

Nestéo

By Marcus Wichelmann, Randy Nguyen,
Simon Oyen, Simon Schwierzeck



Who we are

Frontend Team:

- Simon Oyen - Hochschule Hannover
- Simon Schwierzeck - Hochschule Hannover

Backend Team:

- Randy Nguyen - Grand Valley State University
- Marcus Wichelmann - Hochschule Hannover

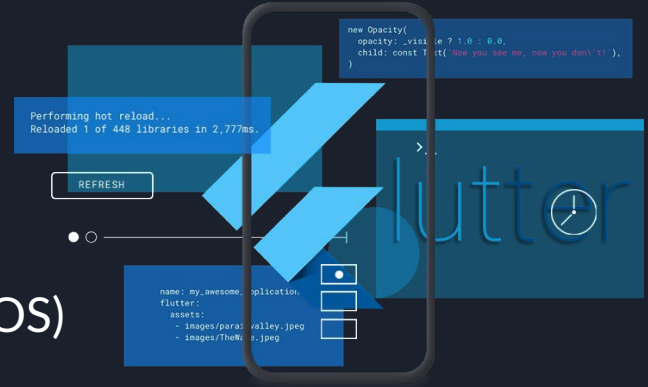


What we built

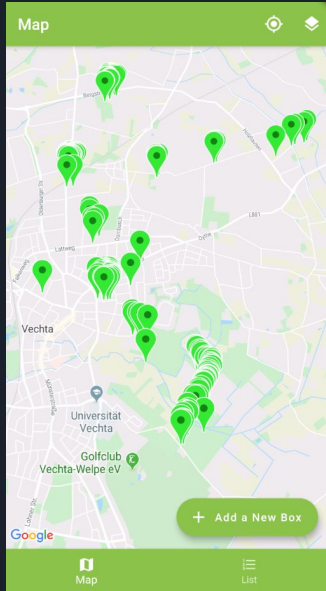
- A system for bird-ringing associations
 - Managing nesting-boxes
 - Monitoring inspection-data of nesting-boxes
- Goals
 - Provide an alternative for the “old” way (Excel-sheets)
 - Easy to use
 - Consistent
 - Open source

What we built

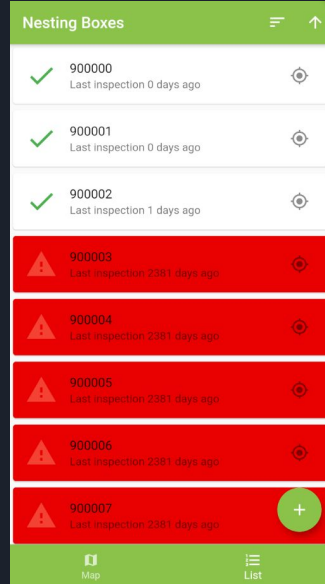
- Frontend: Mobile app for Android (and iOS)
 - Flutter framework, Dart language
 - BLoC pattern for state management
 - Business Logic Component
 - Material Design
- Backend:
 - C# with .Net Core framework
 - MariaDB database
 - Swagger API
 - REST API Communicates with Nesteo App
 - Data Import and Export features



Mobile App

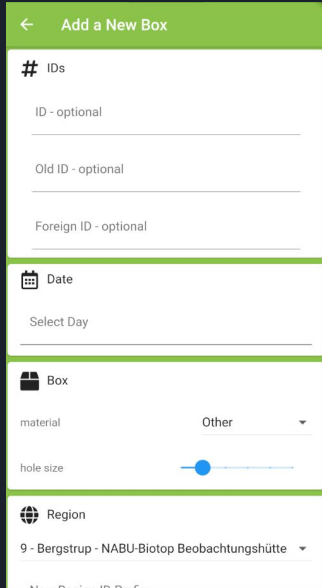


- Map visualisation
 - Shows nesting boxes as markers



- List visualisation
 - Shows nesting boxes in a list
 - Can be sorted by various parameters
 - Color represents inspection-status

Mobile App



← Add a New Box

IDs

ID - optional

Old ID - optional

Foreign ID - optional

Date

Select Day

Box

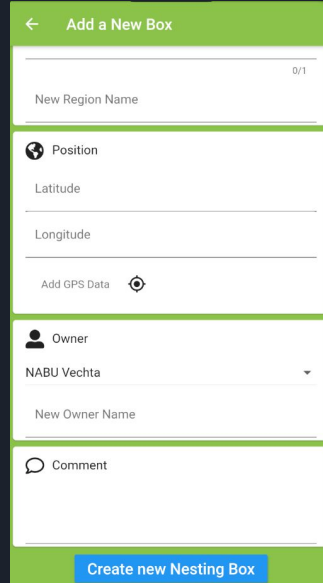
material Other

hole size

Region

9 - Bergstrup - NABU-Biotop Beobachtungshütte

- New nesting box dialogue
 - Allows creation of new nesting boxes



← Add a New Box

New Region Name 0/1

Position

Latitude

Longitude

Add GPS Data

Owner

NABU Vechta

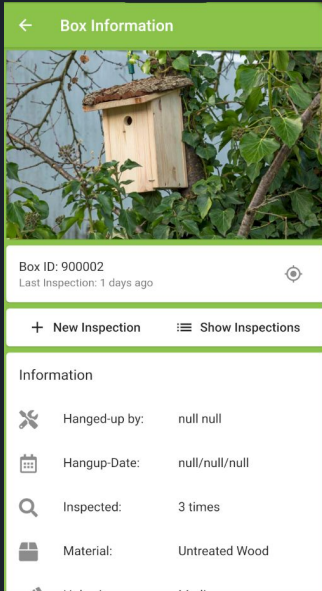
New Owner Name

Comment

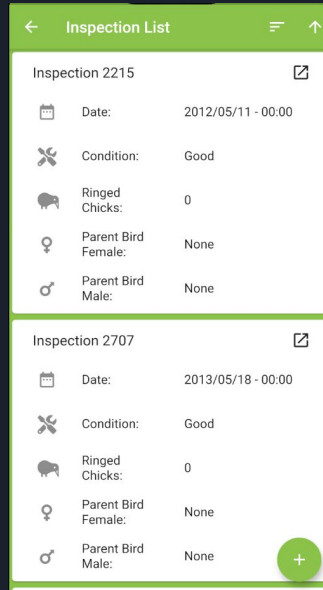
Create new Nesting Box

- Various parameters can be set
- Box-location can be set through GPS
- Region and Owner parameters are stored and can be used for other boxes

Mobile App




- Box Information
 - Shows information about boxes and their inspections
 - Allows creation of new inspections



- Inspection List
 - Shows inspections of a nesting box
 - Each entry can be selected to show even more information

Mobile App

← Inspection Information



Inspection ID: 2215
Last Inspection: 5/11/2012

✕

Cleaned: No

Condition: Good

Repaired: No

●

Contains Eggs: No

- Inspection information
 - Shows detailed information about inspections

← Add a New Inspection

☰ Date
Select Day

✕ Box
Cleaned
Repaired
Condition

● Eggs
Contains Eggs
Number of Eggs

🗨 Birds
Occupied
Number of Chicks

← Add a New Inspection

🗨 Birds

Occupied ☐

Number of Chicks — 0 +

Ringed Chicks — 0 +

Age in Days — 0 +

Female Parent None ▾

Male Parent None ▾

Gartenbaumläufer ▾

New Species

0/1

🗨 Comment

Create new Inspection

- New inspection screen
 - Many parameters that can be set by the user
 - Species are stored and can be used for later inspection

Mobile App

← Inspection Information

Inspection ID: 2215
Last inspection: 5/11/2012

✕

Cleaned: No

Condition: Good

Repaired: No

🥚

Contains Eggs: No

Number of Eggs: 0

🐣

Ringed Chicks: 0

Number of Chicks: 0

Parent Bird Male: None

← Informationen zur Inspektion

Inspection ID: 2215
Last Inspection: 5/11/2012

✕

Gereinigt: Nein

Zustand: Gut

Repariert: Nein

🥚

Enthält Eier: Nein

Anzahl der Eier: 0

🐣

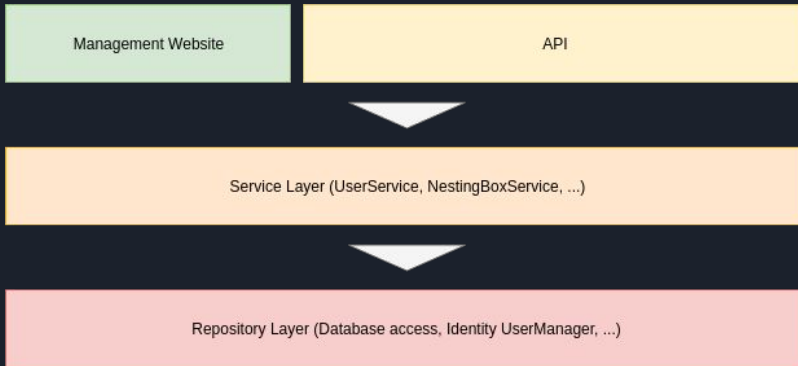
Beringte Jungvögel: 0

Anzahl Jungvögel: 0

Elternvogel männlich: Keiner

- Both in english and german
 - Depending on system language

Backend Organization

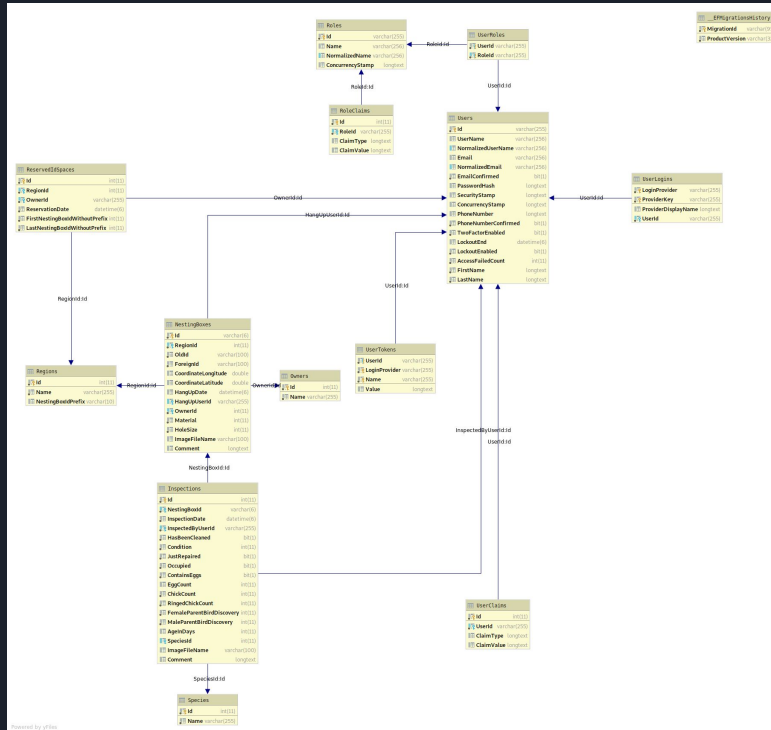


Management Website & API: Communicates with the interfaces of the service layer (for example IUserService). Focused on simple user interface or API tasks like handling authorization or returning appropriate HTTP responses.

Service Layer: Main application logic. It uses a mapping library (AutoMapper) to map between "entity" types (from the service layer) and normal "model" types.

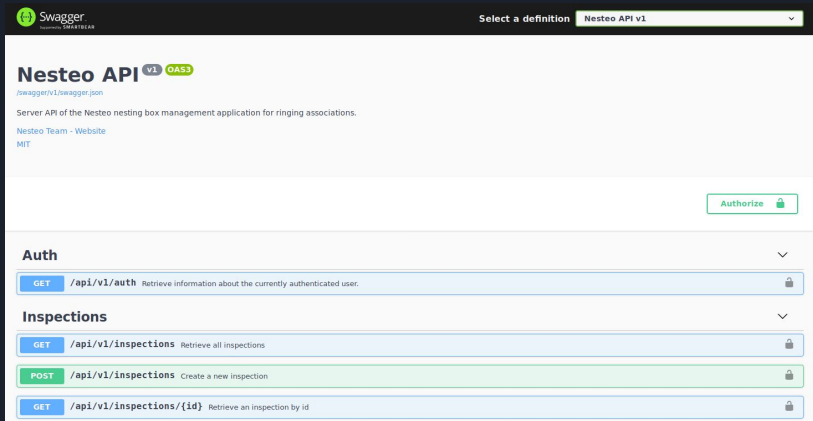
Repository Layer: Directly related with databases and data storage. Implemented by mostly entity classes and the data management logic provided by Entity Framework Core. Contains the user management using the inbuilt Identity framework.

Database Schema



- Stored in MariaDB instance
- Automatically generated/updated by Entity Framework Core migrations
- Tables for user management
- Tables for functional data (Nesting Boxes etc.)

Swagger API



- Test requests
- User can input values into certain requests
 - “Inspections by Id”

API Documentation

Retrieve information about the currently authenticated user.

AUTHORIZATIONS:

basic

Responses

> 200 Success

▼ 400 Bad Request

RESPONSE SCHEMA: application/json

| | |
|--------------|--------------------------|
| errors > | object Nullable |
| type | string Nullable |
| title | string Nullable |
| status | integer <int32> Nullable |
| detail | string Nullable |
| instance | string Nullable |
| extensions > | object Nullable |

GET /api/v1/auth

Response samples

200 400 401 403 404 409 500

Content type
application/json

Copy Expand all Collapse all

```
{
  "errors": {
    "property": [ ... ],
    "property": [ ... ]
  },
  "type": "string",
  "title": "string",
  "status": 0,
  "detail": "string",
  "instance": "string",
  "extensions": {
    "property": { },
    "property": { }
  }
}
```

- Automatically generated
- Describes responses statuses
- Contains response samples