# ECODAT: A WEB-BASED APPLICATION AND DATABASE FOR LIMNOLOGICAL MONITORING DATA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

Ali DEĞERMENCİ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
BIOTECHNOLOGY

SEPTEMBER 2022

Approval of the thesis:

**ECODAT: A WEB-BASED APPLICATION AND DATABASE FOR LIMNOLOGICAL MONITORING DATA**

submitted by **Ali DEĞERMENCİ** in partial fulfillment of the requirements for the degree of **Master of Science** i**n Biotechnology, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences**

Assoc. Prof. Dr. Yeşim Soyer
Head of the Department, **Biotechnology**

Assoc. Prof. Can Özen
Supervisor, **Biotechnology, METU**

**Examining Committee Members:**

Prof. Dr. Meryem Beklioğlu
Biological Sciences, METU

Assoc. Prof. Can Özen
Biotechnology, METU

Prof. Dr. Ilgaz Akata
Biology., Ankara Uni.

Date: 23.09.2022

## ABSTRACT

## ECODAT: A WEB-BASED APPLICATION AND DATABASE FOR LIMNOLOGICAL MONITORING DATA

Değermenci, Ali
Master of Science, Biotechnology
Supervisor : Assoc. Prof. Can Özen

September 2022, 43 pages

This study is focused on the elaboration of database application solutions for saving measured data in limnological monitoring. The purpose of the study is to give background information about monitoring data and the creation of a smart-accessible database system using query language and database management system Microsoft SQL Server 2012 while also providing a web application where researchers can reach, interact and save their fieldwork using C# .NET Core MAUI/Blazor framework. The thesis includes a database solution for saving measured data web application for creating graphs with dynamic queries.

Keywords: Limnology, Web application, Database, Blazor, .NET Core

# ÖZ

## ECODAT: LİMNOLOJİK İZLEME VERİLERİ İÇİN WEB TABANLI BİR UYGULAMA VE VERİTABANI

DEĞERMENCİ, Ali
Yüksek Lisans, Biyoteknoloji
Tez Yöneticisi: Doç. Dr. Can Özen

Eylül 2022, 43 sayfa

Bu çalışma, limnolojik izlemede ölçülen verileri kaydetmek için veritabanı uygulama çözümlerinin detaylandırılmasına odaklanmıştır. Çalışmanın amacı, sorgulama dili ve veritabanı yönetim sistemi Microsoft SQL Server 2012 kullanılarak verilerin izlenmesi ve akıllı erişimli veritabanı sisteminin oluşturulması hakkında arka plan bilgisi vermek ve ayrıca araştırmacıların alan çalışmalarında kullanabilecekleri ve etkileşim kurabilecekleri C# .NET Core MAUI/Blazor tabanlı bir web uygulaması sağlamaktır. Tez, ölçülen verileri kaydetmek için veritabanı çözümünü, dinamik sorgularla grafikler oluşturmak için web uygulamasını içerir.

Anahtar Kelimeler: Limnoloji, Web uygulaması, Veritabanı, Blazor, .NET Core

# ***Apeiron One***

## ACKNOWLEDGMENTS

I am grateful to my dear advisor Assoc. Prof. Can Özen his most valuable guidance and support. He always welcomed me whenever I ran into trouble. Throughout my master's degree, he consistently steered me in the right direction whenever he thought I needed it.

I could not thank Prof. Dr. Meryem Beklioğlu enough for giving me the opportunity to work with her lab. Her unprejudiced, open-minded, and progressive mindset encouraged me to follow my ideas.

I am grateful to the Limnology group and everyone who has come to study in this group since 1997. Their hard work created the Lake Eymir dataset.

I am thankful to Ph.D. candidate Lect. Gülce Yalçın. She helped me immensely throughout the thesis. Her door was always open whenever I had a question about my research. But more importantly, she tolerated my unorthodox mindset and questions.

I am highly indebted to my dear old friend Bioengineer Funda Can, who encouraged me to take a step and keep going. She was there through thick and thin. Without her precious support, I would not be here, and this thesis could not be written.

I want to thank my dear, overly sporty friend Mehmet Can Gürel. His calm but motivating attitude gave me the strength to fight back and take control of my body.

I am very grateful for being here in METU because it brought you, Sena Ezgin, into my life. It would not be the same without your sensible, receptive, and mindful voice, and I would be lost without your guidance and unconditional love. Thank you for being there. Finally, to the woman who gave me her endless patience and love, who raised me to be the man I am now, for which my mere expression of thanks does not suffice. I thank you, mother.

# TABLE OF CONTENTS

## LIST OF FIGURES

# LIST OF ABBREVIATIONS

| ABBREVIATIONS | |
|---|---|
| **DIN** | Dissolved inorganic nitrogen |
| **TP** | Total Phosphorus |
| **SD** | Sechhi depth |
| **DO** | Dissolved oxygen |
| **HTTP** | Hypertext Transfer Protocol |
| **TCP** | Transmission Control Protocol |
| **MVC** | Model View Controller |
| **UI** | User Interface |
| **SQL** | Structured Query Language |
| **MAUI** | Multi-Platform Application UI |

# CHAPTER 1

# INTRODUCTION

## 1.1 Lake Eymir

Turkey's Lake Eymir is positioned approximately 20 kilometers south of Ankara at an elevation of 970m on the Anatolian plateau (39°57'N, 32°53'E) with a varying surface area between 100-130 ha and a mean depth of 3.2m(Beklioğlu et al., 2017). Lake Eymir is a biomanipulated lake, and there are two inflows into the lake. The main inflow comes from Lake Mogan, which is upstream of Lake Eymir, while the second inflow is located at the northern end of the lake and is called Kişlakçı. The second inflow only occurs during spring time, and it's associated with melting snow.

Before biomanipulation, Lake Eymir was in a clear water state with submerged and emergent plants. Sadly, between 1970 and 1995, Lake Eymir received raw sewage effluents from a nearby town called Gölbaşı, located between Lake Eymir and Lake Mogan. This, in turn, led to the eutrophication of the lake. In 1994 the sewage effluent was diverted to Imrahor Valley using a bypass channel(Beklioglu et al., 2000). The bypass channel helped lower the dissolved inorganic nitrogen (DIN) and Total Phosphorus (TP), but even after the diversion, Eymir remained rich in DIN and especially TP(Beklioglu et al., 2003).

In order to lower the biomass of common carp (*Cyprinus carpio*) and tench (*Tinca tinca*) in Lake Eymir, two phases of biomanipulation were carried out from April to October in two different time periods. Local fishers removed fish throughout the week by utilizing multiple-mesh gill nets. This first lake restoration project was conducted between 1998-1999 and aimed to boost the population of piscivorous fish; hence pike(*Esox* lucius) fishing was prohibited(Beklioglu et al., 2003). 30% of the total tench and carp population was removed, resulting in a doubling of Secchi depth(SD) shown in Figure 1.1

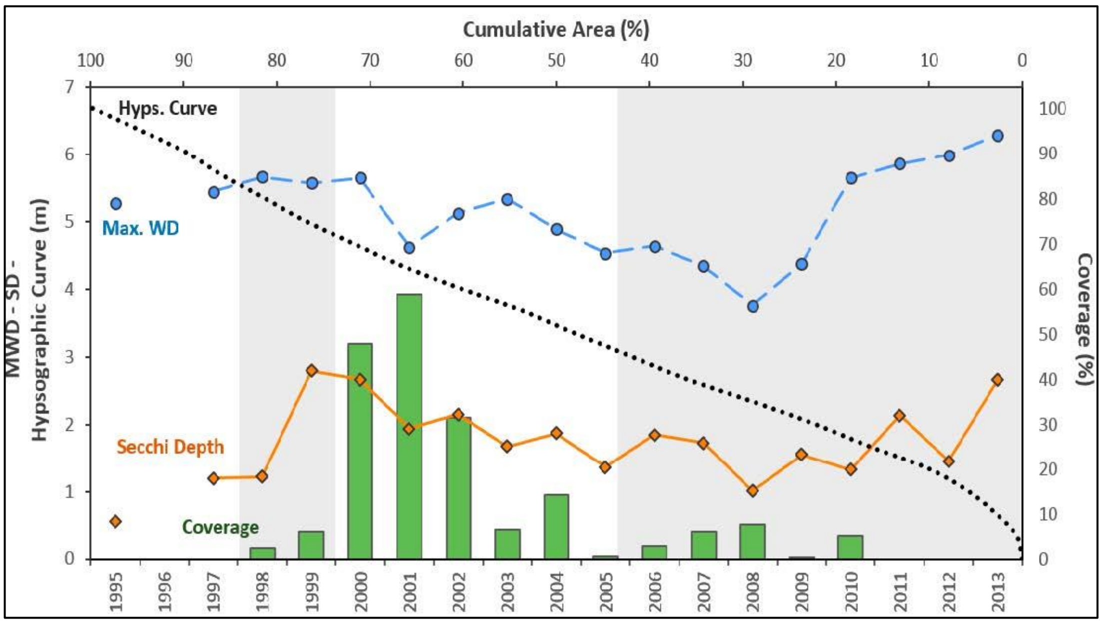


Figure 1.1. Changes in maximum water depth (MWD), Secchi depth (SD), hypsographic curve, and % submerged macrophyte coverage in Lake Eymir (shading indicates the first and second biomanipulation).(Taken from (Beklioğlu et al., 2017))

The biomanipulation of Lake Eymir led to a significant improvement in the water clarity and the recolonization of macrophytes. The recuperation, though, was just temporary. Conditions largely reverted to their pre-biomanipulation level in the drought season between 2003 and 2009, particularly between 2004 and 2005. A reduction in water level and already high nutrient contents were likely to blame for this severe decline(Özen et al., 2010). After this decline, the second biomanipulation plan was initiated between late 2005 to late 2013.

## 1.2 Software

### 1.2.1 Web Environment

The web environment is a stateless environment based on client-server network architecture. This architecture works in such a way that the client sends HTTP requests to which the server sends HTTP responses. From this principle, the server does not know about the client until the client contacts it, and thus the only possibility of communication between the server and the client is through the client's request, to which the server sends a response. The specific feature of this architecture is the client's dependence on the data provided by the server. Figure 1.2 shows the working diagram of the web application with the exchange of data between the server and the client.

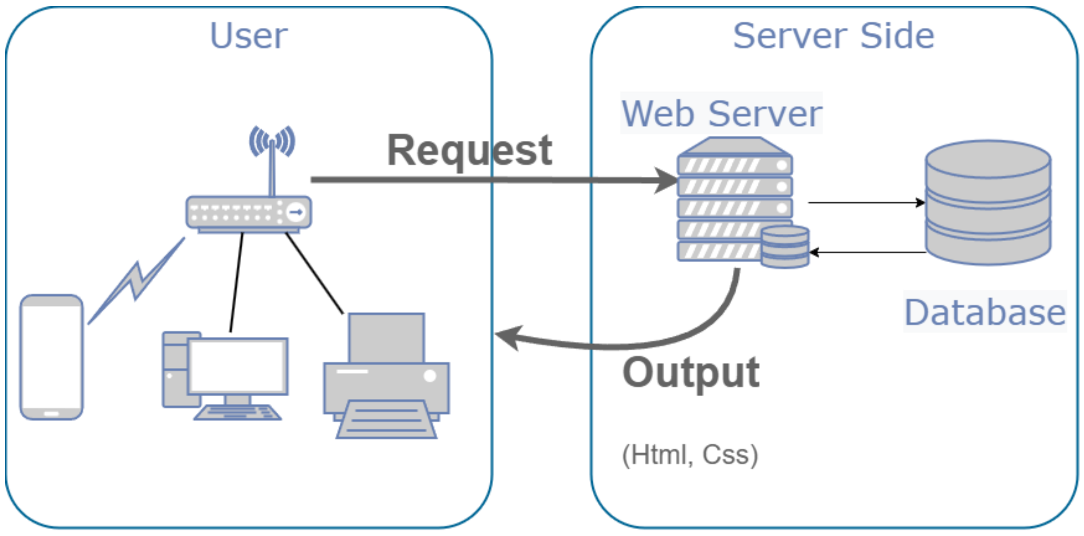


Figure 1.2. Functional diagram of a classic web application

The original idea, where the server "only" provided data that the client displayed in an HTML document, was overcome thanks to client scripting languages, such as JavaScript or VBScript, and more advanced application logic began to be implemented on the client side, which enabled the application to interact with the user. The logic and code of the application are therefore divided into two main parts, namely the server part and the client part, where each fulfills its specific role. This division introduced additional complexity into the development, which many web applications solve using the MVC architecture.

### 1.2.2 HTTP

The HTTP protocol is a stateless protocol that is the basis for communication on the web. It works on the client-server principle, where the client sends an HTTP request to the server, and the server responds with an HTTP response. There are four versions of the protocol: HTTP/0.9, HTTP/1.0, HTTP/1.1, and HTTP/2.0

Originally a single TCP connection was created for each HTTP request. This approach was very inefficient because each new TCP connection caused an unnecessary load. HTTP/1.1 came with the idea of sharing a single TCP connection for multiple HTTP requests. In practice, it recorded the so-called pipe HTTP requests that were blocking. This request pipe functioned as a stack, meaning that the second HTTP request got to the queue only after the first HTTP request was served.

This approach meant that it was advantageous to have as few files as possible in the web application4. Files of the same type (*.js, *.css) were combined into one file called bundle using a technique called bundling.

HTTP/2 brought more practical improvements that enabled web applications to load faster, such as;(Factory.hr, 2019)

Multiplexing (Multiple Request) - multiple HTTP requests through one TCP connection.

- Server Push - if the user sends an HTTP request for a, for example, an HTML file, the server knows that the user will request JavaScript and CSS files after receiving the HTML file, and therefore it sends these files together with the HTML document. Of course, the client has the option to reject these files.
- Binary protocol - instead of the textual representation of commands, their binary representation is used. Binary representation means a simpler implementation of HTTP on the server and client side. It is also more resistant to errors (the text form must solve the escaping and encoding of control characters). There is also less overhead when parsing data and greater efficiency in the use of network resources.

The HTTP/2 specification does not require an encrypted connection (with TLS), but in practice, all browsers for HTTP/2 require it. (*HTTP/2 Frequently Asked Questions*, n.d.; *What Is the HTTP/2 Protocol?*, n.d.)

### 1.2.3 Model View Controller (MVC) architecture

Architectures are used in software development because they make it possible to standardize approaches to solving a given problem and thereby facilitate and make work on a given project easier and more efficient. A project that uses an architecture is better testable and enables the separation of responsibility between individual layers. With this division, the project has a more sustainable development. (Deacon, 2009)

The complexity of the UI increased, and thus the complexity of the presentation layer also increased. This increase resulted in various modifications to the MVC architecture. These MVC-based modifications mostly contain three parts, namely View, Model, and Controller.

Certain modifications of this architectural pattern have been defined with their own name to avoid confusion with existing modifications. The view is a UI representation of data. The model is the data displayed by the View. The Controller is the brain that contains the application logic and takes care of creating the View and the Model.

The entry to the MVC architecture is through the Controller. The controller receives all requests (inputs) and contains the logic for their processing. Based on this logic, it creates a Model that contains the data. The Controller also takes care of initializing the View, which provides a Model with data. The controller can provide different views based on the input arguments of the request.

There are mainly two types of MVCs;

Client MVC

- View => It is an HTML document rendered by the browser and seen by the user.
- Controller => It is responsible for loading data for View and transforming it into DOM6. It is written in JavaScript.
- Model => It is understood as data or data source (webserver).

Server MVC

- View => It is a file that is returned as a response to a client's HTTP request = it can be a file or only an HTTP response.
- Controller => It is responsible for processing the client's HTTP request and creating the subsequent HTTP response.
- Model => It is understood as data.

In practice, this architecture works as follows: The user enters the URL of the web application into the browser. The browser sends a request GET, and the web server returns an HTML document to it. When loading an HTML document, the browser sends additional GET requests to the JavaScript files that are needed. After loading the HTML document and all the scripts, the client controller is initialized from JavaScript. This Controller makes an HTTP request (for example, POST) and sends it to the web server. The web server now acts as a proxy and redirects this request to the server Controller, which processes it and returns an HTTP POST response with a JSON file that contains data for the web application. The Client Controller processes this response and edits the HTML document using the DOM API, and the browser's Render Engine takes care of re-rendering the given web application for the user. (Krasner et al., n.d.; Leff & Rayfield, 2001)

### 1.2.4 Microsoft SQL Server

The origins of MS SQL began in 1985 when Microsoft and IBM announced their upcoming collaboration in the development of software and operating systems. In 1992, Microsoft achieved huge success with the Microsoft Access database product. Microsoft approached Sybase and released the first version of MS SQL Server, SQL Server 1.0, in 1989. It competes not only with Sybase SQL Server but also with Oracle and IBM. The first version in which Microsoft did not use the help of a foreign company was the version of SQL server 6.0 in 1995.

According to (Natarajan et al., 2015), thanks to its properties, robustness, and reliability, Microsoft SQL Server is destined for "mission-critical" applications while also significantly reducing the demands on the infrastructure and its management.

Microsoft SQL Server is comprehensive database software. It covers the needs of organizations in the field of reliable data management and maintenance with the possibility of using integrated transformation and analytical capabilities or extended reporting services.

Like other database management systems, MS SQL Server is now based on SQL as a standardized programming language. SQL Server specifically uses Transactional-SQL, which is SQL extended by a set of custom programming extensions compared to the standard. Originally, as written above, the SQL Server code was developed in the 1980s by Sybase Inc., which is now owned by SAP. (AG, n.d.)

Between 1995 and 2018, Microsoft released eleven versions of SQL Server. The earlier versions were primarily aimed at applications for work in departments and workgroups, but later Microsoft expanded the capabilities of SQL Server and reshaped it to be able to function as an enterprise relational database management system, which is able to compete with for example, Oracle. Over the following years, Microsoft added additional functionality to SQL Server to support new technologies that include support for websites, cloud computing, and mobile devices. (Stonebraker, 2010)

In 2016, SQL Server 2016 was released and developed as part of the "mobile first, cloud first" technology strategy adopted by Microsoft two years earlier. Among other things, with SQL Server 2016 came new features like performance tuning, real-time traffic analysis, and data visualization. Support for so-called big data analysis and other advanced analysis applications has also been added through SQL Server R services, which enable the database management system to run analysis applications written in the R programming language.

#### 1.2.4.1 MS SQL Server architecture

Like other database management system technologies, SQL Server is primarily built on a row-based table structure that links related data elements between different tables, eliminating the redundant need to store data in multiple locations within the database. The relational database model also provides referential integrity and other integrity bindings that ensure data accuracy.

A key component of Microsoft SQL Server is the SQL Server Database Engine, which manages data storage, query processing, and security. It contains a relational "engine" that handles commands and queries and a storage "engine" that handles database files, tables, pages, indexes, buffers, and transactions. Stored procedures, triggers, reports, and other database objects are also created and executed via the Database Engine. (Libkin, 2003; Melton, 1996)

Microsoft kept its support for SQL servers for over 33 years and continued to upgrade the software, and fixed introduced bugs well after its life cycle. The reliability of the Microsoft brand and the performance of the software had key roles when choosing it for EcoDat

### 1.2.5 ASP.NET and ASP.NET Core

ASP.NET was born from ASP technology, which, after the addition of .NET runtime 7, turned dynamic scripting ASP into a statically typed technology that made it possible to write pages in C# or VB.NET languages.

ASP.NET brought Web Forms, which made it possible to develop web applications in a desktop style and while the Web Forms generated required JavaScript quite successfully. There was quite a lot of criticism of Web Forms technology due to the fact that it obscured the natural stateless environment of the web, and therefore many web applications were written in rather poor quality. Web Forms covered the stateless environment of the web using the ViewState concept that maintained the state of the web application. (Rick-Anderson, n.d.-a)

ASP.NET Core is a modern reimplemented version of ASP.NET that uses the open-source, cross-platform implementation of .NET Core instead of the .NET Framework. In this project, .NET MAUI, which is based on ASP.NET Core, was used. (Price, 2019; Rick-Anderson, n.d.-b)

### 1.2.6 ASP.NET Core Blazor

For .NET developers, there are many options for full-stack web application development. Some technologies, such as Silverlight or Web Forms, are no longer officially supported by Microsoft, so the community has started to develop different alternatives.

Microsoft did not develop any front-end framework after the WebForms technology. But after long years and patients from the community, in 2018, Microsoft released the experimental full stack framework Blazor. Shortly after that, on April 18, 2019, a release version was released with official support from Microsoft. (guardrex, n.d.; Joshi, 2019)

### 1.2.7 .NET MAUI

The .NET Multi-Platform Application UI (.NET MAUI) is a cross-platform framework that allows developers to create native mobile and desktop applications in C# and XAML.

From a single shared codebase, developers can use.NET MAUI to create applications that work on Android, iOS, macOS, and Windows.

.NET MAUI is an open-source project that evolved from Xamarin. Forms with totally redesigned UI controls for performance and extensibility from mobile to desktop scenarios. There are numerous parallels between .NET MAUI and Xamarin.Forms. The core difference of this framework is that developers can create multiplatform applications with.NET MAUI using a single project but can also add platform-specific source code and resources if necessary. The ability to build as much of the application logic and UI layout in a single codebase is one of the key goals of.NET MAUI.(davidbritch, n.d.; Vermeir, 2022)

This project is developed by the newly released functionality of .Net MAUI, which could combine the Blazor environment with MAUI, turning the EcoDat application into a cross-platform app that can function as a website too.

## 1.3 Aim of the study

After studying the issues field research is dealing with, this thesis aimed to develop an appliable web application solution for researchers in the field and simultaneously provide a hub for field data to gather. The study also aims to improve data harmonization, cleaning, and standardizing for measured data in limnological monitoring. The goal of the study is also to give an alternative way for the field researcher to upload otherwise scattered files, such as photos, notes, etc. Creation of smart-accessible query creation for the data management system Microsoft SQL Server 2012.

# CHAPTER 2

# MATERIAL AND METHODS

## 2.1 Data Collection

Throughout the study period, manual data collection and measurements were carried out on-site. During the growing season, measurements were taken every two weeks. During the ice-covered period, no sampling was done. The field study template provided by Prof. Dr. Meryem Beklioğlu was used as guidance while gathering the data.

Before the summer of 2022, each digital sampling was taken with a YSI 556 MPS multi-probe field meter (YSI Incorporated, Yellow Springs, OH, USA) providing measurements of water temperature ($^{o}$C), dissolved oxygen concentration (mg/l), salinity (PSU), conductivity (mS), total dissolved solids (g/l), and pH. Later this digital sampling was done with  Aqua TROLL 600 Vented(In-Situ), which provided us with Actual Conductivity (µS/cm), Specific Conductivity (µS/cm), Salinity (PSU), Resistivity (Ω·cm), Density (g/cm$^3$), Total Dissolved Solids (ppt), Turbidity (NTU), pH (pH), pH mV (mV), ORP (mV), RDO Concentration (mg/L), RDO Saturation (%Sat), Oxygen Partial Pressure (Torr), Temperature (°C), Barometric Pressure (mm Hg), Pressure (psi), Depth (ft), Depth to Water (ft), External Barometric Pressure (mbar), AirTemperature (°C), Latitude-Longitude (°). These measurements were taken with 0.5m intervals from the surface through the water column.

Design of the database was based on variables of these measurements.

## 2.2 Hardware

For this application, a local machine was used to develop and store the database. For a test environment, virtualization is taken advantage of. Virtualization allows one physical device to be used for multiple virtual servers. Windows Server 2012 R2 operating system in the Standard edition or Datacenter (in earlier versions of Windows Server Enterprise) installed on a virtual server. The connection was made via built-in Hyper-V. One crucial prerequisite is to check the firewall exceptions on the machine and allow port 1433 for SQL Server communications.

To connect the SQL Server database following connection string was used;

```
$"Data Source=DESKTOP-E3C8TEN; database={srvrdbname}; User ID={srvrusername};Password={srvrpassword};Encrypt=true;TrustServerCertif
icate=true";
```

In the following section, the basic commands of the SQL language are presented, which are used for searching, editing, deleting, or other operations with database data.

**SELECT**

Used to extract data from the table. It has several parameters, both mandatory and optional, which the subsequent data listing can be specified and limited, or sorted in descending or ascending order.

SELECT [column names] FROM [table names] [JOIN TYPE] JOIN [join condition] WHERE [restrictive conditions] GROUP BY [list of fields to group by] HAVING [selection criteria] ORDER BY [list of fields to sort by]

The command above must contain only the SELECT and FROM statements. Other terms are optional.

**INSERT INTO**

Using this command, the user can insert data into the tables.

INSERT INTO [column names] VALUES [inserted values]

**DELETE FROM**

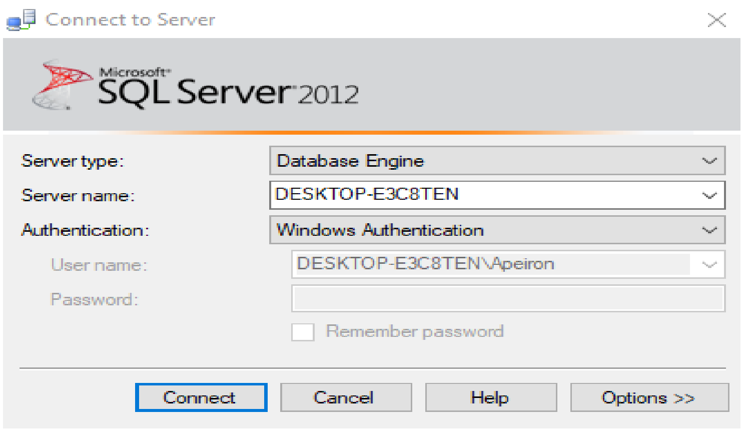
Used to delete records. Either individual records or entire tables.

DELETE FROM [table] WHERE [ID of the record the user wants to delete]

**UPDATE**

Using this command, the user can edit existing records in the database.

UPDATE [table] SET [column name] = [column description] WHERE [ID of being edited record] = [specific id]



Server type: Database Engine
Server name: DESKTOP-E3C8TEN
Authentication: Windows Authentication
User name: DESKTOP-E3C8TEN\Apeiron
Password:
Remember password

Figure 2.1. SQL Server connection

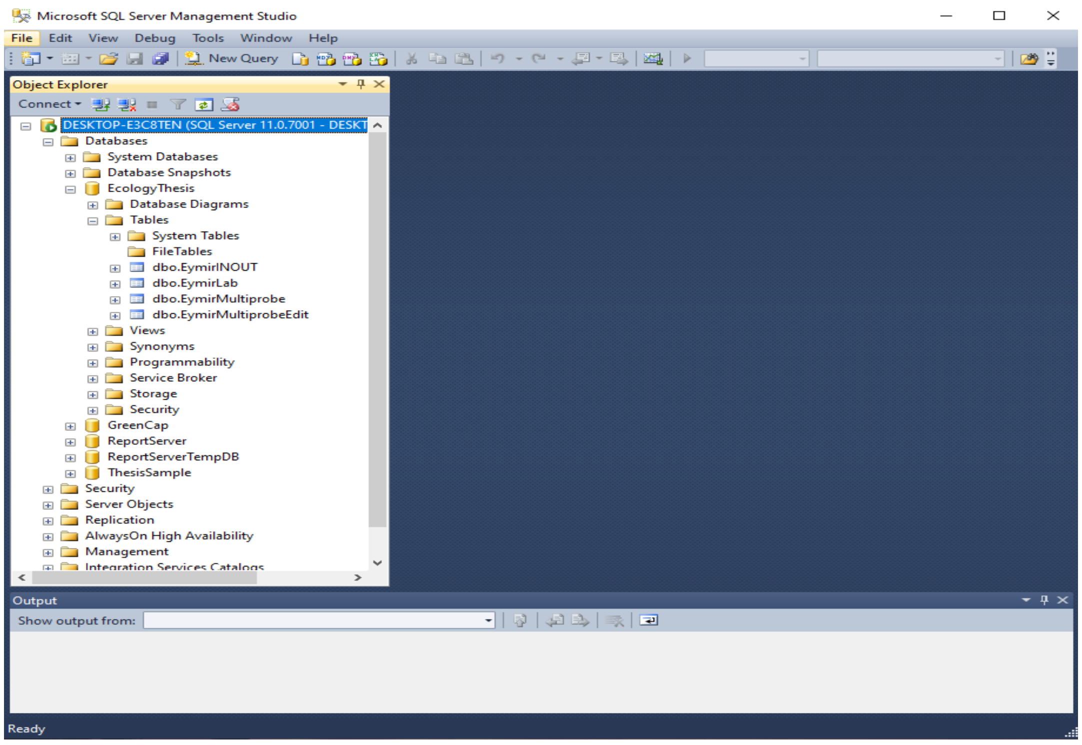
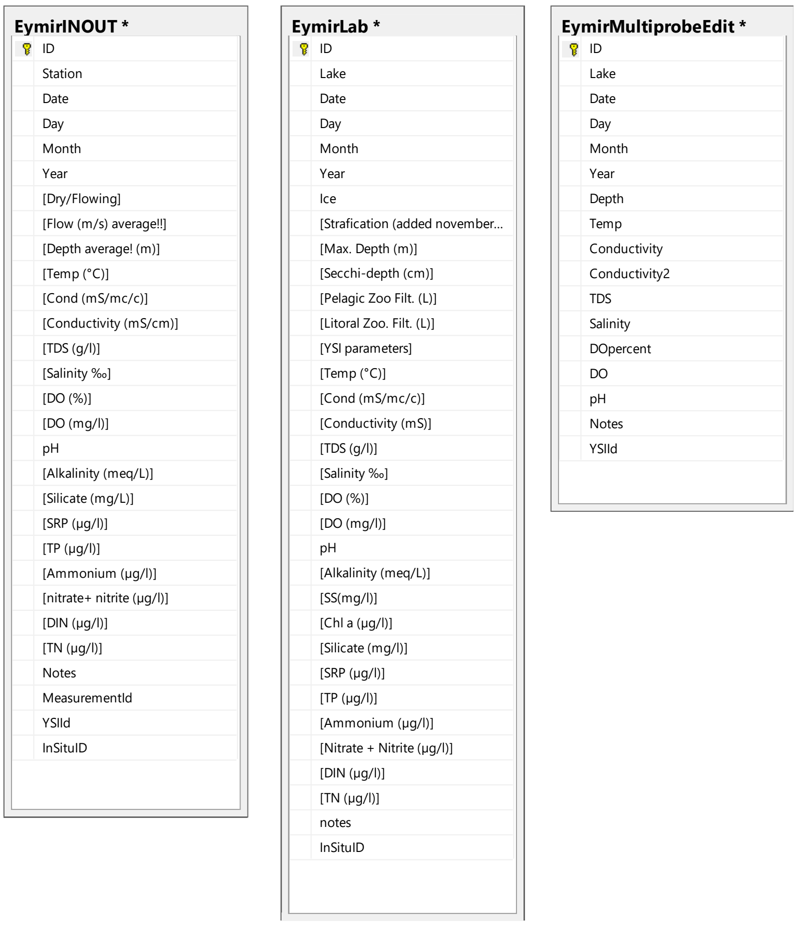


Figure 2.2. MS SQL Server interface


**EymirINOUT ***

ID
Station
Date
Day
Month
Year
[Dry/Flowing]
[Flow (m/s) average!!]
[Depth average! (m)]
[Temp (°C)]
[Cond (mS/mc/c)]
[Conductivity (mS/cm)]
[TDS (g/l)]
[Salinity ‰]
[DO (%)]
[DO (mg/l)]
pH
[Alkalinity (meq/L)]
[Silicate (mg/L)]
[SRP (µg/l)]
[TP (µg/l)]
[Ammonium (µg/l)]
[nitrate+ nitrite (µg/l)]
[DIN (µg/l)]
[TN (µg/l)]
Notes
MeasurementId
YSIId
InSituID

**EymirLab ***

ID
Lake
Date
Day
Month
Year
Ice
[Strafication (added november...
[Max. Depth (m)]
[Secchi-depth (cm)]
[Pelagic Zoo Filt. (L)]
[Litoral Zoo. Filt. (L)]
[YSI parameters]
[Temp (°C)]
[Cond (mS/mc/c)]
[Conductivity (mS)]
[TDS (g/l)]
[Salinity ‰]
[DO (%)]
[DO (mg/l)]
pH
[Alkalinity (meq/L)]
[SS(mg/l)]
[Chl a (µg/l)]
[Silicate (mg/l)]
[SRP (µg/l)]
[TP (µg/l)]
[Ammonium (µg/l)]
[Nitrate + Nitrite (µg/l)]
[DIN (µg/l)]
[TN (µg/l)]
notes
InSituID

**EymirMultiprobeEdit ***

ID
Lake
Date
Day
Month
Year
Depth
Temp
Conductivity
Conductivity2
TDS
Salinity
DOpercent
DO
pH
Notes
YSIId

Figure 2.3. Our initial DB data fields

## 2.3 Web Application

C# source codes are divided into projects. The projects are then combined into a solution which is divided into two projects and one razor library. (Figure 2.4) The razor class library is called RazorLibrary, which contains all elements that can be used and extended to other systems. Most of the interfaces, services, components, utilities, and model data are located in the RazorLibrary project. RazorLibrary, therefore, consists of elements from all three layers and has the role of a common library. The second project, called BlazorServer, is the main project that is translated into an executable application of the web. It imports the RazorLibrary library and contains more specific and more difficult-to-reuse code that sets up and uses the generic library. For example, webview logic, razor codes for a specific page, and web templates are implemented there. The RazorLibrary acts as a hub and takes care of things like authentication, document management, project management, and user management, and offers many interfaces that can or must be implemented by the main project.

Since the number of interfaces, even on a work-in-progress project, is not so small, and it is assumed that it will continue to grow, it makes sense to create a facade over the entire set of interfaces, services, and registrations. A facade is a design pattern whose task is to provide a simpler interface for using a more complex system of services. This method is widely used across the entire ASP NET Core framework. By calling various methods, the necessary services are registered in the background with the container and are automatically configured based on the parameters of the methods.

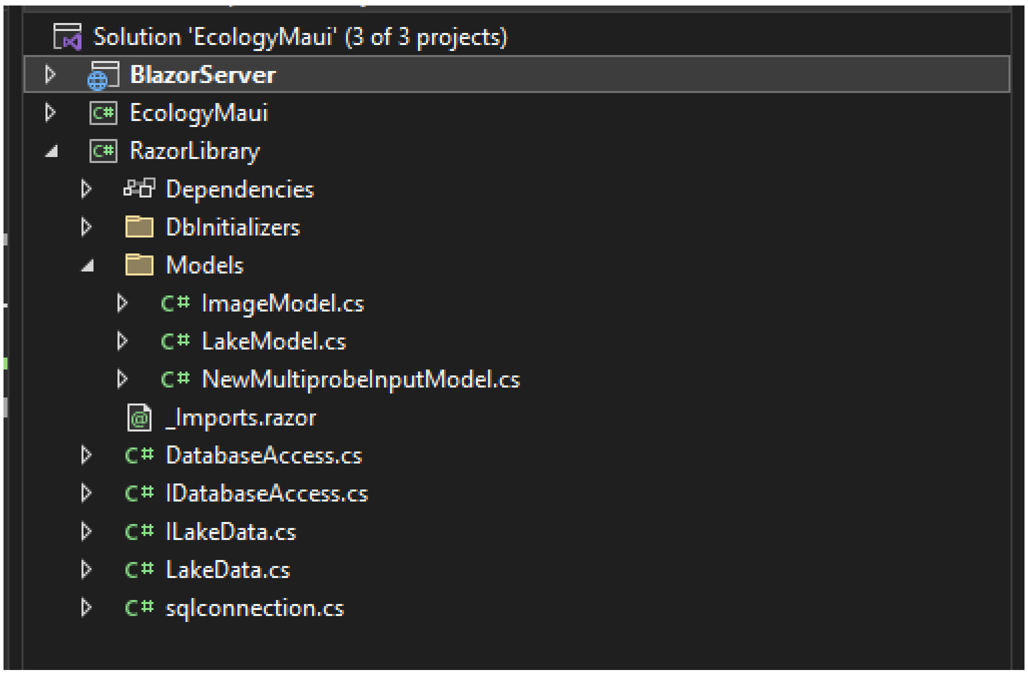
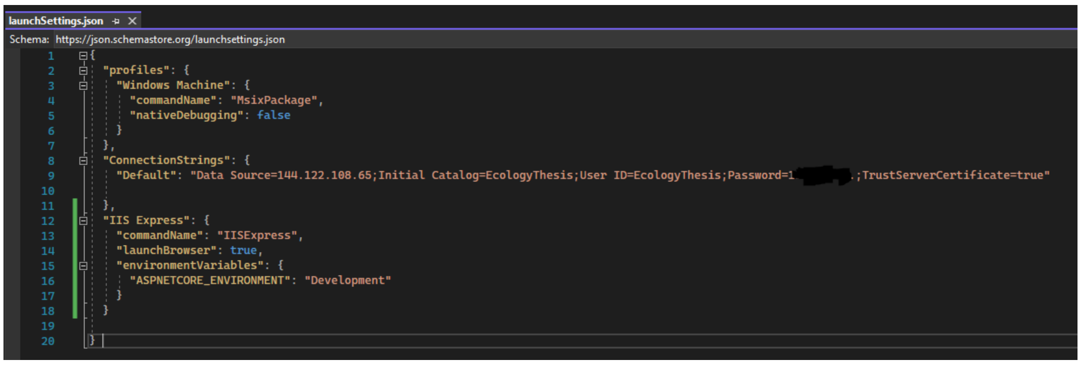

Solution 'EcologyMaui' (3 of 3 projects)
- BlazorServer
- EcologyMaui
- RazorLibrary
  - Dependencies
  - DbInitializers
  - Models
    - ImageModel.cs
    - LakeModel.cs
    - NewMultiprobeInputModel.cs
  - _Imports.razor
  - DatabaseAccess.cs
  - IDatabaseAccess.cs
  - ILakeData.cs
  - LakeData.cs
  - sqlconnection.cs

Figure 2.4. Projects in the same solution of the app

The values of configuration variables, such as the SMTP server address or database credentials, should be kept separate from the application source code so that they can be easily modified when the runtime environment changes. ASP.NET Core uses the options pattern to load these values, which allows you to load configurations from various sources directly into strongly typed C# classes. Thanks to this pattern, services can depend only on the configurations that concern them (Interface segregation principle), and at the same time, the configurations for different parts of the application are independent of each other.

All system configuration is stored in RazorLibrary for general use and launchSettings.json for android development under the EcologyMaui project. Switching between these environments is done by the ASPNETCORE_ENVIRONMENT environment variable, which can be set when the process is started on the operating system. ASP.NET Core takes care of selecting the configuration file according to the current environment, deserializing the data, and mapping the various sections of the configuration to specific objects in the application. Using the data injection technique from the models, these objects can be inserted into the services that need them. The programmer only specifies which sections to map to which classes; the application framework takes care of the rest.

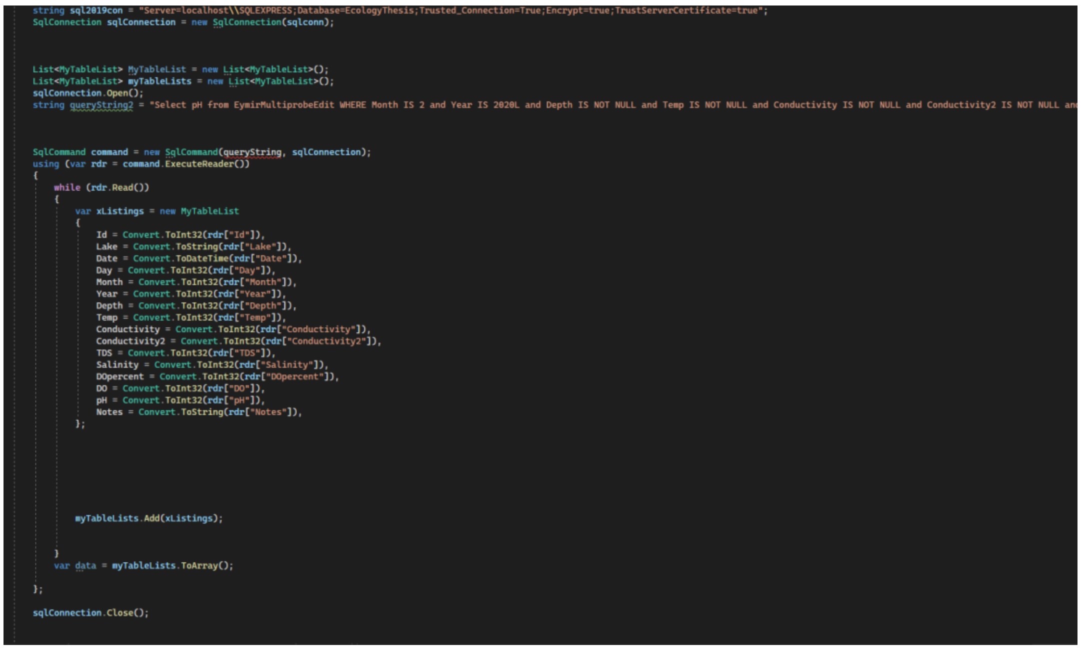


```json
{
  "profiles": {
    "Windows Machine": {
      "commandName": "MsixPackage",
      "nativeDebugging": false
    }
  },
  "ConnectionStrings": {
    "Default": "Data Source=144.122.108.65;Initial Catalog=EcologyThesis;User ID=EcologyThesis;Password=1[redacted];TrustServerCertificate=true"

  },
  "IIS Express": {
    "commandName": "IISExpress",
    "launchBrowser": true,
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  }

}
```

Figure 2.5. launchSettings.json

Database access was managed by DatabaseAccess.razor method under RazorLibrary, which helps to keep individual methods separate from private calls.

Common calls from the database can be made with a simple Microsoft.Data.SqlClient interaction with the code is shown in Figure 2.6. Data call from SQL database using C# syntax.



```
string sql2019con = "Server=localhost\\SQLEXPRESS;Database=EcologyThesis;Trusted_Connection=True;Encrypt=true;TrustServerCertificate=true";
SqlConnection sqlConnection = new SqlConnection(sqlconn);


List<MyTableList> MyTableList = new List<MyTableList>();
List<MyTableList> myTableLists = new List<MyTableList>();
sqlConnection.Open();
string queryString2 = "Select pH from EymirMultiprobeEdit WHERE Month IS 2 and Year IS 2020L and Depth IS NOT NULL and Temp IS NOT NULL and Conductivity IS NOT NULL and Conductivity2 IS NOT NULL an


SqlCommand command = new SqlCommand(queryString, sqlConnection);
using (var rdr = command.ExecuteReader())
{
    while (rdr.Read())
    {
        var xListings = new MyTableList
        {
            Id = Convert.ToInt32(rdr["Id"]),
            Lake = Convert.ToString(rdr["Lake"]),
            Date = Convert.ToDateTime(rdr["Date"]),
            Day = Convert.ToInt32(rdr["Day"]),
            Month = Convert.ToInt32(rdr["Month"]),
            Year = Convert.ToInt32(rdr["Year"]),
            Depth = Convert.ToInt32(rdr["Depth"]),
            Temp = Convert.ToInt32(rdr["Temp"]),
            Conductivity = Convert.ToInt32(rdr["Conductivity"]),
            Conductivity2 = Convert.ToInt32(rdr["Conductivity2"]),
            TDS = Convert.ToInt32(rdr["TDS"]),
            Salinity = Convert.ToInt32(rdr["Salinity"]),
            DOpercent = Convert.ToInt32(rdr["DOpercent"]),
            DO = Convert.ToInt32(rdr["DO"]),
            pH = Convert.ToInt32(rdr["pH"]),
            Notes = Convert.ToString(rdr["Notes"]),
        };




        myTableLists.Add(xListings);


    }
    var data = myTableLists.ToArray();

};

sqlConnection.Close();
```

Figure 2.6. Data call from SQL database using C# syntax.

Entities in the database usually have a primary key by which they are uniquely identified. It is typical to use an integer value that automatically increases as the number of records increases (autoincrement). By utilizing these unique integer keys, one can distinguish individual data from different data.

Access starts when the Blazor code within the page which was injected with the DatabaseAccess method by using "@inject IDatabaseAccess" data calls of async task on any predetermined event. Figure 2.7. Blazor code running on the page shows the OnInitializedAsync task, which is executed right after the call for the page happens (When the user clicks on the page). This task calls for GetLake() function within the LakeData method, which then acts as an intermediatory and returns another call for a task called  LoadData, and sends the required parameters. This double-step process seems unnecessary, but it is crucial for both security and performance reasons.

```
    protected override async Task OnInitializedAsync()
    {

        lake = await _db.GetLake();
        annotationFontColor = NavigationManager.Uri.Contains("dark") || theme ==
Theme.HighContrast ? "#E9ECEF" : "#343A40";

    }
```

Figure 2.7. Blazor code running on the page

```
namespace RazorLibrary
{
    public class DatabaseAccess : IDatabaseAccess
    {
        private readonly IConfiguration _config;

        public string ConnectionStringName { get; set; } = "Default";
        public DatabaseAccess(IConfiguration config)

        {
            _config = config;

        }
        public async Task<List<T>> LoadData<T, U>(string sql, U parameters)
        {
            string srvrdbname = "EcologyThesis";
            string srvrname = "144.1**.1**.*5";
            string srvrusername = "EcologyThesis";
            string srvrpassword = "******";

            string sqlconn = $"Data Source=DESKTOP-
E3C8TEN;database={srvrdbname};User
ID={srvrusername};Password={srvrpassword};Encrypt=true;TrustServerCertificate=tru
e";
            string sqlconn2 = $"Data Source=DESKTOP-
E3C8TEN;database={srvrdbname};User
ID={srvrusername};Password={srvrpassword};Encrypt=true;TrustServerCertificate=tru
e";

            string sql2019con =
"Server=localhost\\SQLEXPRESS;Database=EcologyThesis;Trusted_Connection=True;Encr
ypt=true;TrustServerCertificate=true";

            using (IDbConnection database = new SqlConnection(sql2019con))
            {
                database.Open();
                var data = await database.QueryAsync<T>(sql, parameters);

                database.Close();
                var x = data.ToList();
```

Figure 2.8. DatabaseAccess method

# CHAPTER 3

# RESULTS

## 3.1 Description of EcoDat Application

In this section, the main functions of EcoDat are explained. EcoDat application is based on the .Net 6 framework and .Net Maui, which was released for public use on 17.06.2022 as a novel way of creating an application for multiple platforms. Using .Net Maui Blazor, one project base can be used for a web application as well as native IOS and Android support. EcoDat enables users to interact with the data without any specific platform/device or pre-installed add-ons, giving them tools to visualize their research findings. EcoDat also makes it easier to identify anomalies.

To demonstrate the UI and graph capabilities, a demo page was created. This page contains two predetermined relevant graphs shown in Figure 3.1. Demo page of the application. The graph on the left represents dissolved oxygen value on the Y axis and years on the X axis. In comparison, the graph on the right shows the distribution of Temp and pH values from each month. Graphs shown on this page are interactable and can change their position by dragging and dropping.

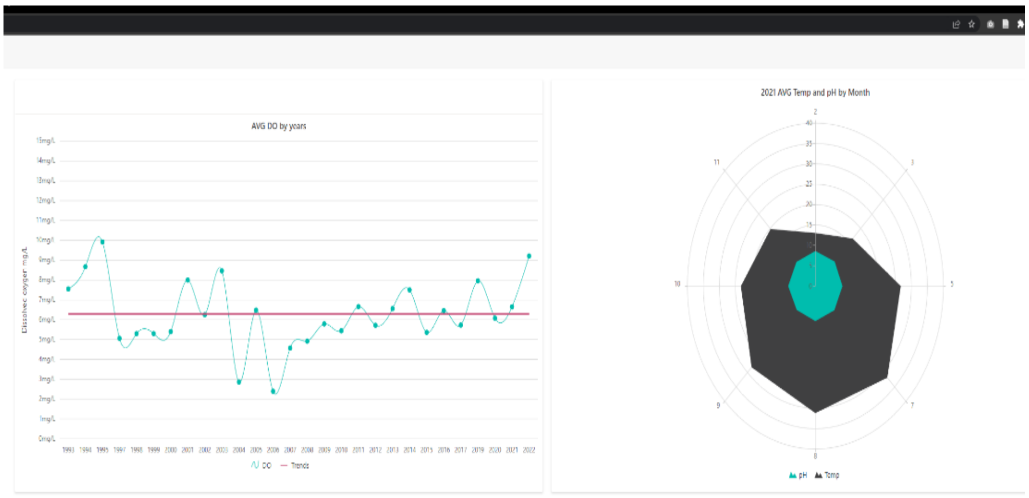


Figure 3.1. Demo page of the application

## 3.2 Data Table

The data table page shows necessary filtering tools implemented using native C# codes. This page enables users to create their SQL query search by utilizing the "Add Group" and "Add Condition" buttons shown in Figure 3.2. Query selector. After the initial selection, other conditions can be added with the "+" button. The resulting three dropdown fields are shown below and in Figure 3.3. Data table dropdown fields;

1) Data type

2) Logic operator

3) Value

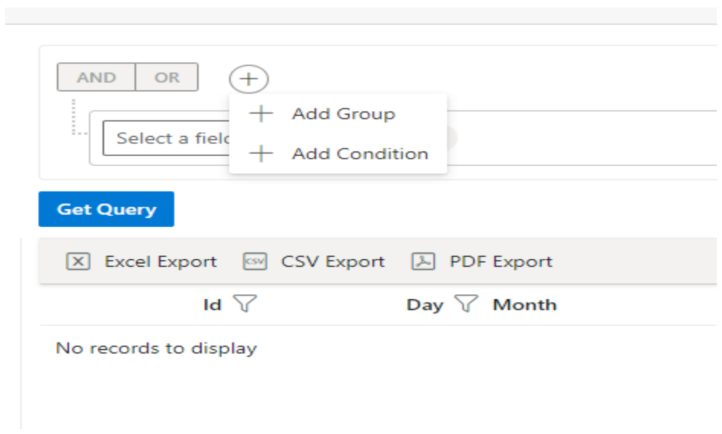


Figure 3.2. Query selector

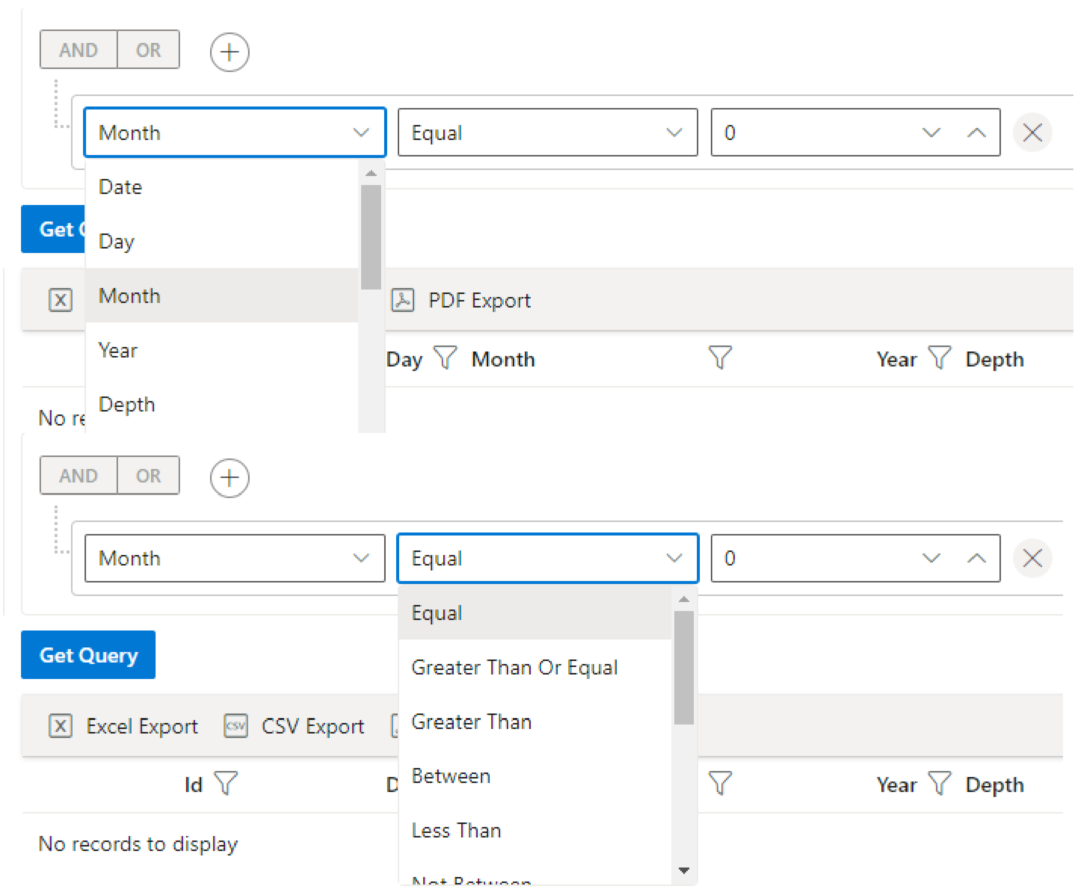


Figure 3.3. Data table dropdown fields

The logic operator field consists of 12 different operators, namely;

Equal

Greater than or equal

Greater than

Between

Less than

Not between

Less than or equal

Not equal

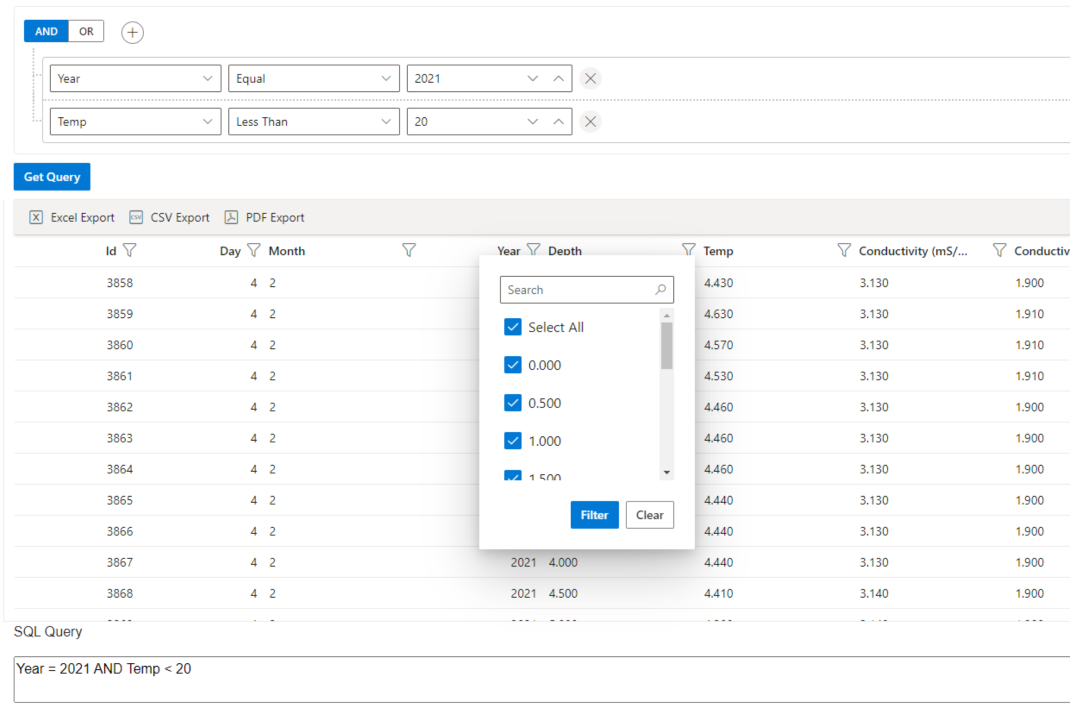
In

Not In

Is Null

Is Not Null

Users can select a data type to interact with and use one of the 12 different operators to filter their data. When the user clicks the query button, the server runs through the database and creates a data grid that meets the conditions set by the user. Values represented in the table can be further filtered by clicking the funnel icon next to the column. This action opens up relevant data that exist within the given parameters. For example, the query search is shown in Figure 3.4. Data creation and filtering requested to see all the data where the year is 2021, and the temperature is below 20 degrees. After this initial search, the researcher can click on the funnel icon and decide to see only measurements for a depth of 1 meter. Furthermore, the resulting data grid can be exported to Excel, CSV, and PDF formats to ease the researcher's needs.



| Id | Day | Month | Year | Depth | Temp | Conductivity (mS/... | Conductiv |
|---|---|---|---|---|---|---|---|
| 3858 | 4 | 2 | | | 4.430 | 3.130 | 1.900 |
| 3859 | 4 | 2 | | | 4.630 | 3.130 | 1.910 |
| 3860 | 4 | 2 | | | 4.570 | 3.130 | 1.910 |
| 3861 | 4 | 2 | | | 4.530 | 3.130 | 1.910 |
| 3862 | 4 | 2 | | | 4.460 | 3.130 | 1.900 |
| 3863 | 4 | 2 | | | 4.460 | 3.130 | 1.900 |
| 3864 | 4 | 2 | | | 4.460 | 3.130 | 1.900 |
| 3865 | 4 | 2 | | | 4.440 | 3.130 | 1.900 |
| 3866 | 4 | 2 | | | 4.440 | 3.130 | 1.900 |
| 3867 | 4 | 2 | 2021 | 4.000 | 4.440 | 3.130 | 1.900 |
| 3868 | 4 | 2 | 2021 | 4.500 | 4.410 | 3.140 | 1.900 |

SQL Query

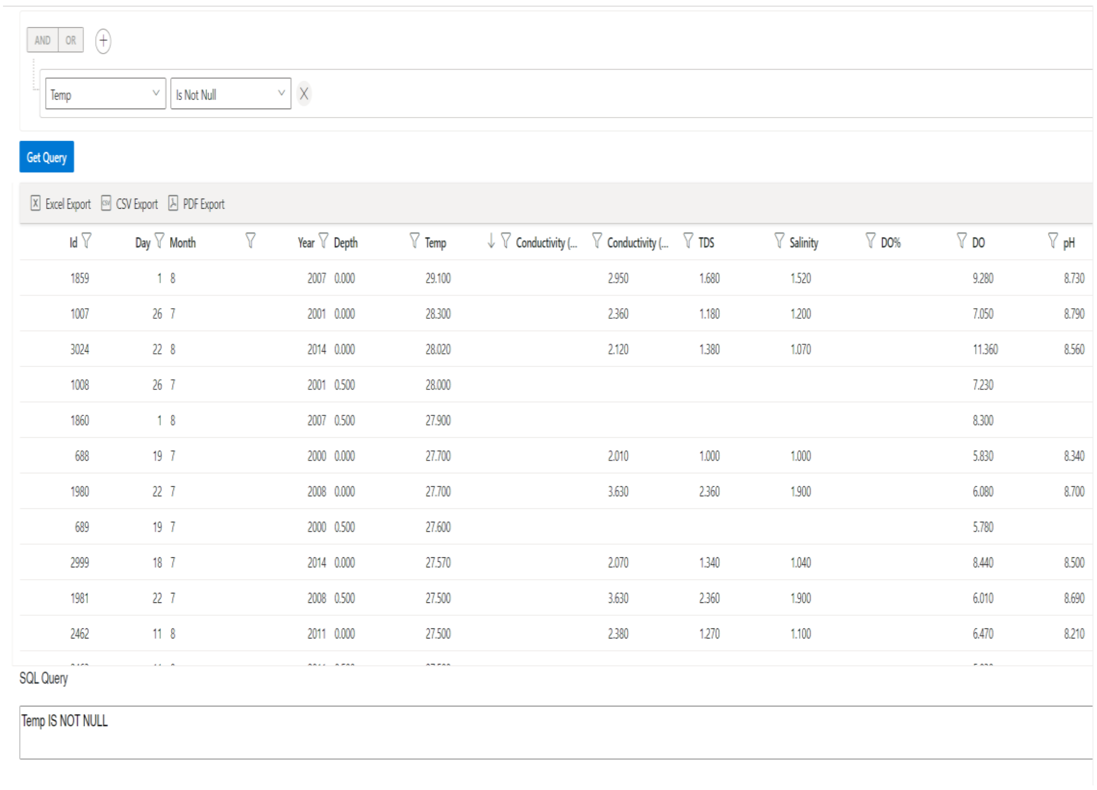
```
Year = 2021 AND Temp < 20
```

Figure 3.4. Data creation and filtering

To elaborate further, four query examples are provided below to demonstrate the capabilities of the data table page.

a) Query Example 1: What is the highest recorded temperature in the dataset?

For this query, the user wants to access the highest recorded temperature reading in the dataset. The page will display all relevant data by selecting "Temperature" in the data type and "Greater than" in the logic operator dropdowns. After the query execution, the user can choose to sort all findings by clicking on the label of the relevant column. (Figure 3.5. Highest recorded temperature)



| Id | Day | Month | Year | Depth | Temp | Conductivity (... | Conductivity (... | TDS | Salinity | DO% | DO | pH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1859 | 1 | 8 | 2007 | 0.000 | 29.100 | | 2.950 | 1.680 | 1.520 | | 9.280 | 8.730 |
| 1007 | 26 | 7 | 2001 | 0.000 | 28.300 | | 2.360 | 1.180 | 1.200 | | 7.050 | 8.790 |
| 3024 | 22 | 8 | 2014 | 0.000 | 28.020 | | 2.120 | 1.380 | 1.070 | | 11.360 | 8.560 |
| 1008 | 26 | 7 | 2001 | 0.500 | 28.000 | | | | | | 7.230 | |
| 1860 | 1 | 8 | 2007 | 0.500 | 27.900 | | | | | | 8.300 | |
| 688 | 19 | 7 | 2000 | 0.000 | 27.700 | | 2.010 | 1.000 | 1.000 | | 5.830 | 8.340 |
| 1980 | 22 | 7 | 2008 | 0.000 | 27.700 | | 3.630 | 2.360 | 1.900 | | 6.080 | 8.700 |
| 689 | 19 | 7 | 2000 | 0.500 | 27.600 | | | | | | 5.780 | |
| 2999 | 18 | 7 | 2014 | 0.000 | 27.570 | | 2.070 | 1.340 | 1.040 | | 8.440 | 8.500 |
| 1981 | 22 | 7 | 2008 | 0.500 | 27.500 | | 3.630 | 2.360 | 1.900 | | 6.010 | 8.690 |
| 2462 | 11 | 8 | 2011 | 0.000 | 27.500 | | 2.380 | 1.270 | 1.100 | | 6.470 | 8.210 |

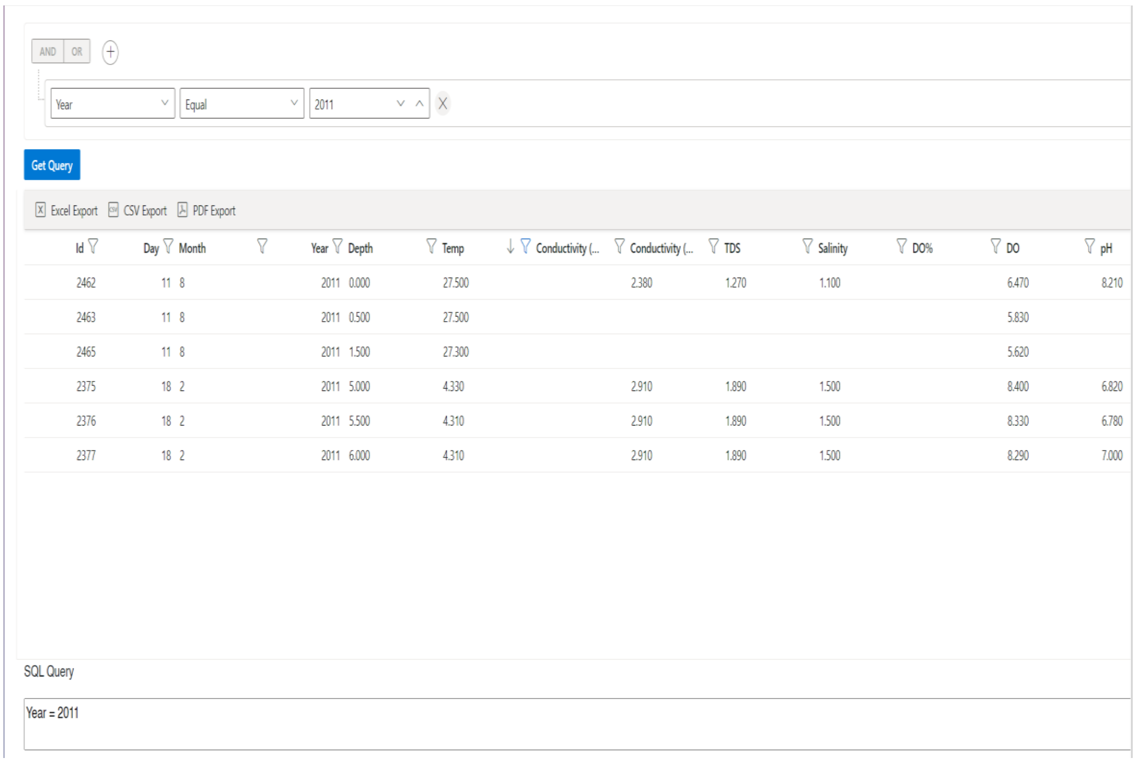
SQL Query

```
Temp IS NOT NULL
```

Figure 3.5. Highest recorded temperature

b) Query Example 2: What are the lowest and highest temperatures in the year 2011?

For this query, the user wants to access the highest and lowest recorded temperature in the year 2011. The page will display all relevant data by selecting "Year" in the data type and the "Equal" operator. After the query execution, the user can filter the results by clicking on the filter icon of temperature and picking the lowest and highest values. Successful query execution results are shown in Figure 3.6. The lowest and highest temperature in 2011



| Id | Day | Month | Year | Depth | Temp | Conductivity (... | Conductivity (... | TDS | Salinity | DO% | DO | pH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2462 | 11 | 8 | 2011 | 0.000 | 27.500 | | 2.380 | 1.270 | 1.100 | | 6.470 | 8.210 |
| 2463 | 11 | 8 | 2011 | 0.500 | 27.500 | | | | | | 5.830 | |
| 2465 | 11 | 8 | 2011 | 1.500 | 27.300 | | | | | | 5.620 | |
| 2375 | 18 | 2 | 2011 | 5.000 | 4.330 | | 2.910 | 1.890 | 1.500 | | 8.400 | 6.820 |
| 2376 | 18 | 2 | 2011 | 5.500 | 4.310 | | 2.910 | 1.890 | 1.500 | | 8.330 | 6.780 |
| 2377 | 18 | 2 | 2011 | 6.000 | 4.310 | | 2.910 | 1.890 | 1.500 | | 8.290 | 7.000 |

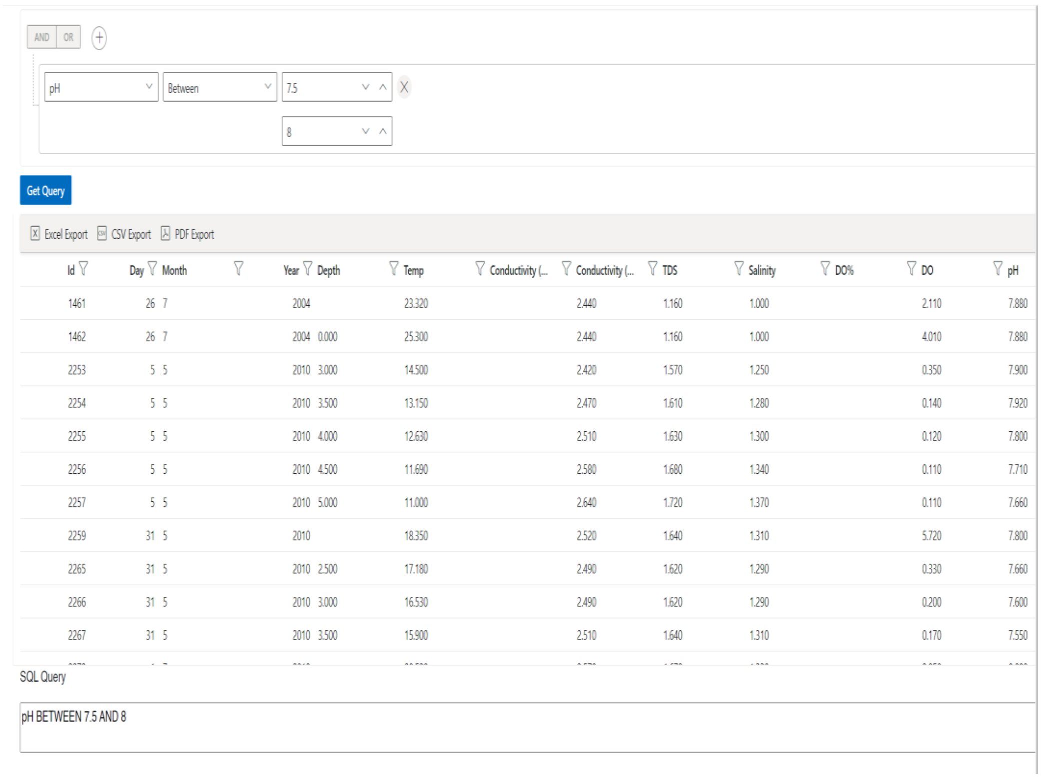
SQL Query

```
Year = 2011
```

Figure 3.6. The lowest and highest temperature in 2011

c) Query Example 3: What were the times when the pH in the lake was between 7.5 and 8.0?

For this query, the user wants to access the measurements where the pH value of the lake was between 7.5 and 8. The page will display all relevant data by selecting "pH" in the data type and the "Between" operator. Successful query execution results are shown in Figure 3.7. pH values between 7.5 and 8.0.



pH Between 7.5 8

| Id | Day | Month | Year | Depth | Temp | Conductivity (... | Conductivity (... | TDS | Salinity | DO% | DO | pH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1461 | 26 | 7 | 2004 |  | 23.320 |  | 2.440 | 1.160 | 1.000 |  | 2.110 | 7.880 |
| 1462 | 26 | 7 | 2004 | 0.000 | 25.300 |  | 2.440 | 1.160 | 1.000 |  | 4.010 | 7.880 |
| 2253 | 5 | 5 | 2010 | 3.000 | 14.500 |  | 2.420 | 1.570 | 1.250 |  | 0.350 | 7.900 |
| 2254 | 5 | 5 | 2010 | 3.500 | 13.150 |  | 2.470 | 1.610 | 1.280 |  | 0.140 | 7.920 |
| 2255 | 5 | 5 | 2010 | 4.000 | 12.630 |  | 2.510 | 1.630 | 1.300 |  | 0.120 | 7.800 |
| 2256 | 5 | 5 | 2010 | 4.500 | 11.690 |  | 2.580 | 1.680 | 1.340 |  | 0.110 | 7.710 |
| 2257 | 5 | 5 | 2010 | 5.000 | 11.000 |  | 2.640 | 1.720 | 1.370 |  | 0.110 | 7.660 |
| 2259 | 31 | 5 | 2010 |  | 18.350 |  | 2.520 | 1.640 | 1.310 |  | 5.720 | 7.800 |
| 2265 | 31 | 5 | 2010 | 2.500 | 17.180 |  | 2.490 | 1.620 | 1.290 |  | 0.330 | 7.660 |
| 2266 | 31 | 5 | 2010 | 3.000 | 16.530 |  | 2.490 | 1.620 | 1.290 |  | 0.200 | 7.600 |
| 2267 | 31 | 5 | 2010 | 3.500 | 15.900 |  | 2.510 | 1.640 | 1.310 |  | 0.170 | 7.550 |

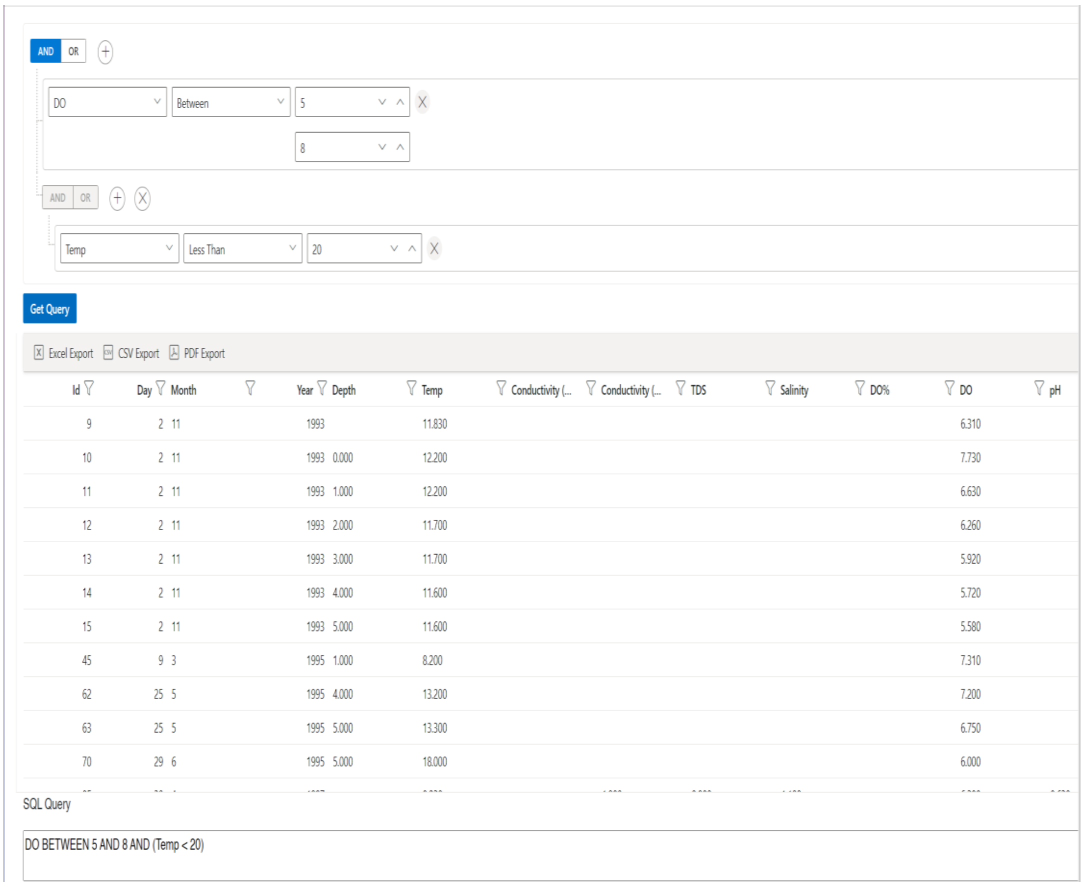
SQL Query

```
pH BETWEEN 7.5 AND 8
```

Figure 3.7. pH values between 7.5 and 8.0

d) Query example 4: What are the times when the DO is between 5 and 8, and the temperature is below 20 °C?

For this query, the user wants to combine filters related to two different data types and tries to find the data points where DO is between 5 to 8 while the temperature is below 20 degrees Celsius. Firstly "DO" in the data type and "Between" in the logic operator are selected; then, by clicking on to "Add Conditions" button, a second group will be formed. This will enable the user to choose "And" or "Or" operators on the menu; by selecting "And" and setting the datatype to "Temperature" and the operator to "Less than," the user can execute both conditions simultaneously. Successful query execution results are shown in Figure 3.8. Times where DO, is between 5 and 8 and the temperature is below 20 °C.


AND OR

DO | Between | 5 | 8

AND OR

Temp | Less Than | 20

Get Query

Excel Export CSV Export PDF Export

| Id | Day | Month | | Year | Depth | Temp | Conductivity (... | Conductivity (... | TDS | Salinity | DO% | DO | pH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 2 | 11 | | 1993 | | 11.830 | | | | | | 6.310 | |
| 10 | 2 | 11 | | 1993 | 0.000 | 12.200 | | | | | | 7.730 | |
| 11 | 2 | 11 | | 1993 | 1.000 | 12.200 | | | | | | 6.630 | |
| 12 | 2 | 11 | | 1993 | 2.000 | 11.700 | | | | | | 6.260 | |
| 13 | 2 | 11 | | 1993 | 3.000 | 11.700 | | | | | | 5.920 | |
| 14 | 2 | 11 | | 1993 | 4.000 | 11.600 | | | | | | 5.720 | |
| 15 | 2 | 11 | | 1993 | 5.000 | 11.600 | | | | | | 5.580 | |
| 45 | 9 | 3 | | 1995 | 1.000 | 8.200 | | | | | | 7.310 | |
| 62 | 25 | 5 | | 1995 | 4.000 | 13.200 | | | | | | 7.200 | |
| 63 | 25 | 5 | | 1995 | 5.000 | 13.300 | | | | | | 6.750 | |
| 70 | 29 | 6 | | 1995 | 5.000 | 18.000 | | | | | | 6.000 | |

SQL Query

```
DO BETWEEN 5 AND 8 AND (Temp < 20)
```

Figure 3.8. Times where DO, is between 5 and 8 and the temperature is below 20 °C

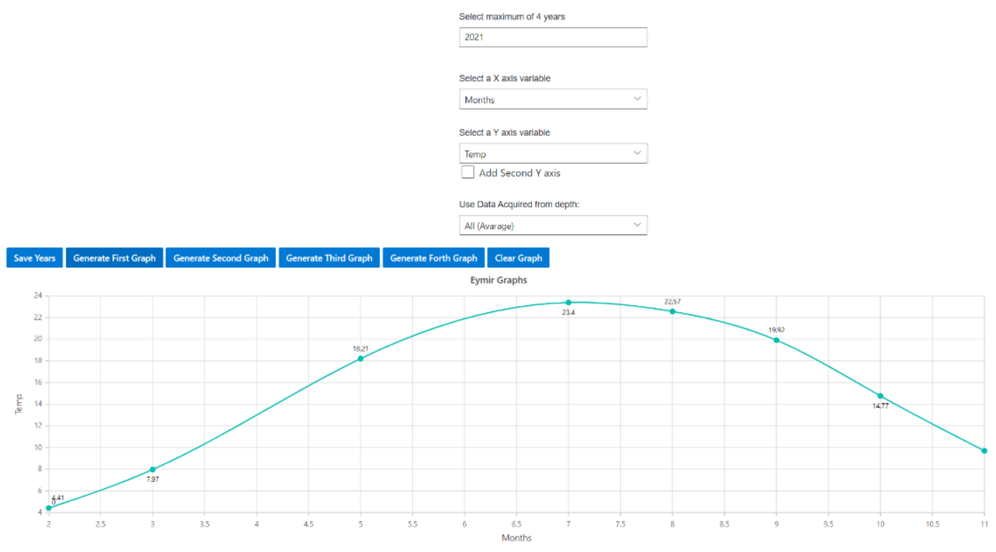
## 3.3 Data Plot

The data plot page shows the necessary selection tools implemented using native C# codes. The graphical representations are generated through the server and fed to the page. This page enables users to create interactive graphs by utilizing provided dropdowns. A maximum of four different years can be selected for query execution. This limit was made to prevent readability issues. After the year selection, X and Y axis variables can be selected using provided dropdowns which consist of Day, Month, Year, Depth (m), Temp(°C), Conductivity (mS/cm//c), Conductivity (mS/cm), TDS (g/l), Salinity ‰, DO (%), DO (mg/l), pH fields in the data. The user can opt to add a second Y axis if desired.

After the configuration process, the users should click on the "Save Years" button to lock in the selected years. When the user clicks each "Generate" button, the server runs through the database, creates the graph that meets the configurations, and generates the corresponding graph. The resulting graph can be interacted with by zooming in and selecting individual values. A logarithmic mean function is used to create single data points from 8 different measurements for a single experiment. This means that the temperature data shown for the given data point represents the logarithmic mean of the water column temperature.

To elaborate further, tree graph examples are provided below to demonstrate the capabilities of the data plot page.

1) Graph of monthly average variation of temperature in a selected year



Select maximum of 4 years: 2021

Select a X axis variable: Months

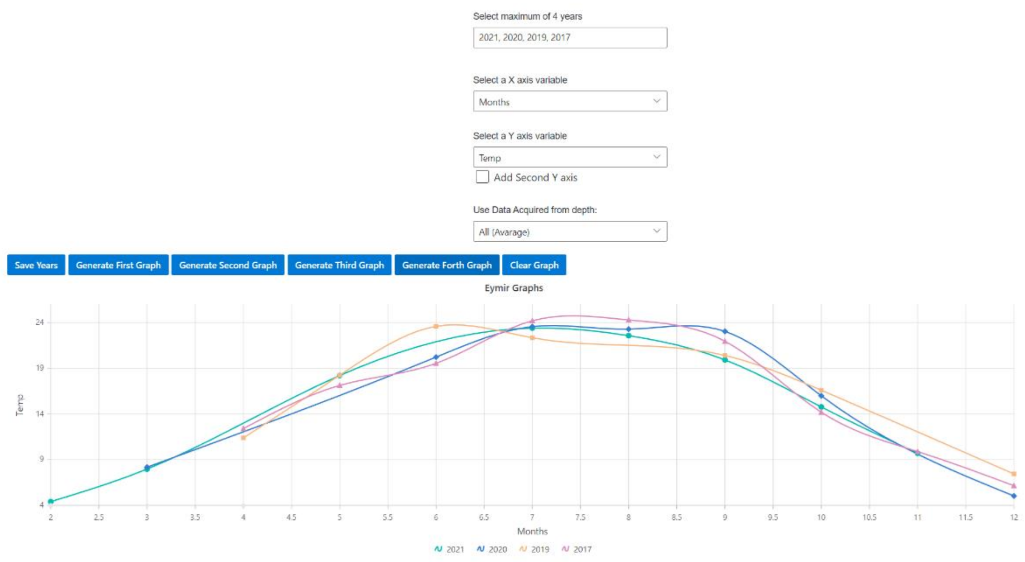
Select a Y axis variable: Temp

Add Second Y axis

Use Data Acquired from depth: All (Avarage)

Figure 3.9. Monthly average variation of temperature in a selected year

2) Monthly average variation graph of overlayed temperature over selected four years


Select maximum of 4 years
2021, 2020, 2019, 2017

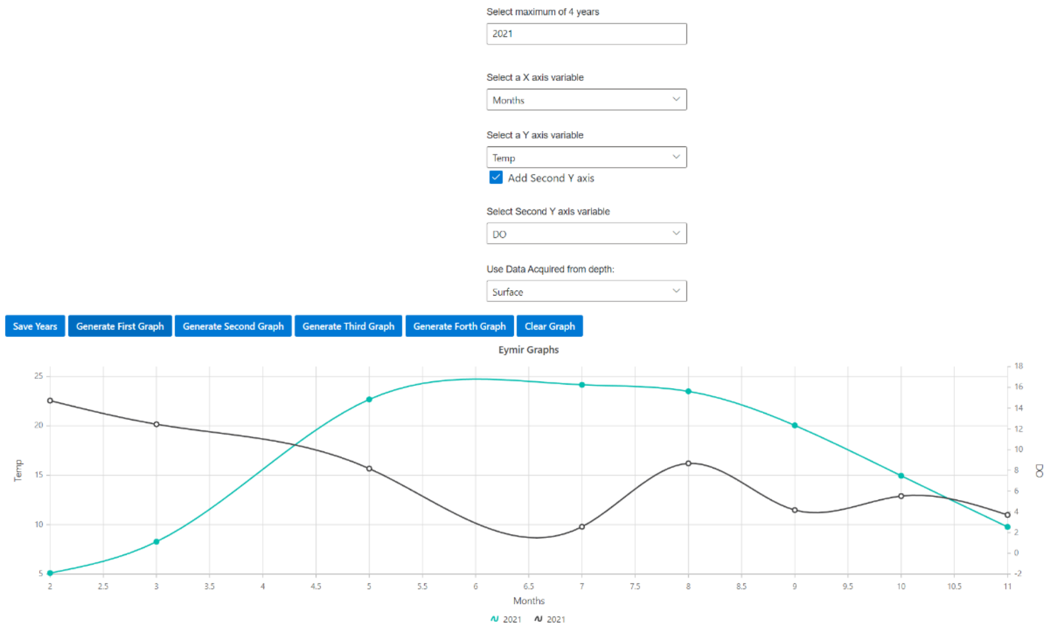
Select a X axis variable
Months

Select a Y axis variable
Temp
Add Second Y axis

Use Data Acquired from depth:
All (Avarage)

Figure 3.10. Monthly average variation graph of overlayed temperature over selected four years

3) Overlay graph of monthly average variation of DO and temperature for samples taken from the surface in a selected year



Select maximum of 4 years
2021

Select a X axis variable
Months

Select a Y axis variable
Temp
✓ Add Second Y axis

Select Second Y axis variable
DO

Use Data Acquired from depth:
Surface

Figure 3.11. Overlay graph of monthly average variation of DO and temperature for samples taken from the surface in a selected year

# CHAPTER 4

# CONCLUSION

In this study, a smart-accessible database system and a web-based application were created for limnological monitoring researchers, allowing them to reach, interact and save their fieldwork using the developed application. Before the actual implementation of the application, technologies in the area of web development were analyzed. Software and coding languages were chosen based on requirements and developer experience. After the theoretical parts of this work, a practical part containing a functional database was processed, which is used to store measured data using a console or WinForms application. The functionality was tested by simulating the data flow and acting in the application as a potential user.

The database solution consists of four main tables containing the measured data, sources of this data, users, and groups assigning rights to users. This database solution can be installed from the enclosed CD carrier, on which the database is located in the form of a .sql file that just needs to be run in Microsoft SQL Server Management Studio.

The web application solution includes a set of windows intended for multiple platforms with the use of a .NET MAUI environment, which will allow the user to measure data, write it to the database, display it, export it to Excel and delete it. Furthermore, It can create input-based graphs that visualize patterns within data and let the user filter it by the desired variable. It is also possible to modify the powers of user accounts so that they only have access to specific areas of the application.

A simple application for Windows was also programmed, enabling basic testing of the application for measuring data and the database, plotting data, and exporting it.

The ECODat application is created with responsiveness in mind and can be used well on mobile devices if desired. The design's color palette is designed with a focus on clarity and lightness. Usability and accessibility aspects such as larger search fields, clear column headers, and contrast ratio have been addressed. ECOData application can create input-based graphs that visualize patterns within data and let the user filter it by the desired variable.

Some parts of the implementation do not align with how they were initially designed. The resulting system lacks an email client and dedicated hosting. However, these steps are not essential, and it is possible to add them later, even though the system is already in operation. Thus, implementing these steps is likely to be a future development subject.

As it is assumed that the resulting application will be further developed, extensions are proposed that would bring benefit or improve the quality of work with the system. The graphical interface for drawing graphs in the application allows the use of a large number of resources, which negatively affects the overall reaction time of the application. However, the foundations of this graphical interface were built for it to allow displaying data in real-time, which is beneficial for possible future development of the application.

The initial intention of this application was to use it for real-time hf monitoring data. A real-time event processing function that could see changes in the measured values in the database could be developed as a proposal to extend the application in the future. This would add significant value to the application. Furthermore, the actions executed in the database are performed by stored procedures. This approach allows other applications to connect to the database and work simultaneously, such as an ASP.NET web application.

Since a research lab could have more than one team working on different projects, a possible improvement could be, for example, the implementation of the dynamic creation of databases based on user input. Furthermore, it would be helpful to extend the definition of access rights to types of entities or specific entities. Implementing mass operations with entities (mass editing, deletion, selection of required lines during import or export) would also be beneficial. A graphical representation of the types of entities and their mutual links and attributes would also be possible. The user interface could be tailored to the team's needs.

# REFERENCES

AG, S. (n.d.). *SAP Completes Acquisition of Sybase, Inc.* Retrieved August 22, 2022, from https://www.prnewswire.com/news-releases/sap-completes-acquisition-of-sybase-inc-99612339.html

Bastviken, D., Tranvik, L. J., Downing, J. A., Crill, P. M., & Enrich-Prast, A. (2011). Freshwater Methane Emissions Offset the Continental Carbon Sink. *Science*, *331*(6013), 50–50. https://doi.org/10.1126/science.1196808

Beklioğlu, M., Bucak, T., Coppens, J., Bezirci, G., Tavşanoğlu, Ü., Çakıroğlu, A., Levi, E., Erdoğan, Ş., Filiz, N., Özkan, K., & Özen, A. (2017). Restoration of Eutrophic Lakes with Fluctuating Water Levels: A 20-Year Monitoring Study of Two Inter-Connected Lakes. *Water*, *9*(2), 127. https://doi.org/10.3390/w9020127

Beklioglu, M., BURNAK, S., & İNCE, Ö. (2000). Benthi-planktivorous Fish-Induced Low Water Quality of Lake Eymir Before Biomanipulation. *Turkish Journal of Zoology*, *24*(3), 315–326. https://doi.org/-

Beklioglu, M., Ince, O., & Tuzun, I. (2003). Restoration of the eutrophic Lake Eymir, Turkey, by biomanipulation after a major external nutrient control I. *Hydrobiologia*, *490*(1), 93–105. https://doi.org/10.1023/A:1023466629489

Beklioglu, M., Meerfhoff, M., Søndergaard, M., & Jeppesen, E. (2011). Eutrophication and Restoration of Shallow Lakes from a Cold Temperate to a Warm Mediterranean and a (Sub)Tropical Climate. In A. A. Ansari, S. Singh Gill, G. R. Lanza, & W. Rast (Eds.), *Eutrophication: Causes, consequences and control* (pp. 91–108). Springer Netherlands. https://doi.org/10.1007/978-90-481-9625-8_4

Benoy, G., Cash, K., McCauley, E., & Wrona, F. (2007). Carbon dynamics in lakes of the boreal forest under a changing climate. *Environmental Reviews*, *15*, 175–189. https://doi.org/10.1139/A07-006

Cole, J. J., Prairie, Y. T., Caraco, N. F., McDowell, W. H., Tranvik, L. J., Striegl, R. G., Duarte, C. M., Kortelainen, P., Downing, J. A., Middelburg, J. J., & Melack, J. (2007). Plumbing the Global Carbon Cycle: Integrating Inland Waters into the Terrestrial Carbon Budget. *Ecosystems*, *10*(1), 172–185. https://doi.org/10.1007/s10021-006-9013-8

davidbritch. (n.d.). *What is .NET MAUI? - .NET MAUI*. Retrieved August 22, 2022, from https://docs.microsoft.com/en-us/dotnet/maui/what-is-maui

Deacon, J. (2009). *Model-View-Controller (MVC) Architecture*. 7.

Downing, J. A., & Duarte, C. M. (2013). Abundance and Size Distribution of Lakes, Ponds and Impoundments☆. In *Reference Module in Earth Systems and Environmental Sciences*. Elsevier. https://doi.org/10.1016/B978-0-12-409548-9.03867-7

Downing, J. A., Prairie, Y. T., Cole, J. J., Duarte, C. M., Tranvik, L. J., Striegl, R. G., McDowell, W. H., Kortelainen, P., Caraco, N. F., Melack, J. M., & Middelburg, J. J. (2006). The global abundance and size distribution of lakes, ponds, and impoundments. *Limnology and Oceanography*, *51*(5), 2388–2397. https://doi.org/10.4319/lo.2006.51.5.2388

Factory.hr. (2019, July 15). HTTP/2: The difference between HTTP/1.1, benefits and how to use it. *Medium*. https://factoryhr.medium.com/http-2-the-difference-between-http-1-1-benefits-and-how-to-use-it-38094fa0e95b

guardrex. (n.d.). *ASP.NET Core Blazor*. Retrieved August 22, 2022, from https://docs.microsoft.com/tr-tr/aspnet/core/blazor/

*HTTP/2 Frequently Asked Questions*. (n.d.). Retrieved August 22, 2022, from https://http2.github.io/faq/

Jennings, E., Jones, S., Arvola, L., Staehr, P. A., Gaiser, E., Jones, I. D., Weathers, K. C., Weyhenmeyer, G. A., Chiu, C.-Y., & De Eyto, E. (2012). Effects of weather-related episodic events in lakes: An analysis based on high-frequency data.

*Freshwater Biology*, *57*(3), 589–601. https://doi.org/10.1111/j.1365-2427.2011.02729.x

Jones, I. D., Page, T., Alex Elliott, J., Thackeray, S. J., & Louise Heathwaite, A. (2011). Increases in lake phytoplankton biomass caused by future climate-driven changes to seasonal river flow. *Global Change Biology*, *17*(5), 1809–1820. https://doi.org/10.1111/j.1365-2486.2010.02332.x

Joshi, B. (2019). Blazor. In B. Joshi (Ed.), *Beginning Database Programming Using ASP.NET Core 3: With MVC, Razor Pages, Web API, jQuery, Angular, SQL Server, and NoSQL* (pp. 337–380). Apress. https://doi.org/10.1007/978-1-4842-5509-4_8

Krasner, G. E., Pope, S. T., & Systems, P. (n.d.). *A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System*. 34.

Leff, A., & Rayfield, J. T. (2001). Web-application development using the Model/View/Controller design pattern. *Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference*, 118–127. https://doi.org/10.1109/EDOC.2001.950428

Libkin, L. (2003). Expressive power of SQL. *Theoretical Computer Science*, *296*(3), 379–404. https://doi.org/10.1016/S0304-3975(02)00736-3

MacIntyre, S., Romero, J. R., & Kling, G. W. (2002). Spatial-temporal variability in surface layer deepening and lateral advection in an embayment of Lake Victoria, East Africa. *Limnology and Oceanography*, *47*(3), 656–671. https://doi.org/10.4319/lo.2002.47.3.0656

Mahmoud, S. H., & Gan, T. Y. (2018). Impact of anthropogenic climate change and human activities on environment and ecosystem services in arid regions. *Science of The Total Environment*, *633*, 1329–1344. https://doi.org/10.1016/j.scitotenv.2018.03.290

Melton, J. (1996). SQL language summary. *ACM Computing Surveys*, *28*(1), 141–143. https://doi.org/10.1145/234313.234374

Moss, B., Battarbee, R. W., & Kernan, M. (2010). Introduction. In *Climate Change Impacts on Freshwater Ecosystems* (pp. 1–14). John Wiley & Sons, Ltd. https://doi.org/10.1002/9781444327397.ch1

Moss, B. R. (2009). *Ecology of Fresh Waters: Man and Medium, Past to Future*. John Wiley & Sons.

Natarajan, J., Bruchez, R., Coles, M., Shaw, S., & Cebollero, M. (2015). *Pro T-SQL Programmer's Guide*. Apress.

Özen, A., Karapınar, B., Kucuk, İ., Jeppesen, E., & Beklioglu, M. (2010). Drought-induced changes in nutrient concentrations and retention in two shallow Mediterranean lakes subjected to different degrees of management. *Hydrobiologia*, *646*(1), 61–72. https://doi.org/10.1007/s10750-010-0179-x

Pearson, A. L., Shortridge, A., Delamater, P. L., Horton, T. H., Dahlin, K., Rzotkiewicz, A., & Marchiori, M. J. (2019). Effects of freshwater blue spaces may be beneficial for mental health: A first, ecological study in the North American Great Lakes region. *PLOS ONE*, *14*(8), e0221977. https://doi.org/10.1371/journal.pone.0221977

Price, M. J. (2019). *C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET using Visual Studio Code, 4th Edition*. Packt Publishing Ltd.

Raymond, P. A., Hartmann, J., Lauerwald, R., Sobek, S., McDonald, C., Hoover, M., Butman, D., Striegl, R., Mayorga, E., Humborg, C., Kortelainen, P., Dürr, H., Meybeck, M., Ciais, P., & Guth, P. (2013). Global carbon dioxide emissions from inland waters. *Nature*, *503*(7476), 355–359. https://doi.org/10.1038/nature12760

Rick-Anderson. (n.d.-a). *ASP.NET overview*. Retrieved August 22, 2022, from https://docs.microsoft.com/en-us/aspnet/overview

Rick-Anderson. (n.d.-b). *Overview of ASP.NET Core*. Retrieved August 22, 2022, from https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core

Stonebraker, M. (2010). SQL databases v. NoSQL databases. *Communications of the ACM*, *53*(4), 10–11. https://doi.org/10.1145/1721654.1721659

Tranvik, L. J., Downing, J. A., Cotner, J. B., Loiselle, S. A., Striegl, R. G., Ballatore, T. J., Dillon, P., Finlay, K., Fortino, K., Knoll, L. B., Kortelainen, P. L., Kutser, T., Larsen, Soren., Laurion, I., Leech, D. M., McCallister, S. L., McKnight, D. M., Melack, J. M., Overholt, E., … Weyhenmeyer, G. A. (2009). Lakes and reservoirs as regulators of carbon cycling and climate. *Limnology and Oceanography*, *54*(6part2), 2298–2314. https://doi.org/10.4319/lo.2009.54.6_part_2.2298

Van de Bogert, M. C., Bade, D. L., Carpenter, S. R., Cole, J. J., Pace, M. L., Hanson, P. C., & Langman, O. C. (2012). Spatial heterogeneity strongly affects estimates of ecosystem metabolism in two north temperate lakes. *Limnology and Oceanography*, *57*(6), 1689–1700. https://doi.org/10.4319/lo.2012.57.6.1689

Vermeir, N. (2022). MAUI. In N. Vermeir (Ed.), *Introducing .NET 6: Getting Started with Blazor, MAUI, Windows App SDK, Desktop Development, and Containers* (pp. 153–176). Apress. https://doi.org/10.1007/978-1-4842-7319-7_6

*What is the HTTP/2 Protocol? Overview & Examples | Upwork*. (n.d.). Retrieved August 22, 2022, from https://www.upwork.com/resources/what-is-http2