.

EVENT

Generic events representing observed actions, transformations, or other processes.

Properties:

- kind (str): Type of event (e.g., "Event").
- eventType (str): Category of the event (e.g., "ObjectEvent", "TransformationEvent").
- eventTime (str): Date and time of the event.

- eventTimeZoneOffset (str): Time zone of the event.
- action (str): Action performed (e.g., "OBSERVE").
- bizLocation (str): Location of the event.
- bizStep (list): Associated business step.
- disposition (str): State or disposition of the event.
- sourceList (list): List of related sources.
- destinationList (list): List of related destinations.
- quantityList (list): Associated quantities.
- extensions (dict, optional): Extended information such as comments or additional data.
- InputQuantityList (list, optional): Input quantities for transformation events.
- outputQuantityList (list): Output quantities for transformation events.

NATIVE SMARTWATER CLOUD EVENT TYPES

Native Smartwater Cloud events are events generated within the Smartwater Cloud application and sent to the SEA2SEE API. Due to the modular design of the SEA2SEE API and its translation layers, we can implement specific translation algorithms for transforming these native events into traceability events.

The event types currently managed are the following:

1. **sea2see.company**

This event contains information about companies or stakeholders. It is translated into a Tilkal MasterData event.

2. **sea2see.productionCenter**

This event contains information about the production centers of stakeholders. It is translated into a Tilkal MasterData event.

3. **sea2see.productionUnit**

This event contains information about production units associated with stakeholders' production centers. It is translated into a Tilkal MasterData event.

4. sea2see.expense

This event represents a purchase activity in S2S and is translated into a Tilkal Biztransaction event.

5. sea2see.lot

This event encompasses three Tilkal events, two mandatory and one optional.

If the batch originates from a supplier purchase, a Tilkal transformation event of type "kind": "Event", "EventType": "TransformationEvent" is generated to represent the transformation of raw material into the cultivated species.

If no purchase is associated, it implies that this event is from a transfer, in which case the Tilkal transformation event is unnecessary.

Additionally, an event of type "kind": "Event", "eventType": "ObjectEvent" is generated, describing a shipping operation.

Lastly, an event of type "kind": "Event", "eventType": "ObjectEvent" is generated, describing a lot receipt operation at the production unit.

6. sea2see.biometry

This event represents a batch biometrics (weighing) operation at the production unit. It is translated into a Tilkal event of type "kind": "Event", "eventType": "ObjectEvent" describing a weighing operation.

7. sea2see.counting

This event represents a lot counting operation at the production unit. It is translated into a Tilkal event of type "kind": "Event", "eventType": "ObjectEvent" describing a counting operation.

8. sea2see.death

This event represents deaths within the lot at the production unit. It is translated into a Tilkal event of type "kind": "Event", "eventType": "ObjectEvent" describing a death event.

9. sea2see.transfer

This event represents the transfer of a lot between production units (total or partial) and generates three events for the Tilkal platform.

This event encompasses three Tilkal events.

First, a Tilkal transformation event of type "kind": "Event", "EventType": "TransformationEvent" is generated to specify the transfer of the cultivated species lot.

Additionally, an event of type "kind": "Event", "eventType": "ObjectEvent" is generated, describing a shipping operation from the source production unit.

Lastly, an event of type "kind": "Event", "eventType": "ObjectEvent" is generated, describing a receipt operation of the lot at the destination production unit.

10. sea2see.sale

Represents a sale by the aquaculturist to another actor in the value chain. Two Tilkal events are generated: one of type MasterData containing information about the buyer, and a transformation event of type "kind": "Event", "EventType": "TransformationEvent" reflecting the quantity sent and received in the transaction.

Once the relevant events have been translated, the application routes the events to the corresponding Tilkal channel associated with the originating Smartwater Cloud actor, based on their ID.

SENDING MESSAGES TO TILKAL PLATFORM

The final module of the Sea2See (S2S) API application handles sending messages to Tilkal using the appropriate channels corresponding to the actors, via HTTP requests after authenticating the application using AWS Cognito.

The TilkalNode class

A class for sending data to the tilkal traceability platform has been developed. This class manages automatically authentication with the AWS Cognito API and management of refresh tokens.

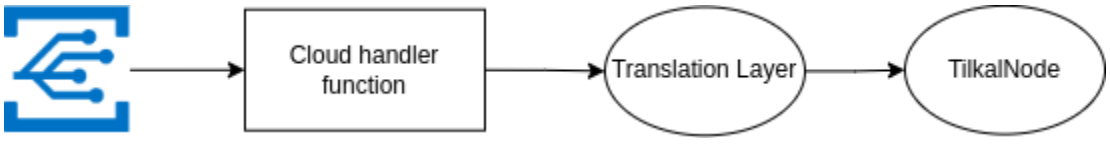
```
class TilkalNode {
    private static _authBaseUrl = Constants.TILKAL_OAUTH_BASE_URL;
    private static _oauthPath = Constants.TILKAL_OAUTH_PATH;

    private static _baseUrl = Constants.TILKAL_BASE_URL;
    private static _publishPath = Constants.TILKAL_PUBLISH_PATH;

    private static _bodyType = "application/x.tilkal.epcis+json";

    private _token?: string;

    private _clientId: string;
    private _clientSecret: string;
    private _scope: string;
```

```
    constructor(clientId: string, clientSecret: string, scope: string) {
        this._clientId = clientId;
        this._clientSecret = clientSecret;
        this._scope = scope;
    }

    private _withParams(path: string, { params = undefined, baseUrl = TilkalNode._baseUrl }:
{ params?: { [key: string]: string }, baseUrl?: string }): URL {
        // Implementation...
    }

    async _requestTheAPI<T>(url: URL, method: string, body: any): Promise<T> {
        // Implementation...
    }

    async refreshToken(): Promise<void> {
        // Implementation...
    }

    async publish(messages: EpcisEvent, channel: string): Promise<void>;
    async publish(messages: EpcisEvent[], channel: string): Promise<void>;
    async publish(messages: EpcisEvent | EpcisEvent[], channel: string): Promise<void> {
        // Implementation...
    }
}

export { TilkalNode };
```

Class definition of the TilkalNode class

2.4. DEFINITION OF THE FUNCTION

Azure function project defines each function within a separated file along with the trigger of the function.



Simplified architecture of the function processing the events from Cloud coming from EventGrid

CLOUD HANDLER FUNCTION

1. Trigger Definition

Registers eventGridCloud as an EventGrid handler in Azure Functions, allowing it to trigger whenever an event of the specified type is received.

```
app.eventGrid('eventGridCloud', {
    handler: eventGridCloud,
});
```

Banner informing how to register a listener of eventGridCloud using AzureFunctions framework.

2. The function definition

```
async function cloudSea2SeeEvent(event: CloudEventGrid, context: InvocationContext):
Promise<void> {

    const eventData = event.data;

    let events: EpcisEvent[] = [];
    switch (event.eventType) {
        case "sea2see.company":
            events = CloudTranslation.Company.translate(eventData);
            break;

        case "sea2see.productionCenter":
            events = CloudTranslation.ProductionCenter.translate(eventData);
            break;

        case "sea2see.productionUnit":
            events = CloudTranslation.ProductionUnit.translate(eventData);
            break;

        //... All OTHER EVENTS ...
    }

    //... Publish events to Tilakal using TilkalNode class
}
```

Banner showing a simplified version of the code used to handle the events that comes from cloud in EventGrid

Description

This function processes specific Sea2See events based on the event type. It translates the data using `CloudTranslation` and, if enabled in `Constants` , publishes the translated events in Tilkal.

Event Types Handled

The types of events processed by the function and the applied translation are as follows: `sea2see.company`, `sea2see.productionCenter`, `sea2see.productionUnit`, `sea2see.expense`, `sea2see.lot`, `sea2see.biometry`, `sea2see.counting`, `sea2see.death`, `sea2see.transfer`, `sea2see.sale`.

3. Cloud Translation layer

Each translation layer is a class implementing the interface `ITranslation` each class transforms the raw event that we receive from EventGrid into a epcis event suitable to be published into the Tilkal platform. Each event type has it's own translation class according to the specifications of the deliverable D3.3.

```
    export const Company: ITranslation = new class {
        isTranslatable(data: any): boolean {
            let is_valid = true;
            //... validations ...
            return true;
        }
        translate(data: CompanyEventGridData): EpcisEvent[] {
            const country = CountryISO[data.address!.country.id];
            const companyId = _epcisCompanyId(data.tenantIntId);
            const location = new Location(companyId, 'en-US')
                .setCity(data.address!.city)
                .setPostalCode(data.address!.zipCode)
                .setCountryCode(country)
                .setStreetAddressOne(data.address!.street)
                .setState(data.address!.state)
                .addAttribute(new
Attribute(`${EPCIS_Prefix.GS1.EPCMasterDataAttributePrefix}#name`, data.companyName))
                .addAttribute(new
Attribute(`${EPCIS_Prefix.Tilkal.MasterDataAttribute}#category`, "Fish Farm Company"));

            return [location];
        }
  }
```

Example of the translation for the sea2see.company event using the tilkal message-builder library.

4. Publishing to Tilkal platform

To publish the events to the Tilkal platform we use the Smartwater's control tower and the TilkalNode class.

```
const tower = new TilkalNode(Constants.TILKAL_CLIENT_ID, Constants.TILKAL_CLIENT_SECRET,
Constants.TILKAL_SCOPE);


let events: EpcisEvent[] = [];
switch (event.eventType) {
//... Handle the event ...
}

const channel = getTilkalChannel();
tower.publish(events, channel);
```

Code using the TilkalNode class to publish the EPCIS events into the tilkal platform

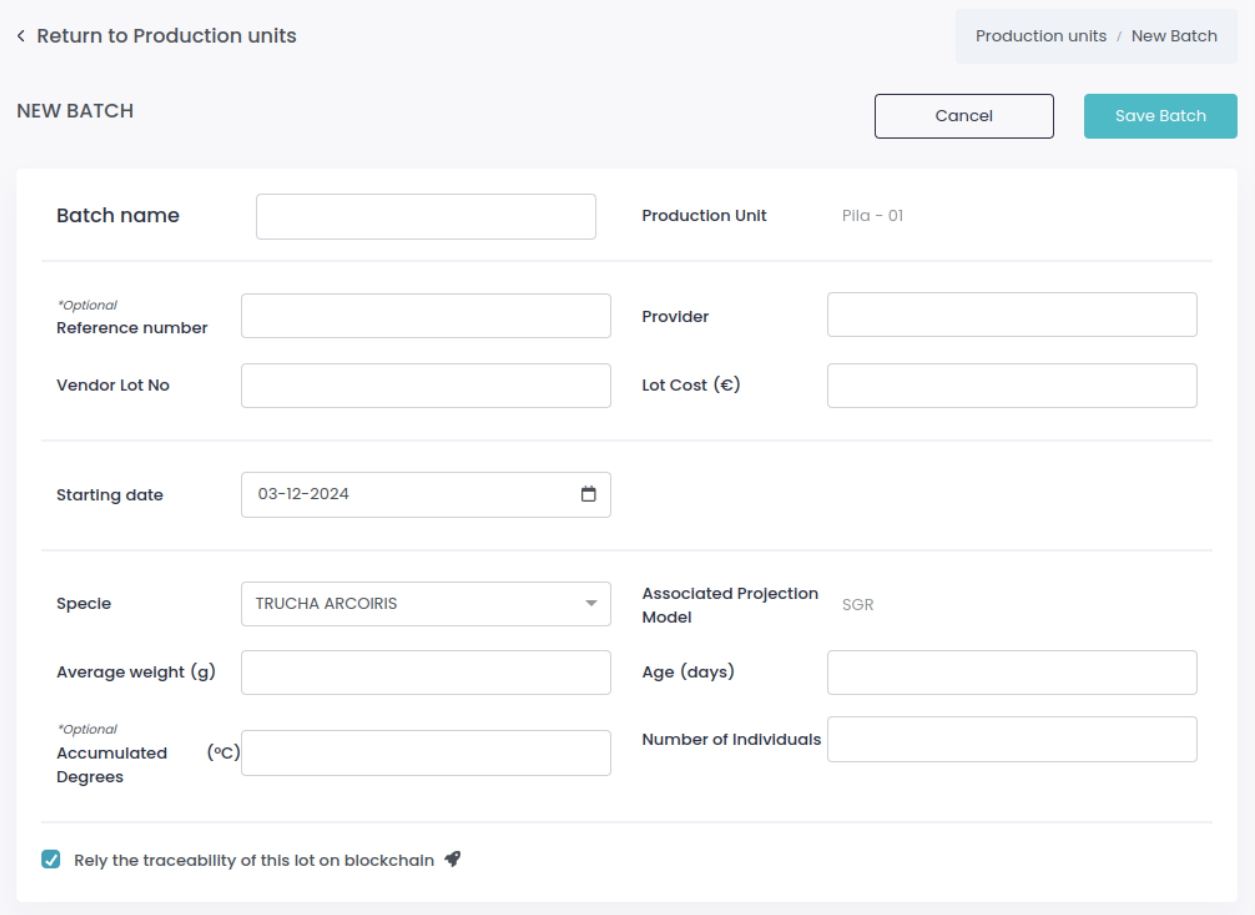
3. USER INTERFACE

A fork of SMARTWATER CLOUD frontend has been deployed in a demonstration environment under the following URL:

https://salmon-ground-04b9aac03-140.westeurope.azurestaticapps.net/

3.1. BATCH CREATION

For creating a batch and getting the system to manage its traceability using the Sea2See Traceability Platform, we must check the "**Rely the traceability of this lot on blockchain**" option box.



Form for creating a lot showing the checkbox for using the blockchain traceability in SMARTWATER CLOUD

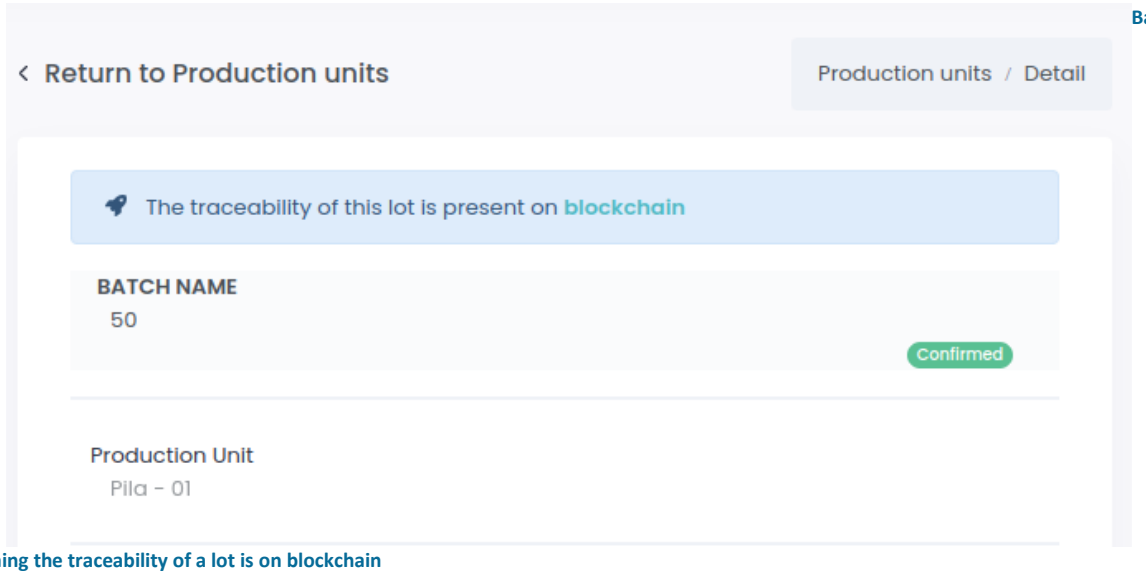
When clicking on "Save Batch" all the traceability of this batch will be registered into the traceability platform in progressive way.

When visualizing the information of a specific batch we can see the mark that says that this batch is available in the blockchain.

At the very first point of the creation of a batch, it will produce the following events and publish them to the S2S traceability platform:

- sea2see.company: Company related data. Contains the master data information of the company which produces fish. It will be translated into a Location event.
- sea2see.productionCenter: Facility related data. Contains the master data information of the facility which production activity takes place. It will be translated into a Location event.
- sea2see.productionUnit: Production unit related data. Contains the master data information of the production unit where fish is stocked, al the production activity events are linked to this production unit. It will be translated into a Location event.
- sea2see.lot: Lot related data, this event contains all the information of the lot, which includes but not limited to: fish specie, quantity and average weight. It will be translated to different EPCIS events: MasterData event for specie and fish material, Commission event for the creation of the batch in the traceability platform, Shipping and Receiving event from the facility to the production unit.
- sea2see.expense: Information related to the purchase made by the facility to the fish provider. It will create several events: Location event of the provider of fish, Shipping and Receiving event from provider to the facility, Transformation event which converts the fish product into an aquaculture fish lot.

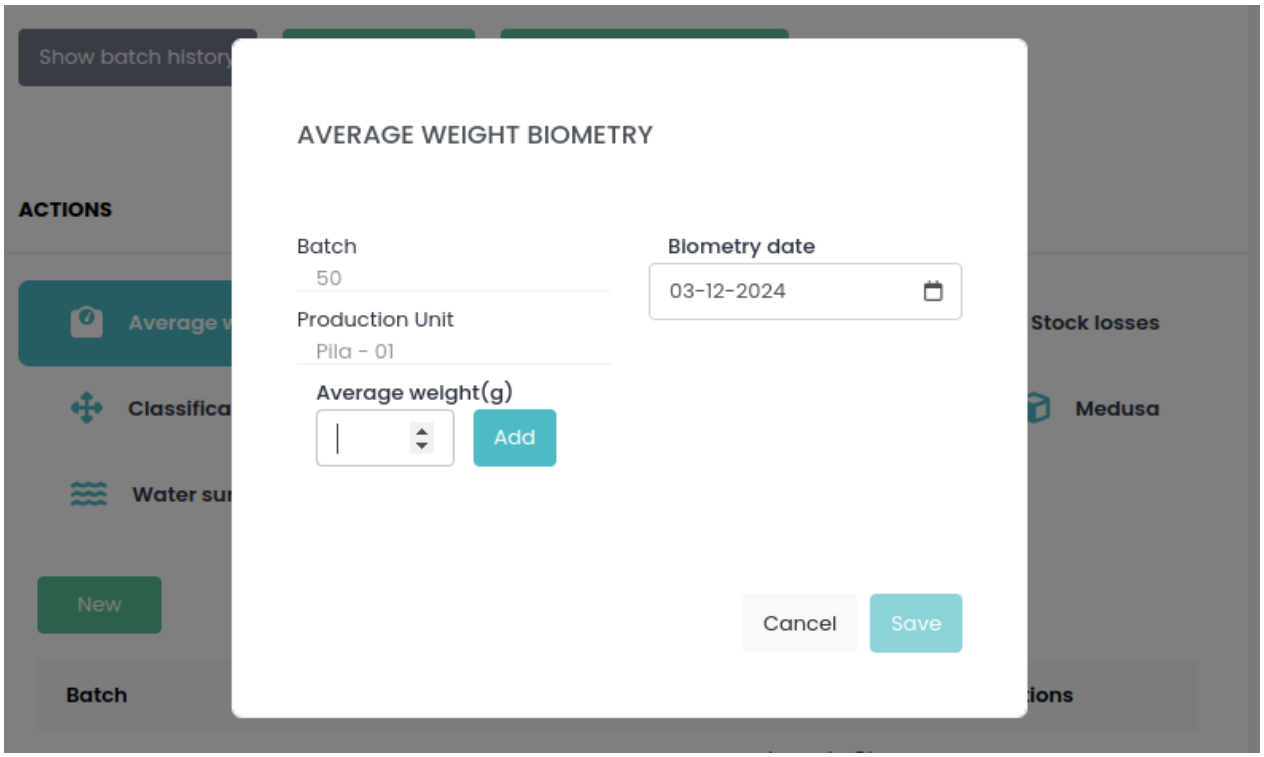
Banner



informing the traceability of a lot is on blockchain

3.2. BATCH EVENTS

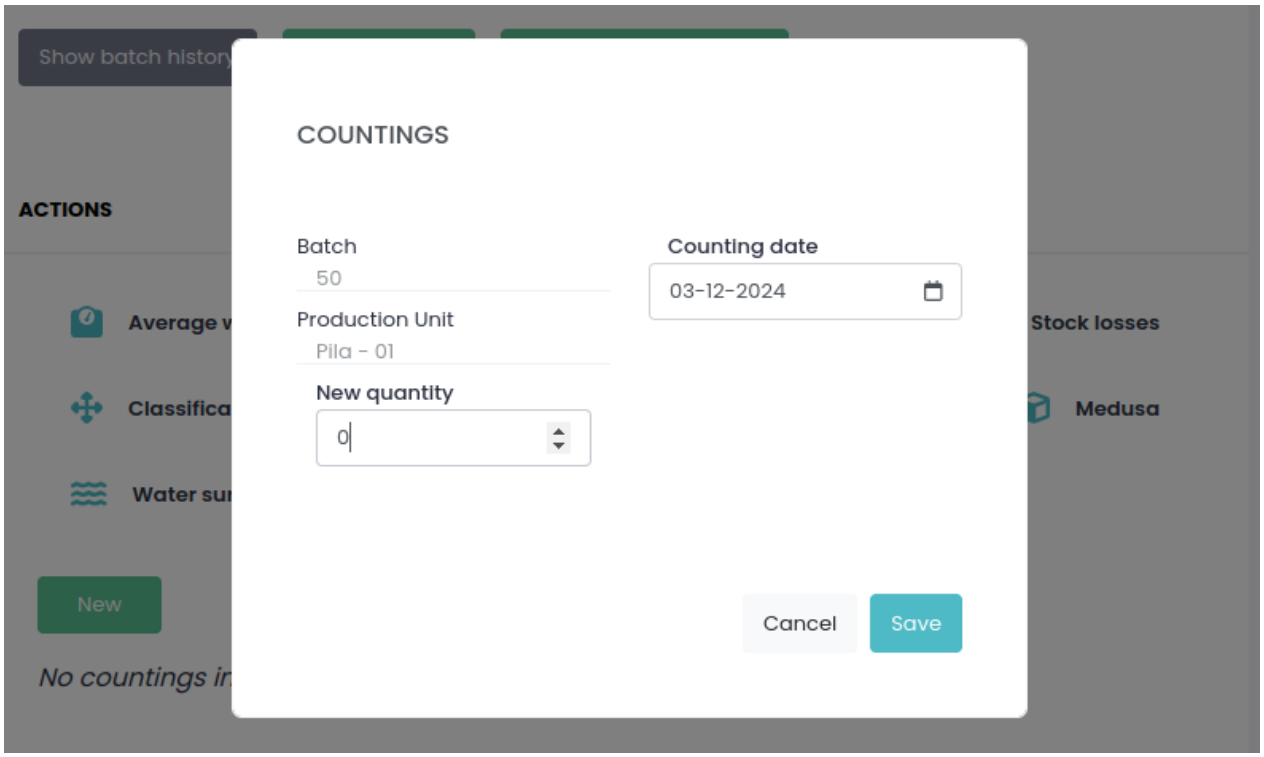
AVERAGE WEIGHT BIOMETRY



Form for creating a stock loss in SMARTWATER CLOUD

Creating an average weight biometry will produce in real-time an event of type `sea2see.biometry` and publish it on the traceability platform.

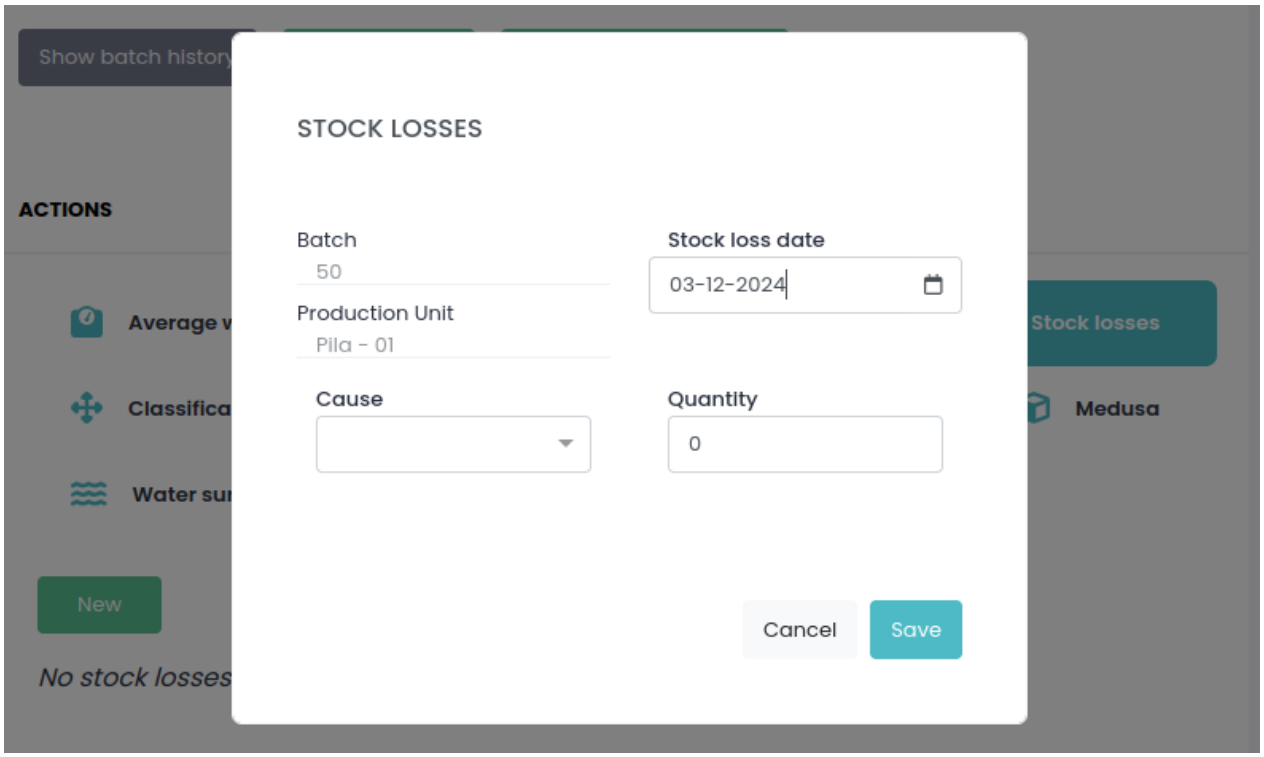
COUNTING



Form for creating a stock loss in SMARTWATER CLOUD

Creating a counting will produce in real-time an event of type `sea2see.counting` and publish it on the traceability platform.

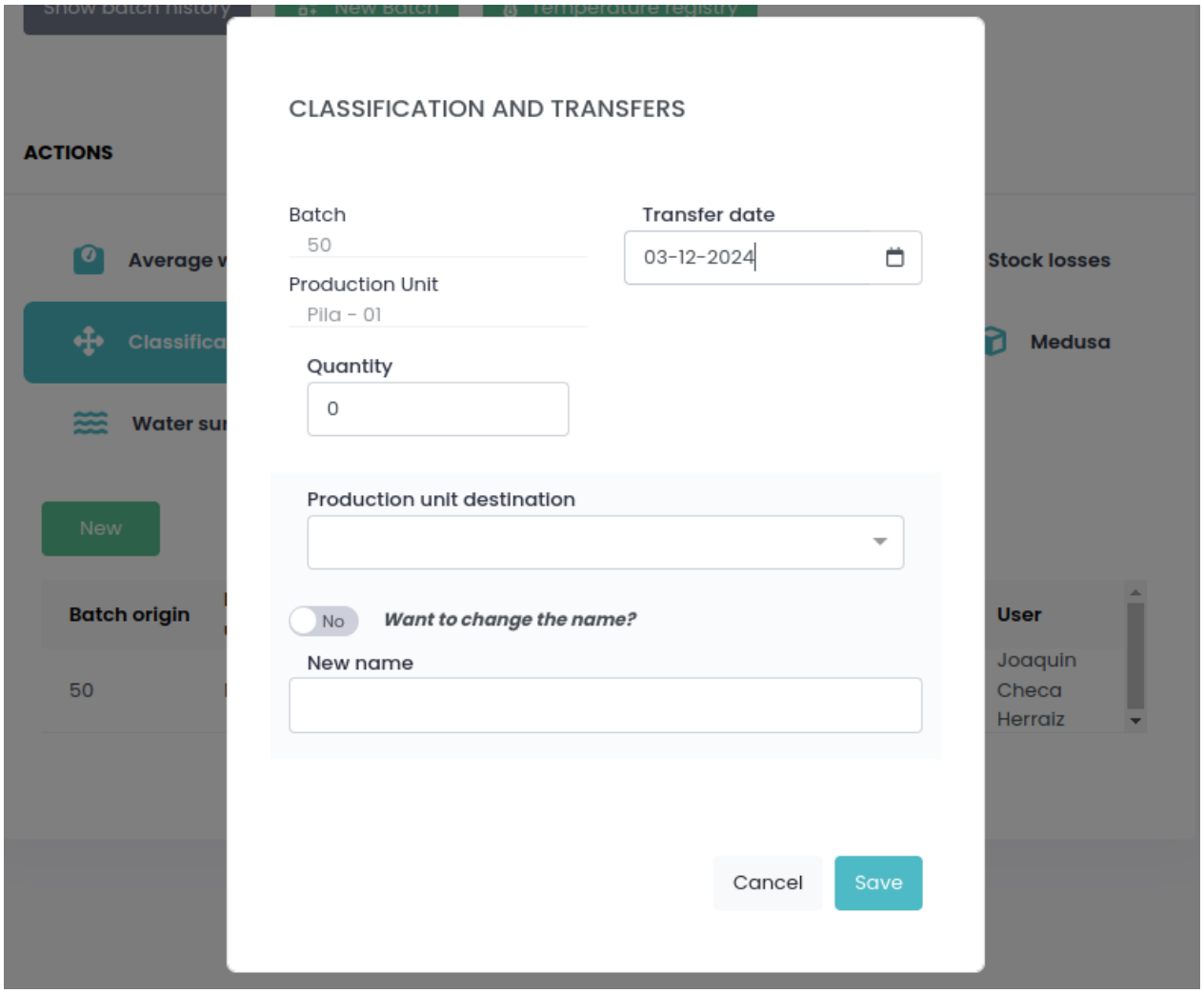
STOCK LOSS (DEATH)



Form for creating a stock loss in SMARTWATER CLOUD

Creating a stock loss or death event will produce in real-time an event of type `sea2see.death` and publish it on the traceability platform.

CLASSIFICATION AND TRANSFERS



Form for creating a transfer in SMARTWATER CLOUD

Creating a transfer will produce in real-time an event of type sea2see.transfer and publish it on the traceability platform.

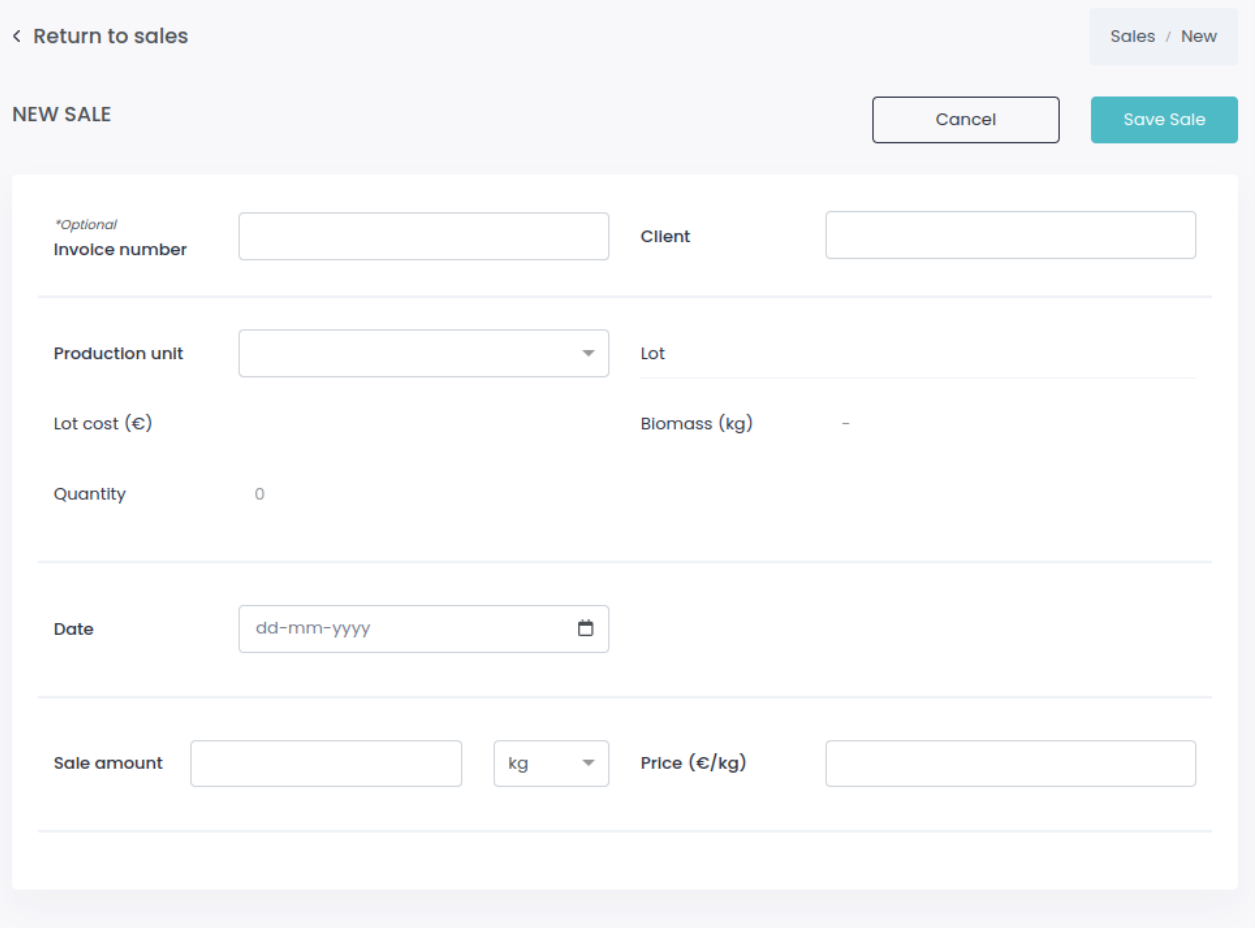
A transfer may create some time new batches, if those new batches come from a batch that has its traceability using the S2S platform, then the new batch will also have its traceability in the S2S platform.

The creation of a transfer will produce different EPCIS events: `Transformation` event, converts the lot in the old production unit into a new lot inside a new production unit, `Shipping` and `Receiving` events which represents the movement of fish form one production unit into other.

3.3. SALE OF THE BATCH

When a fish batch arrives at its commercial size then it can be sold to a customer. From smartwater cloud we can create partial sales so a unique production batch can be sold to different customers.

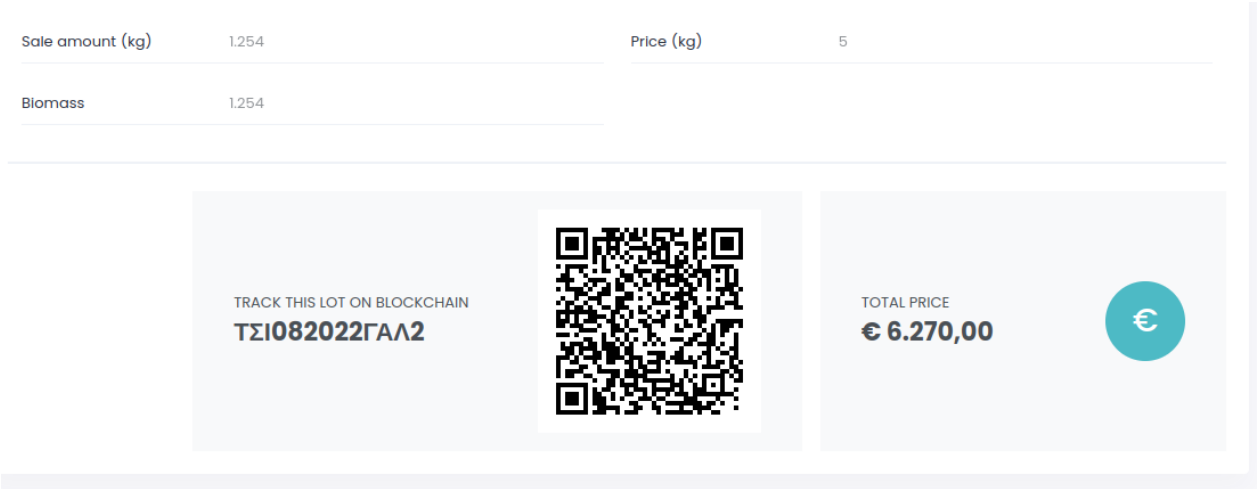
Each Selling event represents a set of events being published into the traceability platform.



Form for creating a sale in SMARTWATER CLOUD

After creating a sale, the quantity sold is subtracted from the fish quantity in the production unit and if the batch relies on its traceability in the S2S platform, then a `sea2see.sale` event is produced and published to the traceability platform. This event will produce the following EPCIS events: `Location` of the customer which buys the fish, a `Transformation` event which converts the production fish batch into a sold batch.

After the sale is made, then a QR code is created so we can continue the traceability steps using the *Set of professional mobile apps developed by Page Up.*



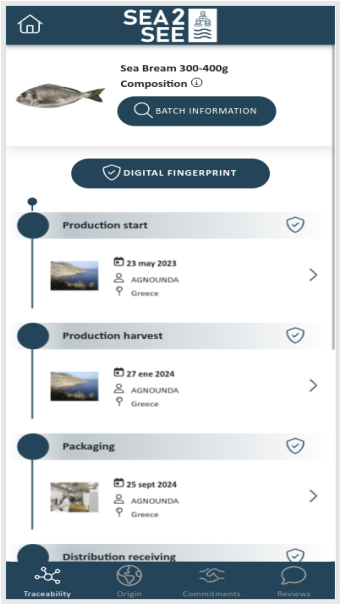
QR Code generated by SMARTWATER CLOUD after creating a sale

3.4. VISUALIZING THE TRACEABILITY IN SPOTLIGHT

After issuing the retail, distribution and/or packaging steps using the apps. We can finally deliver a QR code to the final customer where all the traceability is displayed.

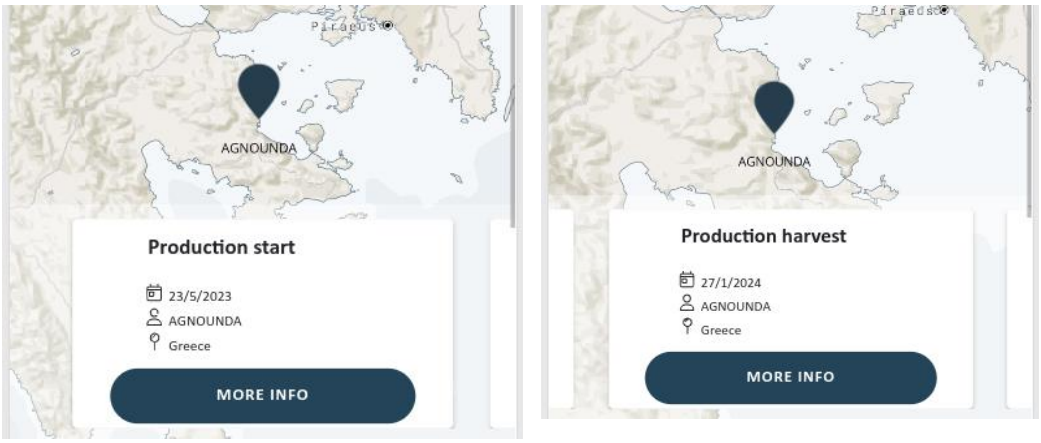
Steps that come from SMARTWATER CLOUD are the following:

- **Production start**: Start of production activity, it comes from the batch creation in SMARTWATER CLOUD.
- **Production harvest**: End of production activity, it comes from the selling of the batch in SMARTWATER CLOUD.



Screenshot of spotlight application showing traceability of a seabream batch

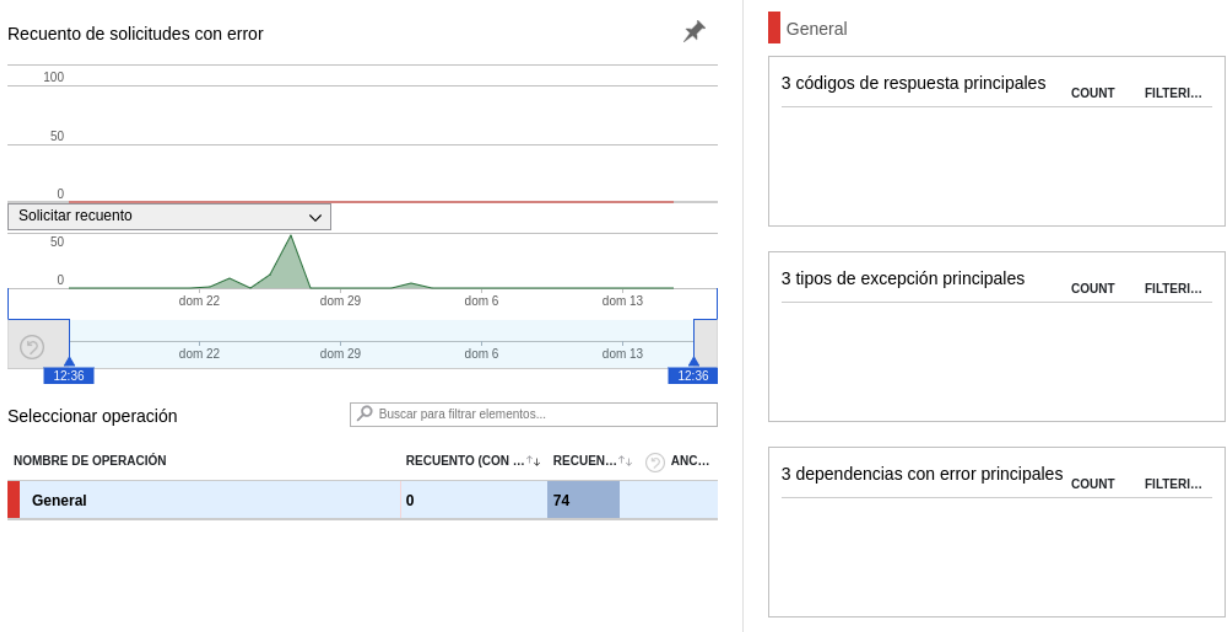
These events also come with geolocation information from SMARTWATER CLOUD.



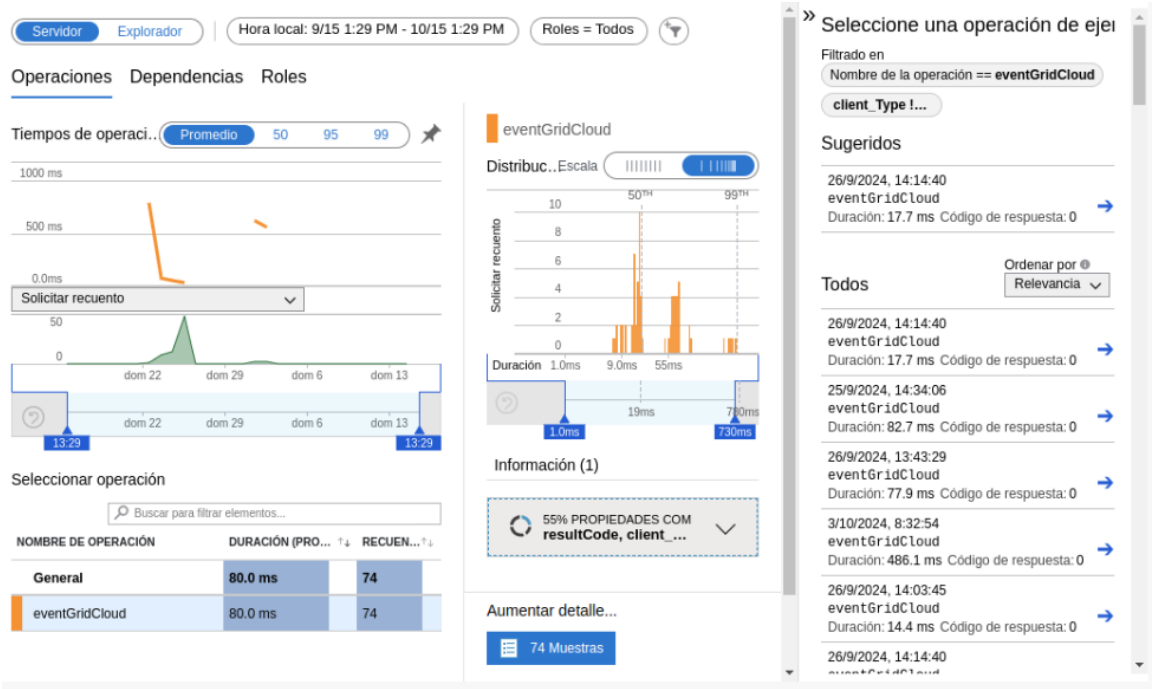
Images of the location of Production start and Production harvest

3. EXECUTION ANALYTICS

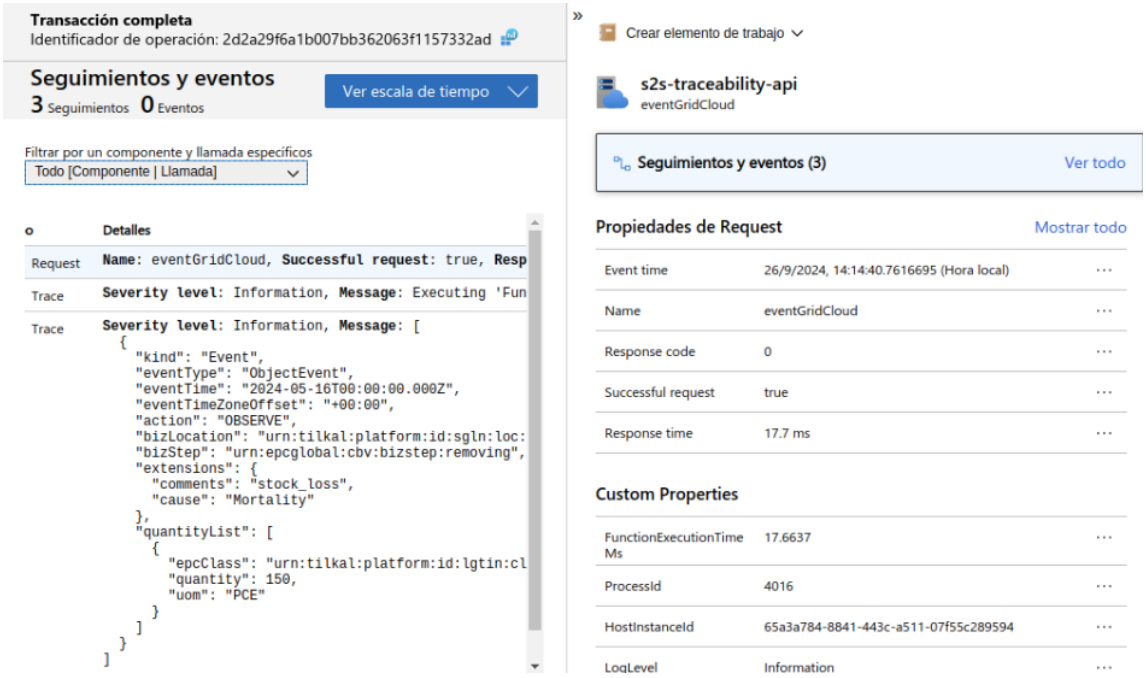
The S2S API is connected to an Azure resource called Application insights, from application insights we can visualize execution metrics and error rate of the execution of every endpoint of the AzureFunctions app.



Screenshot of Application Inshights showing number of errors reported by the application



Screenshot of Application Insights showing number of executions made bay the application



Sample of the information logged by the application when an execution takes place

The use of this tool is very useful because we can monitor the performance of the application, and also detect errors during the execution of the endpoints. This gives us enough information so we can take decisions in optimizing the implementation of the API and fix any bugs that may appear.