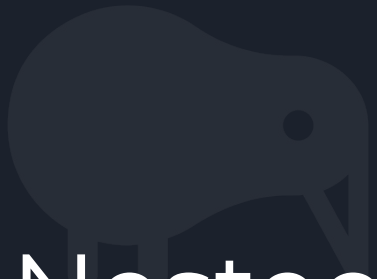


Nesteo

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

Who we are

Frontend Team:

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

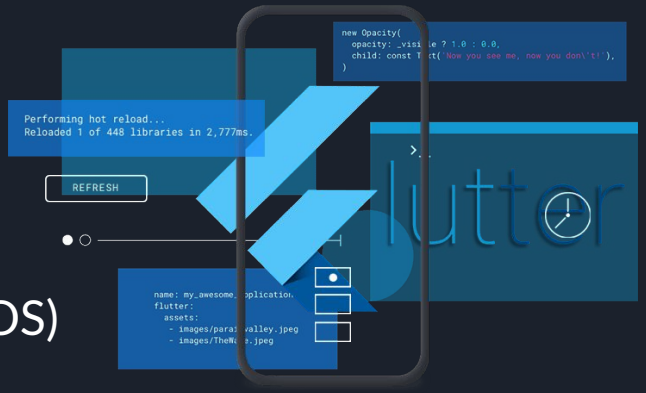
Backend Team:

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

What we built

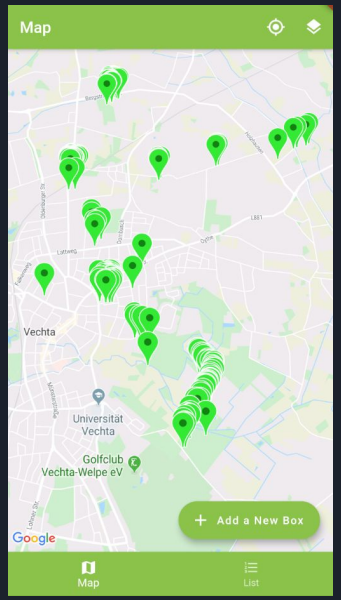
- A system for bird-ringing associations
  - Managing nesting-boxes
  - Monitoring inspection-data of nesting-boxes
- Goals
  - Provide an alternative for the “old” way (Excel-sheets)
  - Easy to use
  - Consistent
  - Open source

What we built

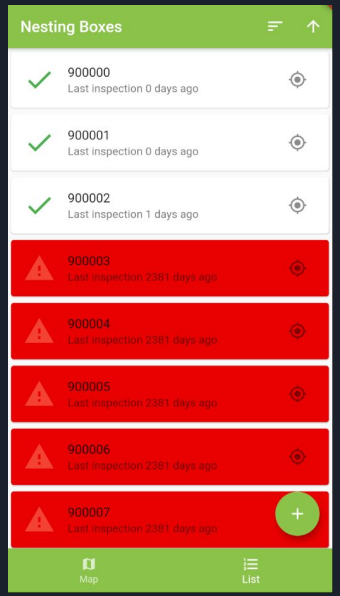


- Frontend: Mobile app for Android (and iOS)
  - Flutter framework, Dart language
  - BLoC pattern for state management
    - Business Logic Component
  - Material Design
- Backend:
  - MariaDB database
  - Swagger API
  - REST API Communicates with Nesteo App
  - Data Import and Export features

Mobile App

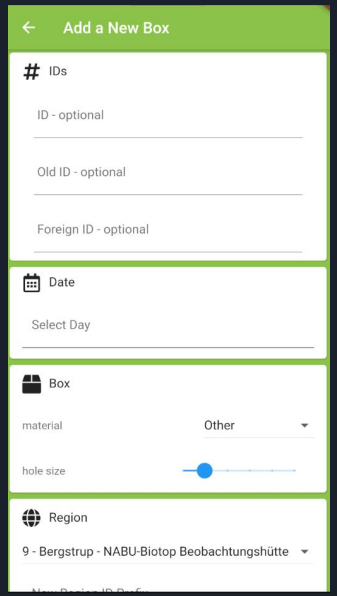


- Map visualisation
  - Shows nesting boxes as markers

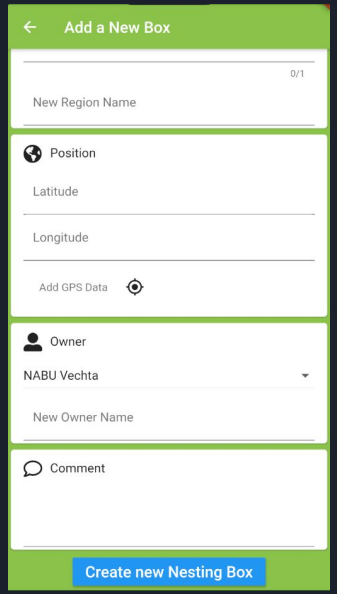


- List visualisation
  - Shows nesting boxes in a list
  - Can be sorted by various parameters
  - Color represents inspection-status

Mobile App

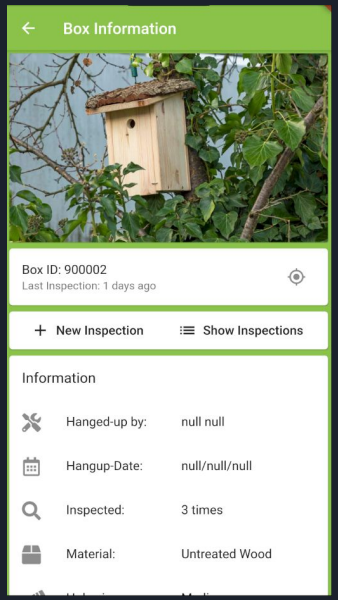


- New nesting box dialogue
  - Allows creation of new nesting boxes

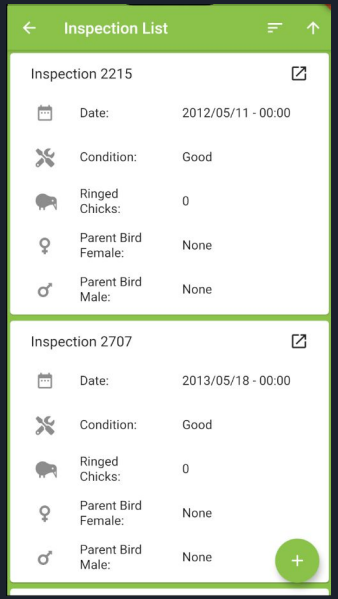


- Various parameters can be set
- Box-location can be set through GPS
- Region and Owner parameters are stored and can be used for other boxes

Mobile App

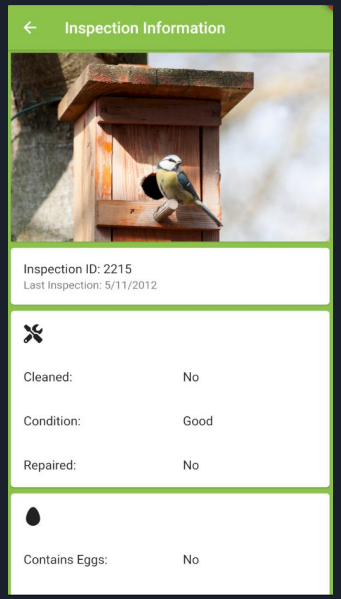


- Box Information
  - Shows information about boxes and their inspections
  - Allows creation of new inspections

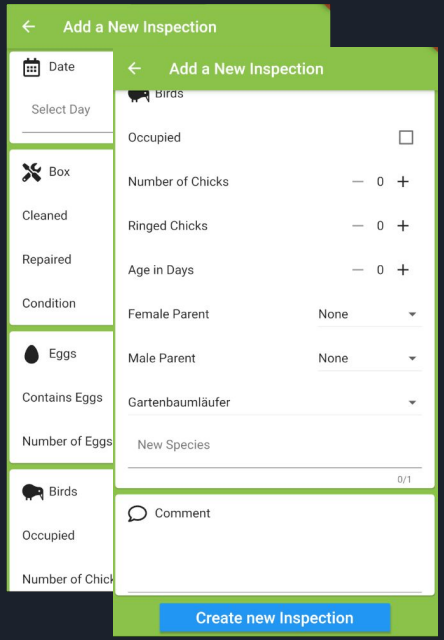


- Inspection List
  - Shows inspections of a nesting box
  - Each entry can be selected to show even more information

Mobile App

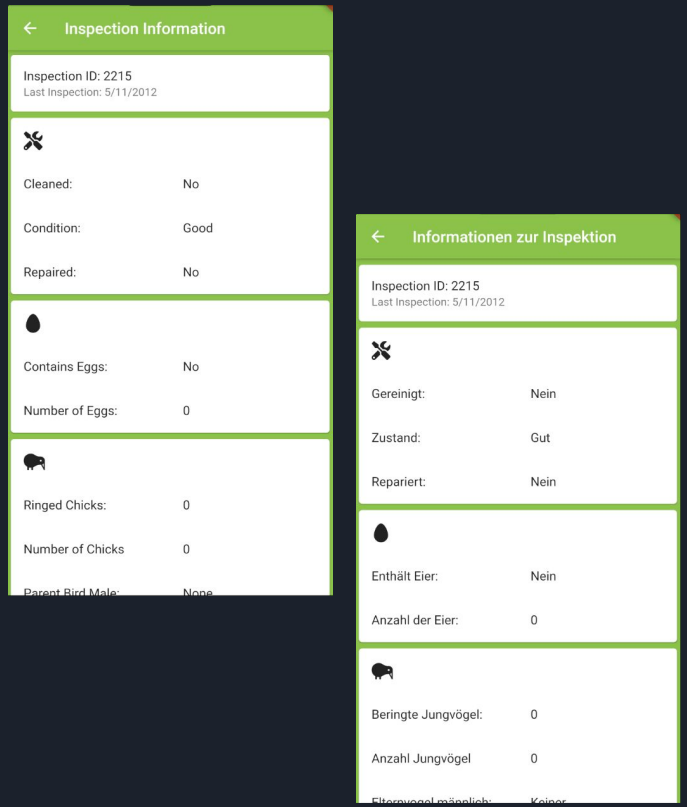


- Inspection information
  - Shows detailed information about inspections



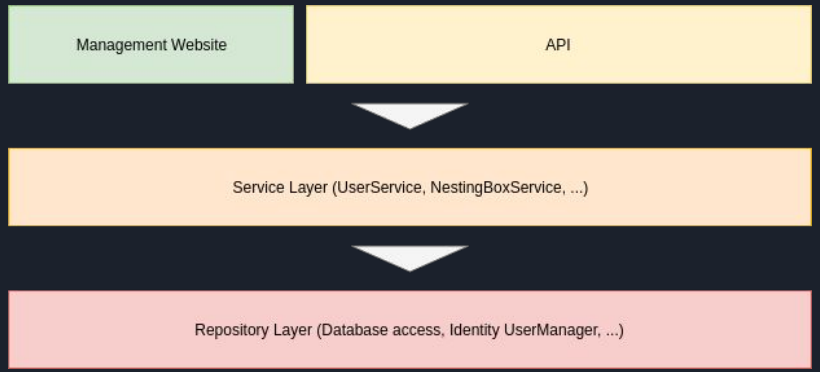
- New inspection screen
  - Many parameters that can be set by the user
  - Species are stored and can be used for later inspection

Mobile App



- Both in english and german
  - Depending on system language

Backend Organization

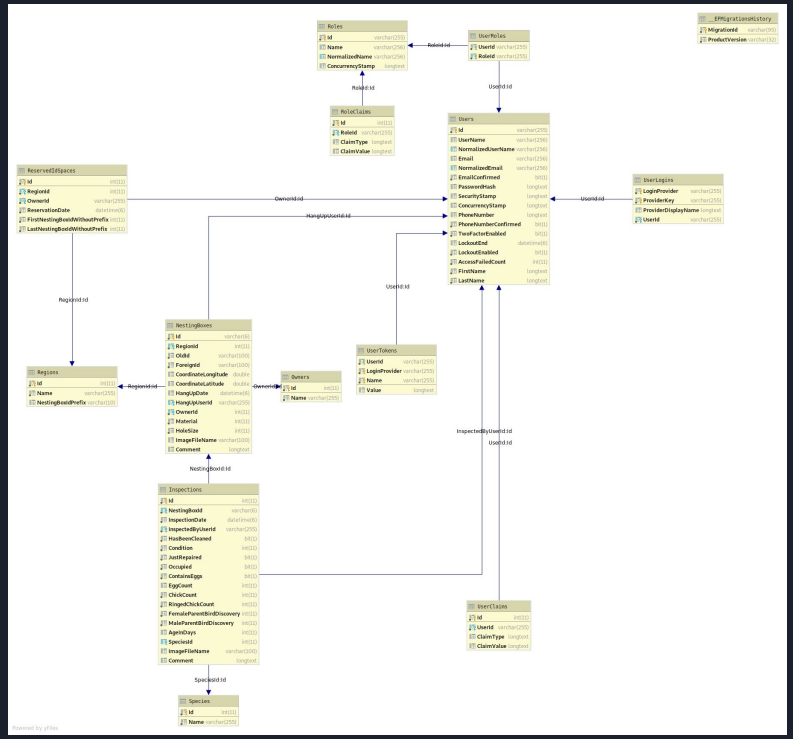


Management Website & API: Communicates with the interfaces of the service layer (for example IUserService). Focused on simple user interface or API tasks like handling authorization or returning appropriate HTTP responses.

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

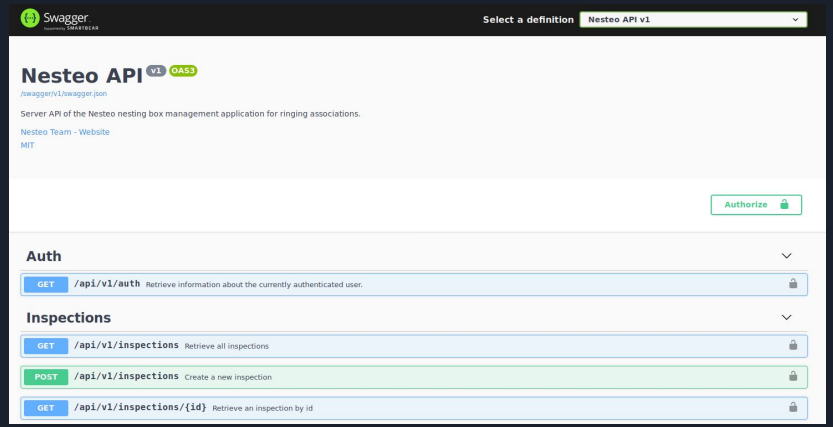
Repository Layer: Directly related with databases and data storage. Implemented by mostly entity classes and the data management logic provided by Entity Framework Core. Contains the user management using the inbuilt Identity framework.

Database Schema



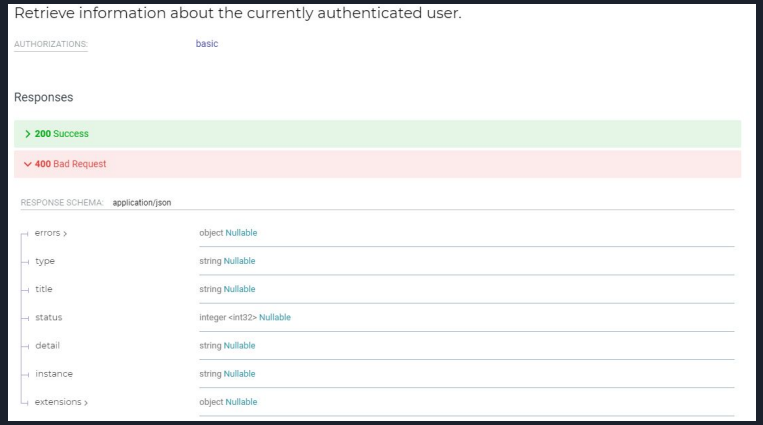
- Stored in MariaDB instance
- Table for Users
- Tables for functional data (Nesting Boxes etc.)

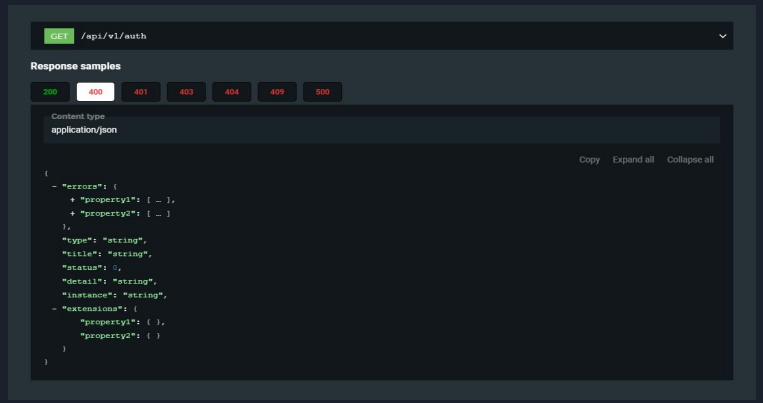
Swagger API



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

API Documentation





- Describes responses statuses
- Contains response samples