İstanbul
Bilgi Üniversitesi

Petting Back Together: Missing Pet Navigation Application Using Image Classification And Image Similarity Matching

by

Yiğit Can Ayaz, 118202068
Begüm Alioğlu, 119200073

Supervised by

Asst. Prof. Murat Orhun

Submitted to the
Faculty of Engineering and Natural Sciences
in partial fulfillment of the requirements for the

Bachelor of Science

in the
Department of Computer Engineering

January, 2023

Abstract

Pets hold a significant place in our lives. Statistics suggest that pets are likely to go missing at least once over a five-year span. Pet owners resort to various methods to locate their lost companions, with some still relying on traditional tactics such as banners, which have been largely deemed inefficient. We firmly believe that the solution to this issue lies in digital transformation. To that end, we have delineated the technology that we will harness to develop our application. Furthermore, we have detailed the strategies that our application will employ to address and rectify the missing pet dilemma.

Table of Contents

List of Figures

List of Abbreviations

PVC	Polyvinyl chloride
HTTP	Hyper Text Transfer Protocol
PHP	Hypertext Preprocessor
SQL	Structured Query Language
DBMS	Database Management System
API	Application Programming Interface
ASP	Active Server Pages
NET	Network
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
UML	Unified Modeling Language
TCP	Transmission Control Protocol
SSE	Server-Sent Events
OOP	Object Oriented Programming
AOP	Aspect Oriented Programming - değiştirdim
DTO	Data Transfer Object
ORM	Object Relational Mapping
DevOps	Developers and Operations
CI/CD	Continuous Integration/Continuous Delivery
CRUD	Create, Read, Update, Delete
DTO	Data Transfer Object
JSON	JavaScript Object Notation
JWT	JSON Web Token

1 Introduction

In Turkey 4 million cats and 2 million dogs have their owners [1]. Pets can provide companionship and a sense of connection, especially for people who live alone or are isolated [2]. Interacting with pets can lower stress and anxiety levels and improve overall mood. Owning a pet, can encourage physical activity through activities such as walking or playing. This can help improve cardiovascular health and reduce the risk of obesity and other health conditions. The first demonstration of an association between pets and health was an early study of 92 heart-attack victims in which 28% of pet owners survived for at least a year as compared to only 6% of non–pet owners [3]. A study of over 2,400 cat owners concluded there was a significantly lower relative risk for death due to cardiovascular diseases, including stroke and heart attack, compared to non-owners during a 20-year follow-up [4].

Between 11-16% of dogs and 12-18% of cats are likely to go missing at least once in five years [5]. People may feel a sense of panic and fear, especially if they are not sure what has happened to their pet or if they believe their pet may be in danger. It is likely that pets may experience anxiety, fear, confusion, and sadness. Some pets may also feel a strong bond with their owners and may become distressed when separated from them. %34 of the missing cats and %6 of the missing dogs are not found[5]. %60 of the cats returned home on their own and only %1 of the cats found via microchip[5]. This proofs that people were not taking the missing pet to animal shelter when they saw.

Using banners to search for a missing pet can be a controversial and potentially ineffective method. Here are a few reasons why using banners may not be the best way to find a missing pet: limited reach, no centralized location, outdated information, inaccurate descriptions and it is illegal. It is also not environmentally friendly. Flyers are often printed on paper, which requires the use of natural resources and energy to produce. When flyers are distributed and then discarded, they contribute to waste and litter, which can have negative impacts on the environment. The production of paper can contribute to deforestation, as trees are often cut down to make paper products. This can lead to habitat destruction for wildlife and contribute to climate change. The production and distribution of flyers also generates carbon emissions, which contribute to climate change. These emissions come from the energy used to produce and transport the flyers, as well as the fuel used by people who are distributing the flyers.

2 Related Works

Today, with the development of technology, there are dozens of social media platforms. People use these applications to share a message on social media asking for help to find their pet. Examples of these applications are Lost Pet, Missing Pet Finder, Animal Search, Gören Duyan, Kayıp Dostlar. All these applications will be explained and their contents will be analyzed. Also, people put banners on the public property, contact the local animal shelters and rescue groups or ask the neighbors and local businesses. They may use tracking devices or microchips.

2.1 Applications

2.1.1 Lost Pet

This platform is a dedicated resource established in New Zealand, expressly intended to address the issue of lost and found pets. Its primary functionality lies in the provision of an interactive map, providing real-time visibility of reported missing and located pets. The distinctive feature of this application lies in its search system, which is primarily based on microchip identification. Consequently, the reach of this service is somewhat limited, as it only extends to pets that are microchipped. For pets without a microchip, the platform doesn't provide any particular recourse. Moreover, the application maintains an extensive database where pet owners can register their pets for a fee of $15. This registration is a preventive measure which can facilitate quick reunification in case the pet is lost and subsequently found. However, it's important to note that the platform doesn't offer a simple mechanism for users who find a pet. Rather than just snapping a photo and reporting it, the user is required to transport the found pet to a veterinary clinic. This process not only ensures a thorough check of the pet's health but also allows for accurate identification via microchip scanning.

2.1.2 Missing Pet Finder

This mobile application operates on a concept similar to Instagram, with a localized focus setting it apart. It enables users to post adverts about missing pets, which are subsequently sent as notifications to other users within the vicinity. This feature fosters a community effort in locating the missing pet. Users also have the capacity to engage in discussions under the adverts. Moreover, the application provides an option to generate a flyer, fostering the possibility of a broader reach through social media sharing.

2.1.3 Animal Search

This platform originates from the United Kingdom and offers a comprehensive solution to the challenges surrounding missing pets. It enables users to report a missing pet, register a found pet, or add their pet to an existing database. Their business model comprises of four distinct subscription levels, ranging from a free service to a premium one which includes a dedicated search team. In an effort to broaden their outreach, they create posters and maintain a robust presence on social media with over 50,000 followers, facilitating a wider network to aid in the search and recovery of lost pets.

2.1.4 Gören Duyan

"Gören Duyan" transcends the typical bounds of a missing pet application, as it also extends to other categories such as misplaced motorcycles and cars. This platform allows users to post an advert for their missing pets, as well as share information about any lost pets they may encounter outdoors. To initiate contact with other users, registration is necessary. Furthermore, this platform integrates social media sharing capabilities, allowing users to amplify their adverts beyond the application itself.

2.1.5 Kayıp Dostlar

"Kayıp Dostlar" aligns closely with its counterparts, enabling users to report both lost and found pets. This application offers two distinctive options: it enables users to report a pet they found on the street, as well as a pet they found and decided to take in. Aside from these nuances, "Kayıp Dostlar" shares many similarities with other applications in the same space.

2.2 Banners

Despite the common practice of pet owners hanging banners on streets to locate their missing pets, it's crucial to note that such an approach is not only illegal but also detrimental to the environment. As per Turkish law, individuals posting papers or similar posters in public places or street sides can face administrative fines ranging from one hundred to three thousand Turkish Liras [6]. In addition, PVC scrim, which is frequently used for such advertising banners, poses a significant environmental concern. Regrettably, PVC banners are not easily recyclable and often end up in landfills, releasing harmful toxins for many ensuing [7].

2.3 Animal Shelters

In the United States, the issue of lost and stray pets is substantial. Each year, approximately 6.3 million animals find themselves in animal shelters across the country. Among these, around 810,000 stray animals are successfully reunited with their owners. This figure includes 710,000 dogs and an estimated 100,000 cats [8]. These statistics underscore the importance of efficient systems for reporting lost pets and reuniting them with their owners.

3 Methodology

3.1 Structure

The client-side, server-side, and database are the fundamental pillars of any digital application, working in harmony to create a fluid and seamless user experience. The client-side is primarily concerned with user interface and presentation, as depicted in Figure 1. Communication with the server-side occurs through HTTP/TCP requests and responses. The server-side, on the other hand, oversees the management of logic and data necessary to power the client-side functionalities. Its duties include the receipt of client requests and the delivery of appropriate responses. Image processing can be efficiently executed on the server-side, employing a range of programming languages and libraries. Within the server-side context, image processing algorithms can be devised as functions that receive an image, conduct specific operations, and return the processed image as output. In essence, the application is a compilation of code executing on the server, responsible for managing the website's logic, including data retrieval from the database, request processing, and generation of responses.

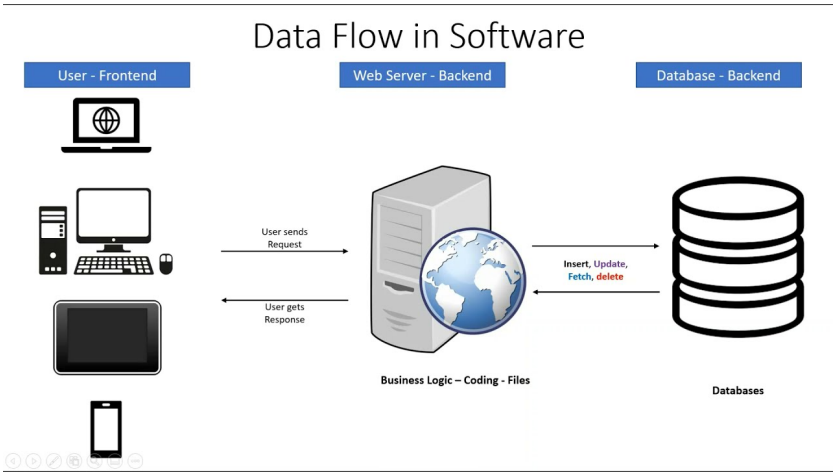

Figure 1: Data Flow

3.2 Database

A database represents a meticulously organized collection of data, stored electronically for easy accessibility. Its primary objective lies in the storage, organization, and management of voluminous data in a manner that ensures simple access and analysis. The key functionalities of a database typically encompass data storage, organization, management, querying, and access provision. These functionalities are instrumental in creating an efficient data management system that meets the diverse requirements of users.

3.2.1 Microsoft SQL Server

Microsoft SQL Server is a robust database management system (DBMS) designed for efficient storage, organization, and management of substantial amounts of data. SQL Server incorporates a suite of components, including the SQL Server database engine, reporting services, integration services, and analysis services. Collectively, these features make SQL Server a flexible and powerful DBMS, employed extensively for data storage and management across a spectrum of applications.

3.3 Image Processing

Image processing, a subset of computer science, leverages algorithms and techniques to manipulate, analyze, and interpret images. This discipline entails performing operations on an image, such as enhancement of contrast or sharpness, noise reduction, and detection and extraction of features or objects. The applications of image processing are manifold, encompassing fields as diverse as medical imaging, surveillance, satellite imagery, and digital photography.

3.3.1 Image Classification

Image classification is a subset of machine learning, dedicated to assigning appropriate labels or classes to input images based on their content. As depicted in Figure 2, a rudimentary image classification system might categorize an image of a dog as "dog", and an image of a cat as "cat". Image classification algorithms generally operate by training a model on a voluminous dataset of labeled images, thus enabling the model to discern the distinguishing features or characteristics of different image classes. During the training phase, the model processes a multitude of images along with their associated labels, adjusting its internal parameters to accurately predict the labels of new, unseen images.

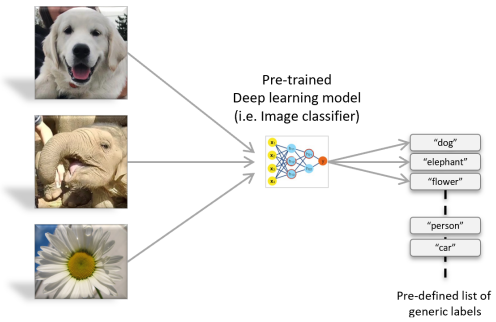

Figure 2: Image Classification

3.3.2 Image Similarity

An image similarity model is a specialized form of machine learning model that quantifies the similarity or dissimilarity between two or more images. This model ingests a set of images and subsequently generates a score indicative of the level of similarity between these images. Common methodologies for assessing image similarity encompass Pixel-level, Feature-based, and Deep learning-based comparisons. Image similarity models have diverse applications, such as image search, duplicate detection, and content-based image retrieval. They can be developed using a variety of machine learning algorithms and techniques, contingent on the unique requirements of the task at hand. Importantly, these similarity models are trained to output embeddings that map items onto a metric space, wherein similar items are proximate and dissimilar items are distant, as depicted in Figure 3.

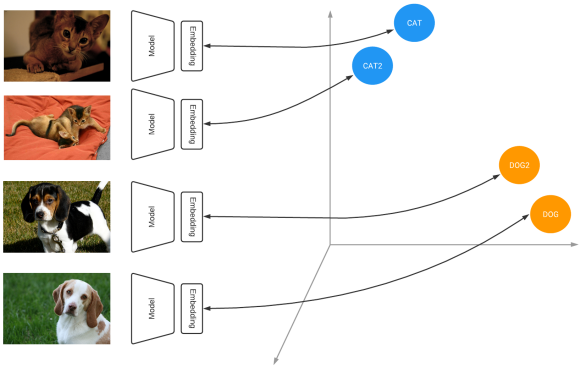

Figure 3: Image Similarity

3.4 Server-side

A server-side application is a program executed on a server, tasked with the performance of specific functions or the provision of particular services to clients. Such applications are typically devised using server-side programming languages, including PHP, Python, or Java, and their execution takes place on the server rather than on the client device. Tasks undertaken by a server-side application might encompass data processing, data storage, communication, authentication, and authorization. In essence, a server-side application is a program responsible for executing specific tasks or providing specific services to clients from the server. These applications are integral components of numerous client-server systems and find utility across a wide array of applications, such as web and mobile applications, among others.

3.4.1 ASP.NET Core API Application

An ASP.NET Core API application represents a web application that unveils a collection of API endpoints for client access and interaction. An API, or "Application Programming Interface", serves as a conduit for different applications or systems to communicate and exchange data. Typically, this includes a set of controllers that process incoming client requests and deliver responses in the form of data or other resources. An ASP.NET Core API application can be employed to build a myriad of API-based applications, encompassing web APIs for mobile or web applications, microservices for distributed systems, or server-side services for web or desktop applications.

3.4.2 Hangfire

Hangfire is an open-source job scheduling library for .NET applications. It provides a simple way to perform background processing or recurring tasks in your application. Hangfire allows you to offload time-consuming or resource-intensive operations from the main thread or request and execute them in the background. Jobs are organized into queues, allowing for prioritization and resource allocation. Hangfire requires persistent storage to store job information and state, supporting various options like SQL databases and Redis. The web-based dashboard enables monitoring and management of jobs, displaying their status, execution history, and any errors. Hangfire supports both fire-and-forget and recurring jobs, ensuring timely execution.

3.4.3 MediatR

The MediatR pattern is a pattern and library used in .NET platform for software development. This pattern facilitates message-based communication and makes the code base more flexible and maintainable. When used in conjunction with the Onion architecture, it offers several benefits. Onion architecture is based on the principles of abstraction and inversion of dependencies, organizing the application into layers around the core. MediatR reduces dependencies in this architecture and prevents clients from directly communicating with other layers. Following the principle of creating a separate class for each operation or command, MediatR supports the Single Responsibility Principle and enhances code base maintainability. Moreover, it enables easy addition of new operations or commands and modification of handlers, thereby improving the flexibility and extensibility of the application. MediatR also facilitates isolation and testing of handlers, enhancing testability. This combination allows for writing more readable, maintainable, and testable code.

3.4.4 EMGU.CV

EMGU.CV is an open-source cross-platform .NET wrapper for the popular computer vision library OpenCV. It serves as a bridge between .NET applications and OpenCV, allowing developers to harness the extensive capabilities of OpenCV within their .NET projects. With EMGU.CV, developers can perform a wide range of computer vision and image processing tasks, leveraging functions for image filtering, feature detection, object recognition, video processing, and more. EMGU.CV is designed to optimize performance, offering seamless integration with .NET frameworks and efficient memory management. It provides comprehensive documentation, code samples, and an active community, making it an ideal choice for developing powerful and efficient computer vision applications in the .NET ecosystem.

3.4.5 Entity Framework Core

Entity Framework Core is a lightweight, open-source object-relational mapping framework developed by Microsoft. It simplifies database access and manipulation by providing a high-level abstraction layer between applications and databases. With Entity Framework Core, developers can work with relational databases using object-oriented programming concepts, mapping database tables to .NET objects. It supports multiple database providers, integrates seamlessly with language integrated euery for querying data, and includes a migration feature for managing database schema changes. Entity

Framework Core is cross-platform and optimized for performance, offering extensibility and reducing the need for boilerplate code.

3.4.6 Autofac

Autofac is an open-source, lightweight, and extensible dependency injection container for .NET applications. It simplifies dependency management and follows the Inversion of Control principle, allowing for loosely coupled and modular design. Autofac provides a fluent API for registering and resolving dependencies, supporting various lifetimes to control object creation and disposal. It promotes modularity and composition through the use of modules, facilitating the composition of application components from reusable parts. Autofac integrates seamlessly with different .NET frameworks and technologies, enabling easier integration into various application types. With its flexibility and extensibility, Autofac enhances code maintainability, scalability, and testability by managing and resolving dependencies in a clean and efficient manner.

3.4.7 ML.NET

ML.NET stands as an open-source machine learning framework meticulously crafted by Microsoft, designed to cater specifically to the requirements of .NET developers. This dynamic framework is equipped with an intuitive API and is deeply integrated with the .NET ecosystem, thereby empowering developers to seamlessly infuse machine learning competencies into their applications. ML.NET furnishes developers with the capacity to train and employ machine learning models for a diverse range of tasks, encompassing classification, recommendation, and more. It advocates flexibility in model generation and deployment, offers robust capabilities for data transformation, and is furnished with evaluation metrics along with explainability features. Being backed by a vibrant and active community, ML.NET stands as a potent asset for .NET developers endeavoring to harness the power of machine learning in their applications.

3.4.8 Swashbuckle

Swashbuckle is an open-source library that seamlessly integrates with ASP.NET Web API or ASP.NET Core, enabling the automatic generation of interactive API documentation and client SDKs. Its primary objective is to streamline API documentation, offering an intuitive user interface for effortless exploration and testing of APIs. Additionally, it facilitates the generation of client SDKs to enhance the consumption of APIs with ease.

3.4.9 Automapper

AutoMapper is an open-source library designed to streamline the mapping of data between diverse object models within .NET applications. It offers an automated approach to the mapping process, diminishing the reliance on manual mapping code and elevating code maintainability. By leveraging AutoMapper, developers can significantly simplify the complex task of transferring data between various structures, enhancing overall productivity and ensuring efficient application development.

3.4.10 Fluent Validation

FluentValidation is an open-source library that empowers developers to define and execute validation rules on data models within .NET applications in a clear and expressive manner. By providing a fluent API, it simplifies the process of writing validation code and offers support for a wide range of validation scenarios. With seamless integration into popular .NET frameworks, Fluent Validation fosters data integrity and enhances the overall user experience by enforcing compliance with pre-defined validation rules. Its versatile capabilities enable developers to create robust and maintainable validation logic, promoting the delivery of high-quality software.

3.4.11 Log4net

Log4net is a powerful open-source logging framework that empowers developers to seamlessly integrate comprehensive logging capabilities into their .NET applications. With its flexible logging levels, versatile output destinations, and customizable log formats, Log4net facilitates efficient logging and log management. By capturing detailed log information, Log4net plays a crucial role in application troubleshooting and debugging, enabling developers to effectively analyze and resolve issues. Its robust features and extensibility make it an invaluable tool for enhancing application reliability, performance, and maintainability.

3.5 Client-side

A client application refers to a software program that operates on a client device and facilitates the interaction with a server to access resources or services. Typically developed using client-side programming languages like HTML, CSS, and JavaScript, client applications are executed on the client device rather than the server. They play a pivotal role in client-server systems and are utilized across various domains, including web applications, mobile

applications, and more. Client applications perform diverse tasks such as data presentation, user input management, communication with the server, and local processing. As a fundamental component of numerous software systems, client applications contribute significantly to the seamless delivery of services and user experiences.

3.5.1 Angular

Angular is a cutting-edge front-end web development framework widely employed for constructing high-performance single-page applications. Developed and maintained by Google, Angular leverages TypeScript, an enhanced version of JavaScript, to enable seamless application development. Its core objective is to simplify the creation of dynamic, interactive, and responsive web applications by providing a comprehensive suite of tools and frameworks. Angular encompasses robust features, including powerful data binding, routing capabilities, and a user interface development toolkit. Moreover, it offers developers essential tools for building scalable and maintainable applications, such as dependency injection, modularization, and a reactive programming model. With its emphasis on performance and developer productivity, Angular has emerged as a go-to framework for developing sophisticated web applications.

3.6 DevOps

3.6.1 Azure Portal

The Azure Portal serves as a critical web-based user interface for effectively managing and deploying resources within the Azure cloud environment, playing a pivotal role in the DevOps process. This powerful tool empowers developers, operations teams, and other stakeholders to efficiently create and oversee cloud resources, configure deployment pipelines, monitor application health and performance, enforce robust security controls, seamlessly integrate with diverse Azure services, and optimize costs. With its user-friendly interface and seamless integration with continuous integration and continuous deployment (CI/CD) tools, the Azure Portal provides a centralized and collaborative platform for efficient collaboration, monitoring, and management. By facilitating streamlined application lifecycle management in the Azure cloud, it enables organizations to embrace efficient DevOps practices, driving enhanced productivity and successful cloud-based deployments.

3.7 Web Socket

Represent a sophisticated protocol engineered to enable a full-duplex communication channel via a single TCP connection. The protocol is architected to support instantaneous, bidirectional communication between a client and server. For instance, as depicted in Figure 4, this feature enables the server to instantaneously push data to the client, eliminating the need for constant data requests from the server end. This protocol is of significant use in applications that demand real-time communication such as online gaming, chat applications, and collaborative tools. Furthermore, in scenarios demanding low-latency communication like financial trading platforms or live streaming applications, WebSockets stand as an optimum choice. Utilizing WebSockets necessitates support from both the client and server, and it's the client that triggers the connection via an HTTP request, subsequently upgraded to a WebSocket connection upon server approval. Consequently, real-time communication between the client and server is established, paving the way for efficient interactions and the creation of interactive and responsive applications.

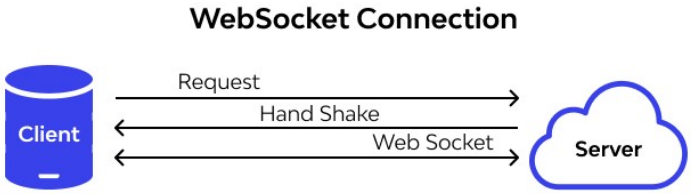

Figure 4: WebSocket Connection

3.7.1 SignalR

Pioneering real-time communication library specifically designed for .NET applications. Utilizing a diverse range of technologies and protocols, including WebSockets, Server-Sent Events (SSE), and long polling, SignalR successfully provides real-time communication. It boasts of an automatic transport mechanism selection feature, which considers the client's capabilities and network conditions. Developers can easily integrate SignalR into their .NET applications, leveraging its APIs to establish a robust connection between the client and the server. With the connection in place, the server can transmit real-time messages to the client and vice versa. Considering its vast applications in real-time communication scenarios, SignalR is a potent tool for crafting applications demanding real-time, low-latency, and bidirectional communication.

3.8 Message Queue

In the software development landscape, message queues play an instrumental role in fostering asynchronous communication between different components or systems. By offering a decoupled, asynchronous messaging protocol, message queues enable producers to publish messages and consumers to retrieve and process those messages at their convenience. Message queues guarantee the reliable delivery and durability of messages, preserving their order, and allowing for load balancing across multiple consumers even under circumstances involving failures or restarts. This mechanism significantly contributes to scalability and resilience in distributed systems. Hence, message queues are extensively utilized, presenting a reliable and efficient communication mechanism among software components while promoting loose coupling, scalability, and fault tolerance. You can see an example of this in the figure 5.

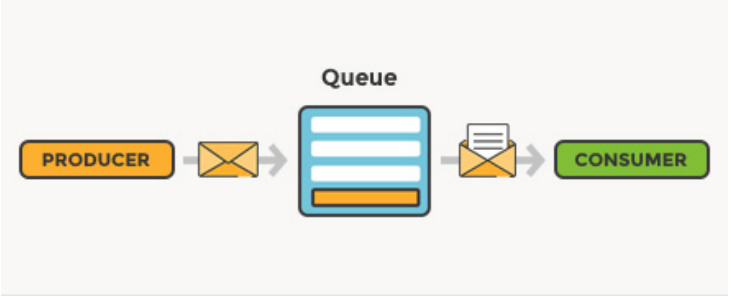

Figure 5: Message Queue

3.8.1 RabbitMQ

RabbitMQ serves as an influential open-source message broker that diligently facilitates seamless communication within distributed systems. Functioning as a middleman, RabbitMQ enables producers to transmit messages to exchanges, subsequently routed to various queues, adhering to predefined binding rules. Consumers, on the other hand, subscribe to these queues and receive messages to be processed. Its support for diverse exchange types and its provision for reliable message delivery, acknowledgments, and message persistence emphasizes RabbitMQ's resilience. Offering a flexible and scalable messaging solution, it aids in decoupling components and fostering asynchronous communication in a plethora of domains, including enterprise-level applications and microservices architectures. RabbitMQ's rich array of plugins and integrations further amplify its versatility, allowing for customizations and extensions to satisfy specific messaging requirements.

3.9 Object Oriented Programming

Object-oriented Programming (OOP) is a progressive programming paradigm anchored on the notion of "objects", encapsulating both data and the code manipulated to perform particular tasks. OOP uniquely structures software around objects that symbolize real-world entities like individuals, locations, and things. These objects converse via methods, which are functions inherently associated with a specific object. Additionally, OOP advocates for encapsulation, essentially concealing an object's internal details from external view. This approach paves the way for modifications to an object's implementation with minimal disruption to other system components, underscoring OOP's elegance and robustness.

3.10 Aspect Oriented Programming

Aspect-oriented Programming (AOP) is a dynamic programming paradigm aimed at enhancing modularity through the effective segregation of cross-cutting concerns. A cross-cutting concern intrudes on the conventional hierarchy of modules within a software system, impacting multiple system components. Examples of these concerns encapsulate logging, security, and performance monitoring. In AOP, such cross-cutting concerns are executed using "aspects", which are modules of code woven into the principal application code. This separation process fosters a more modular and maintainable system by segregating the cross-cutting concern from the main application code. AOP generally complements other programming paradigms, such as object-oriented programming or functional programming, providing a valuable toolset to handle concerns challenging to modularize through conventional techniques.

4 Application Design

4.1 Advert

When creating an advert, the first step is to select the type of advert you are looking for. We offer two types of adverts: "missing" adverts and "found" adverts. For both types, when the user uploads an image of the pet, a server-side method is triggered. This method is connected to an Image Classification Model, which accurately determines whether the pet is a cat or a dog. If the response from this model falls below the specified score threshold, an advert cannot be created.

```
1  CatDogImageClassification.ModelInput sampleDataSmall = new
       CatDogImageClassification.ModelInput()
2  {
3      ImageSource = imageBytes,
4  };
5  var prediction = CatDogImageClassification.Predict(
       sampleDataSmall);
6  float score = (prediction.Score[0] > prediction.Score[1] ?
       prediction.Score[0] : prediction.Score[1]);
7  if (score < .95)
8  {
9      return new ErrorDataResult<CreatePetDto>(PetMessages.
       PetClassificationLowScore);
10 }
```

Otherwise, The response from this model is then saved to the database. The Image Classification Model has been trained using a comprehensive dataset of dogs and cats. Additionally, the user is required to provide relevant information in the description section of the advert. The location information will be automatically gathered from the user's device or location settings. You can see an example of this in the figure 6.

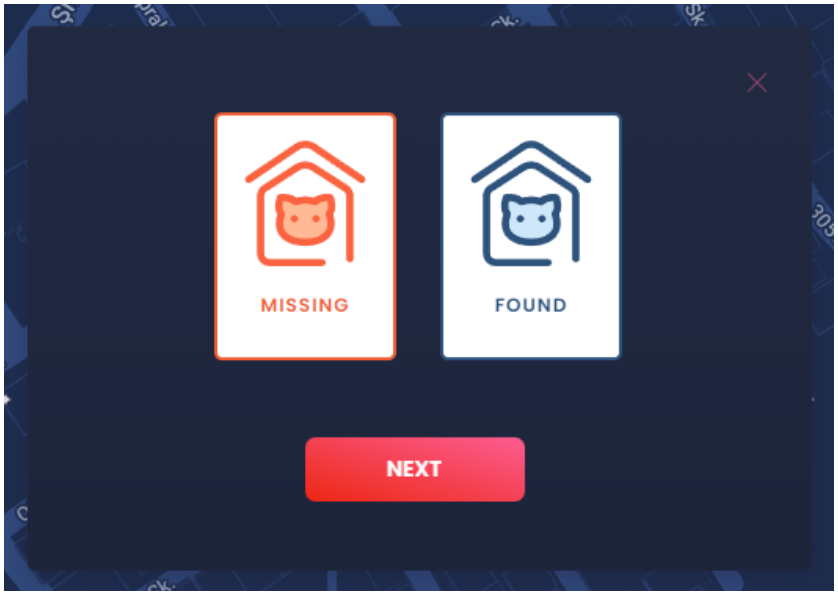

Figure 6: Creating an advert

4.1.1 Missing Advert

When it comes to creating a "missing" advert, users are required to have an account. This specific advert type is intended for individuals who are searching for their lost pets. The user creating the advert can provide detailed information about the missing pet. Additionally, the user has the ability to update the advert, declaring that the pet is no longer missing and specifying the user who found it. This feature allows for effective communication and collaboration between the pet owners and those who have found the missing pet.

4.1.2 Found Advert

When it comes to creating a "found" advert, users are not required to have an account. This particular advert type is designed for individuals who have come across a lost pet on the street. The person creating this advert is not obligated to physically transport the found pet anywhere. It allows anyone to create the advert, documenting the details of the found pet without the need for accompanying the animal.

4.2 Interactive Map

This component serves as the core of the application. It encompasses a tailor-made, interactive map that leverages the powerful Google Maps API.

4.2.1 All Adverts

Within this map, all adverts are displayed as markers, as depicted in Figure 6, based on their respective coordinates. You can see an example of this in the figure 7.

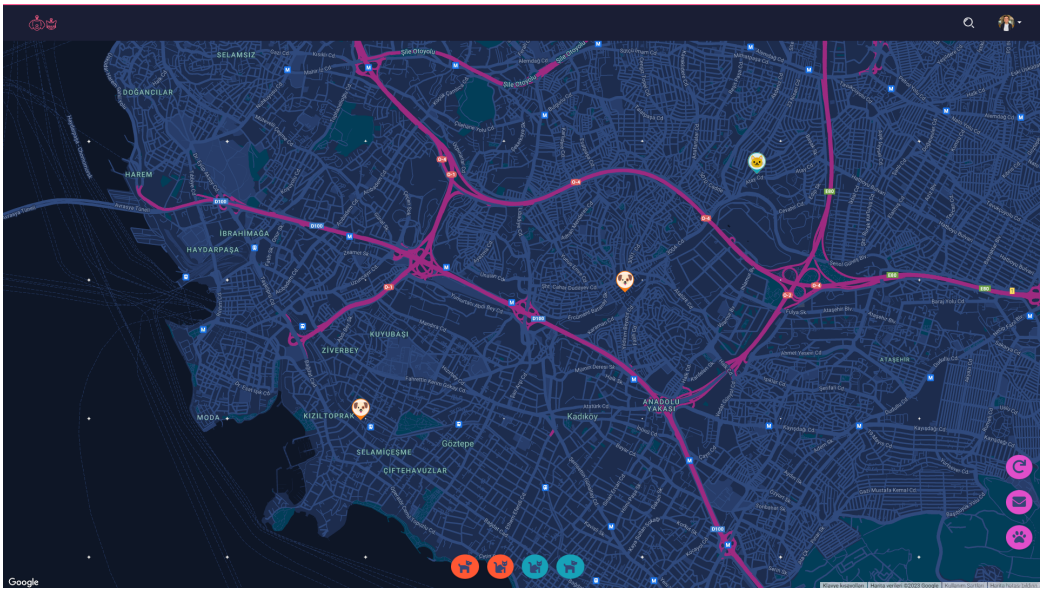

Figure 7: Interactive Map

4.2.2 Advert Details

Users have the ability to click on these markers to view the comprehensive details of the adverts. In addition, they have the option to share it with others through the social media buttons. You can see an example of this in the figure 8.

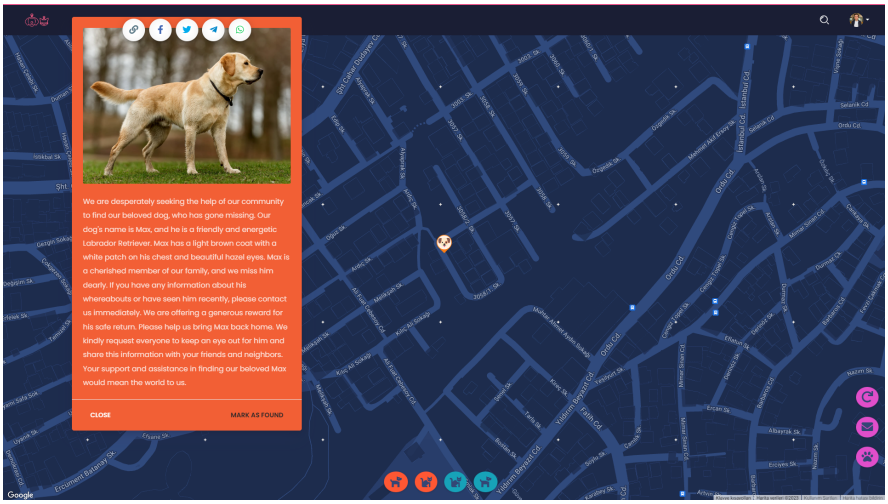

Figure 8: Advert Detail

4.2.3 Filters

In order to make the information more accessible and organized, there are four buttons available that allow users to filter the markers on the map according to their preferences. These buttons help users focus on specific types of markers, such as reset, missing dogs, missing cats, found dogs, or found cats.

4.3 Smart Search

The smart search conveys two pieces of information to the server: the user's location and the photo of their pet. Firstly, the server compiles a list of nearby pets, followed by the utilization of our developed image classification model to identify same types of pets. Once this identification process is complete, the comparison takes place between the input photo and the other posted adverts, considering three aspects: proximity of location, posting date of the advert, and the matching score of the photos. Subsequently, the server sends this list, along with their respective scores, back to the client. You can see an example of this in the figure 9.

```
double angle = 2 * Math.Atan2(Math.Sqrt(a), Math.Sqrt(1 - a));
var distanceScore = angle * RADIUS;
distanceScore = Math.Max(50 - (distanceScore / 100), 0);
if (distanceScore == 0)
{
    return null;
}
var timeScore = Math.Max(30 - (int)(Math.Abs((pet.CreatedDate -
    currentTime).TotalMilliseconds) / (double)TimeSpan.FromDays
    (1).TotalMilliseconds), 0);
using (Mat modelImage = CvInvoke.Imread(outputPath, ImreadModes.
    Grayscale))
using (var matches = new VectorOfVectorOfDMatch())
using (Mat observedImage = CvInvoke.Imread(Environment.
    CurrentDirectory + "\\wwwroot\\Images\\" + pet.ImagePath,
    ImreadModes.Grayscale))
{
    EmguCvImageSimilarity.FindMatch(modelImage, observedImage,
    out long matchTime, out VectorOfKeyPoint modelKeyPoints, out
    VectorOfKeyPoint observedKeyPoints, matches, out Mat mask,
    out Mat homography, out score);
}
```

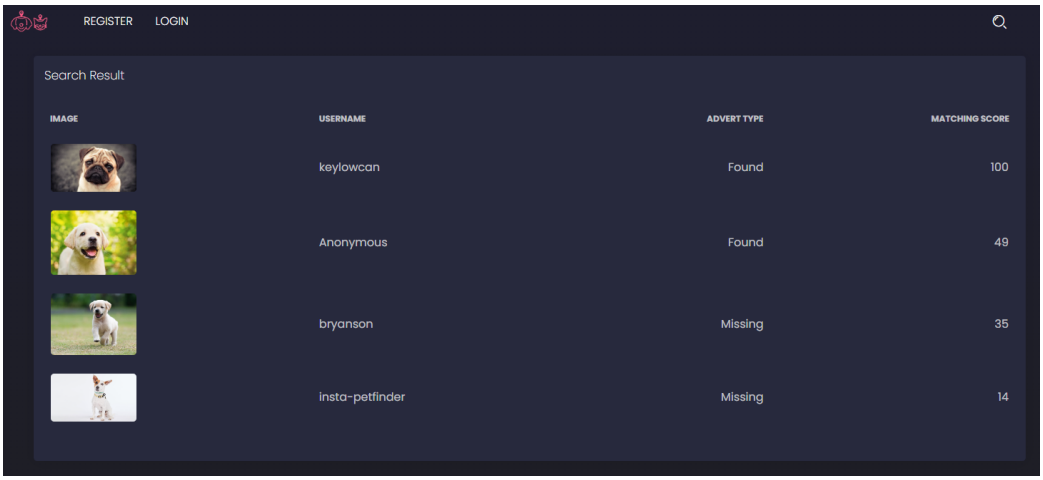

Figure 9: Smart Search Results

4.4 Auth

We have built a structure where some of the backend API endpoints will only allow authorized requests.

4.4.1 Register

The values obtained on the registration screen are being sent to the server, including variables FirstName, LastName, UserName, PhoneNumber, BirthDate, CountryId, Email, Image, and Password. You can see an example of this in the figure 10.

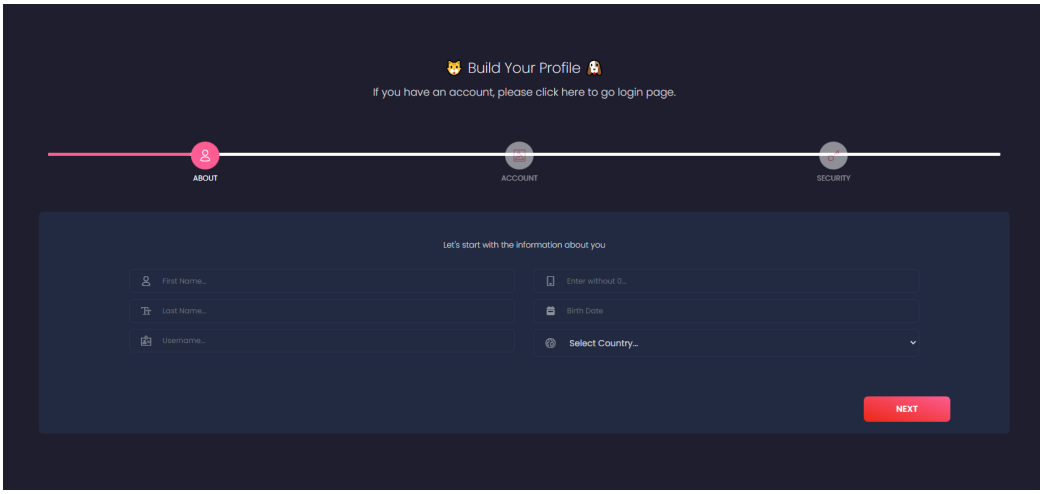

Figure 10: Register Screen

4.4.2 User Login

The values obtained on the user login screen are being sent to the server, including variables Email and Password. You can see an example of this in the figure 11.

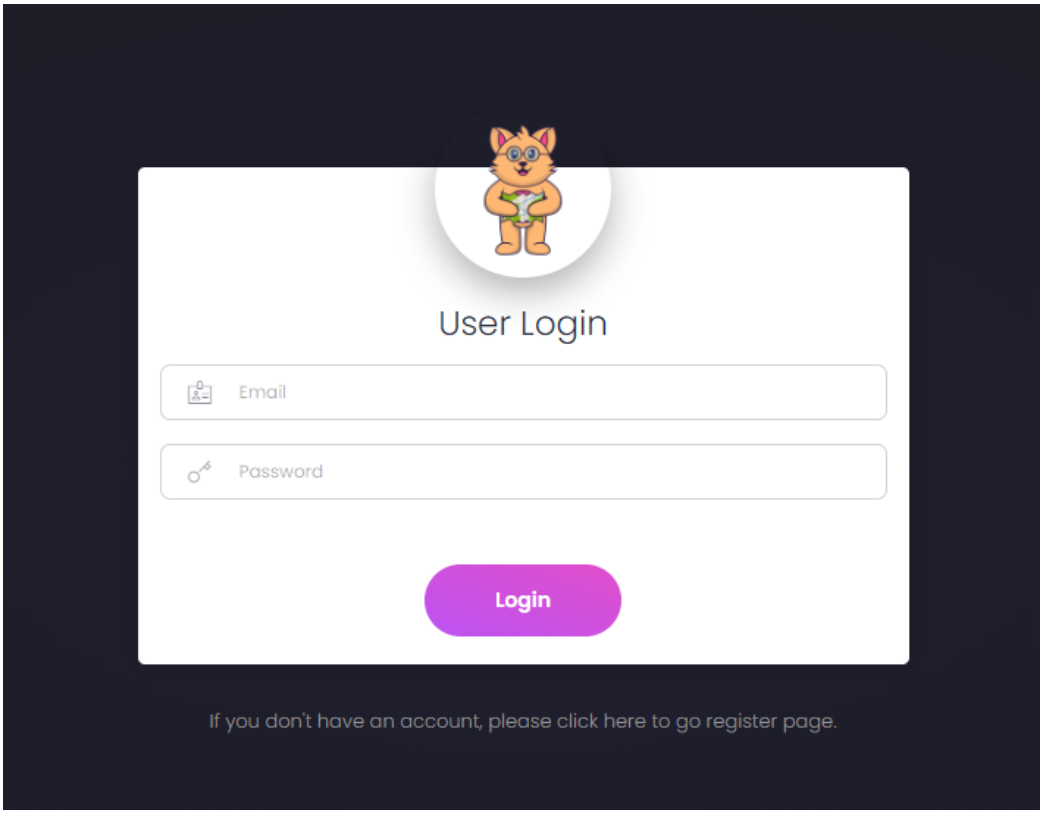

Figure 11: User Login Screen

4.4.3 Admin Login

Same with user login, the values obtained on the admin login screen are being sent to the server, including variables Email and Password.

4.5 User

A user section is automatically generated for every account created, enabling access to numerous features within the application. You can see an example of this in the figure 12.

Figure 12: Badges

4.5.1 Profile

In this section, users can view their own profiles as well as those of other users. The badges that they earn from completing tasks are also displayed on this screen. As this was designed to be a social welfare project, instead of allowing money transfers, we are introducing a gamification method known as a rewarding system. Users can earn various types of badges based on the number of pets they have found. In addition, the posted missing and found pets by the user are displayed in a list format. Pets that users have found from the missing adverts posted by others are also shown here. Users can click on a pet to go to the detailed page of the advert. You can see an example of this in the figure 13.

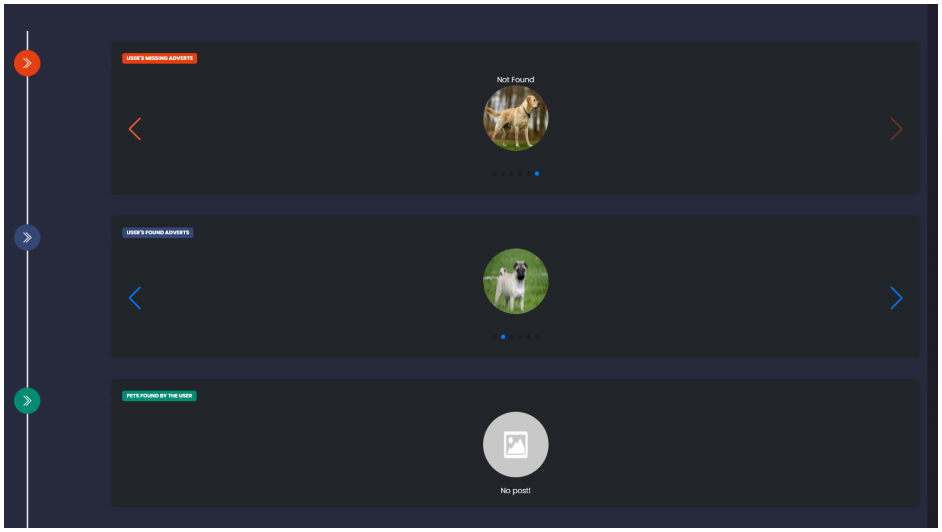

Figure 13: User Profile

4.5.2 Settings

The Settings page is where we can update our profile. From here, we can reset our password and also activate our user account. It's necessary to activate the account to access all features such as sending messages. Activation codes are generated on-the-spot and stored in the 'ActivationCodes' table for reference in our database. You can see an example of this in the figure 14 and 15.

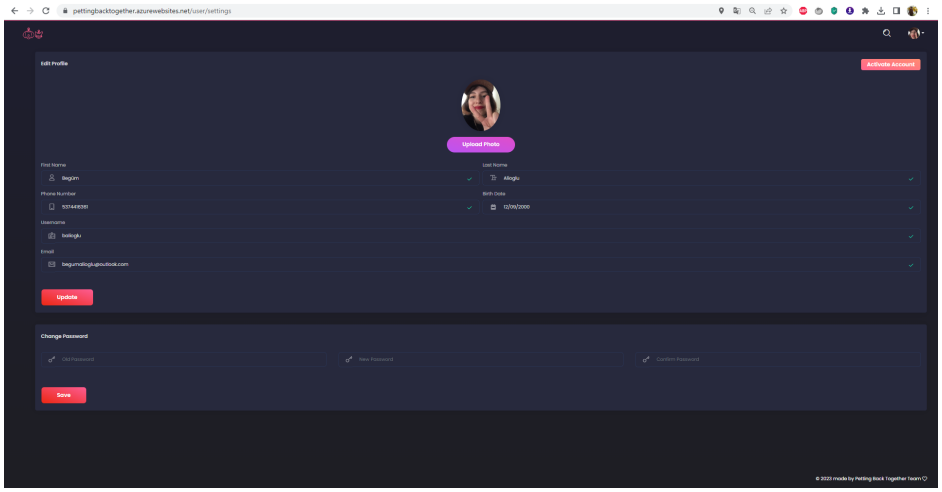

Figure 14: Settings Page

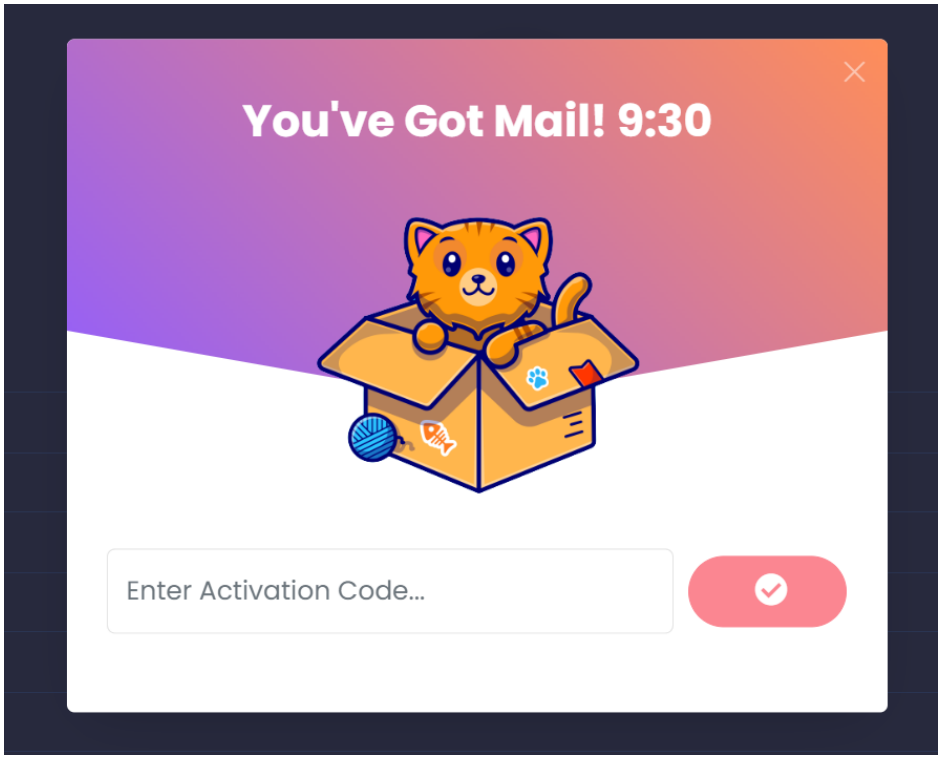

Figure 15: Activation Code Form

```
1  [UserActiveAspect(false)]
2  public async Task<IDataResult<Token>> Handle(
       ActivateUserCommandRequest request, CancellationToken
       cancellationToken)
3  {
4      var userId = _userService.GetCurrentUsersId();
5      var activationCodeResult = await _activationCodeService.
       CheckIfActivationCodeForUserIsCorrectAsync(userId, request.
       Code);
6      if (activationCodeResult.Success)
7      {
8          var user = await _userService.ActivateUserAsync(userId);
9          if (user.Success)
10         {
11             return _userService.CreateAccessToken(user.Data,
       activationCodeResult.Message);
12         }
13         else
14         {
15             return new ErrorDataResult<Token>(user.Message);
16         }
17     }
```

```
18      else
19      {
20          return new ErrorDataResult<Token>(activationCodeResult.
        Message);
21      }
22  }
```

4.5.3 Real Time Chat

Real-time chat, developed using SignalR, is an integral component of a service, enabling seamless and synchronous communication among users. This feature-rich functionality includes instant messaging, allowing users to exchange concise messages in real-time. Additionally, it supports the utilization of online chat platforms to facilitate these interactions. To access and fully leverage the real-time chat feature, it is imperative for users to possess a valid user account, ensuring secure and personalized communication experiences. You can see an example of this in the figure 16.
Client Side:

```
1  start(hubUrl: string) {
2    hubUrl = this.baseSignalRUrl + hubUrl;
3    const builder: HubConnectionBuilder = new HubConnectionBuilder
       ();
4    const hubConnection: HubConnection = builder
5      .withUrl(hubUrl, {
6        accessTokenFactory: () => localStorage.getItem("
       accessToken"),
7        skipNegotiation: true,
8        transport: signalR.HttpTransportType.WebSockets,
9      })
10     .withAutomaticReconnect()
11     .build();
12   hubConnection
13     .start()
14     .then(() => console.log("Connected"))
15     .catch((error) => setTimeout(() => this.start(hubUrl), 2000)
       );
16   hubConnection.onreconnected((connectionId) => console.log("
       Reconnected"));
17   hubConnection.onreconnecting((error) => console.log("
       Reconnecting"));
18   hubConnection.onclose((error) => console.log(error));
19   return hubConnection;
20 }
```

Server Side:

```
1  private readonly static ConnectionMapping<int> _connections =
       new ConnectionMapping<int>();
2  protected IHubContext<MessageHubService> _context;
3  public MessageHubService(IHubContext<MessageHubService> context)
4  {
5      _context = context;
6  }
7  public async Task SendMessage(string text, int receiverUserId,
       int senderUserId)
8  {
9      foreach (var connectionId in _connections.GetConnections(
       receiverUserId))
10     {
11         await _context.Clients.Client(connectionId).SendAsync(
       ReceiveFunctionNames.SendMessage, text, senderUserId,
       receiverUserId);
12     }
13 }
```

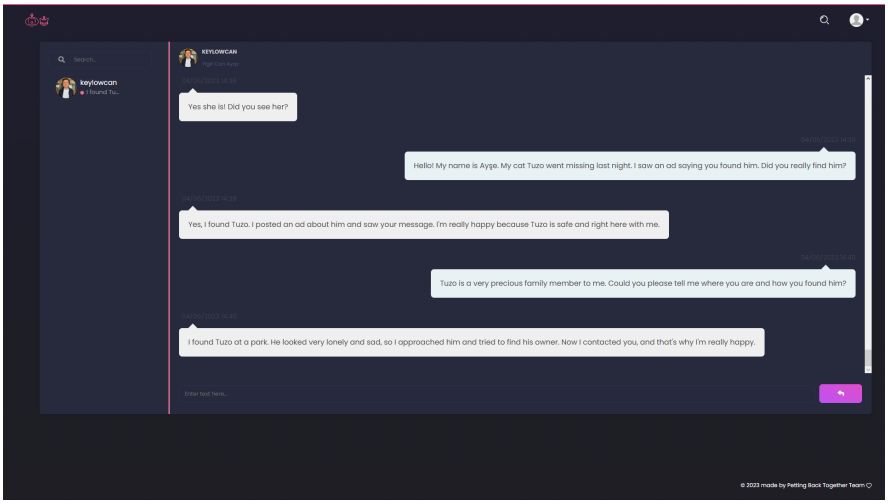

Figure 16: Messages

4.6 Admin Panel

An admin panel, alternatively referred to as an administration panel, is a sophisticated web-based interface designed to facilitate the efficient management and supervision of diverse functions and features within a website or application. This comprehensive tool empowers users with the capability to seamlessly oversee content, users, and other crucial aspects of the site or application. Typically utilized by website administrators, content editors, and authorized personnel possessing the requisite permissions, admin panels play a pivotal role in enabling authorized individuals to make necessary modifications to the site or application. It is worth noting that access to this page is restricted solely to users with an admin role as defined in the UserOperationClaims table. You can see an example of this in the figure 17.

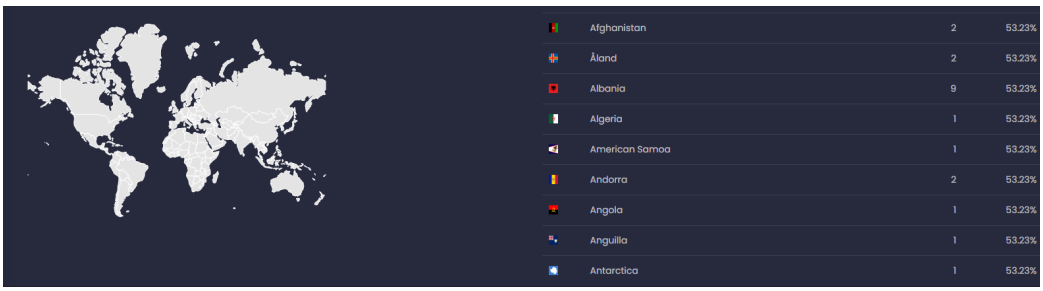

Figure 17: Dashboard

4.6.1 Managing Data

Data management encompasses the systematic organization, secure storage, and continuous maintenance of the data amassed by an organization. This multifaceted endeavor entails a range of tasks, including the acquisition and consolidation of data from diverse sources, its meticulous storage in a secure and structured manner, and the diligent upkeep to ensure its accuracy and currency. Within software applications, the management of data revolves around performing CRUD operations, which stands for Create, Read, Update, and Delete. These fundamental operations serve as the cornerstone for data management in databases, enabling the creation, retrieval, modification, and removal of data. In our specific context, we undertake CRUD operations for User Operation Claims, Operation Claims, Badges, Advert Types, and Species, enabling efficient management and control over these entities within our system. You can see an example of this in the figure 18.

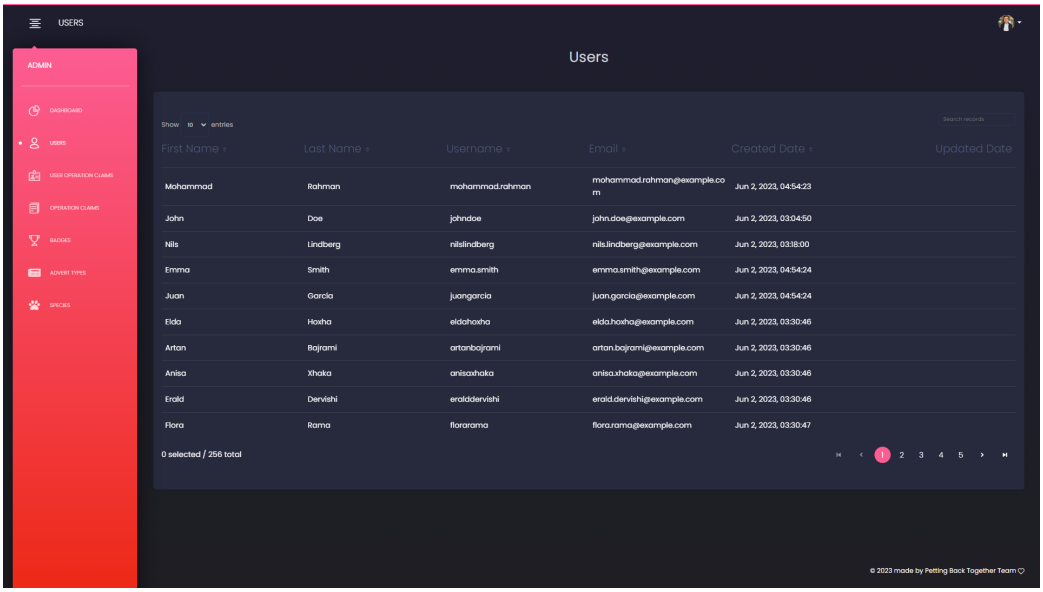

Figure 18: Managing Data

4.7 Smart E-mail Notification Service

Instead of manually checking the map every day after a person enters a missing advert, we have developed an algorithm that automatically compares the photos in the vicinity of where the advert was posted. When a "found" advert with matching photos above a certain score is entered, an email is sent to the person who submitted the "missing" advert. You can see an example of this in the figure 18.

```
1  var similarPets = await _imageSimilarityService.CheckSimilarity(
       request.Image, closePets.Data);
2  if (similarPets.Data.Count > 0)
3  {
4      var currentUserEmail = _userService.GetCurrentUsersEmail();
5      var emails = similarPets.Data
6          .Where(p => p.PostedUserEmail != currentUserEmail)
7          .Select(p => p.PostedUserEmail)
8          .Distinct()
9          .ToArray();
10     if (similarPets.Success && similarPets.Data != null &&
       emails.Length > 0)
11     {
12         await _mailService.SendBulkMailAsync(emails, "You have
       match!", MailMessages.SmartEmailNotification + " <a href='
       https://pettingbacktogether.azurewebsites.net/pets/" + pet.
       Data.Id + "'> Matched Pet </a>");
13     }
```

```
}
```

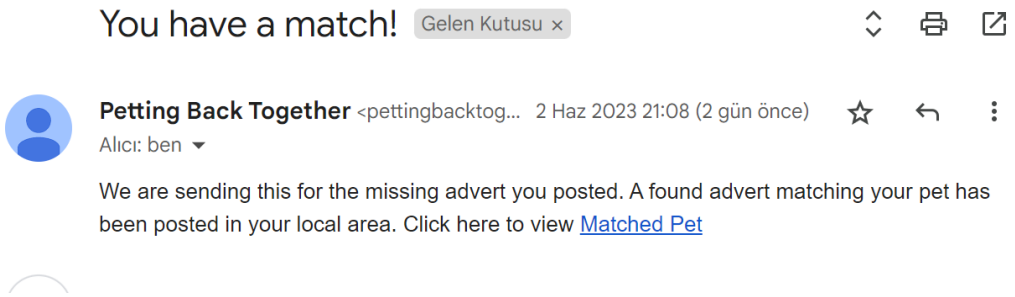

Figure 19: Mail Example

4.8 Subscription Service

Users residing in Turkey can stay informed about missing adverts in their districts by subscribing to their respective districts. The functioning principle of this system is as follows: when a missing advert is posted, it is added to a message queue (RabbitMQ). Thanks to Hangfire, the consumer method of this queue runs every 23 hours, identifying the districts of the entered missing adverts and sending a collective email to the subscribers in those districts. You can see an example of this in the figure 20 and 21.
Code for adding missing adverts to the message queue

```
public void AddToQueue(SubscriptionQueueDto subscriptionQueueDto
    )
{
    var connection = _factory.CreateConnection();
    var channel = connection.CreateModel();
    channel.QueueDeclare(queue: "ptb-subscription-mail",
                         durable: true,
                         exclusive: false,
                         autoDelete: false,
                         arguments: null);
    var properties = channel.CreateBasicProperties();
    properties.Persistent = true;
    var bodyMessage = Encoding.UTF8.GetBytes(JsonConvert.
    SerializeObject(subscriptionQueueDto));
    channel.BasicPublish(exchange: "",
                         routingKey: "ptb-subscription-mail",
                         basicProperties: properties,
                         body: bodyMessage);
    channel.Close();
    connection.Close();
```

```
19 }
```

The part where we process the missing adverts that we pull from the queue

```
foreach (var districtPet in districtPets)
{
    int districtId = districtPet.Key;
    var emails = await _subscriptionService.
    GetAllEmailsByDistrictIdAsync(districtId);
    List<int> petIds = districtPet.Value;
    string bodyMail = MailMessage.SubscriptionBody;
    foreach (var petId in petIds)
    {
        Debug.WriteLine(petId);
        bodyMail += "<br/> <br/> <a href='https://
    pettingbacktogether.azurewebsites.net/pets/" + petId + "'>
    Matched Pet </a>";
    }
    await SendBulkMailAsync(emails, "Daily missing adverts
    notification", bodyMail);
}
```

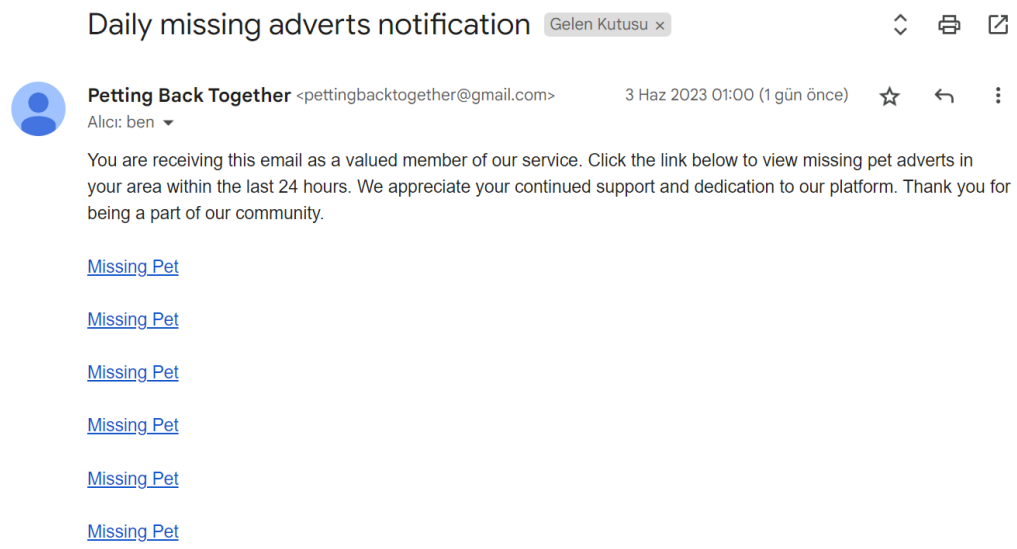

Figure 20: Subscription Mail

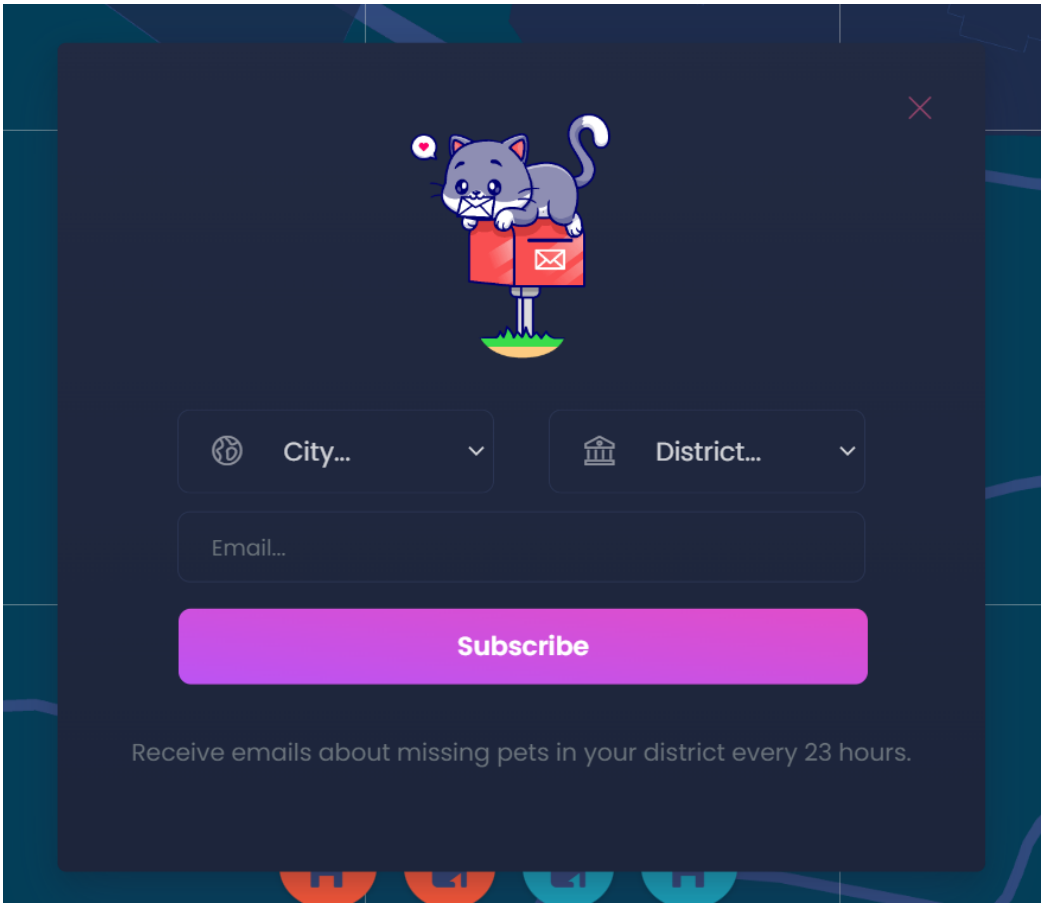

Figure 21: Subscription Form

4.9 Onion Architecture

Onion Architecture is a popular software design approach in the industry, particularly on platforms like .NET. This approach organizes different layers and dependencies of a software system, creating a hierarchical structure akin to an 'onion.' Central to its philosophy is arranging dependencies from the outside in, allowing the outer layers to depend on the inner ones, while keeping the latter independent. This arrangement isolates the core business logic of the software from external influences, enhancing its testability, flexibility, and maintainability.

Adhering to Onion Architecture allows for a modular and clean separation of concerns, resulting in a more robust and adaptable software system. It offers several benefits such as promoting modularity and low dependency. This facilitates the independent development and modification of components, fostering a modular code base and increased reusability. The architecture also enhances testability by controlling dependencies between layers, which allows for easier and independent testing of software components, especially the business logic layer.

Onion Architecture's flexibility means changes in one layer have minimal impact on others, which simplifies adapting to evolving requirements. Additionally, it improves manageability by organizing the code base into distinct layers, making it more comprehensible, organized, and scalable. Consequently, Onion Architecture is a preferred approach in .NET projects, leading to the creation of cleaner, more organized, and testable code bases. You can see an example of this in the figure 22.

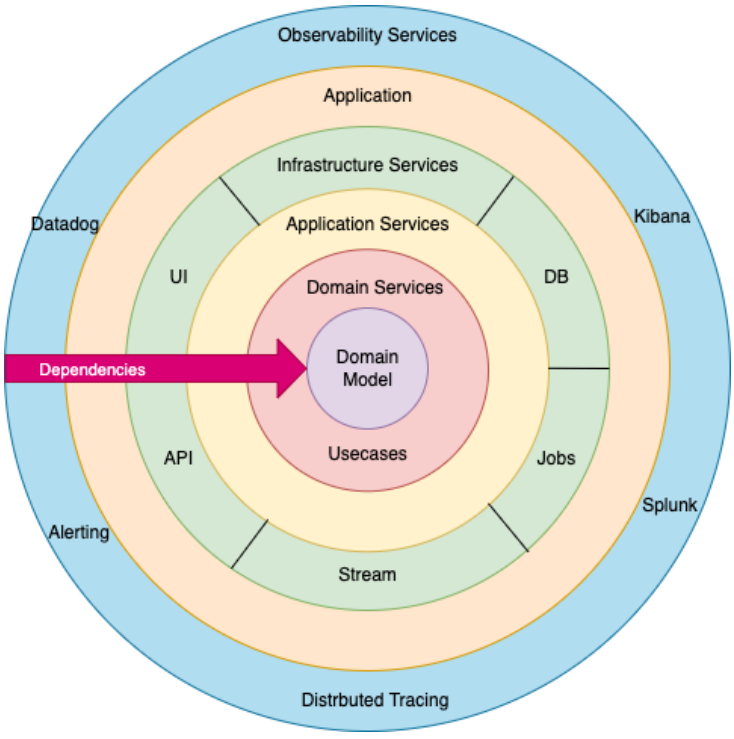

Figure 22: Onion Architecture

4.9.1 Application Layer

The Application Layer is a crucial component in the software architecture, encapsulating all interfaces within the application. It plays a pivotal role as a conduit, orchestrating interactions between various software components. Instead of allowing direct communication with Persistence/Infrastructure layers, WebAPI is utilized, acting as an intermediary to facilitate communication and data flow. This layer hosts a multitude of functionalities and design patterns for proficient software operation. Among them, Constants and Data Transfer Objects (DTOs) are used to ensure reliable data exchange between layers, maintaining integrity and consistency. The MediatR Pattern is utilized in this layer to implement a mediator for handling requests and responses, providing a robust means for decoupling components and creating maintainable, loosely coupled systems. Hub Interfaces are included, helping to provide real-time communication capabilities. Mapping, particularly through tools like AutoMapper, is implemented to efficiently convert data between different object models, enhancing the scalability and maintainability of the software. Repository Interfaces abstract the data layer, promoting loose coupling and separation of concerns, enabling the application to switch between different forms of data storage. Request Parameters are used to regulate and filter data sent from the client to the server, enforcing business rules and improving security. Finally, Service Interfaces define contracts for services, fostering a clear separation of responsibilities and enabling dependency injection, which in turn increases the testability and scalability of the application. This layered approach, focusing on separation of concerns, promotes a maintainable, scalable, and robust software architecture that can efficiently handle complexity and growth over time.

4.9.2 Domain Layer

The Domain Layer, a fundamental part of any software architecture, is where the entities reside. Entity instances correspond to individual records in the database tables. Their properties map directly to table columns, and Navigation properties manage relationships between entities, reflecting associations like one-to-one, one-to-many, or many-to-many. We use these entities throughout our project in every aspect.

4.9.3 External Layer

The External Layer of a software architecture serves as the repository for the foundational files of a project. One distinct characteristic of this layer is its universality; the code written here should be independent of the project and reusable in other projects. This means that no code within this layer is written according to specific business rules, ensuring its broad applicability. Various components find their home in the External Layer. Aspects and Interceptors, for instance, facilitate cross-cutting concerns such as logging, caching, or transaction management, without cluttering the business logic. Cross Cutting Concerns itself is a concept that encapsulates concerns that span multiple parts of a system, like security and communication, which are independent of the business functionality. Extensions and Utilities provide generic functions that can be used across the application, reducing code repetition and enhancing maintainability. Similarly, Helpers offer support functions or classes that simplify complex operations, aiding in code readability and efficiency. On the security front, elements like Encryption, Hashing, and JSON Web Tokens (JWT) are employed. Encryption and Hashing ensure data integrity and confidentiality, while JWT provides a method for securing user sessions on a web application without needing to handle or store sensitive information like passwords.

4.9.4 AI Layer

In this layer, we train the image classification model here, enabling it to determine whether an image contains a cat or a dog. The AI layer in the software architecture is responsible for integrating and implementing AI capabilities within the system. Within this layer, we store files that are developed using ML.NET, Microsoft's machine learning framework, to accomplish the task of image classification.

4.9.5 Infrastructure Layer

The Infrastructure Layer primarily caters to services that are not directly interacting with the database. This layer encapsulates the services that support the application's operations, including the use of external libraries and APIs, without direct data persistence responsibilities. A prime example within this layer is the ImageSimilarityService, developed using the Emgu.CV library. Emgu.CV is a .NET wrapper to the OpenCV image processing library, and it facilitates the creation of complex image processing services. The ImageSimilarityService leverages this capability to enable image similarity comparisons within the application, a feature that augments the system's capabilities

without necessitating direct database interactions. Additionally, this layer handles the integration of external APIs, such as the BackgroundRemovalApi from OpenWish. This integration allows the application to leverage external functionalities like background removal in images, further enhancing the application's feature set.

4.9.6 Persistence Layer

The Persistence Layer, in the context of software architecture, is fundamentally concerned with data persistence operations, including data storage and retrieval. It houses all classes related to database operations, thereby serving as the bridge between the application and the database. One key component in this layer is Contexts. In the realm of Object-Relational Mapping (ORM) technologies, such as Entity Framework, the Context is the primary class that interacts with the database. It represents a session with the database, allowing querying and saving of data. Repository Classes also reside within the Persistence Layer. The Repository Pattern is a well-adopted design pattern that abstracts data access behind collection-like interfaces. Each repository class is usually associated with one entity type and provides methods for retrieving and storing that particular type of entity, hence isolating the rest of the application from the specifics of data access code. Service Classes in this layer typically encapsulate business logic related to data access, thereby preventing the domain classes from being cluttered with data access code. They provide an additional layer of abstraction over the repositories, often offering a place to implement caching, logging, and other cross-cutting concerns.

4.9.7 Presentation Layer

Responsible for interacting with the end-user or client. This layer hosts our API Controllers, which define and manage the endpoints for the system, serving as the gateway to the application's data and functionalities. These API Controllers are designed to handle various types of requests, including GET, POST, PUT, and DELETE operations. Each request type correlates to a different action within the application. GET requests are used to retrieve data, POST to send data, PUT to update data, and DELETE to remove data. The API Controllers' role is to receive these requests, interact with the underlying services and layers as needed, and ultimately return an appropriate response to the client. Also, it incorporates security mechanisms like the Authorize Aspect. This function checks whether incoming requests are authorized to access the data or functionalities they are requesting. In essence, it acts as a gatekeeper, ensuring that only valid and authorized requests are

processed.

4.9.8 SignalR Layer

The SignalR Layer in a software architecture is the segment dedicated to establishing real-time communication within the system. This is achieved using the SignalR library, a software library developed by Microsoft for ASP.NET developers that simplifies the process of adding real-time web functionality to applications. Real-time web functionality enables server-side code to push content updates to connected clients instantly as they become available. This means the server can actively push updates to the client, rather than the client having to request updates from the server. Within this layer, hubs like the MessageHub and AdvertHub are developed and maintained. A hub in SignalR is a high-level pipeline that allows a client and server to call methods on each other.

4.9.9 WebApi Layer

The WebApi Layer which is using ASP.NET Core 7 is the launching pad for the application. This layer primarily manages the references to other projects and acts as the glue that binds different layers and modules of the system together, effectively orchestrating the overall operation of the application. A crucial responsibility of the WebApi Layer is to handle various integrations and implementations that enhance the system's functionality and manageability. For instance, Hangfire, a framework that enables background job processing in .NET and .NET Core applications, is integrated within this layer. Another key component in this layer is the DependencyResolver. Dependency resolution, or Dependency Injection (DI), is a design pattern that helps manage dependencies between objects, promoting loose coupling and enhancing testability. By implementing DI in the WebApi Layer, we facilitate efficient management of dependencies across the entire application. Additionally, this layer incorporates Swagger, a tool used for building APIs with a common language that everyone can understand. Swagger provides a user interface to call our API endpoints, making it easier to test and debug them.

4.10 Aspects

Aspects are constructs that are applied to methods, allowing for a cleaner organization of the code block. By placing aspects on top of methods, we

ensure that the internal structure of the code remains tidy and uncluttered. We make use of various custom aspects to enhance our development process.

4.10.1 Cache Aspect

The aspect first checks if the results for the given input parameters are already available in the cache. If so, the cached result is returned immediately, saving the time and resources it would take to run the function again. If the result is not in the cache (a "cache miss"), the function is executed, and its result is stored in the cache before being returned.

4.10.2 Cache Remove Aspect

"Removing a Cache" typically refers to invalidating or deleting a cached value in the cache data structure. This may be done for a variety of reasons, such as when the cached value is no longer valid or accurate, or when the cache needs to be refreshed with new data.

4.10.3 Exception Log Aspect

This functionality is incorporated into every method in a service, serving as a universal error interceptor. In case an exception occurs while a method is being executed, the ExceptionLogAspect intercepts the error. Instead of the application crashing or falling into an undefined state, the error is logged along with pertinent information, such as the error's location, type, input parameters that triggered it, and possibly a stack trace. This information offers a comprehensive understanding of what went wrong and where, greatly facilitating the detection and resolution of errors in a faster and more efficient manner.

4.10.4 Log Aspect

The LogAspect serves the purpose of logging to the database. It is commonly used in service methods such as Create, Update, and Remove. It records a JSON file consisting of details, date, and audit information to the Logs table in the database.

4.10.5 Performance Aspect

We integrate the PerformanceAspect component into all of our business services. This feature enables us to monitor and track the execution time of the

methods involved, providing valuable insights into the performance characteristics of our services. By doing so, we can ensure efficient operation and optimization wherever necessary.

4.10.6 Secured Operation Aspect

The SecuredOperationAspect component is used to conduct role-based method authorization. Unlike the 'Authorized' functionality which only checks whether a user is logged in or not, the SecuredOperationAspect goes a step further to also consider the user's role. This is particularly important for API endpoints that are intended to be used solely by administrators. These endpoints must incorporate the SecuredOperationAspect to ensure that only users with the appropriate roles, such as admins, have access to the functionalities provided.

4.10.7 Transaction Scope Aspect

This code is used to ensure that methods that need to perform multiple operations on a database do so within a transaction, and that if any of the operations fail, none of them are committed to the database. This helps maintain data consistency and integrity.

4.10.8 User Active Aspect

This checks whether the user account is active or not. In order to access all features within the application, it is necessary for the user's account to be activated.

4.10.9 Validation Aspect

Validation aspect is an aspect that is used to implement data validation logic for an application. Validation aspects are used to ensure that the data being processed or persisted by an application is accurate and conforms to certain rules or constraints.

4.11 Customized ORB Detector

After identifying the type of animal, we need to determine the domestic animal it most closely resembles within its own species. To achieve this, we've developed an algorithm using the ORB class found in EMGU.CV. The FindMatch method is employed to identify a match between a model image and an observed image. To locate the match between the two images, we use

the ORB (Oriented FAST and Rotated BRIEF) feature detector and Brute-Force matcher. The ORB (Oriented FAST and Rotated BRIEF) is a widely used algorithm in computer vision applications for feature detection and descriptor computation, deriving its name from two distinct algorithms: FAST (Features from Accelerated Segment Test) and BRIEF (Binary Robust Independent Elementary Features).
FAST: This algorithm is used to detect features in images, commonly referred to as corners. These features are usually distinctive and unique points in the image, hence they are used in comparing images or matching from one image to another. FAST is designed to perform the feature detection operation quickly.
BRIEF: This algorithm computes a descriptor for a feature point using the intensity information of pixels around that feature point. This descriptor is used to determine whether the feature point can be matched somewhere else in the image or in another image. BRIEF descriptors are popular due to their speedy calculations and robust matching capabilities.
The algorithm we've developed using ORB takes two photographs as parameters, modelImage and observedImage. We need to send the modelImage without a background because the comparison image should only contain the animal's picture. Hence, we used PicWish's Background Removal API. After running this algorithm, it provides us with a match score for the two images. You can see an example of this in the figure 23.

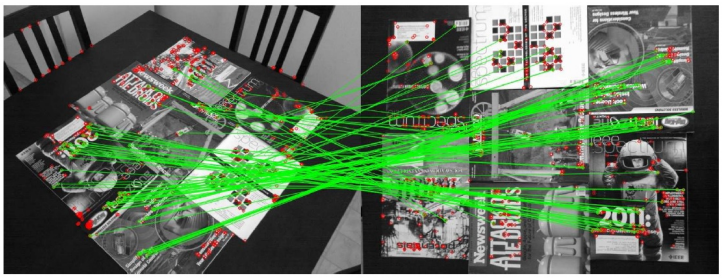

Figure 23: ORB Detector Matching Example

```
using (BFMatcher matcher = new BFMatcher(DistanceType.Hamming))
{
    matcher.Add(modelDescriptors);
    matcher.KnnMatch(observedDescriptors, matches, k, null);
    mask = new Mat(matches.Size, 1, DepthType.Cv8U, 1);
    mask.SetTo(new MCvScalar(255));
    Features2DToolbox.VoteForUniqueness(matches,
    uniquenessThreshold, mask);
    score = 0;
```

```
9       for (int i = 0; i < matches.Size; i++)
10      {
11          var value = (byte)mask.GetData().GetValue(i, 0);
12          int number = value;
13          if (number == 0) continue;
14          foreach (var e in matches[i].ToArray())
15              ++score;
16      }
17      int nonZeroCount = CvInvoke.CountNonZero(mask);
18      if (nonZeroCount >= 4)
19      {
20          nonZeroCount = Features2DToolbox.
    VoteForSizeAndOrientation(modelKeyPoints, observedKeyPoints,
    matches, mask, 1.5, 20);
21          if (nonZeroCount >= 4)
22              homography = Features2DToolbox.
    GetHomographyMatrixFromMatchedFeatures(modelKeyPoints,
    observedKeyPoints, matches, mask, 2);
23      }
24  }
```

4.12 Data Set

We will use Cats and Dogs Classification Dataset by SCHUBERTSLYSCHUBERT to train our image processing model. We will use this model to distinguish which pet is from which species [9].

4.13 API Endpoints

The followings are API Endpoints of our application. These pages are generated using Swagger. (Figures 25, 26, 27, 28)

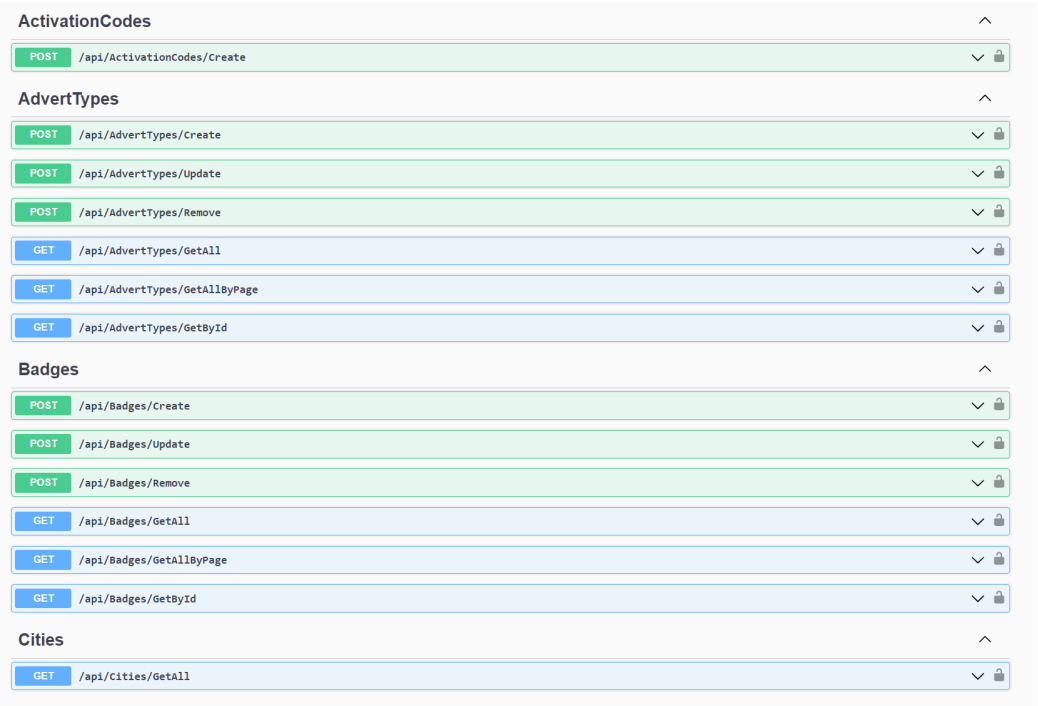

Figure 24: ActivationCodes, AdvertTypes, Badges, Cities Endpoints

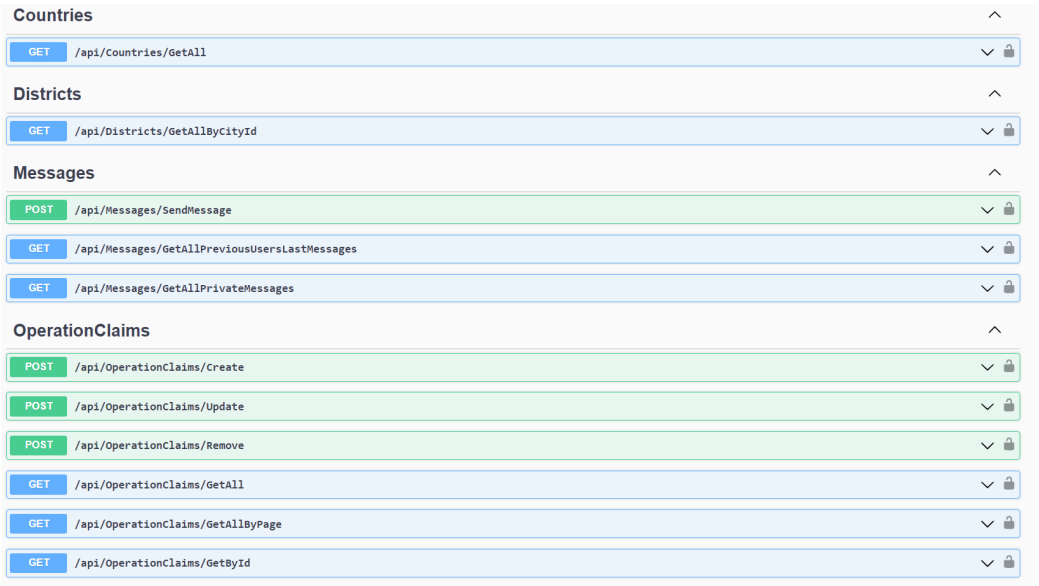

Figure 25: Countries, Districts, Messages, Operation Claims

Figure 26: Pets, Species Endpoints

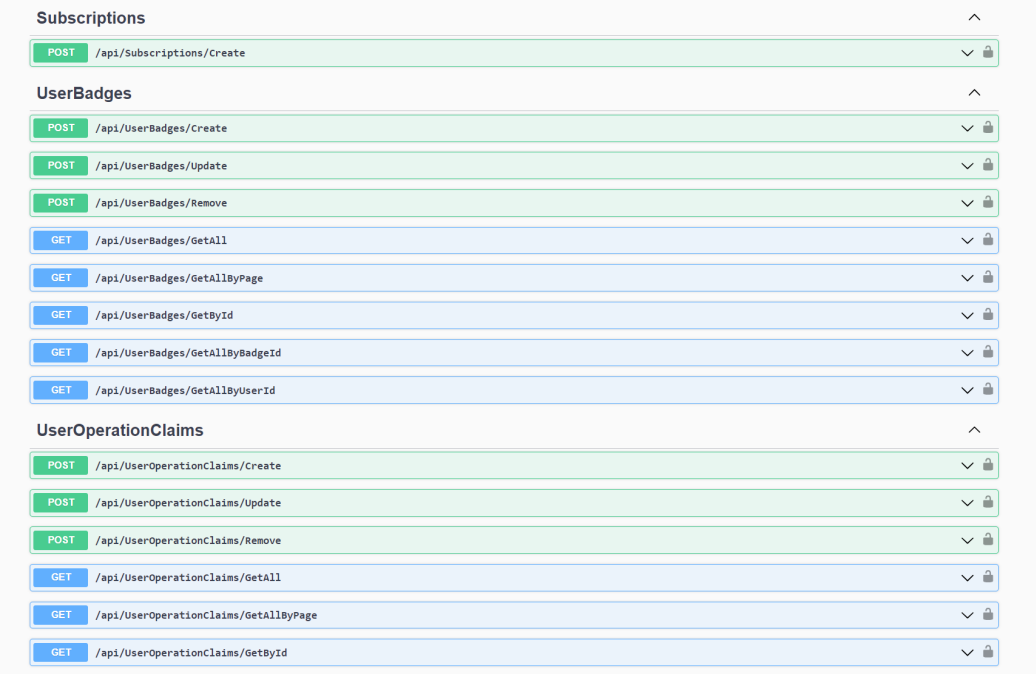

Figure 27: Subscriptions, UserBadges, UserOperationClaims Endpoints

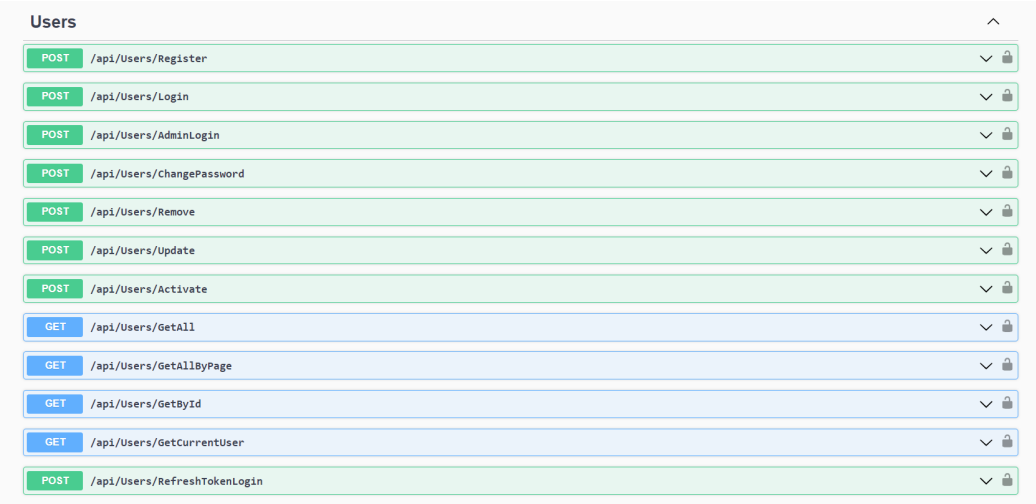

Figure 28: Users Endpoints

4.14 Database Diagram

Final version of the database diagram is shown in figure 9.

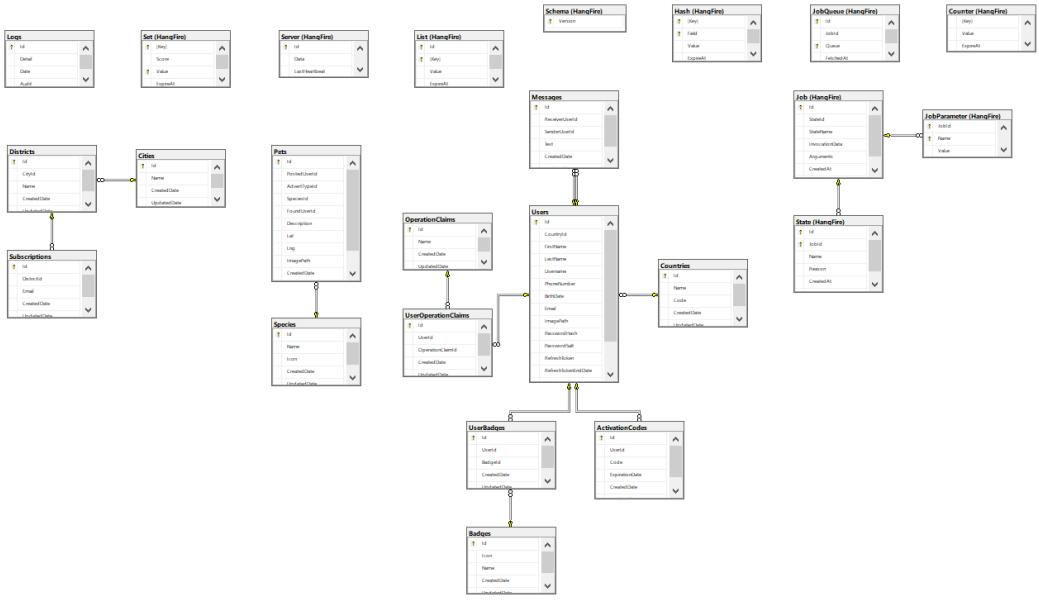

Figure 29: Petting Back Together Database

4.15 UML Diagram

Activity diagram for the application is given in figure 10.

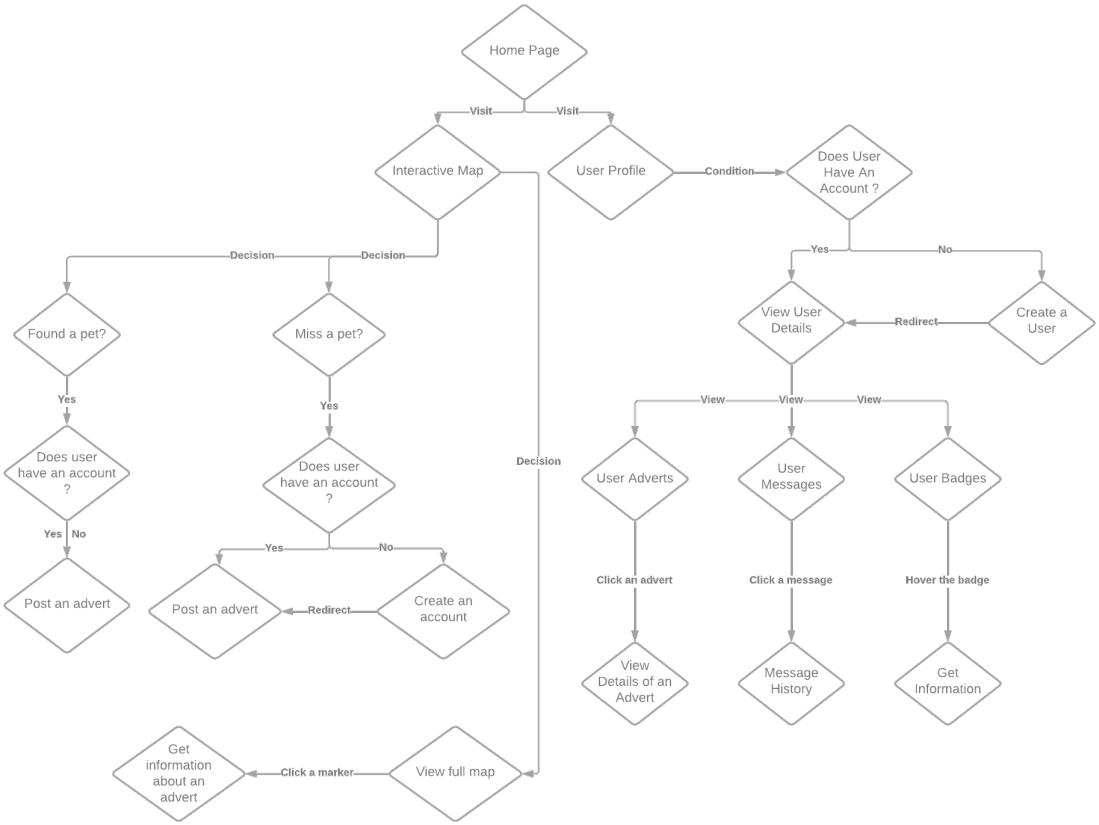

Figure 30: Activity Diagram

4.16 Security

4.16.1 Hashing and Salting

We ensure the security of passwords by applying a process known as hashing and salting. Hashing transforms the password into an irreversible and encoded form, making it practically impossible to decode. However, there exist pre-computed tables called rainbow tables that contain hash values for common passwords. To mitigate this vulnerability, we incorporate an additional layer of security called salting. When a user enters their password during the login process, we hash the password and apply salt to it. The salt is a random value unique to each user. The resulting hash is then compared with the stored hash and salt values in the database. If there is a match, the system generates a token. This token allows for authentication procedures to be performed securely. To implement this process, we utilize the HMACSHA512 algorithm, a widely recognized cryptographic function that ensures the integrity and authenticity of the hashed and salted passwords. This algorithm provides a robust security mechanism for protecting user passwords and defending against various forms of attacks, such as rainbow table attacks.

```
1  public static bool VerifyPasswordHash(string password, byte[]
       passwordHash, byte[] passwordSalt)
2  {
3      using (var hmac = new System.Security.Cryptography.
       HMACSHA512(passwordSalt))
4      {
5          var computedHash = hmac.ComputeHash(Encoding.UTF8.
       GetBytes(password));
6          for (int i = 0; i < computedHash.Length; i++)
7          {
8              if (computedHash[i] != passwordHash[i])
9              {
10                 return false;
11             }
12         }
13         return true;
14     }
15 }
```

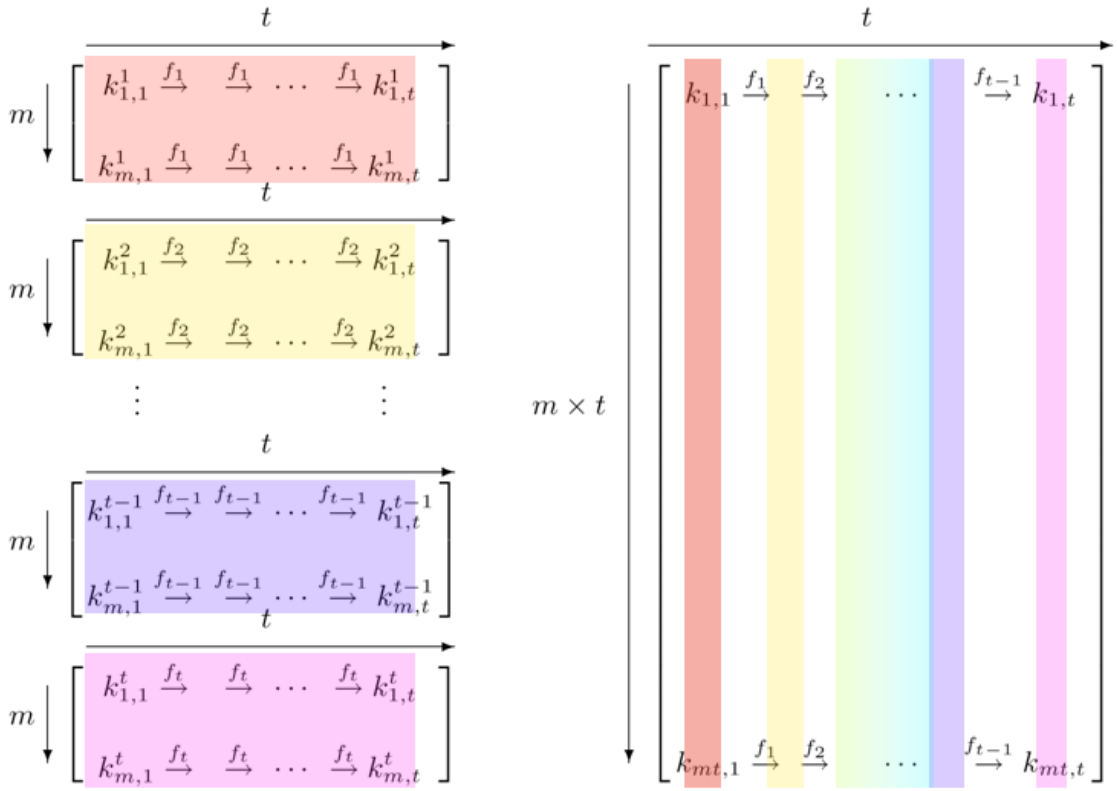

Figure 31: Rainbow Table Attack

4.16.2 Tokens

Bearer tokens are used to ensure security between the server and client. They eliminate the need to store sensitive credentials on the client-side and reduce the risk of unauthorized access. Bearer tokens are encrypted and digitally signed to prevent tampering. They also have a short lifespan and expiration mechanism to enforce re-authentication and limit the window of opportunity for attackers. Bearer tokens can be scoped and restricted to specific resources or actions, allowing for fine-grained access control. Typically, in such a system, two types of tokens are used: access token and refresh token.

Access Token: An access token is a short-lived token used to authenticate and authorize a user. It is usually valid as long as the user's session is active. The access token is used to access the resources (APIs, services, etc.) that the user is authorized to access. In an Angular or client-side application, the access token is provided with each request to gain access to resources on the server.

Refresh Token: A refresh token is a long-term token used to obtain a new access token after the user's session has expired. The refresh token is used to authenticate the user and obtain a new access token from the server. This allows the user to obtain access without renewing their session when the current access token has expired. The refresh token can have a longer expiration time and should be securely stored.

```
public Token CreateToken(User user, List<string>
    operationClaimStrings)
{
     _accessTokenExpiration = DateTime.Now.AddMinutes(
    _tokenOptions.AccessTokenExpiration);
     var securityKey = SecurityKeyHelper.CreateSecurityKey(
    _tokenOptions.SecurityKey);
     var signingCredentials = SigningCredentialsHelper.
    CreateSigningCredentials(securityKey);
     var jwt = CreateJwtSecurityToken(_tokenOptions, user,
    signingCredentials, operationClaimStrings);
     var jwtSecurityTokenHandler = new JwtSecurityTokenHandler();
     var token = jwtSecurityTokenHandler.WriteToken(jwt);
     return new Token
     {
          AccessToken = token,
          Expiration = _accessTokenExpiration,
          RefreshToken = CreateRefreshToken()
     };
}
```

4.17 Validation and Defensive Programming for requests

With Fluent Validation, we script rules concerning the structure of data that we receive as requests. Additionally, we implement defensive business logic. This not only ensures data integrity by preventing the storage of erroneous data, but also enhances the robustness and reliability of our system.

```
public class RegisterUserCommandValidator : AbstractValidator<
    RegisterUserCommandRequest>
{
     public RegisterUserCommandValidator()
     {
          RuleFor(u => u.FirstName).NotEmpty();
          RuleFor(u => u.LastName).NotEmpty();
          RuleFor(u => u.Username).NotEmpty();
```

```
8           RuleFor(u => u.PhoneNumber).NotEmpty();
9           RuleFor(u => u.BirthDate).NotEmpty();
10          RuleFor(u => u.CountryId).NotEmpty();
11          RuleFor(u => u.Email).NotEmpty();
12          RuleFor(u => u.Password).NotEmpty().MinimumLength(5);
13      }
14  }
```

```
1  var businessRulesResult = ResultBusinessRules.Run(await
       _userService.CheckIfUserUsernameExists(request.Username),
       await _userService.CheckIfUserEmailExists(request.Email),
       await _userService.CheckIfUserPhoneNumberExists(request.
       PhoneNumber), _userService.CheckIfUserIsNotUnderage(request.
       BirthDate));
2
3  if (businessRulesResult != null)
4  {
5      return new ErrorDataResult<Token>(businessRulesResult.
       Message);
6  }
```

5 Future Work

In the future, we aim to make several improvements to our algorithm that compares the similarities of the domestic animals we currently use. While the algorithm generally functions well, it can sometimes be confusing. Therefore, our goal is to enhance this algorithm to make it more reliable.

Currently, our application sends a list of missing pet adverts in the chosen region to our users every 23 hours. We aim to develop an enhancement that will expand this feature beyond a single region, allowing users to receive listings from multiple areas.

Regarding the design of our application, we have intentionally made the decision to prevent updates to the description sections of the submitted adverts. However, we are planning to make improvements to allow for updates in the future.

Our application is not currently a fully operational and widely used platform, which means that there are several legal aspects that need to be addressed. For instance, during the user registration process, we do not require users to provide consent to any text. We have ambitious plans for the development of our application and aim to address these legal aspects in the future.

At present, our registered users are unable to close their accounts or unsubscribe from the mailing list if they have opted-in. We intend to implement improvements to address these issues and provide users with better control over their memberships and mailing preferences.

6 Conclusion

The main purpose of the application is to reunite the missing pets with their families as soon as possible because they are more likely to face death during their stay outside. The missing pets you see on the outside and missing pets you can't find will be gathered on one platform. With the solutions we offer in the application, it will now be easier to find lost pets. When the user uploads an image of the missing pet an image classification model will run and will distinguish the species of the pet. After getting the species of the pet a similarity algorithm will run and will get the best matching adverts on that specific species. Then the adverts will be send to the user based on the location information in an descending order. Other solution that we are offering is notifying the person with e-mail who posted the missing pet advert, when another found advert posted with the same species and enough matching score in a close location.

As a result of our research and solutions, we have identified a few more features that we are likely to implement in the future to make the app even better. One of them is a service works with Image Similarity Model which compares the input image with other images in the adverts that users uploaded. This service will return images in an descending order according to their closest matching above 50% from our database.

Another feature we think we can do in the future is to add money transfer to the application. The money transfer can be between the person who finds the lost pet and the person who lost it, or it can be made to us by users to donate animal food through the application.

The last future feature that we can add is to create a pet database. In this database we will give a unique id to every pet that we store along with the the necessary information about the pet and the owner. In the future, it will be easier for the owner to find one of the pets in this database when the pet is missing.

References

[1] Vikipedi. "Evcil hayvan" https://tr.wikipedia.org/wiki/Evcil_hayvan

[2] Grajfoner D, Ke GN, Wong RMM. The Effect of Pets on Human Mental Health and Wellbeing during COVID-19 Lockdown in Malaysia. Animals (Basel), 2021. 14;11(9):2689. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8470955/

[3] Friedmann, E., Katcher, A., Lynch, J., & Thomas, S. Animal companions and one-year survival of patients after discharge from a coronary care unit. Public Health Reports, 95, 307–312. https://www.researchgate.net/publication/241645032

[4] Anna M. van Heeckeren, DVM. 10 Mental & Physical Health Benefits of Having Pets, 2021, 307–312. https://www.onehealth.org/blog/10-mental-physical-health-benefits-of-having-pets

[5] Weiss E, Slater M, Lord L. "Frequency of Lost Dogs and Cats in the United States and the Methods Used to Locate Them". Animals, 2012. 2(2):301-315. https://doi.org/10.3390/ani2020301

[6] Inan, A., Demir I. Aciklamali - Gerekceli kahabatler Kanunu, 2014 343. https://www.tbb.gov.tr/online/yayinlar/kabahatler_kanunu/files/kabahatler%20kanunu.pdf

[7] Giles, C. Banner Recycling, 2019. https://www.bannerworld.co.uk/blog/banners/banner-recycling/

[8] Admin. Pet Statistics, 2021. https://www.aspca.org/helping-people-pets/shelter-intake-and-surrender/pet-statistics

[9] SCHUBERTSLYSCHUBERT. Cat and Dog, 2018. https://www.kaggle.com/datasets/tongpython/cat-and-dog