Algorithmic Design and Implementation of Information Database, for International Freight Transport

Petra Opitz, Tamás Haidegger and Enikő Nagy

John von Neumann Faculty of Informatics, Obuda University, Budapest, Hungary; petra.opitz@irob.uni-obuda.hu; nagy.eniko@nik.uni-obuda.hu

School of Computing, Queen's University, Kingston, ON

University Research and Innovation Center (EKIK), Obuda University, Budapest, Hungary; University Research and Innovation Center (EKIK), Obuda University, Bécsi út 96/b, H-1034 Budapest, Hungary, haidegger@uni-obuda.hu

Abstract: International freight transportation represents a rapidly expanding sector, playing a pivotal role in the global economy and everyday life. Efficient management of transportation processes requires the collection and processing of substantial amounts of information, presenting significant technical and IT security challenges. While the market experienced a slight decline due to the COVID-19 pandemic, it has shown steady growth, particularly driven by the increased demand for online shopping (B2C). The primary transportation modes – air, rail, road, and sea – differ in terms of factors that influence the optimal choice, requiring consideration of both intrinsic and extrinsic elements. These factors are most effectively managed through an information database that functions as a decision support system. This article specifically addresses the challenges associated with road transport orchestration and data management. The proposed database architecture offers a model that can benefit transport managers, financial staff within transport companies, and even the clients, by providing access to key information related to individual transportation units, or "tours", that represent discrete transport activities.

Keywords: optimal freight algorithm; data-driven transport; road transport database optimization; global transport sustainability

1 Introduction

Logistics became an essential component of global business, serving and catering properly for consumer society is one of the most important and challenging industry processes. The current global freight transportation system faces several significant shortcomings, being addressed by the research community. Machine Learning (ML) algorithms, predictive analytics, and real-time data processing are essential for leveraging data to enhance transport operations [1]. Meanwhile, sustainable transportation aims to minimize the environmental, social and economic impact associated with the worldwide flow of individuals and products. Research in this area focuses on developing environmentally friendly transportation technologies, promoting alternative fuels (e.g., electric and hydrogen), improving energy efficiency, reducing greenhouse gas emissions and fostering sustainable urban mobility practices. It also encompasses policy analysis, regulatory frameworks and international collaborations to address challenges related to climate change, air pollution, congestion and resource depletion in the transportation sector on land, at sea and in air [2]. Many of these domains can be affected via optimized freight orchestration and management, largely driven by software solutions. The study is based on addressing the current complex situations by seeking prompt action and innovative solutions from the R&D community. It involves investigating various processes, including comparable systems and transport methods, and evaluating different data storage methods such as relational and NoSQL databases.

The most popular modes of transport are air, road, sea, and rail transports, being exposed to the transformative effects of Artificial Intelligence methods and intelligent software solutions [3]. The market is growing and there is continuous demand for advanced global freight market solutions, where the total market value is estimated to be $192.50 billion. According to Globe Newswire, the global freight trucking market, valued at $2.2 trillion in 2022, is expected to grow at a compound annual growth rate (CAGR) of 5.4%, reaching $3.4 trillion by 2030 [4].

1.1 State of the Art and Major Challenges

Current complex situations require prompt action and new solutions from the R&D community, and smart transportation and Transportation 4.0 have been a key research topic of our group in the past, focusing on intrinsic and extrinsic challenges of the domain (Figure 1):

- Congestion: Ports, highways, railways and airports often experience congestion due to increased global trade volumes, and adverse events, such as the recent accidents of the container ships Ever Green and Dali showed. Congestion leads to delay, increased costs, and inefficiencies in the transportation network, calling for more efficient scheduling methods.
- Infrastructure Limitations: Many regions lack adequate infrastructure to support efficient freight transportation. This includes outdated ports, insufficient road and rail networks, and limited warehousing facilities. Upgrading infrastructure requires substantial investment and coordination and may only change over long period of time.
- Environmental Impact: Freight transportation substantially contributes to greenhouse gas emissions. The reliance on fossil fuels in trucks, ships, and

planes contributes to climate change and poses health risks to communities living near to transportation routes [5].

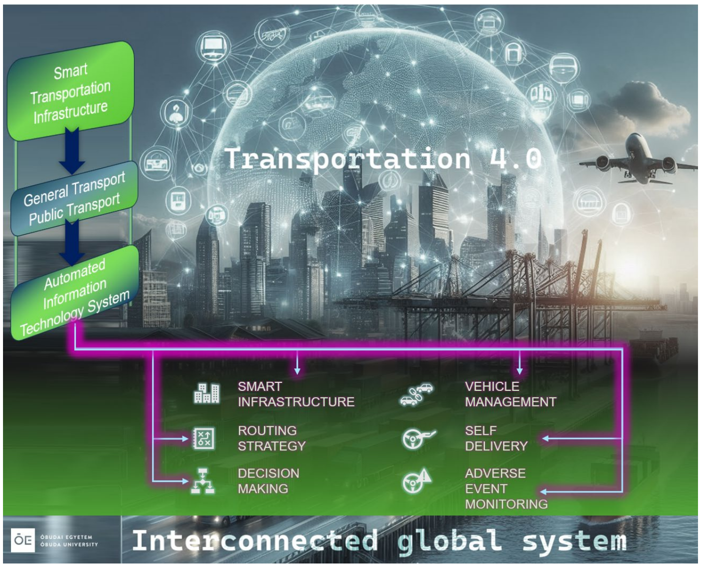
Smart Transportation Infrastructure
General Transport Public Transport
Automated Information Technology System
Transportation 4.0
SMART INFRASTRUCTURE
VEHICLE MANAGEMENT
ROUTING STRATEGY
SELF DELIVERY
DECISION MAKING
ADVERSE EVENT MONITORING
ÓBUDAI EGYETEM ÓBUDA UNIVERSITY
Interconnected global system

Figure 1
Illustration of the IT system support for inter-connected global freight transportation system, leading to the concept of Logistics 4.0

- Safety and security: Freight transportation is vulnerable to security threats such as theft, piracy, and terrorism. Protecting shipments from these risks require robust security measures and international cooperation.
- Regulatory Compliance: Freight transportation involves navigating a complex web of regulations, tariffs and customs procedures across different countries and regions. Compliance with these regulations adds complexity and costs to the transportation process, especially in the case of foods [6].
- Labor Issues: The freight transportation industry faces challenges related to labor shortages, driver retention, and workforce diversity. Automation and technological advancements may also impact job availability and require retraining of workers; the transitions may be significantly sped up by the development of robots for logistic purposes [7].
- Supply Chain Disruptions: Natural disasters, geopolitical conflicts, trade disputes, and pandemics can disrupt global supply chains and lead to delay in freight transportation. Building resilience and flexibility into supply chains is essential for mitigating these disruptions, while this typically increases the costs [8].

In the advent of Logistics 4.0, a new, integrated approach to fully digital global transportation system, addressing these challenges requires collaboration among governments, industry stakeholders, and international organizations to develop sustainable, efficient and resilient freight transportation systems (Figure 1). This may involve investments in infrastructure, adoption of cleaner technologies, implementation of regulatory reforms, and promotion of advancements in logistics and supply chain management, also being addressed at Obuda University [9].

1.2 Algorithmic Challenges of Industry 1.0-4.0

Logistics traditionally includes various algorithmic problems, primarily starting at route optimization. *Routing algorithms* are mathematical and computational techniques designed to identify the most efficient paths or sequences for transporting goods or services across a network, minimizing cost, time, or energy while adhering to specific constraints like capacity, time windows, or traffic conditions. These algorithms play a critical role in logistics, including international freight transport, by ensuring effective use of resources [10] [11]. Key scientific categories of such algorithms include:

1. **Graph-Based Algorithms**:
   - **Dijkstra's Algorithm**: Utilizes graph theory to calculate the shortest path between nodes, leveraging edge weights that represent cost metrics like distance or travel time.
   - **A* Algorithm**: An enhancement of Dijkstra’s that integrates heuristics, optimizing both pathfinding speed and accuracy by estimating the remaining distance to the goal.
2. **Optimization Models for Complex Networks**:
   - **Mixed-Integer Linear Programming (MILP)**: Used for the Vehicle Routing Problem (VRP), formulating routes as optimization problems with integer and continuous variables under constraints.
   - **Heuristic and Metaheuristic Methods**:
     - **Clarke-Wright Savings Algorithm**: Focuses on combining individual routes into fewer, consolidated ones, reducing overall cost.
     - **Genetic Algorithms (GA)**: Inspired by evolutionary biology, this method generates a population of solutions, iteratively improving them through selection, crossover, and mutation.
3. **Dynamic and Real-Time Optimization**:
   - Ant Colony Optimization (ACO): Simulates the behavior of ants to find optimal routes, particularly effective in dynamic environments with changing traffic patterns.

- Machine Learning Algorithms: Predict travel times or optimize fleet movements using historical and real-time data.

In industrial applications, like those studied in robotics and logistics research, such algorithms ensure seamless coordination of resources, meeting both technical and economic goals. Furthermore, *network flow optimization* is essential for throughput optimization, where typical algorithm classes include:

1. **Linear Programming (LP):** Optimizes resource allocation in supply chains, like transportation and warehousing
2. **Max-Flow/Min-Cost Algorithms**: For balancing costs and maximizing capacity in freight networks
3. **Integer Programming (IP):** Addresses discrete decision-making, such as container loading or fleet assignments

Demand Forecasting and Inventory Management focused algorithms already include a wider range of solutions, including Time Series Analysis (e.g., ARIMA), Regression Models, Exponential Smoothing, Machine Learning (ML) models and Neural Networks (NN), including Deep Learning Methods [12]. For *Freight Consolidation and Load Optimization*, there exists numerous solutions, heuristics and metaheuristics, Simulated Annealing or the Knapsack Problem Algorithms.

1.3 Processes under Current Investigations

Transport management services are frequently adopted by companies to optimize logistics operations, creating a mutually beneficial system in which transport firms secure consistent employment for drivers while customers benefit from centralized task allocation. In this model, customers submit shipment requests – including origin, destination, cargo specifications (e.g., temperature requirements), and pricing preferences – to a freight forwarder. The transport manager then solicits offers from hauliers, selects the optimal bid, and issues a transport contract. During execution, the dispatcher maintains continuous communication with the carrier. Upon delivery, the carrier invoices the dispatcher, who subsequently invoices the client at the agreed rate, with the intermediary margin constituting the dispatcher's profit.

Operational data are organized into *tours*, each representing a planned route comprising multiple shipping orders executed by a single driver and vehicle. Given the volume and importance of logistics data, a centralized web application was designed to provide stakeholders with unified access to key tour information – such as departure and arrival points, vehicle registration numbers, driver identity, and responsible transport operator.

In our current work [13], A NoSQL database architecture was selected for its scalability and efficient data retrieval capabilities. The system architecture comprises a C# based, API layer developed in Visual Studio, interfacing between the database and a web frontend. Data visualization is implemented using Microsoft Power BI. The resulting system supports the creation, modification, and deletion of tour records – functions essential for operational continuity and for compliance with the company's quality management system, ensuring accurate freight tracking and documentation.

According to the open science theory, guaranteeing the reproducibility of a software tool for e.g., logistics support in the field of IT involves several key practices and methodologies. To support open science, the authors have made the source-code available via GitHub: https://github.com/haidegger/OU-transport-DB, released under Creative Commons CC0 v1.0 Universal.

1.4 Market Overview

According to Statista the European road freight transport market reached €324.5 billion in 2020. For 2021, the market grew further to around €352.4 billion. This growth was in line with the positive trend observed between 2010 and 2020. The figures showed an increase in each year, except in 2012. Overall, the road freight market grew by 27.5% by the end of 2021 compared to 2010 [15].

Also, Statista states that in 2019, goods were transported on road in Europe for approximately 1.764 billion kilometers. In recent years, the share of road freight transport has gradually increased, while the share of rail transport has decreased.

In Hungary, road freight transport in 2021 increased by 4.9 ton-kilometers (TKM), or about 15.14%, compared to 2020, i.e. in 2021 the transport volume amounted to 37.1 billion ton-kilometers [16]. TKM is a measure of the transport of goods, which represents the transport of one ton of goods (including the tare weight of packaging and intermodal transport units) by a given mode of transport (road, rail, air, sea, inland waterway, pipeline, etc.) over a distance of 1 km.

1.5 State-of-the-art Comparable Solutions

The exact name of this type of application is "Transportation Management Systems" (TMS). These are centralized platforms on which all actors of the supply chain can collaborate to execute the transportation and the distribution processes. The end users of TMS include freight forwarders, carriers, warehouse managers, transport managers, transport intermediaries, and supply chain managers. Current providers offer various services, typically focusing on just a limited set of market domains. As a growing trend, sustainability and social responsibility are becoming a more and more recognized part of the value proposition [14]:

- Descartes Aljex stands as the leading freight management platform for freight operators, operating on a SaaS-based model. This scalable cloud-based TMS serves as a hub for a vast network of brokers, 3PLs (third-party logistics or outsourced logistics), freight forwarders, and combiners.

- Turvo serves as a cooperative TMS platform fostering connections between individuals and companies, empowering freight operators and shippers to revolutionize their processes through cloud-based software and mobile applications.
- MyCarrierTMS empowers shippers by leveraging digital workflows, organizations can streamline the comparison of shipping costs and delivery schedules from various carriers, making the booking process easier with an adaptive learning system that automatically fills in frequently used order details. Additionally, it automatically generates electronic waybills or delivery notes.
- Oracle Transportation Management Cloud empowers organizations to manage and diminish expenses, enhance service levels, endorse sustainability initiatives, and automate adaptable business processes throughout transportation and logistics networks.

2 Materials and Methods

Our developed software solution focuses exclusively on road freight transport. For the implementation, we chose NoSQL database server to store the data and one of the most popular open-source applications, MongoDB. To facilitate the creation of new data entries, we implemented a Create function for each collection.

This straightforward method highlights the flexibility of MongoDB's NoSQL architecture. It ensures that the system can rapidly accommodate new records with minimal overhead, which is essential in logistics operations characterized by frequent updates and high data volumes.

It is a cost-effective method as it stores data in key-value pairs so it saves memory, which speeds up queries and supports replication making the database more fault-tolerant. Compared to relational databases, we intended to exploit the flexibility of NoSQL systems in terms of storage as well as the faster write speed which is an important factor due to the huge amount of incoming data and the huge storage capacity requirement as data must always be up to date. We could also integrate the data into Power BI so there was no barrier for using it.

We intended to advance a Rest API, that we could implement in Microsoft C# .NET Core 6 using the C# knowledge already acquired and the new info we wanted to obtain from this current research.

After investigating several methods by which data store would be the most efficient to work with, we finally decided to use the MongoDB document store also since

Visual Studio has a MongoDB add-on "MongoDB for VS Code" which allowed us to easily integrate databases into the application.

Finally, we implemented the front-end part using ASP.NET(MVVM) and we used a NuGet Package named Swagger to test each HTTP access and other test cases in the form of nUnit tests. We also implemented data visualization using PowerBI for different query cases.

2.1 Data Storage

Several methods and applications for storing data have been investigated earlier. We had to consider several aspects, such as the type of data to work with, the free of charge availability of the data storage applications, and most importantly, their compatibility with the Visual Studio, a NuGet package or a built-in tool that would allow us the easy import of the generated data.

First, we had to choose a data storage technique with a changeable schema, and that was the NoSQL. The next consideration was fast access to the data. It is important that the huge amount of stored data can be accessed by the user as quickly as possible and the NoSQL is better in terms of speed. The scalability of the NoSQL databases is also an advantage and the fact that there is no single point of failure is also an advantage, so this storage method was chosen. Next, we selected the MongoDb document repository, as the actual data storage application, which could most efficiently store our data and modify them if necessary.

When creating the data, at first, we tried to find a suitable data repository on the internet, but none of them met our expectations, so we created our own one using the Spawner application. Spawner is a tool used to generate sample or test data for databases. It can be customized to produce either delimited text files or SQL statements. It can also be inserted directly into a MySQL 5.x database. It contains several field types, most of them are configurable. Because we had to generate large amounts of data for some tables (hundreds of records), we created the following tables and data (Figure 2):

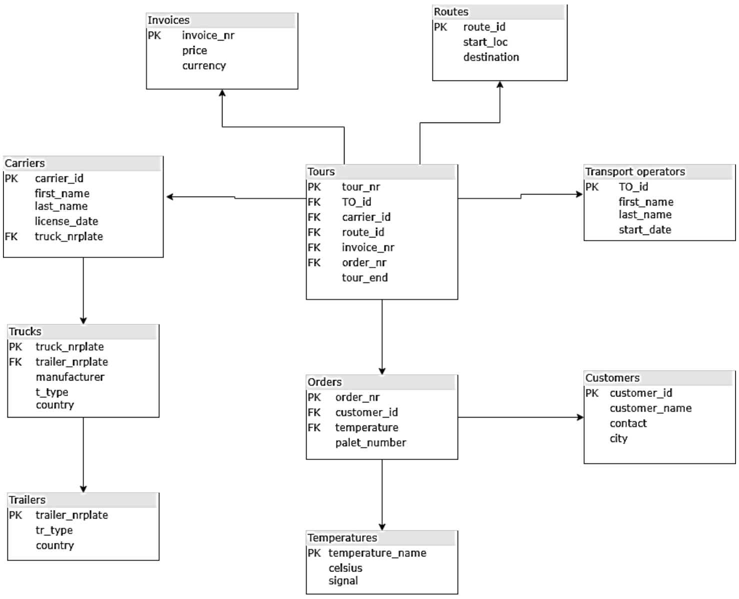

Figure 2

Designed database structure fitting the domain specialties

2.2 Backend and Frontend

For the implementation of the backend, we have already chosen Visual Studio and within it we developed the Rest API layer in C#, which created a kind of communication layer between the database and the frontend. We worked with the latest release of Visual Studio 2022 using .Net Core Web API, version 6 of .NET Core.

Initially, we created a "Models" folder in which we formed the classes corresponding to each collection and in it we created the variables corresponding to each column. Since there were deserialization problems, we needed MongoDB specific commands and components, so we wanted to install the "MongoDB for VS Code" add-on, which was designed for this purpose. Then evolved our next problem: it is available only in VS Code not in Visual Studio 2022. The tools we needed was in the Visual Studio NuGet packages so we installed "MongoDB. Driver," which is the official MongoDb runtime utility package in Visual Studio. It is essentially equivalent to "Mongodb for VS Code" in its functionality and the "MongoDB.Bson" Nuget packages. The latter is the Bson library for MongoDb, we needed this later as MongoDb stores data in BSON format, which is binary JSON. These could help us interact with our database and the Bson data types. Here, we allocated the MongoDB.Bson, MongoDB.Bson.Serialization. Attributes directives to help us. Using these we created a BsonId and a BsonRepresentation

(BsonType.ObjectId) attribute for the Id variables, which MongoDb automatically created. The first one makes it more RestApi friendly and the latter works locally in .NET Core as a string but could be stored in MongoDb as an ObjectId. We performed this with all collections. At a later running, we found an error that if the column name was not written exactly as was given in the class, it resulted an error. A solution could be the Map attribute that would allow us to nominate each property differently, but we did not use this, as having some other format was not important for us since the database format was perfect.

Then, we created a new public class entitled "MongoDbSetup.cs", where we created only 3 variables, ConnectionURI could store the path, DatabaseName that was verbose, could store the name of the database, which in this case was "transportdb" and we declared a variable nominated CollectionName that was responsible for storing the collection names, these were carriers, customers, invoices, operators, orders, routes, temperatures, tours, trailers, and trucks. We have assigned getters and setters to each of these, i.e., they could be queried and rewritten, and we have also specified that they could not be "null", i.e., they could not be empty. These contained the information needed to establish and maintain a connection to the database.

After the above, we defined the variables mentioned above in the file "appsettings.JSON". To do this, we created a new JSON object. It was important to keep the names consistent. Here, we set the value of the variables, i.e., we retrieved the Connection string from MongoDb Compass, passed this to the ConnectionURI variable. However, we realized that since we were connecting with localhost, the database would probably not be accessible from another devices, so the current MongoDb Compass solution would not work. The MongoDb Atlas became necessary here, which is the most advanced cloud database service on the market, with unique data distribution and mobility between AWS, Azure and Google Cloud, built-in automation to optimize resources and workloads, and much more [9]. This solution is chargeable, but a cluster with basic configurations can be built and used for free of charge. In order to achieve this, we created an account and started a new cluster with Amazon AWS as provider, starting this again to create the collections we had already defined and importing the individual documents. Within the previously installed MongoDb Compass, we could edit and view our collections in the same way, as in the case of localhost, so we continued to work with the cluster and collection in the cloud, in Atlas.

Then, we created a new folder entitled "Services", and a class entitled "MongoDBService.cs" in it. In this, we coded the service itself, the CRUD operations, we needed some additional function to use our collections in MongoDb to manage them as efficiently as possible, so we have enabled the MongoDb.Driver and MongoDb.Bson namespaces. We declared several variables to represent each collection, each of which were type of "IMongoCollection". In the constructor, we passed the path, the database name, and the corresponding collection to access the MongoDb database via the MongoSetup class.

Once we did this, we added the Models and Services namespaces in "Program.cs" and used the builder configuration to connect to this class as well. In the Program class, we set up the use of Swagger.

Next, came the actual CRUD methods, which were created in a separate Controllers folder in the Controller class of each collection. To enable the modification of existing records, we implemented an Update function. This method ensures that the correct document is identified and updated based on its unique identifier. By leveraging MongoDB's ReplaceOneAsync function, the entire document can be efficiently replaced with its updated version. Such functionality is crucial for maintaining data integrity in real-time logistics systems, where existing entries often require modifications due to dynamic operational changes.

These CRUD operations form the backbone of our application's backend, providing robust and scalable support for real-time logistics data management. Therefore we created the ReadAll and ReadOne functions, as well as the delete, add, and overwrite methods. These were coded in detail in the MongoDBService class, and the methods in the Controller classes. MongoDBService implemented the IMongoDBService interface. As a result, after running the program, Swagger and the application ran at the same time and after testing the execution of the encoded operations, we found that they all worked correctly, so we could move on to implement the frontend.

We created a WPF project, which automatically created a xaml and a xaml.cs. We installed the Nuget packages we needed which were WebAPi.Client, SignalR.Client and MVVM. WebApi.Client was helping with formatting and content negotiations to System.net.Http, SignalRClient was downloaded so that data change was always up-to-date during data manipulation in several windows at the same time. Furthermore, MVVM provided the necessary components to implement the Model-View-ViewModel method.

In the MVVM design pattern, there are three key components: the model, the view, and the view model. Each element plays a distinct role within the pattern. The view understands the view model, the view model understands the model, but the model doesn't interact with the view model, and the view model doesn't interact with the view. The view's responsibility is to define how the content is structured, laid out, and visually presented to the user on the screen.

Ideally, each view is defined in XAML, with minimal code-behind that does not include business logic. The view model contains properties and commands that the view data can connect to, updating the view about any state changes through notification events. Model classes, on the other hand, are non-visual entities that hold the application's data.

We created the model class earlier as part of the Backend, created the View as xaml and created the ViewModels. In these, the Command objects for each CRUD method were created and set up so by pressing the appropriate button the desired action would be performed. This was done in each Collection's ViewModel. In the ViewModel we also created a "Selected" variable and property that would store the currently selected document. When the above variable was null then it could not be deleted and the button was not functioning. Also, the variable was important for the Update method since in that case the input was also an instance of the given collection. Furthermore, it was implemented to automatically display the data of the selected document in the component created below such as TextBoxes and DataPickers. Next, we defined the DataContexts in the xaml files which for each window became the corresponding ViewModel and after creating the corresponding components (TextBox, DataPicker) we connected them with the help of "Bindig" to the corresponding Commands or documents, data types.

In order to the Swagger and WPF would be able to communicate smoothly, we used a RestService.cs utility class from one of our previous projects. Obviously, we had to modify this in several cases, for example so that it would try to reach Swagger and the collections on the right endpoint in each case and so that the CRUD methods ask for and run with the right inputs. We used this RestService class for the ViewModel classes, always supplemented with the appropriate collection type.

After the required implementation we used the application several times so that we could make sure it was working fine and we could say it worked as it was described.

2.3 Testing

For testing we decided to follow the principles of TDD, i.e. first define the test cases and write the code based on that. For this, we created a new Class Library referred as "Testing" and in it a new class referred as "TestingClass-cs," which will contain the tests. The first and most important step was to install the Microsoft.Net.Test.Sdk, NUnit and NUnit3TestAdapter Nuget and Moq packages. Unit testing consisted of the following steps: Arrange, Act, Assert. This test illustrates the "Arrange-Act-Assert" principle of unit testing. It ensures that the GetOrderById function works as expected by mocking the MongoDB dependencies and verifying the correctness of the returned data. Such tests are critical in validating the application's behavior under different scenarios.

In the Arrange phase, we created and configured the system under test. This system could be a method, a single object or a graph of connected objects. It was okay if the Arrange phase was empty, for example, in case we tested a static method. In the Act phase, we tested the system by a selected method. Upon delivery of the results, it would be checked for errors or unintended side-effects. The Assert phase made our unit test pass or fail. Here, we checked whether the behavior of the method was consistent with the expectations [32]. Using the Moq Nuget package, we created an instance of the MongoDbService class by setting the MockBehaviour property to Strict, thus forcing the created Moq object to operate as its original class would do, which was important for exception handling, for example. We also created the individual Controller classes, into which we manually and locally loaded data in order not to load or modify the real database, but another database created for this purpose. We implemented this in the Setup method with the Setup attribute, which served to summarize the first step of Unit tests, the "Arrange" phase, in one method, so we did not have to create the corresponding objects for each test. After creating the location collections, we specified to the Moq object that the GetAll methods return the local collections asynchronously and then we could start with the Test methods.

At testing, we did not test the operation of the methods themselves, but we examined them from different business points of view. It may happen that various restrictions are in effect, for example, if a customer or supplier did their work improperly, they may be blacklisted, but in this case, the program must use exception management to prevent the creation of the given entity or even the modification of an existing one. All test methods have a Test attribute, so they can be run accordingly.

We wrote 7 tests in total, all of them resulted in green, which meant all of the determined exceptions were managed. In the future, new AI-provided methods may bring advantage to this phase, providing systematic and standardized methods for advanced autonomous feature development [17].

2.4 Data Visualization

For Data Visualization we used PowerBI. This is believed to be a really important software. There must be some kind of visualized feedback for a company's progress.

We chose the Power BI because it is widely accessible as it is free of charge (with limited options). It is also code free and user-friendly with very intuitive user interface which helps the non-experts to easily modify any results [18]. Apart from Power BI, Tableau was also a runner-up for Data Visualization for this project as it also has a user-friendly interface, and it is similar to Power BI in many aspects.

Although these are easily usable without coding, both Tableau and Power BI offer R and Python programming options when necessary. Power BI is available only for Windows while Tableau can also be used with Mac OS among other systems. Although Power BI is more affordable than Tableau at the moment nevertheless both have free of charge versions, namely Power BI Desktop and Tableau Public. The Tableau's disadvantage is that it requires specific skills to use them efficiently and the dashboards are limited.

There is also Qlik sense which is a non-code cloud service and a business analytics platform that offers easy installation, easy usage and workflows. Qlik has a significant disadvantage compared to Power BI, since it can't handle a large number of data efficiently and it causes window freezing according to users. The above-mentioned tools, however, are all market leaders at this time [19]. The brain processes images much faster than text – about 60000 times more quickly. A study conducted by University of Minnesota researchers in 1986 discovered that presentations with visual aids were 43% more convincing than those without. Data storytelling, with the aid of data visualization, is a great way to explain, analyze, and interpret data [20]. We created two examples (Figures 3, 4) which are very common. In the first, the statement presents the annual orders. In the first case, we visualized how many orders arrived annually for the past three years.

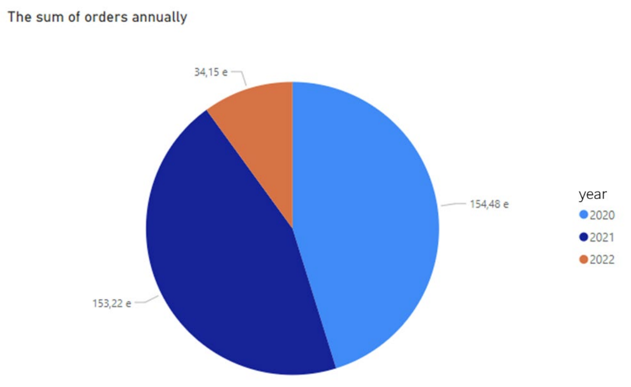

Figure 3
The sum of orders annually

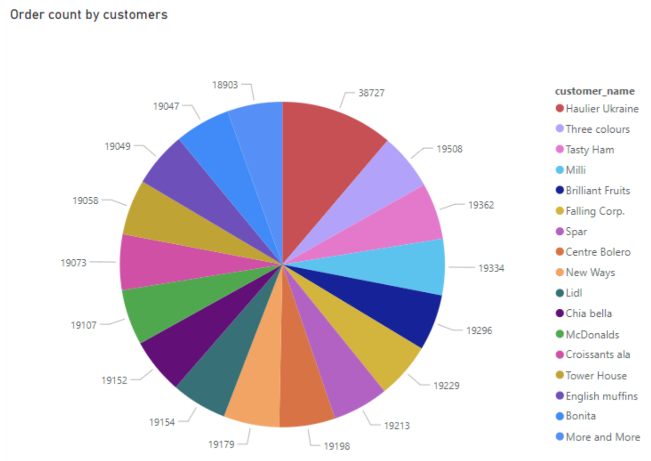

Figure 4
Order numbers by customers

The second statement shows the details and explains exactly how many orders were given by each customer. These two cases almost certainly appear in real cases, and we wanted to illustrate that these data are much easier understandable in this form than in terms of text for e.g., non- professionals.

3 Results and Future Work

After thorough considerations, we can state that we have implemented and achieved every planned aspect. All of the CRUD methods are working properly and asynchronously. We presented seven green test cases. We realized quite some errors during the development phase but with much effort we were able to analyze them.

The program is in use now, in its pilot phase, relying on the implementation outlined in Section 3. In more details, there is a MainWindow (Figure 5) that pops up immediately when the program is initialized. Here, the user can select the set he wants to modify. Clicking on the button “Carriers” brings up another window for that set, where different data manipulation options can be selected, which are Create, Update and Delete. On the left side of the window, one can see the corresponding buttons, as well as a Back button, which allows to return to the main page without stopping the program (Figure 6). In the center of the pop-up window, a form is displayed to facilitate data entry. Here, one can find Labels and TextBoxes or, if one of the data types in the collection is date format, a DatePicker. Finally, on the right edge of the window, there is a ListBox, that lists the existing items in the set (Figure 6).

In addition, there could be user profiles for each entity’s representatives such as carriers and especially operators. They would be able to access only that parts of the application that was relevant for them. For example, carriers could review only the tours and the related information that were assigned for them, but transport operators would be able to manage the CRUD methods too.

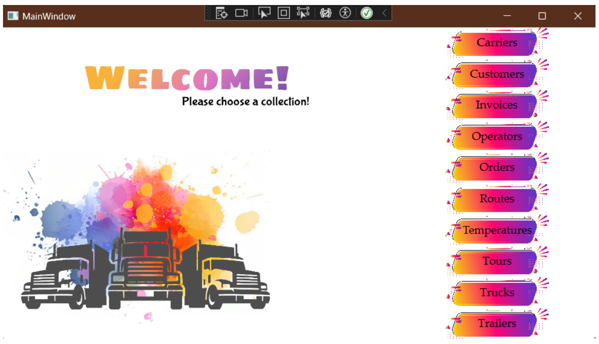

Figure 5
Main Window of the completed user interface

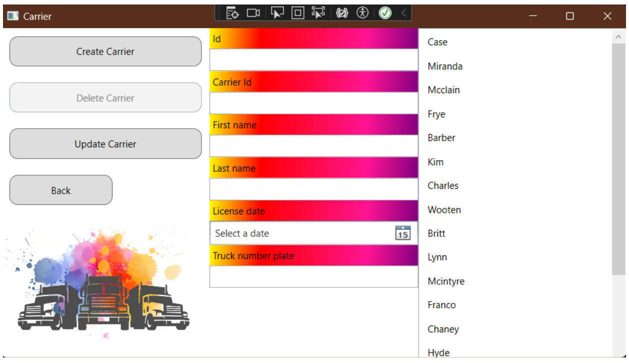

Figure 6
Carrier Window, allowing the user to customize the database

Also, for the tours there would have a lot of related documents such as incoming orders from customers, outgoing orders from us to carriers, PODs like CMR which were absolutely necessary for import-export transportations. Some kind of ERP system could be used and we could integrate financial components and also visualization. There could be an invoice booking opportunity too, so in case an invoice would arrive, a financial colleague could automatically book and pay that.

Creating a Workflow environment could also help the work of the employees, it would be great to create a separate user interface for each work section and create some kind of marker that would indicate the status of the job or the path that the whole process could go through, and people could assign each other for implementing those jobs. Such implementation would also offer notifications which would allow the users to notice the changes immediately. Such new features could greatly benefit from the recent AI-supported development of software tools, enabling the wide spread of Logistics 4.0 [21] [22].

These two cases almost certainly appear in real cases, and we wanted to illustrate that these data and processes are feasible.

4 Discussion and Limitations

Overall, we managed to create a purpose-driven application, that was systematically planned from the beginning. We ran into many obstacles during the development, which we solved with research and a lot of debugging. We think the created application is rudimentary, but it could be a very good basis for a further developed application, or even for an Enterprise Resource Planning system, which was partly introduced in prior publications [10] [23].

There are some potential positive impacts for the users of this application. First of all, it would create a company-wide platform for every user to simultaneously see all information for every transportation and be aware of any changes immediately. It would be useful for financial decisions too, since the program stores incomes and could provide feedback the hauler's performance. It could alleviate various business decisions for leaders.

This program offers basic data storage and manipulation; therefore, it has some limitations with respect to the results. The users are not capable of monitoring changes and do not know who made those changes, though, that would be essential for responsibility issues. Also, there is no way of changing the price of a tour into a different currency, as there is no implemented changing method. Another limitation is that the program currently only accepts 6-figure Number Plates, however, there are cases worldwide, where the number of the characters is more (or less) or various formats are authorized (e.g., in Singapore). Before scaling the project, it would be essential that the program become multilingual, but for now, it is only in English – relying on online translation tools.

Conclusions

Our improvement was designed to enable basic operations while meeting domain-specific requirements. Further developments are aimed at keeping pace with today's technological and professional developments and the latest advances in Artificial Intelligence. The development of global transport and freight optimization software tools is an emerging area. Significant developments in logistics and supply chain management require new approaches and agile development. By integrating advanced algorithms, data analytics and real-time monitoring capabilities, our software offers the potential to increase the efficiency, sustainability and reliability of freight transport, starting at the local level.

The challenges of managing complex supply chains across diverse geo-graphical regions are formidable, but the capabilities provided by AI-supported intelligent software empower stakeholders to make well-informed decisions and maximize the efficient use of resources. By leveraging AI technology, organizations can optimize operations, reduce costs and increase customer satisfaction, while simultaneously addressing environmental concerns and regulatory requirements. Additional modern communication systems and optimizations can further assist in making superior decisions, which are supported by a number of technologies [24] [25].

The aim of our work was to meet domain-specific requirements, but the application in its current form, can only perform the most basic operations and is not suitable for more complex tasks. Further improvements are needed to keep pace with today's technological and professional developments, as well as with the latest advances in AI. This has already been pointed out by numerous professional researchers [26] [27]. The development of global transportation and freight optimization software tools represents an evolving field. The significant advancement in the field of logistics and supply chain management requires new approaches and nimble development. Through the integration of advanced algorithms, data analytics, and real-time monitoring capabilities, our software offers the potential to increase the efficiency, sustainability and reliability of freight transportation, starting on a local scale.

The challenges inherent in managing complex supply chains across diverse geographical regions are formidable, but the capabilities provided by AI-supported intelligent software, enable stakeholders to make informed decisions, optimize resource utilization and mitigate risks more effectively. By harnessing AI technology, organizations can streamline operations, cut costs and improve customer satisfaction, while addressing the various environmental and regulatory concerns.

Acknowledgment

T. Haidegger is a Consolidator Researcher, receiving financial support from the Distinguished Researcher program of Obuda University.

References

[1] Heinbach, Christoph, Jan Beinke, Friedemann Kammler, and Oliver Thomas. "Data-driven forwarding: a typology of digital platforms for road freight transport management." Electronic Markets 32, No. 2 (2022): 807-828

[2] Schweiger, Karolin, and Lukas Preis. "Urban air mobility: Systematic review of scientific publications and regulations for vertiport design and operations." Drones 6, No. 7 (2022): 179

[3] Liang, J., Li, Y., Yin, G., Xu, L., Lu, Y., Feng, J., Shen, T. and Cai, G., (2022) A MAS-based hierarchical architecture for the cooperation control of connected and automated vehicles. IEEE Transactions on Vehicular Technology, 72(2), pp. 1559-1573

[4] Research and Markets: Global Freight Trucking Market to Reach $3.4 Trillion by 2030, Driven by Truck Trailer Segment and Post-Pandemic Recovery. Online: https://tinyurl.com/opitznagy1

[5] Takács, K., Mason, A., Cordova-Lopez, L. E., Alexy, M., Galambos, P. et al., (2022) Current Safety Legislation of Food Processing Smart Robot Systems–The Red Meat Sector. Acta Polytechnica Hungarica, 19(11), pp. 249-267

[6] Lukács, E., Levendovics, R., & Haidegger, T. (2023) Enhancing autonomous skill assessment of robot-assisted minimally invasive surgery: A comprehensive analysis of global and gesture-level techniques applied on the JIGSAWS dataset. Acta Polytech. Hung, 20(8), 133-153

[7] Mason, A., Haidegger, T., & Alvseike, O. (2023) Time for change: The case of robotic food processing. IEEE Robotics & Automation Magazine, 30(2), 116-122

[8] Haidegger, T.; Mai, V.; Mörch, C.; Boesl, D.; Jacobs, A.; Rao R, B.; Khamis, A.; Lach, L.; Vanderborght, B. Robotics: Enabler and inhibitor of the Sustainable Development Goals. Sustainable Production and Consumption 2023, 43, 422-434, https://doi.org/10.101 7546/j.spc.2023.11.011. 755

[9] Haidegger, T. P., Galambos, P., Tar, J. K., Kozlovszky, M., Zrubka, Z., Eigner, G., & Rudas, I. J. (2024) Strategies and outcomes of building a successful university research and innovation ecosystem. Acta Polytechnica Hungarica, 21(10), pp. 13-35

[10] Fu, L., Sun, D., & Rilett, L. R. (2006) Heuristic shortest path algorithms for transportation applications: State of the art. Computers & Operations Research, 33(11), 3324-3343

[11] Liang, J., Tian, Q., Feng, J., Pi, D. and Yin, G., (2023) A polytopic model-based robust predictive control scheme for path tracking of autonomous vehicles. IEEE Transactions on Intelligent Vehicles

[12] Rudas, Imre J., and Jozef K. Tar. "Computational intelligence for problem solving in engineering." In IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society, pp. 1317-1322, 2010

[13] Opitz, P., Haidegger, T., & Nagy, E. Concept and Considerations for the Development of an Information Database for Efficient International Freight Transport. In 2025 IEEE 12th Intl. Conf. on Computational Cybernetics and Cyber-Medical Systems (ICCC) pp. 209-214, 2025

[14] Houghtaling, M. A.; Fiorini, S. R.; Fabiano, N.; Gonçalves, P. J.; Ulgen, O.; .; et al. Standardizing an ontology for ethically aligned 395 robotic and autonomous systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems 2024, 54, 1791-1804

[15] Statista: Size of the road freight market in Europe from 2010 to 2025. Online: https://tinyurl.com/opitznagy2

[16] Statista: Amount of freight transported by road in Hungary from 2008 to 2022. Online: https://tinyurl.com/opitznagy3

[17] E Prestes, MA Houghtaling, PJS Gonçalves, N Fabiano, O Ulgen, et al. The First Global Ontological Standard for Ethically Driven Robotics and Automation Systems. IEEE Robotics & Automation Magazine 28 (4), pp. 120-124

[18] Ma, Y., Amiri, A., Hassini, E. and Razavi, S., (2022) Transportation data visualization with a focus on freight: a literature review. Transportation Planning and Technology, 45(4), pp. 358-401

[19] Vogel, D. R., Dickson, G. W. and Lehman, J. A., (1986) Persuasion and the role of visual presentation support: The UM/3M study

[20] Torre-Bastida, A. I., Del Ser, J., Laña, I., Ilardia, M., Bilbao, M. N. and Campos-Cordobés, S., (2018) Big Data for transportation and mobility:

recent advances, trends and challenges. IET Intelligent Transport Systems, 12(8), pp. 742-755

[21] E. Sós, and P. Földesi: Application of the QFD Technique Method in Logistics Strategy. Acta Polytechnica Hungarica Vol. 20, No. 2, 2023, pp. 145-164

[22] Ngampravatdee, Chayaporn, Koorosh Gharehbaghi, Amin Hosseinian-Far, Kong Fah Tee, and Kerry McManus. "Strategic initiatives for large transport infrastructure planning36: reinforcing sustainability in urban transportation through better stakeholder engagement." Sustainability 15, No. 18 (2023): 13912

[23] Sun, X., Yu, H., Solvang, W. D., Wang, Y. and Wang, K., (2022) The application of Industry 4.0 technologies in sustainable logistics: a systematic literature review (2012-2020) to explore future research opportunities. Environmental Science and Pollution Research, pp. 1-32

[24] Molnár, Gy., New learning spaces? M-learning's, in particular the IPad's potentials in education, International Journal of Interactive Mobile Technologies 7: 1, 2013, pp. 56-60

[25] Tick, A. and Molnár, Gy., Improving Efficiency: P-Graph Modeling for Business Workflows with Dynamic and Fuzzy Extensions, Acta Polytechnica Hungarica 22 : 1, 2025, pp. 79-100

[26] Nagy, E.; Horváth, F., Molnár, Gy., CMDB Reporting Introduction and Quality Improvement at a Given Hungarian Company, Acta Polytechnica Hungarica 21 : 9, 2024, pp. 109-127

[27] Molnár, Gy. and Nagy, E., Current Issues in Effective Learning: Methodological and Technological Challenges and Opportunities Based on Modern ICT and Artificial Intelligence, EAI/Springer Innovations in Communication and Computing 1 2025, pp. 1-11, Paper: Chapter 1