Route Optimization Based on Traffic Data with Google Maps API

Mikayil Sadigzade
Dept. of Computer Science
Dokuz Eylul University
Izmir, Turkiye
mikayilsadigzade@ogr.deu.edu.tr
0000-0001-9446-9368

Efendi Nasiboglu
Dept. of Computer Science
Dokuz Eylul University
Izmir, Turkiye
efendi.nasibov@deu.edu.tr
0000-0002-1889-6410

***Abstract*—This study introduces a route optimization application based on traffic density data using the Google Maps API. Developed specifically to meet companies' daily logistics needs, this system aims to help users save time and fuel. The application utilizes API services written in C# and is developed for the Android platform using Kotlin. The study provides optimal solutions with a user-friendly interface by accelerating route planning processes.**

Keywords—Google Maps services, Geocoding, Route optimization, Kotlin, C#.

I. Introduction

Traffic congestion is a significant issue that negatively impacts daily life, especially in large cities. This problem causes individual users and logistics companies to experience time loss and increased costs. An efficient route planning system has the potential to address these issues, providing both time and cost savings.

In recent years, various studies have been conducted on route optimization. Smith et al. (2021) demonstrated how dynamic traffic data can improve route planning processes. Their study modeled how traffic congestion affects delays in certain areas and found that generating alternative routes based on this data reduced total travel time by 20%. Johnson and Lee (2022) emphasized that traffic congestion and alternative route analysis are crucial in enhancing energy efficiency, highlighting that fuel consumption in major cities could be reduced by 10%.

In his study titled "Modeling Traffic Networks Using Integrated Route and Link Data," Zhao used data obtained from real-time navigation services such as Google Maps to model traffic networks. This research aims to improve the accuracy of travel time predictions and provide more reliable results in traffic network analysis.

Garcia and colleagues (2023) reported that route optimization in the logistics sector reduces operational costs by 15%. Additionally, Chen and Wong (2021) stated that tools like Google Maps API analyze real-time traffic data to provide users with optimized routes, significantly accelerating business processes. Furthermore, Lee and Kim (2022) compared route planning algorithms in the logistics sector and found that the most effective methods are systems integrated with traffic data. The findings from these studies demonstrate the effectiveness of digital solutions for improving traffic management.

II. Description of the Processing System

In this regard, Google's open source APIs were used:

A. Obtaining an API Key

An API key can be obtained by creating a project through the Google Cloud Console with the following steps:

Log into the Google Cloud Console and either create a new project or select an existing one.

In the APIs & Services > Credentials section, use the "Create Credentials" option to generate an API key.

Store your API key securely and use it only in authorized applications.

The following APIs must be enabled:

- Directions API
- Distance Matrix API
- Traffic Layer
- Geocoding API

Activate these APIs through the APIs & Services > Library section in the Google Cloud Console.

B. Logic of the APIs Used

Directions API:

The Directions API provides route information between two or more points. This API supports the following operations:

Determining the fastest or shortest route between two points.

Offering route suggestions based on different modes of transportation (car, walking, cycling, etc.).

Calculating the estimated time and distance for the route.

Logically, the Directions API takes the start and destination points provided by the user, converts these points into geographic coordinates, and then calculates the optimal route using the current road network. The route data is returned in JSON format.

Distance Matrix API

The Distance Matrix API calculates the distances and estimated travel times between multiple starting and destination points. This API is particularly useful in applications such as logistics and delivery optimization.

The API performs individual calculations for each starting and destination point. Dynamic results are generated by considering road conditions, traffic density, and travel mode. The working principle of the Distance Matrix API is based on a matrix: starting points are represented in the rows,

destination points in the columns, and each cell contains the distance and duration between two points.

Traffic Layer

The Traffic Layer visualizes real-time traffic density data on the map. The data is obtained from anonymized user movements and other traffic sensors. Traffic density is expressed through color codes (green: smooth, yellow: moderate, red: congested). It is integrated into the map, allowing users to quickly analyze the traffic situation. This layer operates as an additional information layer added to the map.

Geocoding API

The Geocoding API converts addresses into geographic coordinates (latitude and longitude) or coordinates into addresses. This API is used to standardize user inputs and accurately position locations on the map.

- Forward geocoding: Creating coordinates from addresses.
- Reverse geocoding: Creating addresses from coordinates.

The API analyzes user-provided textual addresses and returns the most appropriate coordinates from mapping databases.

Google Maps APIs offer a wide range of capabilities for developing map-based applications. In this article, a map service integrated with a C# infrastructure using Google Maps API is discussed. This service includes functionalities like geocoding, autocomplete, route calculation, and location search. The code snippet allows users to perform address-based location searches, create route plans, and obtain geographic information.

The service is structured with a C# class, GoogleMapsService, which makes API requests using HttpClient. Each functionality processes data by calling the relevant Google Maps API endpoints and provides the data to clients via appropriate DTOs (Data Transfer Objects). The main components of the service are as follows:

Autocomplete

The autocomplete feature allows users to receive search results related to an address or location dynamically. This method makes an API call that lists suggested places based on the words entered by the user. Search results can be restricted to a specific country (e.g., Turkey).

- **Usage area:** Providing quick search suggestions.
- **Output**: Returns a list containing address and location information to the user.

Finding a Location (Geocoding)

It is used to obtain the latitude and longitude information of an address. By using the Geocoding API, the geographic coordinates of an address are obtained. This feature is commonly used in map-based applications.

- **Usage area**: Creating bookmarks based on addresses.
- **Output**: DTOs containing address and coordinate information are returned.

Search by Place ID (Place ID)

It is used to get detailed information about a place based on its unique identity (Place ID).

This method retrieves detailed address and location information via the Place Details API.

- **Usage area**: Provide detailed location information based on user selection.
- **Output**: The address and coordinate information of the selected location is returned.

Route Calculation (Directions)

Calculates a route between a given start and destination.

This method makes an API call that supports optimized routes and waypoints. Optimization can be done based on distance or time based on user preferences.

- **Usage area**: Planning a route by taking into account traffic conditions and alternative routes.
- **Output**: Information such as route summary, total distance and time.

Finding Location with Coordinates (Reverse Geocoding)

Retrieves address information based on a latitude and longitude pair.

This feature allows users to click on a point on the map to get relevant address information.

- **Area of Use**: Dynamic address representation in map-based distributions.
- **Output**: DTO containing address information.

Security and API Key Management

The API key (apiKey) is statically defined within the service. Since this can create security vulnerabilities, it is recommended to retrieve the key from environment variables or a secure configuration file.

The working logic of the application

In the written code, the user first finds the addresses to be visited by typing their names. They can also select locations by manually dragging them (Fig.1).

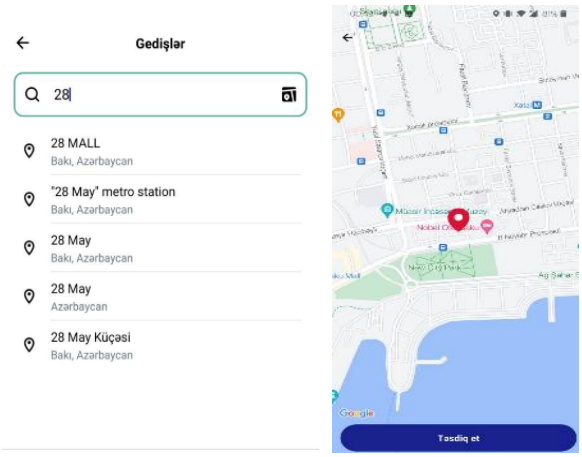


Fig.1. Determination of addresses and locations.

1. Determining the routes to be taken

After selecting the addresses to go, it is presented to the user in the most optimal way by pressing the sort button. Before sorting, it is necessary to choose whether it is optimal by kilometer or time (Fig.2).

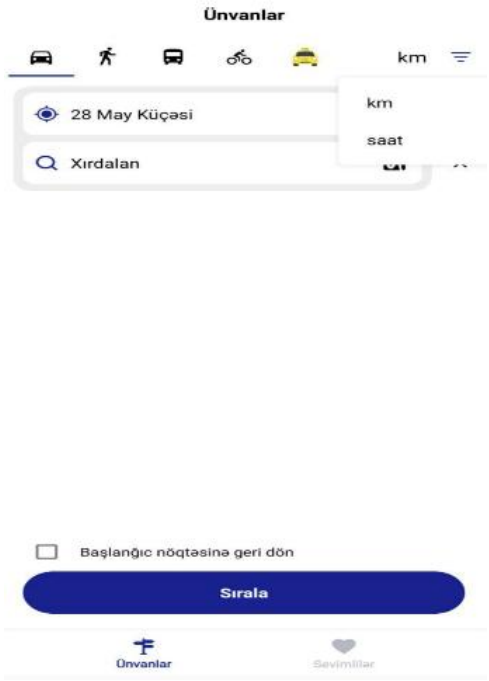


Fig.2. Optimization by kilometer or time.

2. Determining the sequencing logic of routes

After the routes are sorted, they appear in order on a new screen. On the destination screen, the user can see the estimated time, distance, and the sequence of the route. By tapping the icon, they can choose their preferred map application and continue with it. After completing each route, the user must return to the app and manually select the next one. Additionally, by tapping the heart icon in the top right corner, they can add the route to their favorites. They can also assign a custom name to the route while adding (Fig.3).

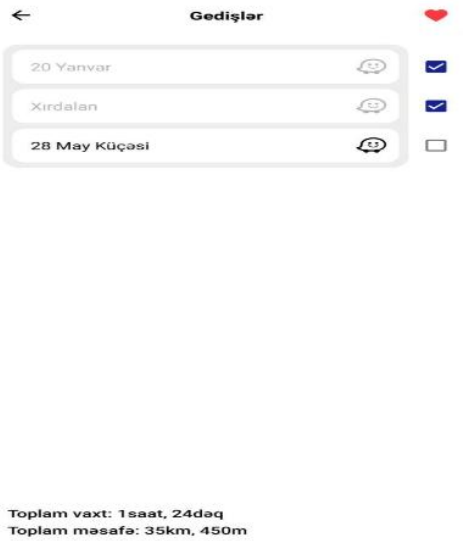


Fig.3. Adding the route to the favorites.

3. Route sorting

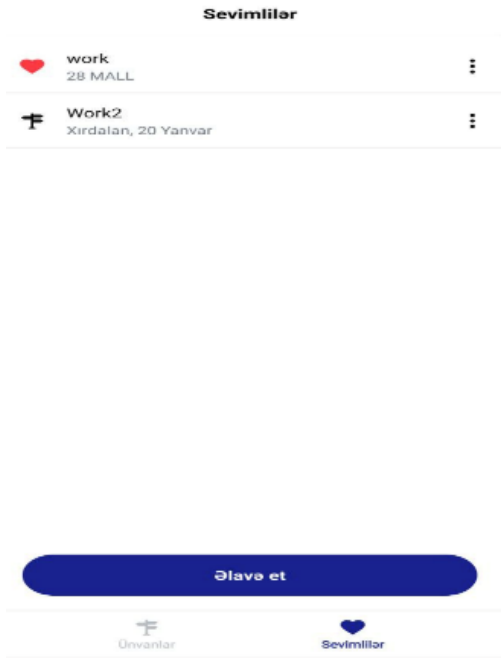
The favorites tab displays added favorite addresses and routes. On this screen, users can add, edit, and delete favorite entries (Fig.4).

4. Favorite page

C. Multi-Layered Architecture with .NET Core REST API and Android Kotlin Integration

In this section, you can discuss the fundamental principles of developing RESTful APIs using .NET Core. Then, explain how you develop an Android application using Kotlin and communicate with these APIs.

- **.NET Core Rest API**
  - Creating a REST API in .NET Core
  - Layered architecture (Controller, Service, Repository)
  - Database operations (EF Core)
- **API Consumption with Android and Kotlin**
  - HTTP requests with Retrofit
  - JSON data model
  - Security and authentication (JWT esc.)
-
-
-
-
-



- Fig.4. Editing favorite entries.

III. Conclusion

This article presents a structured approach to building a Google Maps service for location search, geocoding, and route optimization. Leveraging Google Maps APIs (Places, Geocoding, and Directions), this service enables seamless integration of advanced location-based functionalities into applications.

Key features include address-based location search, coordinate-based location retrieval, and optimized route calculation with customizable travel modes and preferences. This application demonstrates how modern mapping solutions can address complex location-based challenges in software projects.

By abstracting API calls into reusable methods, the service enhances code readability, maintainability, and scalability. In the future, additional API support, such as real-time traffic updates or user-defined points of interest, can further improve its usability in dynamic environments. In conclusion, this Google Maps service provides a solid foundation for developing location-based applications, allowing businesses to offer personalized and efficient solutions to their users.

REFERENCES

1. Mith, J., Brown, A., & Taylor, K. (2021). Real-time Traffic Data for Route Optimization. Journal of Transportation Systems, 34(2), 45-58.

2. Johnson, M., & Lee, H. (2022). Energy Efficiency in Traffic Management through Alternative Routes. International Journal of Logistics Research, 40(3), 120-135.

3. Garcia, P., Martinez, L., & Fernandez, R. (2023). Cost Reduction in Logistics Using Route Optimization Techniques. Logistics and Supply Chain Innovations, 29(1), 75-90.

4. Chen, L., & Wong, P. (2021). Enhancing Logistics Efficiency with Real-time Traffic Data. Transportation Research and Analytics, 25(4), 88-102.

5. Lee, S., & Kim, J. (2022). Comparative Analysis of Route Planning Algorithms in Logistics. Advanced Logistics Studies, 18(3), 210-223.

6. Liu, Y., & Zhang, T. (2023). Dynamic Traffic Data Integration in Route Optimization. Journal of Urban Transportation, 50(1), 65-80.

7. Wu, Q., Gao, L., & Chen, R. (2022). Carbon Emission Reduction through Real-time Traffic Management. Environmental Logistics Review, 12(4), 102-115.

8. Gao, X., & Chen, P. (2022).Traffic Congestion-Based Route Optimization for Logistics. International Journal of Transport Systems, 19(2), 135-149.

9. Zhang, H., & Li, F. (2023). *Real-time Traffic Data Impact on Driver Decision Making. Transportation Safety Journal, 27(3), 56-74.

10. Yang, Z., & Liu, M. (2023). Mobile Application Innovations in Route Planning. Smart Logistics Applications, 14(2), 98-110.

11. Zhao, X., & Spall, J. C. (2018). Modeling traffic networks using integrated route and link data. *Transportation Research Part C: Emerging Technologies, 92*, 123-135.