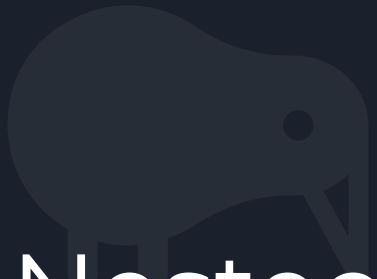


Nesteo

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

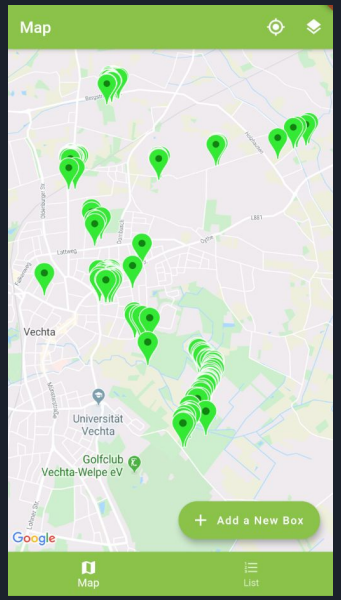
What we built

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

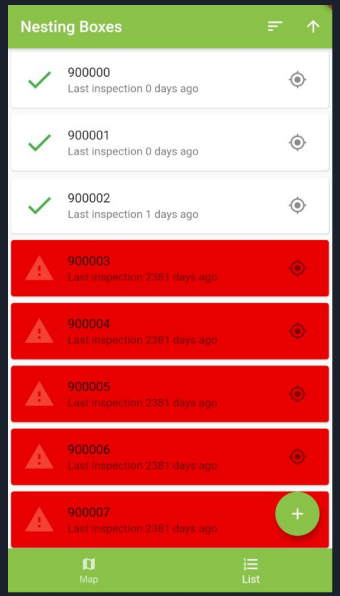
Mobile App - Technical background

- Flutter framework
  - Allows Android and iOS apps from the same code
  - UI is created inside the code
- Material Design components
- State Management with BLoC design pattern
  - Events -> Business Logic Component -> State
  - Scales very good
  - Uses dependency injection
- Requests to the backend
  - Service-Classes make http-requests and return JSON
  - Repository-Classes create objects from JSON
  - Code generation

Mobile App

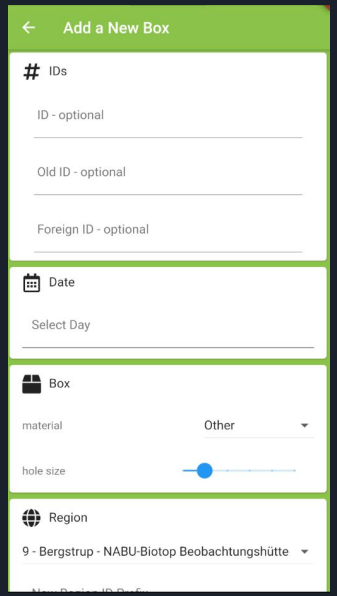


- Map visualisation
  - Shows nesting boxes as markers

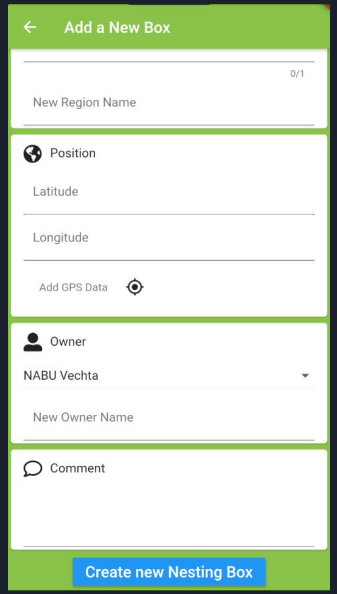


- List visualisation
  - Can be sorted by various parameters
  - Color represents inspection-status

Mobile App

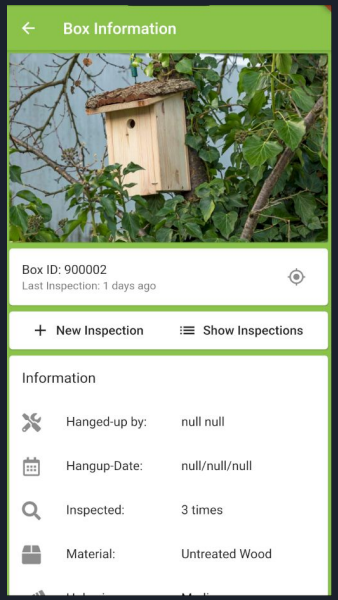


- New nesting box dialogue
  - Allows creation of new nesting boxes

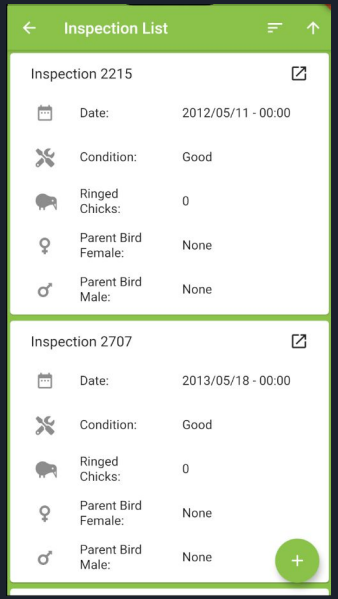


- Various parameters can be set
- Box-location can be set through GPS
- Region and Owner parameters are stored and can be used for other boxes

Mobile App

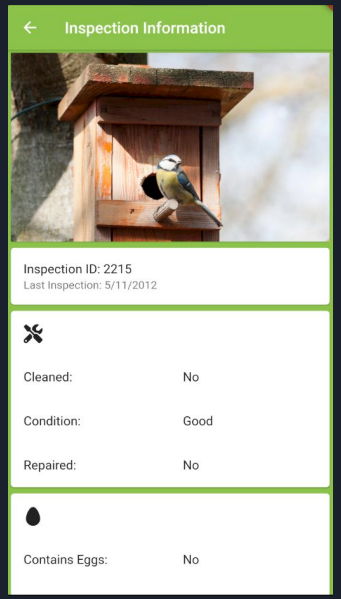


- Box Information
  - Shows information about boxes and their inspections
  - Allows creation of new inspections

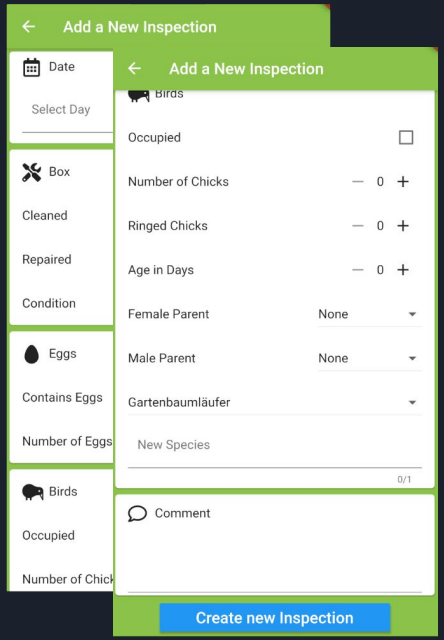


- Inspection List
  - Shows inspections of a nesting box
  - Each entry can be selected to show even more information

Mobile App

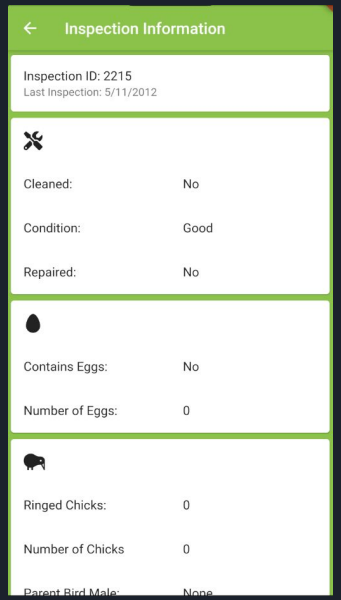


- Inspection information
  - Shows detailed information about inspections

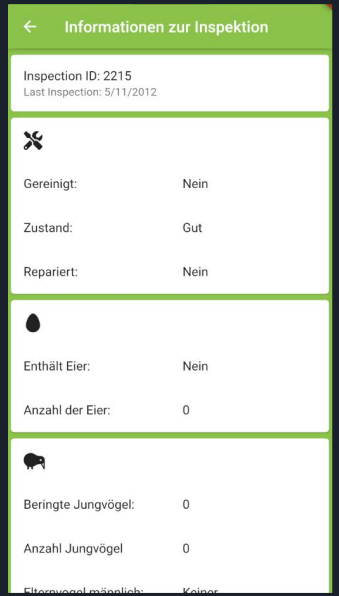


- New inspection screen
  - Many parameters that can be set by the user
  - Species are stored and can be used for later inspection

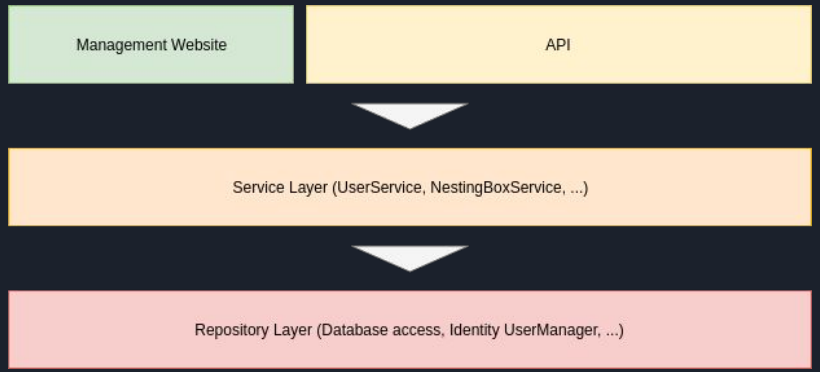
Mobile App



- Both in english and german
  - Depending on system language



Backend Organization

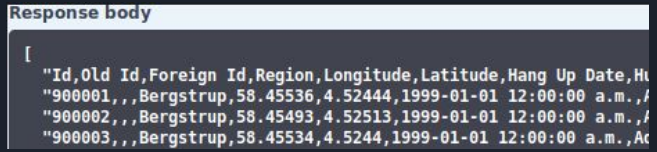


Management Website & API: Communicates with the interfaces of the service layer (for example IUserService). Focused on simple user interface or API tasks like handling authorization or returning appropriate HTTP responses.

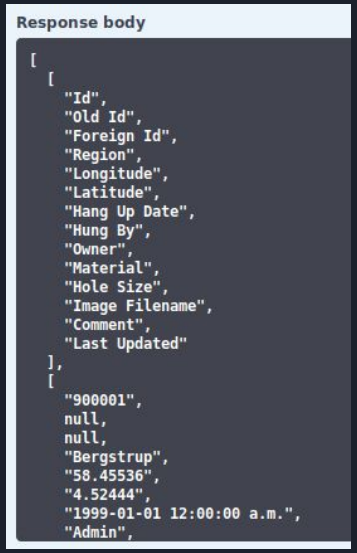
Service Layer: Main application logic. It uses a mapping library (AutoMapper) to map between "entity" types (from the service layer) and normal "model" types.

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.

Data Export



Exported as String



Exported as String[]

Tried Packages:
CsvHelper
LINQtoCSV
ServiceStack

Data Export

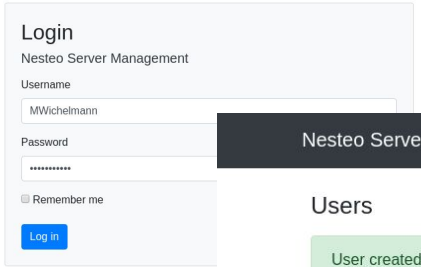
```
// Convert field values to strings
IEnumerable<string> encodedFieldValues = fields.Select(value => value?.ToString() ?? string.Empty);

// Escape double-quotes in value
encodedFieldValues = encodedFieldValues.Select(value => value.Replace( oldValue: "\"",  newValue: "\"\""));

// Enclose fields containing reserved characters with double-quotes
encodedFieldValues = encodedFieldValues.Select(value => {
    if (value.Contains("\r\n") || value.Contains("\"") || value.Contains(","))
        value = $"\"{value}\"";
    return value;
});

// Join fields to a single CSV row
return string.Join( separator: ",", encodedFieldValues);
```

Manually Define CSV Serialization

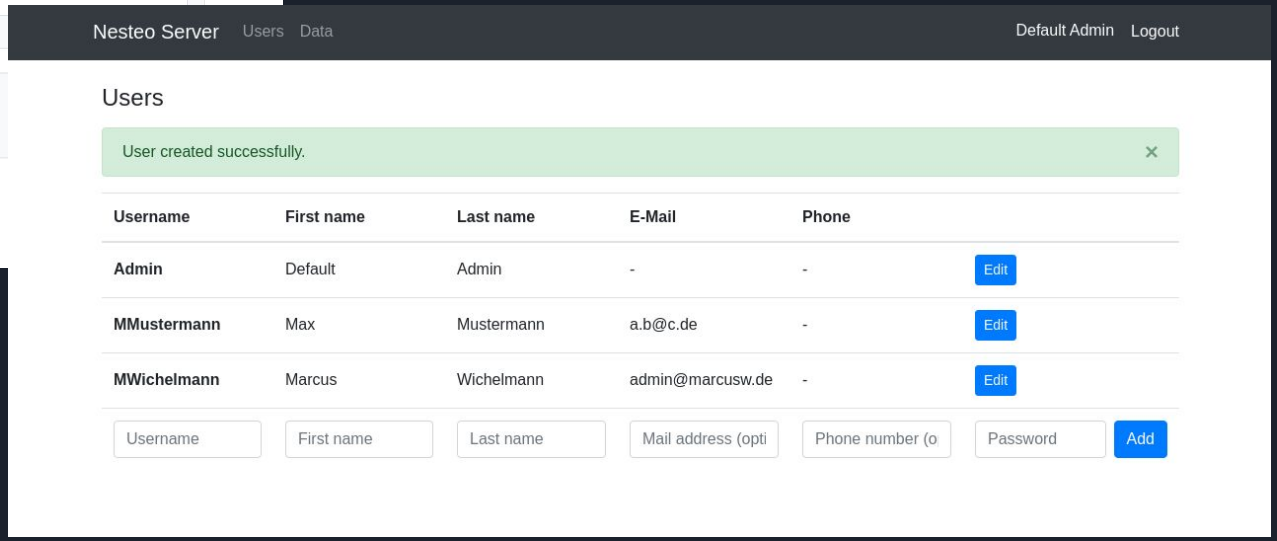
```
// Set response content type
response.ContentType = "text/csv";

// Set content disposition to make this a download
var contentDisposition = new ContentDispositionHeaderValue("attachment");
contentDisposition.SetHttpFileName(_fileDownloadName);
response.Headers[HeaderNames.ContentDisposition] = contentDisposition.ToString();

try
{
    // Write CSV records to response stream
    await using var streamWriter = new StreamWriter(response.Body, Encoding.UTF8,  bufferSize: -1,  leaveOpen: true);
    await foreach (string record in _records)
        await streamWriter.WriteLineAsync(record);
}
```

Change Response from JSON to CSV

Server Management Interface

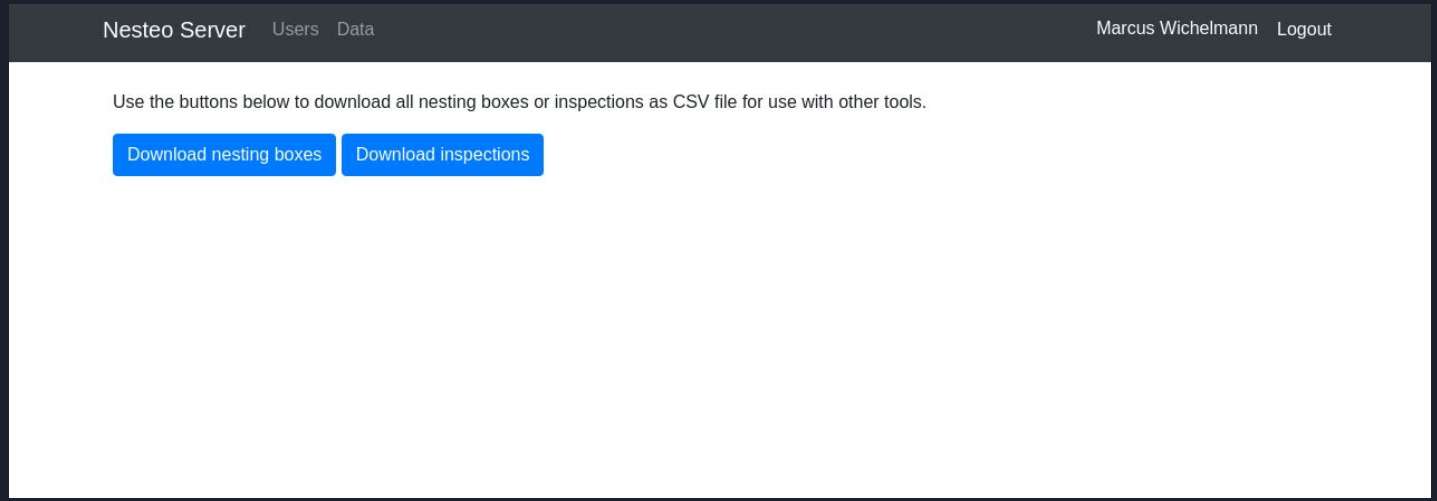


Easy management of all users who are allowed to use the app

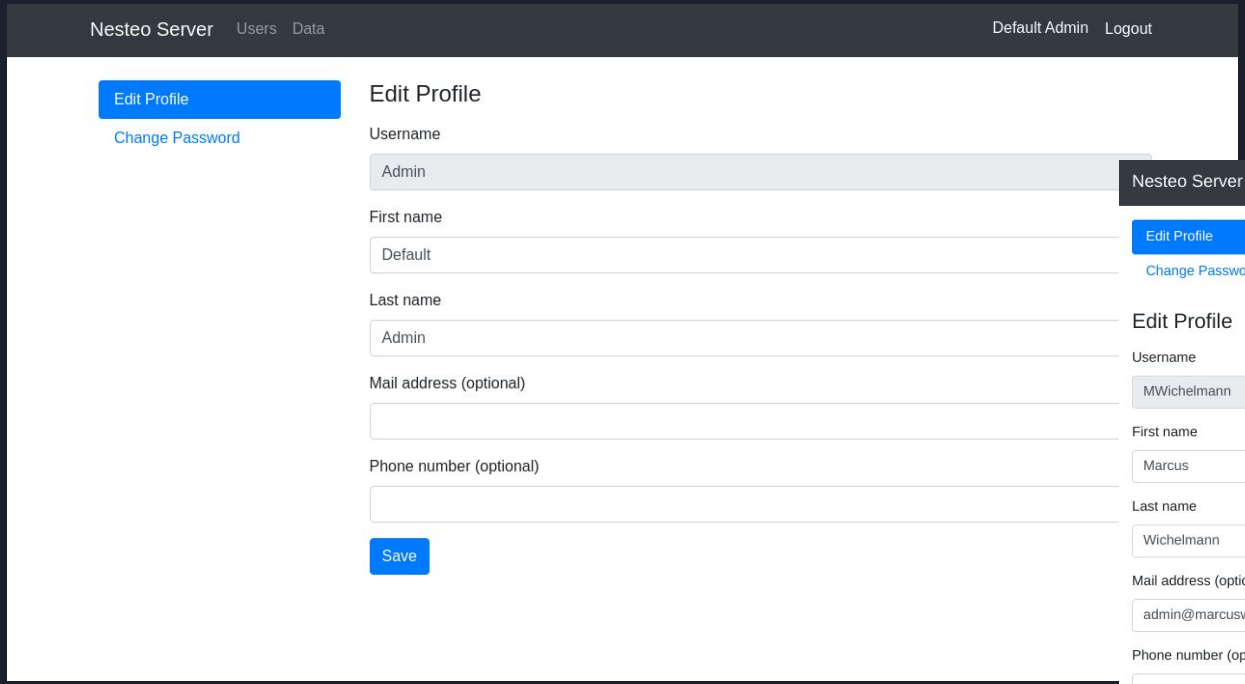


Server Management Interface

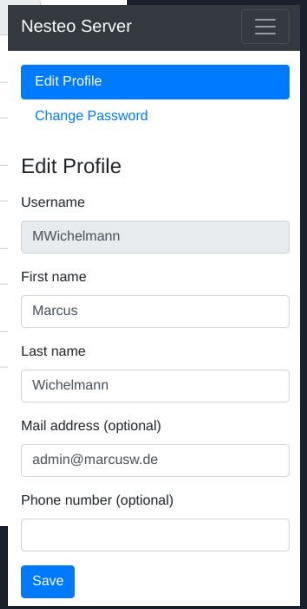
Download all nesting boxes and inspections as CSV file



Server Management Interface



Personal profile management



Technical Growth

Randy Nguyen

- Gained frontend and backend experience
- BLoC design pattern
- Agile team
- Documentation
- Time management

Technical Growth

Marcus Wichelmann

- Entity Framework Core
- Leading the backend
- Communication
- Language skills
- Agile team
- Time management

Technical Growth

Simon Oyen

- Mobile development
  - UI, business-logic, backend-communication
- Working with dependency-injection
- Using code-generation
- Dart/Flutter
- BloC design pattern for state management
- Planning / time management
  - Estimating issues, planning sprints ahead

Technical Growth

Simon Schwierzeck

- Mobile Development
- Dart/Flutter
- Usability
- Github/Zenhub
- Teamwork
- Communication

Teamwork Reflection

- Language barrier was no problem
- Time Zones makes collaboration difficult
- Problems with 2 former team members
- Restructuring the team
- Visit at GVSU
- Conclusion on teamwork

Principles

“Manage personnel and resources to enhance the quality of working life”

- International team, 6-hour time difference
- Tasks have been specifically split up to not be dependent on each other
- when close collaboration was necessary
    - German members worked late
    - American members worked early

Principles

“Create opportunities for members of the organization to grow as professionals”

- Project was an educational opportunity
- Teams were assigned by interests and country
  - Everyone should work with someone from the other country
- Rebalancing the teams later allowed one member to learn a wider range of technology

Conclusions

- Frontend
  - Flutter/Dart works great, would recommend
  - Continuous Integration with Flutter works, used it more for builds than tests though
  - App in the playstore is a nice motivation
- Backend
  - Entity Framework Core and .Net Core ecosystem is very performant
  - Management website gives user-friendly way of managing accounts and data
- General
  - GitHub with ZenHub works great
    - Could have created more issues for deliverables etc.
  - Slack works good if everybody is active
  - Time-Management was difficult later in the project
  - Presentation / Report rhythm works very good