JOURNAL OF SPATIAL INFORMATION SCIENCE
Number N (YYYY), pp. xx–yy
doi:10.5311/JOSIS.*TBD*

RESEARCH ARTICLE

# A Web Application for Wildlife Sightings - How Accessible Is This Technology Using Modern Frameworks?

Julian Kraft

LSFM, ZHAW, Switzerland

*Received: December 18, 2024; returned: TBD; .*

**Abstract:**
The project aimed to evaluate the accessibility of modern technology for creating a web application that records geotagged data. Using modern web frameworks, a prototype was developed to document wildlife sightings. This application, accessible online and installable on smartphones across major operating systems, exemplifies how a straightforward and purpose-driven tool can be built with moderate programming knowledge, guided support, and persistence. The project's workflow encompassed backend development in Go, frontend design with the Ionic framework, and database integration using MariaDB. Additionally, the system incorporates a public data visualization tool and direct database access. While the project highlights the potential of current technologies and AI-powered tools like GitHub Copilot, it underscores challenges in managing interconnected components. The result is a versatile platform, demonstrating both the promise and complexity of custom web application development for geospatial data collection.

**Keywords:** web application development, wildlife monitoring, geospatial data, user authentication, frontend frameworks, backend programming, database management, mobile web apps, ionic and angular frameworks

## 1 Introduction

At the heart of any project involving geospatial data are the data themselves. If these data are not yet available, the goal is to find an appropriate method to collect them. In the field

of environmental sciences, data collection still often relies on manual methods. However, this fieldwork can also be simplified through the use of technology. A suitable tool for this purpose is something almost everyone possesses – a smartphone. A quick online search reveals that there are already many apps available that can accomplish this. However, these are often relatively expensive, offer far more features than actually needed, and control over the data often lies with the app provider. With all the modern frameworks and programming languages available today, it should not be that difficult to write a custom app that does exactly what is required. And so, the idea to create a custom app was born. In this case, an example to record wildlife sightings was chosen as the use case. The general idea is to create an individual toolkit from which future use cases can be derived. This work aims to examine just how challenging it really is to develop an app, particularly with very limited prior knowledge, a lot of trial and error, extensive research, the use of AI, immense patience, and significantly more support from an experienced developer.

It was with this support that the project began. To avoid having to figure everything out from scratch, the process started with a consultation. During this meeting, the idea was presented, and discussions were held on which technologies, programming languages, and frameworks would be practical, feasible, and sensible. In a subsequent session, the server was configured, and the development environment was set up together. A not-so-small crash course then provided a solid foundation for diving into the project.

# 2 Methods

The app essentially consists of three parts: the web-based frontend, the backend server, and the database. These three components are interconnected and communicate with each other. The frontend is the user interface displayed on the smartphone, the backend handles the logic running in the background, and the database serves as the storage location for all data that is retrieved or saved. There is a separate frontend component, a static page, for the data view which is publicly available and can be accessed via a web browser.

## 2.1 Frontend

The frontend was created using the Ionic framework. Ionic is an open-source toolkit for developing mobile user interfaces. It is based on Angular, another open-source framework developed by Google. Ionic enables the creation of web applications that run on all platforms – Android, iOS, and web – while offering native app-like performance and experience [2]. The framework makes it relatively easy to start a project and test it in a development environment, where the app is displayed in the browser. The actual programming is done using the programming languages HTML, CSS, and TypeScript. TypeScript is a programming language developed by Microsoft that is based on JavaScript. CSS is a stylesheets language used for designing the user interface, while HTML is a markup language used for structuring the user interface.

There are two main pages implemented in the frontend. The home page deals with user authentication and the animals page where the user can input data. The home page is the first page displayed when the app is opened. There are two options: to create a user or to log in. Upon successful authentication, the user is directed to the animals page. Here there is a stepper at work in the background. Upon completion of a step, the stepper is increased

by one, triggering a different view for the next step. This is all handled by the frontend itself. Only fetching and saving data are handled by the backend.

## 2.2 Backend

The backend was created using the programming language Go. Go is a programming language developed by Google that is used for the development of web applications and servers. It is easy to learn and use and offers good performance [1]. It is also worth mentioning that GitHub Copilot provides excellent suggestions for the Go programming language, which significantly simplifies programming. The main logic handled by the backend in the app includes user authentication, accessing the database to check for available users or retrieve options for animals, and transforming and saving new data in the database.

It is structured into a few files. There is the main file `main.go` where the server is started and the routes are defined. And there are three different handler files to handle processes concerning login, animal sightings, and the requests of the static page for the data view.

## 2.3 Database

The database used is MariaDB. MariaDB is a relational database derived from MySQL and developed by the open-source community. It is easy to install and use and offers good performance [3]. In this case, Docker was used to install the database on the server and run it in a so-called container. The database consists of three tables: `users`, `tierarten`, and `sichtungen`, where `tierart_id` and `user_id` are used as foreign keys in the `sichtungen` table. The ER diagram of the database is shown in Figure 1.

## 2.4 WebMap

A final component used in the app is a WebMap. This is retrieved directly from the frontend via Google Web Services and embedded into the app. To enable this, a Google developer account is required to create a WebMap, which can then be accessed using a MapID. Configurations regarding the appearance of the map can be made directly in the Google developer account.

## 2.5 Design

The design of the app is really quite simple. In general, it just uses the default styling of the Ionic framework with very minor adjustments. The only part where some effort was put in was the logo. The logo was designed by a fellow student and is shown in Figure 2. The idea is to vertically show the general purpose of recording geo-tagged data. Horizontally it symbolizes the data content of the app. So for future use cases, the logo can be easily adjusted to fit the new content.

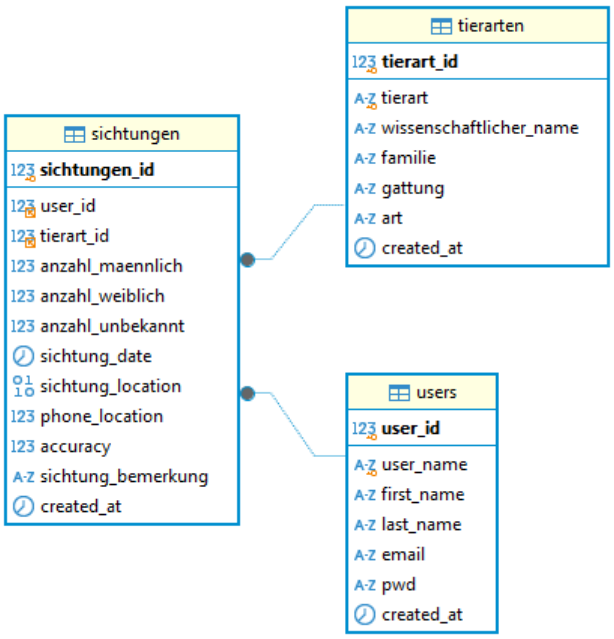


Figure 1: ER Diagram of the database

Figure 2: Logo for the app

## 3 Results

The app was successfully developed and deployed. It will be available for a few months after the submission of this report. It seems a bit pointless to describe the app in detail here, as it is available online and can be tested by anyone. This part of the report will therefore be kept short. The actual use of the app is supposed to be experienced by the user.

## 3.1 Web Application

To install the app on a smartphone just use the QR code in Figure 3 or visit the website https://wildtierapp.juliankraft.ch/app/. On opening the page a prompt will suggest the installation of the app. It works best using the Google Chrome browser. In order for the app to work, location services must be enabled on the smartphone and the app must be granted access to the location.

Figure 3: QR Code to install the app

## 3.2 Data View

In addition to the app, a data view was created to display the data collected by the app. It is publicly available and can be accessed at https://wildtierapp.juliankraft.ch/. This works best using the Google Chrome browser on a computer.

## 3.3 Data Access

This data view is only a frontend to display the data. In order to access the data for further processing, a direct database access is available. A Python script to access the data and a config file for a read-only user is available on the GitHub repository for the backend under `./db_setup/`. Feel free to use this script and config file to access the data for testing. However, the data is not valid since it was created for testing purposes only and anyone testing the app can add random data to the database.

## 3.4 Code Base

All the code for the app this document and the presentation is publicly available on GitHub:

- Frontend: https://github.com/juliankraft/WildtierSichtungsApp_front
- Backend: https://github.com/juliankraft/WildtierSichtungsApp_back
- Documentation: https://github.com/juliankraft/WildtierSichtungsApp_documentation

It is worth mentioning that the code is not perfect – at this point, it is not more than a working prototype. Still, the code is open source and can be used by anyone to do whatever under the creative commons license CC0 1.0 Universal.

# 4 Discussion

It seems that the technology to create a web application to record geo-tagged data is accessible to everyone with a bit more than basic programming knowledge, some help, and some insane persistence. The app was successfully developed and deployed. The data view was created and the data can be accessed via a direct database access. Still, it was a real struggle to get to this point, and the hours put into this project are countless. The technology is accessible, but the learning curve is steep. It is hard to determine if the accessibility is more a result of modern frameworks or the fast-improving AI tools like GitHub Copilot and ChatGPT.

The biggest challenge was to keep an overview of all the components and how they interact with each other. For every part of the app, there are countless files and modules somehow connected to each other. Especially the frontend was a real struggle. After the simple command `ionic start` and taking a few initial decisions, the project folder initiated by the framework is already filled with a lot of files and folders rather cryptic to a beginner. While AI is very strong in writing functions or limited parts of the code, it is not very helpful in keeping an overview of the whole project. That seems to be where the real challenge lies.

## 4.1 Next Steps

The App is, at this point, a working prototype. A first step would be to further clean up the code and make it more readable before adding more to it. There are a few things that would need some more work. An especially important one would be the spatial data and its accuracy. Questions to follow up on would be: Is there a way to maximize the accuracy of the phone location? How to handle the accuracy of the manually input location? Another important thing to work on would be the app control elements. While the idea was to keep the app as simple as possible, a bit more control would be nice. For example, the possibility to go back to the previous screen to change the input. A data view within the app could be a nice feature especially the option to delete data points in order for the user to correct mistakes.

In order for these things to be implemented and the project being taken further, an actual use case would be needed. It can be a bit frustrating to work on a project, intense like this one, without a real use case. With an actual use case, there would be a clear goal and therefore a clear path to follow. For example, the data structure would not be as generic and some real considerations there could improve the app. Another very important thing provided by a real use case would be an actual user group testing the app.

# Acknowledgments

I would like to express my gratitude to Ramon Ott – the experienced developer – for his help and support during this project. Without his help, this project would not have been possible.

I very much appreciate the help of my fellow student Annalisa Berger for her great work on the design of the logo for the application.

## AI Declarations

- The creation of this document was supported by ChatGPT and GitHub Copilot.
- AI played a significant role in the development of the prototype.
- The Introduction and the Methods sections were translated from German to English using ChatGPT since they were written before realizing this report had to follow the JOSIS template, specifically defining the language.
- The Abstract was initially generated by ChatGPT and then edited by the author.
- A complete spellcheck was performed using ChatGPT.

## References

[1] GO PROGRAMMING LANGUAGE. The Go Programming Language. https://go.dev/, 2024. Last Accessed December 17, 2024.

[2] IONIC FRAMEWORK. Ionic Framework - The Cross-Platform App Development Leader. https://ionicframework.com/, 2024. Last Accessed December 17, 2024.

[3] MARIADB FOUNDATION. MariaDB Foundation. https://mariadb.org/, 2024. Last Accessed December 17, 2024.