Paisii Hilendarski University of Plovdiv

Faculty of Mathematics and Informatics

Department of Computer Systems

Development of intelligent tools for working with virtualized cultural and historical sites

ABSTRACT

of a doctoral dissertation for the award of the educational and scientific degree of Doctor (PhD) in the field of higher education 4. Natural Sciences, Mathematics and Informatics; professional field 4.6. Informatics and Computer Science;

Doctoral program Informatics

PhD student: Ilia Iliev Nedelchev

Scientific supervisor: Prof. Dr. Stanimir Stoyanov

Plovdiv, 2025

The dissertation work was discussed and directed for defense before a scientific jury at a meeting of the Department of "Computer Systems" at the Faculty of Mathematics and Informatics of the "Paisii Hilendarski" University, Plovdiv, on 07.11.2025

The dissertation work contains 144 pages. The bibliography includes 116 sources. The list of author's publications consists of 2 titles.

The defense of the dissertation work will take place on 06.02.2026 in the Conference Hall of the New Building of the "Paisii Hilendarski University", Plovdiv.

The materials for the defense are available to those interested in the secretariat of the Faculty of Mathematics and Informatics - room 330 in the New Building of the "Paisii Hilendarski" University, every working day from 08:30 to 17:00.

Scientific jury:

Internal Members:

1) Prof. Dr. Asya Georgieva Stoyanova-Doycheva – Faculty of Mathematics and Informatics, Plovdiv University „Paisii Hilendarski“.

2) Assoc. Prof. Dr. Veneta Veselinova Tabakova-Komsalova – Faculty of Mathematics and Informatics, Plovdiv University „Paisii Hilendarski“.

External Members:

3) Prof. Dr. Vladimir Vasilev Monov – Institute of Information and Communication Technologies, Bulgarian Academy of Sciences (IIKT-BAS).

4) Prof. Dr. Evdokia Nikolaeva Sotirova – University “Prof. Dr. Asen Zlatarov”, Burgas.

5) Assoc. Prof. Dr. Emil Georgiev Delinov – Thrace University – Stara Zagora.

Reserve Members:

6) Prof. Dr. Sotir Nikolov Sotirov – University “Prof. Dr. Asen Zlatarov”, Burgas.

7) Assoc. Prof. Dr. Georgi Nikolov Cholakov – Faculty of Mathematics and Informatics, Plovdiv University „Paisii Hilendarski“.

Author: Ilia Iliev Nedelchev

Title: "Development of intelligent tools for working with virtualized cultural and historical sites"

Plovdiv, 2025

INTRODICTION

The dissertation is dedicated to the development of intelligent, knowledge-based tools for the digital presentation of our cultural and historical heritage.

The primary objective of the dissertation is to establish a platform for the digitization of our cultural and historical heritage. A key component of this platform will be a personal tourist guide. To achieve the goal, the following tasks have been formulated:

- Analysis of the latest trends in building such information systems.
- Proposal for improving the platform architecture.
- Development of a concept, model, and architecture of a personalized tourist guide.
- Implementation of a prototype of a personal tourist guide and conducting experiments with the prototype.

The research conducted during the doctoral program goes through the following stages:

- Creation of a general concept, model, and architecture of a basic tourist guide.
- Development of a prototype of the basic tourist guide, which can be adapted for a specific tourist destination.
- Parameters are specified, taking into account personal characteristics of tourists, which can be used to personalize the basic tourist guide.
- The possibility of adapting the basic tourist guide for a different application area is demonstrated - in this case, for monitoring air quality in the Plovdiv region.

In accordance with the goal and objectives, an appropriate methodology for conducting the research is proposed. The dissertation consists of an introduction, four chapters, and a conclusion.

CHAPTER ONE: STATE OF THE PROBLEM

In accordance with the topic of the dissertation, the following are of interest:

- The specific aspects of modern tourists.
- The peculiarities of the virtual tourist guide.
- Intelligent agents.

The specific aspects of tourist services and tourism are analyzed. The strategic goals in the management of the tourist destination are related to improving its competitiveness, increasing tourist satisfaction, and achieving a balance in the interests of all subjects (national and local authorities, tourist business, local community, and tourists) [1], as well as to achieving sustainable tourism, requiring responsible behavior of all parties. Essential for the development of tourism is the quality of the tourist product, which encompasses various goods and services designed to meet tourists' needs [2]. The market for tourist services is extremely specific due to the provision of various types of services, including information [3]. Technologies in tourism are a multidimensional construct, and their effectiveness is assessed based on four attributes: accessibility, informativeness, interactivity, and personalization [4].

The features of the virtual tourist guide are presented. The main approach in creating virtual tourist guides is to create a system that will facilitate the mechanism of traditional information search and ticket booking/purchase. The systems created should reduce the activities performed by tourists by searching for information about tourist attractions, their working hours, a suitable period for visiting, and a way to reach them, which will allow for more effective time management by tourists. The virtual tourist guide needs to have a user-oriented design to meet their needs and facilitate the process of using the application. The design of the application, which is user-oriented, should be interactive, requiring the opinion of users during its development to create a usable and accessible product. The main meaning of the virtual tourist guide is to help the tourist find the information they need at the right time. The long time it takes tourists to navigate an unfamiliar city and identify

tourist attractions and their locations can be significantly reduced by using a virtual tour guide.

The specifics of artificial intelligence and intelligent agents are analyzed. Definitions of the essence of artificial intelligence are different, as it is defined as machine learning, which includes programming approaches that use "various algorithms and methods, such as linear regression, decision trees, Bayesian networks, evolutionary algorithms, and artificial neural networks" [5], as well as a solution that increases the power of computer systems. Artificial intelligence is also the ability to create such software, through which to "demonstrate thinking comparable to that of humans, with the artificial intelligence being able to perceive, analyze and interact with its environment, learn from previous experience and solve complex problems autonomously without human intervention" [6].

Intelligent agents are, by their nature, independent programs or objects that interact with their environment by perceiving it through sensors. In artificial intelligence, intelligent agents are autonomous entities using sensors and actuators to explore the environment and perform various activities that are rational, i.e., maximize efficiency and achieve the best outcome.

From the review of the state of the issue, conclusions were drawn that guided the conduct of the study, the results of which are presented in the dissertation.

Regarding the specifics of modern tourists:

- Shifting the focus of the tourist offer from traditional to individual tourist offers.
- Change in the preferences of tourists from the consumption of unified products to the use of individual offers to achieve specific experiences.
- Tourists are distinguished into those who use organized tourist trips and those who individually organize their visit to the tourist destination.

Regarding the virtual tourist guide:

- The virtual tourist guide should provide the opportunity to adapt to different regions.
- To verify the performance of the results, the implemented system was adapted for the city of Gabrovo.
- The main goal of creating a virtual tourist guide is to solve the above-mentioned problems related to the awareness of tourists about different tourist destinations (Gabrovo was chosen as a tourist destination for testing the guide).

- The virtual tourist guide will support the work of tourist information centers (in this case, Gabrovo), which is why it is necessary to be directed at the individual tourist and provide him with the opportunity to create an individual optimal route, according to personal preferences.
- The virtual tourist guide must adapt to changes in the environment, such as changing weather conditions, deviation of the tourist from the set route, traffic jams on the road, etc.
- The main scenario of the virtual tourist guide is to create a tourist route, using environmental data such as time, location of a person, as well as parameters set by the tourist – budget, distance and time available. When creating the tourist route, the guide will provide information to the tourist about the most highly rated tourist sites.
- Although in recent years the development of more information systems and applications in the field of tourism has been established, they have been developed for specific tourist services, which require tourists to use several applications when visiting tourist destinations. Few applications have been developed that provide tourists with access to all tourist services, contain all the necessary information, and use tools for analyzing the situation in the tourist destination in real time.

For the successful implementation of the planned virtual tourist guide, a thorough review and analysis of the existing state of intelligent agents (Jason, Natural Language Processing, NLU – Rasa, LLM – OpenAI, and Vips) is necessary.

CHAPTER TWO: MODEL AND ARCHITECTURE OF THE PERSONAL TOUR GUIDE

The region of the city of Gabrovo was chosen to test the proposed model, architecture, and prototype implementation of a tourist guide. The prototype is adapted to serve tourists visiting this region. Gabrovo has a strategic location in terms of accessibility and attracting tourists, both from the Black Sea coast and from Western Bulgaria. The general awareness of Gabrovo as a tourist destination can be described as unsatisfactory. The main drawback is the chaotic presentation of information about individual tourist sites, such as places to stay, eat, tourist attractions, transportation options, etc., and limited opportunity for online communication. Mobile applications are missing, which is a hindering factor in attracting tourists. The main goal of creating a virtual tourist guide is to solve the above-mentioned problems related to tourists' awareness of the tourist destination. The virtual tourist guide will support the work of the tourist information center in Gabrovo, which is why it is necessary to be directed at the individual tourist and provide him with the opportunity to compile an individual optimal route, according to personal preferences. Gabrovo and the surrounding area contain numerous tourist sites and attractions, which, due to their historical, cultural, and natural riches, should be visited by every tourist.

The main functionalities of the virtual agent are related to the organization of an optimal tourist route, which will be tailored to the tourists' free time, the type of transport they have, the budget, and the distance they are willing to travel. Unlike traditional information applications, the virtual tourist guide will not only present a list of all tourist sites in Gabrovo and the region, but will also prepare a personal tourist plan.

The virtual tourist guide is a developed mobile application that is available for:

- mobile devices with Android operating system;
- website (https://exploring-gabrovo.com/).

The system consists of several parts:

- mobile application;
- public website;
- content management panel.

The created mobile application is extremely intuitive, creating an individual plan for users in two steps. Figure 1 presents the steps that the user

of the mobile application should take in order to have a plan prepared for them.

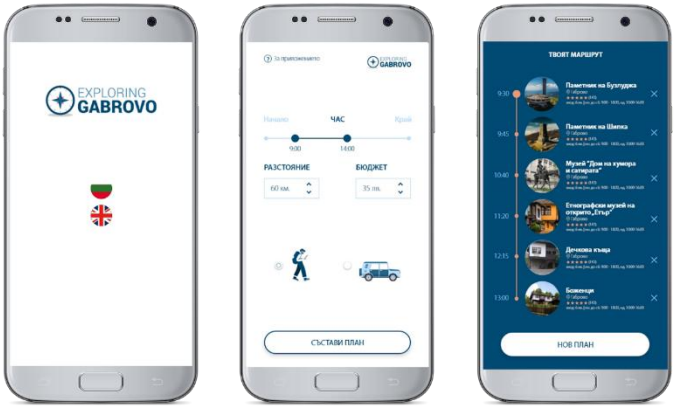

Figure 1. Steps for preparing a tourism plan

The steps for preparing a travel plan are:

- First, a language must be selected, because the application is planned for both Bulgarian and foreign tourists. For this purpose, there is an option in the interface to select Bulgarian or English.
- Second, the user must indicate the time interval for his plan, the distance he is willing to travel, his budget, and his mode of transportation.

After pressing the "Create a plan" button, a screen with a generated tourist plan appears. The user is given the opportunity to exclude objects, thereby generating a new, more optimal plan than the previous one. The virtual tourist guide allows, when marking a given object by the user, to visualize additional information about it, which includes the location of the object, entrance fee, weather conditions in which the visit is appropriate, a description of the object, and a gallery. When generating the tourist plan according to the parameters set by the user, the current location of the user is also considered through the geolocation function of the mobile phone. The virtual tourist guide also considers the meteorological situation at the time of generating the plan, selecting only tourist objects that are suitable for visiting according to the specifics of the weather. The selection of tourist objects that

are included in the virtual tourist guide is based on the rating received for them from Google ratings.

The public website has the same functionality as the mobile application and includes a list of all tourist objects that are available in the system.

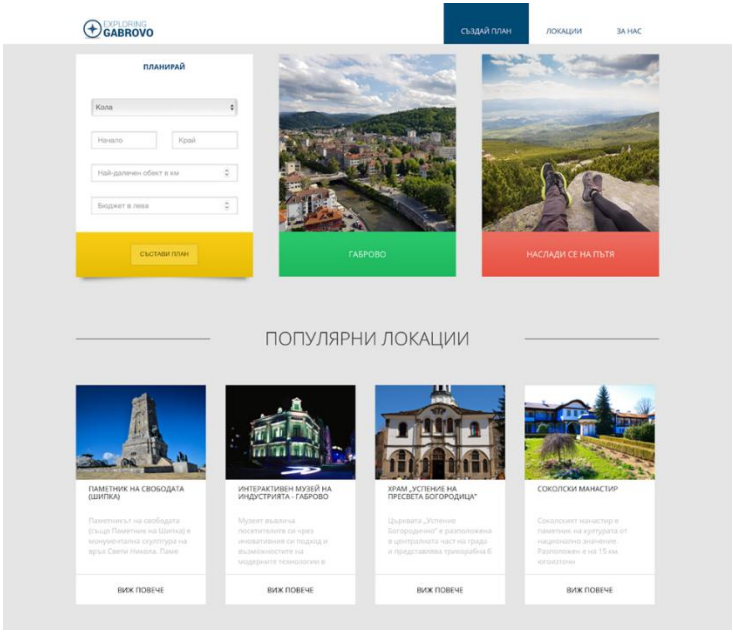

Figure 2. Home page of the public website

Fig. 2 shows the home page of the public website, which also contains the form through which users can generate a tourist route. The home page also presents a list of popular locations. The public website contains a list of all tourist sites in Gabrovo and the region. The sites in the list are sorted by site rating, starting with the highest rated sites.

The site management panel is intended only for users who have the right to edit and add content. Such users can be system administrators, employees of a tourist information center, or employees of a specific tourist site. In the content management panel, new sites can be added, the content of already added sites can be edited, and existing ones can be deleted. In this panel, users can be added and removed from the administrators, with each user having either an administrator or an editor role. The site management panel screen

includes a list of all sites, with an edit button and a delete button next to each site. On the same page, there is also a button to add a new object. When you click add or edit, a window opens to fill in or edit information about the object.

Fig. 3 presents the architecture of the tourist guide. The mobile application makes requests to a web resource to retrieve the necessary information about the objects and generate a tourist route for the user. The web application makes requests for a database in which information about objects is stored. The mobile application was implemented using Flutter technology, which is an open-source user interface software development kit created by Google [7]. Applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web from a single code base were also used for development. The web application was implemented using ASP.NET Core technology – a free open-source web platform and successor to ASP.NET, developed by Microsoft [8]. The web application consists of a website and a REST API, with the REST API processing the requests sent by the mobile application [9].

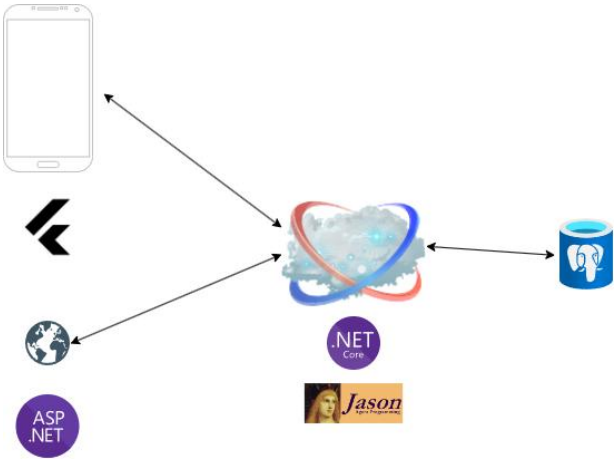

Figure 3. Application architecture visualization

The web application is hosted on the Heroku cloud platform, and the virtualization tool is Docker [10]. Docker is a set of platform-as-a-service products that use OS-level virtualization to deliver software in packages called containers. The open-source relational database PostgreSQL is used as the database, which supports multiple data types, transactions, encryption,

regular expressions, etc., and is widely used, including by large companies [11]. The architecture of the virtual tour guide contains intelligent agents, the decision being made due to the fact that one of the ideas behind the creation of intelligent agents is that they should not simply follow a certain algorithm, as is the case with "typical" computer programs, but should act as a human would. The main properties of intelligent agents are autonomy, proactivity, reactivity, and social skills. The virtual tour guide is a multi-layered system [12] that includes a multi-agent environment implemented on the Jason framework. Agent Environment Service is a Java environment for Jason agents. It includes a Rest Api implemented using Javalin. This allows agents to communicate with the outside world.

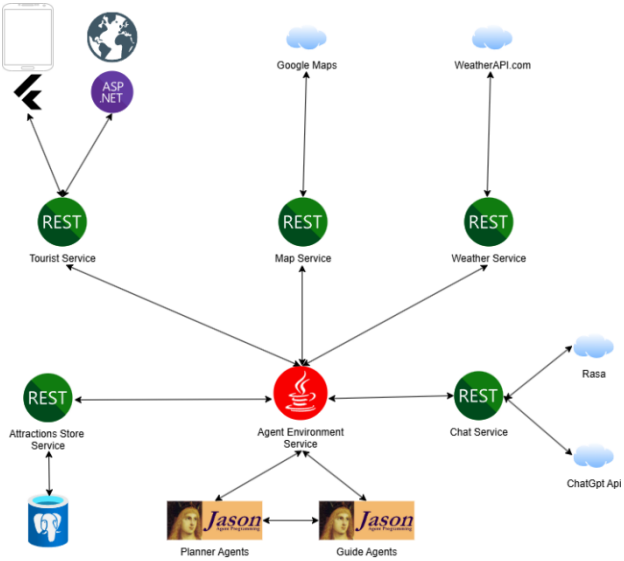

Figure 4. Multi-agent environment of the virtual tour guide

Two types of Jason agents are used:

- Planner Agents – The role of which is to manage the Guide agents when a request for a new plan comes. Planner Agents create a Guide agent. They also take care of releasing the Guide Agent when it is not needed.
- Guide Agent – this is an agent who takes care of the individual plan of the tourist. For each tourist, there is a separate Guide agent; this

agent generates a plan, monitors changes in the environment, location, weather conditions, working hours, and, based on these changes the plan if necessary. Its main goal is to take the user through the maximum rated attractions until the end of the time available to the user. This agent reacts proactively to changes in the environment.

The BDI–based architecture (Belief-desire-intention) was chosen for the architecture of the agents. The specified choice was made due to the fact that this type of architecture is as close as possible to human behavior when making decisions.

User part – initially, the mobile application for users was developed with Xamarin – it allows the development of platform-independent applications, which means that after development, there will be support for both Android and iSO. Xamarin is owned by Microsoft and supported by Microsoft, this is one of the reasons for choosing this technology. After testing Xamarin for the mobile technology, Flutter was chosen. An ASP.NET Core MVC application with Blazor was used to develop the website. Blazor is compiled to Web Assembly, which is executed on the client's web browser. The mobile application and the website will share a large part of the logic, which will be written in the C# programming language. Both applications will address a Web application, which will store user data and objects. The website will also contain an additional part that will serve for the administration of users and objects by users with the administrator role.

Server part – the server part consists of a web application, a database, and a Java REST application with the JSON environment [13]. The web application uses ASP.NET Core technology to implement a REST application. This application will store and administer data about users and objects and will make requests to the Jason environment for preparing individual tourist plans. PostreSQL is used as a database. PostreSQL is an open source relational database. The Jason environment will be integrated with the Spark library, through which the REST application will be implemented.

In addition to the multi-agent environment, there are 4 more web applications and one mobile application. Attractions Store is implemented using ASP:

- NET Core is an application that contains information about tourist attractions. It has a connection to a Sql Server database, which contains information about tourist attractions and their characteristics, such as opening hours, a photo gallery, whether they

are suitable for rainy weather, whether they are outdoors or indoors, GPS location.

- Tourist Service is also implemented using ASP.NET Core technology. The purpose of this web application is to accept requests from tourists made through the mobile application, as well as regular information about changes in the tourist (such as current location and changes in the requirements for drawing up the plan).
- Map Service takes care of storing the distances between the attractions, and can return information if the user is not close to a given attraction. The application also connects to the Google Routes service, which helps calculate how long it will take the user to reach a given attraction based on whether they are by car or on foot.
- Weather Service returns information about the weather conditions at a given object and where the tourist is located. This service makes calls to WeatherApi.Com at a certain interval and caches the information, thus reducing calls to WeatherApi.Com.
- Chat Service manages the chat interaction with the user, as the user can always ask a question in a chat window and receive an answer. There are two types of chat. Rasa uses predefined scenarios, such as Rasa Nlu (Natural Language Understanding), to understand the user's intention. If the Rasa scenario cannot process the given user request, the chat is forwarded to the ChatGp API, which uses LLM to process a general request. When the user approaches a given object, its GPS coordinates are close to the coordinates of the object, and a chat is triggered for the corresponding object, and Rasa scenarios for the corresponding object are triggered again first, but if the conversation goes outside the defined scenario, a call is made to Chat GPT.

The mobile application is implemented using Flutter technology, constantly sending the user's coordinates after starting the plan execution. Thus, if necessary and if the user deviates, the Guide agent sends a message to the Tourist Service, which notifies the user accordingly. It also allows him to change the plan when changing the direction, weather conditions, or when a tourist site closes for the day.

When processing a request for a plan, the following activities are included:

- Request from a mobile application or website to create a plan. The user ID, location, plan start time, and plan end time are sent.
- A request is made to the database to retrieve the user profile by ID.

- Response to the request to the database for the user profile.
- A request is made to the Jason environment to create a travel plan for the user with the user profile, location, plan start time, and plan end time.
- Response that the plan was successfully accepted and placed in the agent's queue for processing.
- Request from the Jason environment to the Web application for information about objects near the location and working hours in the plan interval.
- Request to the database for information about objects that are near the location and have working hours in the given interval.
- Response from the database to the web application with information about the objects.
- Response from the web application to the Jason environment with information about the objects.
- Sending the prepared plan from the Jason environment to the web application.
- Response from the web application that it has received the plan successfully.
- Response from the web application to the user, including the prepared plan.

The processes in the virtual tour guide system include:

- Process before the user registers or authenticates in the system. This is the process in which the user enters the system, but not through their profile (or does not yet have one). He will see the registration and login form, will be able to view the various objects, but will not have access to the route preparation module.
- Process of registration and filling out a profile for a new user. In this process, the user will create their profile and fill in their preferences through a questionnaire.
- Process of preparing a route for a logged-in user. This process can be started once the user has entered their profile and, based on the data provided by them, a personal route can be prepared for them.
- Process of administering content in the system. This process can only be performed by a user with the administrator role.

THREE: PROTOTYPE IMPLEMENTATION

Jason, a tool for building a multi-agent system, was used for the programmatic implementation of the prototype. Jason is an open source interpreter that extends AgentSpeak, allowing the creation of an environment for agents using the Java programming language. The agents themselves are described using Prolog. Advances in artificial intelligence will require the development of agents that seamlessly combine reactive and proactive behavior. These agents will not only respond to immediate needs, but will also anticipate and prepare for future challenges [14]. The above-mentioned properties of intelligent agents allow the tour guide to effectively collaborate with individual agents. The achievement of the tour guide's goals is also based on a multi-agent environment. It is important to note that the agents used can adapt to changes in the state of the various tourist sites that are used.

The architecture used by the tour guide in terms of agents is BDI - Belief-desire-intention. This architecture is a major theme in many studies on intelligent agents. In the BDI architecture, beliefs represent the agent's information about its environment, other agents, and itself, goals (desires) are states to be achieved, and intentions are commitments to achieve specific goals [83]. The architecture of the prototype system is presented in Figure 5.

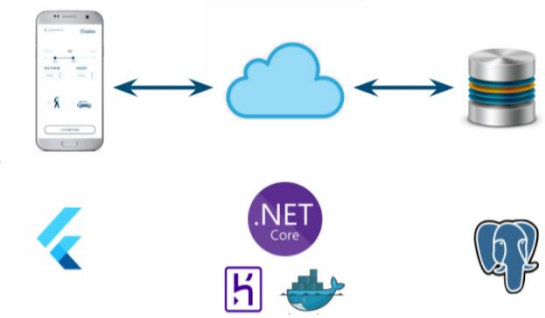

Figure 5. Prototype system architecture

As can be seen from the architecture presented in Figure 5, the mobile application makes requests to a web resource to retrieve information about the objects and to generate a tourist route for the client. In turn, the web application makes requests to a database in which the information about the

objects is stored. The mobile application is implemented using Flutter (an open source user interface software created by Google that is used to develop applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia and the web from a single code base).

The web application is implemented using ASP.NET Core technology – a free open source web platform and successor to ASP.NET developed by Microsoft. The web application consists of a website and a REST API, with the REST API processing the requests sent by the mobile application. The web application is hosted on the Heroku cloud platform, and the virtualization tool is Docker (a set of platform-as-a-service products that use OS-level virtualization to deliver software in packages called containers). The database uses Post-greSQL (an open source relational database that supports multiple data types, transactions, encryption, regular expressions, etc.).

Figure 6 shows a diagram of the database schema.

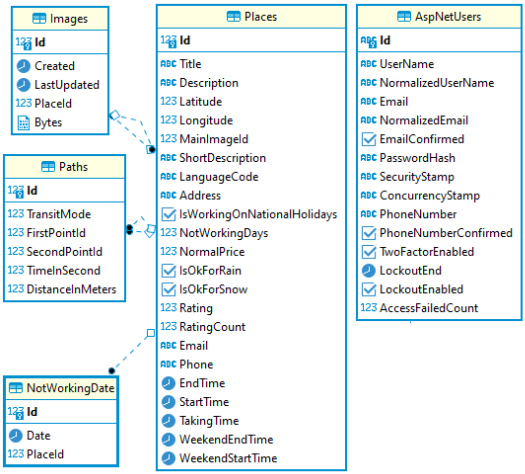

Figure 6. Database schema diagram

The following tables are used:

- The Images table stores the photos of the tourist sites (stored as byte arrays in the database).
- The Paths table contains the distance between two tourist sites for a given mode of transport.

- The NotWorkingDate table stores the dates during which the site is not working.
- The Places table stores information about the tourist sites. The AspNetUsers table stores information about the users.

Flutter was used to create the mobile application on the client side – allowing the creation of a platform-independent application, which means that after development, there is support for Android for iSO. The main reason for choosing Flutter is that it is owned by Google and is supported by the company. An ASP.NET Core MVC application with Blazor was used to create the website on the client side. Blazor is compiled to Web Assembly, which is executed on the client's web browser. The mobile application and the website will share a large part of the logic, which will be written in the C# programming language. The two applications address a web application that stores data about the user and tourist objects. The website also contains an additional part aimed at allowing the administration of users and objects by the user, with the role of administrator.

The server part of the tourist guide consists of a web application, a database, and a Java REST application with the Jason framework. The web application uses ASP.NET Core technology to implement a REST application that stores and administers data about users and objects and makes requests to the Jason environment to prepare individual tourist plans. PostreSQL is used as a database. – an open source relational database. The Jason environment is integrated with the Spark library, through which the REST application is implemented. The components of the tourist are:

- Planning Agents – When a new traveler starts using the mobile app, they are assigned a planning agent. The primary task of the planning agent is to create a tour guide agent that is specifically tailored to the traveler’s needs.
- Tour Guide Agents – are responsible for all aspects of trip planning, including gathering information, generating plans, monitoring the user’s time and location, and adjusting plans as needed. They use the Jason Framework to manage their beliefs, desires, and intentions, which allows them to adapt their plans based on various factors.
- Information Collection – The tour guide agent collects all the necessary data for trip planning using various RESTful API services. This includes data related to tourist attractions and weather conditions. The agent uses this data to make informed decisions and provide appropriate recommendations.

- Mobile App – serves as the primary interface for the user to interact with the tour guide agent. Through the mobile app, users can enter their preferences, such as budget, time frame, and distance constraints. They can also review the generated travel plan, provide feedback, and request changes.
- Plan Generation – Based on the user's preferences and real-time data collected from various API services, the tour guide agent creates an initial travel plan that takes into account the user's constraints and factors, such as tourist attractions, accommodations, meals, and transportation.
- Weather Monitoring and User Location Tracking – The tour guide agent continuously monitors weather conditions and the user's location using API services and GPS data from the user's device. This allows the agent to adjust the travel plan in real-time based on changing weather conditions or the user's movements.
- Plan Adjustment – The tour guide agent can adjust the travel plan based on feedback, requests, or changes in real-time data from the user. For example, if the user decides to extend their stay at a particular location, the agent can change the plan to accommodate this change, while also taking into account the user's preferences and constraints.

In this system, the agent planner focuses on creating tour guide agents, while the tour guide agents handle all aspects of trip planning and user communication through the mobile application. The combination of Jason Framework MAS and RESTful API services allows for the development of a highly adaptive and personalized tour guide application that can respond to changing conditions and user input, providing an optimal trip tailored to the unique needs and preferences of each user.

A planner agent is an intelligent software object that coordinates and manages tasks by creating, monitoring, and controlling other agents or processes. Using reasoning and decision-making capabilities, planner agents process incoming requests, allocate resources, and set goals for subordinate agents. In multi-agent systems, planner agents play a key role in managing complex tasks and achieving system goals by dynamically adapting to changes in the environment and adjusting the behavior of individual agents accordingly.

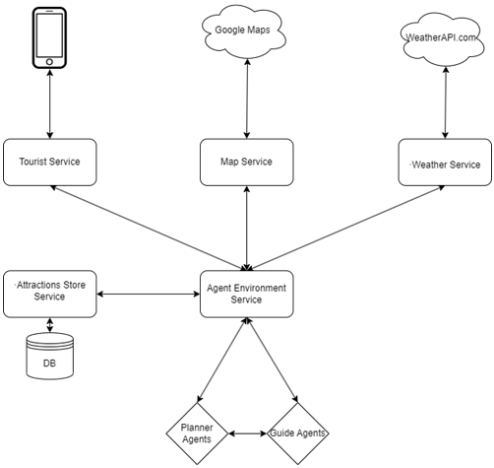

Figure 7. Service and Agent Architecture

By effectively orchestrating tasks and resources, scheduling agents can improve the overall performance and resilience of the system. A segment of the scheduling agent implementation is shown in Fig. 8.

```
+tourist(ID,X,Y,W,B,T):true
<- .print("нов турист");
.create_agent(ID, "guide.asl");
.send(ID,tell,init(ID));
.send(ID,tell,gps(X,Y));
.send(ID,tell,weather(W));
.send(ID,tell,budget(B));
.send(ID,tell,time(T));
.send(ID,tell,startGuide(ID)).
+finishPlane(X):true
<- .print("финиш на гида");
.kill_agent(X).
```

Figure 8. Segment of the agent implementation program

A guide agent is an intelligent software object designed to assist users, such as tourists, by providing personalized recommendations and plans based

on their preferences, constraints, and environmental factors. Using advanced reasoning and decision-making capabilities, guide agents adapt to dynamic environments and tailor their guidance to create a satisfying experience for users. In multi-agent systems, guide agents interact with other agents, such as planning agents, to coordinate and efficiently perform tasks. By leveraging their knowledge base and adapting to real-time changes, guide agents provide a high level of personalization, improving the user experience and providing valuable assistance in various areas. Figure 9 shows a code fragment describing a Jason guide agent. This code fragment describes a Jason guide agent that assists tourists by creating personalized plans based on the information provided.

```
@init[atomic]
+init(ID): true
<- init(ID).
// Ново местоположение.
+gps(X,Y): true
<- gps(X,Y).
+weather(W): true
<- weather(W).
+budget(B): true
<- budget(B).
+time(T): true
<- time(T).
+startGuide(ID): true
<- startGuide(ID);
!createPlan.
// Завършване на туристическия план или време.
+finish(X): true
<- .send(planer,tell,finishPlane(X)).
+visit(A): true
<- visit(A).
+checkTime(T):true
<- adjustPlanAccordingTime(T).
+changeDirection(X,Y):true
<- adjustPlanAccordingDirection(X,Y).
+changeWeather(W):true
<- adjustPlanAccordingWeather(W).
+!createPlan:true
<- createPlan.
```

Figure 9. A code snippet describing a tour guide agent in Jason

The multi-agent travel guide, powered by Rasa, Jason, ChatGPT, and the .NET 6 REST API, is a testament to the symbiotic relationship between technology and tourism, not only disseminating information but also fostering engagement, providing not only directions but also cultural exploration of Gabrovo. The latest version of the app continues to develop a mobile app as a multi-functional travel guide system designed to offer travelers a comprehensive experience when exploring Gabrovo. The app offers a wide range of features, including navigation, information, and local resources, but the integration of a chatbot and voice bot enhances its usefulness. The app serves as a versatile tool that meets multiple traveler needs. It encompasses functionalities such as navigation assistance, event planning, local cuisine recommendations, and real-time updates on attractions and events. Users can access a wealth of information about Gabrovo's history, culture, and traditions through an intuitive and easy-to-use interface.

The latest version of the app includes a chatbot and voice bot as valuable additions to the mobile app, offering a conversational layer that seamlessly integrates with the broader app functionalities. These AI-powered agents act as responsive and informative companions, enriching the traveler's experience with personalized interactions. When users access specific menus in the app, it automatically opens a conversation mode. This mode provides a set of features and information about Gabrovo and its attractions. Travelers can then initiate inquiries, seek recommendations, or engage in casual dialogue to improve their understanding of the region.

The system uses two advanced conversational AI technologies, Rasa and ChatGPT, in an orchestrated manner. When a user initiates a conversation or query, the system first consults Rasa for intent recognition and dialog management. If Rasa determines that the user's query cannot be adequately answered, it gracefully switches to a fallback strategy by invoking the ChatGPT API. Rasa Open Source is an open source conversational AI platform that allows you to understand and conduct conversations, as well as connect to messaging channels and third-party systems through a set of APIs. It provides the building blocks for creating virtual (digital) assistants or chatbots. Training Rasa models to conduct intelligent conversations is a key aspect of the multi-agent travel guide. To achieve this, we use the standard Rasa REST API for training and inference, but our unique architecture includes a custom-built Flask REST API to facilitate communication with Rasa agents, each of which is loaded with separate models tailored for different purposes. Initial activities involve training Rasa models, a critical step in ensuring that the chatbot and voice bot are well-prepared to effectively

engage users. Rasa's universal framework allows for fine-tuning of models for precise intent recognition, object extraction, and dialogue management. This extensive training process enables agents to understand a wide range of user inputs, ranging from general queries about Gabrovo to highly specific questions related to individual attractions.

The core of the architecture is based on a custom Flask REST API, meticulously developed to organize communication with Rasa agents. This API serves as a central hub through which user requests, queries, and interactions are routed to the appropriate Rasa model, providing a contextually relevant and accurate response. Furthermore, the Flask REST API seamlessly communicates with the .NET 6 REST API, further enhancing the capabilities of the system.

The code used defines a Flask web application that serves as a communication bridge between a custom application and Rasa chatbot agents (Fig. 10). This code defines a Flask web application that serves as a communication bridge between your custom application and Rasa chatbot agents. It initializes and manages Rasa agents for various scenarios and allows you to send messages to these agents to generate responses. The code provides a RESTful API using Flask to initialize and interact with Rasa chatbot agents for various scenarios. It allows you to send messages to these agents and retrieve their responses, making it a key component of your multi-agent travel guide system. The multi-agent travel guide uses SQL Server as a reliable solution for storing Rasa models. The table structure adopted for this purpose consists of two main elements: model headers and the corresponding physical disk paths. This structured approach to model storage increases efficiency, scalability, and reliability, contributing to an improved user experience. Essentially, this code represents the key interaction between our travel guide app and the OpenAI GPT-based chatbot. It allows users to seamlessly search for information, ask questions, and engage in meaningful dialogues about Gabrovo and its attractions. This interaction is a key element of our multi-agent travel guide, enriching the user experience by providing informative and contextually relevant answers.

```
from flask import Flask, jsonify, request
import rasa.core.agent
from waitress import serve
def create_app(test_config=None):
app = Flask(__name__)
agents = dict()
endpoints = "endpoints.yml"
@app.route('/initialize/<int:scenario_id>', methods=['POST'])
async def initialize(scenario_id):
model_path = request.get_json()['modelPath']
endpoint_config = rasa.core.utils.read_endpoints_from_path(endpoints)
# Умишлено презаписване на настройката на модела в конфигурацията
на крайната точка засега
endpoint_config.model = None
agents[scenario_id] = await rasa.core.agent.load_agent(
model_path = model_path,
endpoints = endpoint_config)
return '', 204
@app.route('/send-message/<string:conversation_id>/<int:scenario_id>',
methods=['POST'])
async def send_message(conversation_id, scenario_id):
message = request.get_json()['message']
if scenario_id not in agents.keys():
return '', 403 results = []
responses = await agents[scenario_id].handle_text(message, None,
conversation_id)
for response in responses:
if "text" in response:
results.append(response["text"])
return jsonify(results)
return app
serve(create_app(), host="rasa.local", port=8080)
```

Figure 9. A code snippet describing a tour guide agent in Jason

In summary, the approach used integrates state-of-the-art technologies, with the inclusion of Rasa for intelligent chatbot functionality and the integration of ChatGPT for dynamic and context-aware conversations. The foundation of the system is Jason, a robust agent-based framework that plays

a key role in organizing interactions and delivering highly personalized user experiences.

Notable aspects of the created system include the implementation of a .NET 6 REST API to facilitate seamless communication, allowing Jason to interact with Rasa and ChatGPT. The critical role of SQL Server for the efficient management of model titles and physical locations is also highlighted, which contributes significantly to the speed and reliability of our travel guide.

The system adheres to a proactive approach, initiating conversations based on the user's GPS location, thus ensuring that travelers receive timely and contextually relevant information about nearby attractions. Furthermore, we seamlessly integrated a mobile application enriched with voice and chatbot functionalities, improving user engagement and accessibility.

During the development process, we clarified the practical application of Rasa and ChatGPT, demonstrating their key roles in facilitating conversations, answering queries, and providing detailed information about Gabrovo and its attractions.

Our multi-agent travel guide is a fusion of cutting-edge technology and pragmatic utility, designed to offer travelers an intelligent and engaging companion for exploring Gabrovo and the surrounding regions. By leveraging the potential of LLM, NLU, and agent-based frameworks, we created a system capable of responding to the diverse demands of tourists, providing personalized and information-rich experiences.

CHAPTER FOUR: PROTOTYPE OF A MULTI-AGENT PLATFORM TO SUPPORT OBJECTIVE AIR QUALITY ANALYSIS

The chapter demonstrates the possibilities of the developed architecture and prototype of a personal tour guide to adapt to a different application area – air monitoring in the Plovdiv region.

The proposed prototype is based on the fact that, due to various reasons – such as geographical location, climate, and climate change, being the center of vegetable production in Bulgaria, as well as the presence of a large copper-zinc plant and a waste landfill in the nearby area – Plovdiv is among the most polluted regions in Bulgaria. Air quality protection in Plovdiv is becoming an increasingly pressing issue. Since air quality monitoring is a sensitive topic for Plovdiv, there are often contradictory measurements, assumptions, opinions, and publications regarding the reliability of data and the location of monitoring instruments. The development of the prototype aims to create a multi-agent platform that, given this context, will support objective air quality analysis by:

- Reporting our measurements.
- Incorporating additional information from external sources.
- Comparing data and analyzing deviations and discrepancies.

The platform is hybrid, as it integrates methods from both symbolic and subsymbolic AI. Its core consists of two personal assistants: one based on symbolic AI methods, and the other on subsymbolic AI methods. In the evolution of AI, these two main approaches have often been seen as competing and mutually exclusive. However, the concept of integrated AI has recently gained momentum, highlighting the benefits of combining methods from both paradigms. For the purposes of developing the prototype, we believe that a hybrid platform is a suitable solution.

The first assistant, designed to process our measurements and work with our knowledge base, was developed as a BDI agent using JaCaMo technology. The second assistant, tasked with incorporating external sources and performing analysis, was implemented as a RaAct agent using LangChain technology.

The ACreM (Air Credible Monitoring) platform is agent-centric, hybrid, and regional [15]. It is agent-centric because the two main components of the platform are implemented as agents. The platform is primarily intended for use in the Plovdiv region. For this purpose, region-specific parameters have

been prepared that the platform uses in its work. The architecture of the ACreM platform is given in Fig. 10.

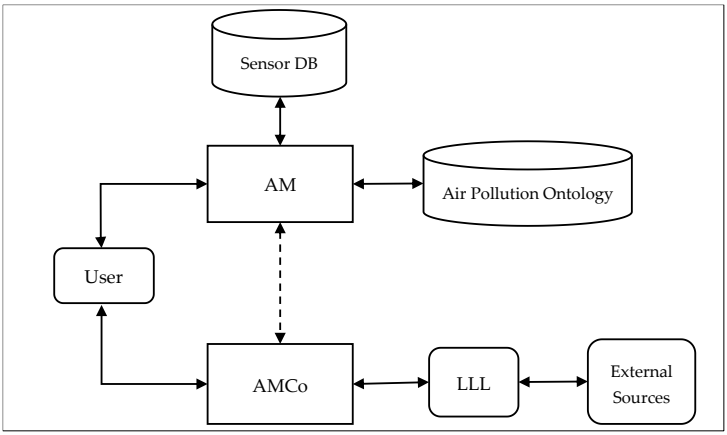

Figure 10. Architecture of the ACreM platform

Air Monitor (AM) is a personal assistant designed to identify and localize various air quality anomalies based on our measurements. AM was developed as a BDI (Beliefs-Desires-Intention) agent in the JaCaMo development environment. The BDI model has its roots in the philosophical tradition of understanding practical reasoning in humans. Practical reasoning is action-oriented – the process of searching for what to do. Practical deliberation is a matter of weighing conflicting considerations for and against competing options, depending on the agent's desires, concerns, or value judgments. As a process, practical reasoning involves two distinct activities. The first, known as deliberation, is to decide what state of affairs (goal) we want to achieve. The second, called planning, is to decide how to achieve that state of affairs (goal). One of the advantages of this architecture is the ability to flexibly represent and work with the agent's environment, which is essential for our purposes. Another advantage is that AM works with proven expertise in our area of interest (in this case air pollution). This circumstance is at the same time a disadvantage of the ReAct agent, which works with free text, created by non-professionals such as journalists, politicians and interested citizens. Therefore, we believe that combining the two approaches is an adequate way to achieve the goal of the study, namely the search for reliable information about the state of the air in the city of Plovdiv. Fig. 11

shows the general architecture of AM, located in its working environment. The diagram shows the main components of AM and its environment.

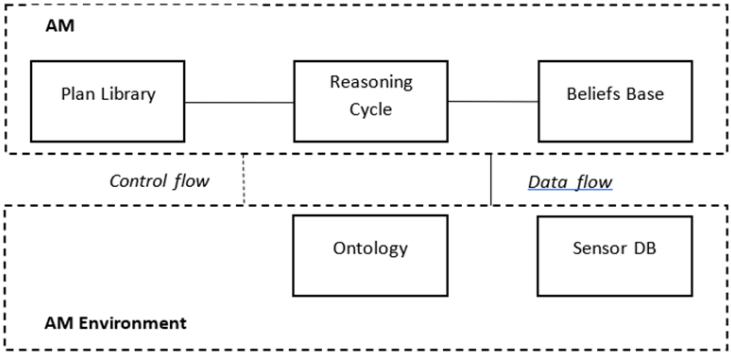

Figure 11. AM architecture

The belief base is a discrete structure that models the agent's perception of its environment. Individual beliefs are represented as predicates (in the style of the logic programming language Prolog). The structure consists of two basic components, static and dynamic. Any change in the belief base (deletion or addition of a belief) triggers an actual event. This component is updated at a time interval specified by us. An example segment of the belief base is given in Fig. 12.

```
/* Beliefs Base -  represents the agent's initial knowledge: the sensor values from the Sensor DB and from Air Pollution
Ontology. CO2 level in parts per million (ppm), as well as PM2.5 level in micrograms per cubic meter (µg/m3) */
        sensor_value(co2, current_value).
        sensor_value(pm25, current_value).
        co2_threshold(1000).
        pm25_threshold(15).
/* Initial goal causes the initial triggering_event to be generated, in this case to initialize and start the air quality
monitoring process */
        !start_air_monitoring.
```

Figure 12. Example segment of the Belief Base

Two main repositories are located in the AM environment. The basic knowledge for understanding and monitoring air quality is stored in an ontology called Air Pollution Ontology, developed in accordance with the global air quality guidelines of the World Health Organization, the Bulgarian

Environmental Executive Agency and the Environmental Monitoring System of the Municipality of Plovdiv offer comprehensive information on air quality in Bulgaria, including measurements of key pollutants such as PM10, PM2.5, SO_2, O_3, CO and NO_2, available on their official website. The specified data serve as a basis for drawing conclusions about air quality by comparing it with dynamically entered data in the databases. The second repository is a relational database that stores our measurements. The databases contain the dynamic data obtained from different IoT nodes or from real-time data entered by the platform users. The data flow starts with its beginning – measurements performed by devices in the sensor group (Fig. 13).

The recorded values are then read and consolidated into a single data packet by a dedicated controller. This controller acts as an intermediary, adapting between the various sensor devices, each with its own specific interface, and the wider system. At regular intervals (typically every 1 to 5 minutes), the controller transmits the data packet to the data endpoint of the software system. Upon arrival, the data packet is directed to a data transformation module. Its main function is to split the data packet into a series of individual measurements, each associated with its respective value and metric. The implementation of the data transformation module uses Node Red streams. For each measurement, the data transformation module initiates a discrete REST request to the OGC SensorThings API, which in turn stores the measurement in the data store. The Fraunhofer IOSB implementation serves as the backbone for the SensorThings API. PostgreSQL, enhanced with the TimescaleDB extension, serves as the designated data store. Once the data is securely stored in the data store, it can be retrieved using various SensorThings API reading methods. These methods cover a wide range of filters, allowing for detailed customization of the retrieved data. Applications, such as Grafana, can use the SensorThings API reading methods to monitor, visualize, or export data. Grafana, for example, has a convenient pre-configured data source for seamless integration with the SensorThings API.

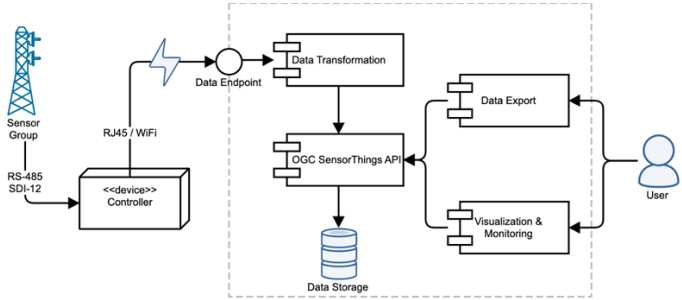

Figure 13. Architecture Components

The interaction of AM with its environment is based on the concept of A&A (Artifacts and Agents). Air Monitor Copilot (AMCo) is a copilot agent with ReAct architecture, which we use to explore and evaluate air pollution data in the Burgas region from external sources. The ReAct framework uses a combination of task decomposition, reasoning loops, and multiple problem-solving tools. ReAct agents from the LangChain library can support a complete request processing process. The introduction of the platform aims to contribute to the objective assessment of air quality and, therefore, to effective decision-making. In the current version, AMCo presents the results in an unstructured format (text), which is easy to read by a human user. In future interactions with AM, the data exchange should also be in a structured or semi-structured (e.g. JSON) format.

CONCLUSION

The results of the research presented in this dissertation can be briefly summarized as follows:

- A concept, model, and architecture of a basic tourist guide have been developed.
- A working prototype of a tourist guide has been created.
- The possibilities for adapting the prototype for real applications in the main application area (tourism) and in a different application area (environmental protection) have been demonstrated.

In addition, the following conclusions can be drawn from the results of the study:

- Regarding the features of the virtual tourist guide - the main approach in creating virtual tourist guides is to create a system that will facilitate the mechanism of traditional information search and ticket booking/purchase.
- Regarding the model of the personal tour guide and basic functionalities - the virtual tourist guide is a developed mobile application, which is available for mobile devices with the Android operating system and a website (https://exploring-gabrovo.com/). The system consists of several parts: a mobile application, a public website, and a content management panel.
- Regarding the guidelines for future development of the virtual tourist guide - the created tourist guide will support the work of the tourist information center in Gabrovo, therefore it is necessary to be directed at the individual tourist and provide him with the opportunity to compile an individual optimal route, according to personal preferences. The virtual tour guide must adapt to changing environmental conditions, such as changing weather conditions, the tourist's deviation from the set route, traffic jams on the road, etc.

LIST OF PUBLICATIONS CITED IN THE DISSERTATION

of Ilia Iliev Nedelchev
PHD STUDENT AT THE DEPARTMENT OF COMPUTER SYSTEMS

1. Stoyanov, S.; Doychev, E.; Stoyanova-Doycheva, A.; Tabakova-Komsalova, V.; Stoyanov, I.; Nedelchev, I. A Regional Multi-Agent Air Monitoring Platform. Future Internet 2025, 17, 112. https://doi.org/10.3390/fi17030112
2. I. Nedelchev, V. Tabakova-Komsalova, I. Stoyanov, S. Stoyanov, V. Ivanova and T. Kazashka, "Supporting Digitization of a Cultural and Historical Heritage Platform," 2024 International Conference Automatics and Informatics (ICAI), Varna, Bulgaria, 2024, pp. 490-493, doi: 10.1109/ICAI63388.2024.10851643.

REFERENCED LITERATURE

[1] Божинова, М. Личев, Т. Тодорова, Л. Павлов, П. Георгиева, К. Съвременни измерения на управлението на туристическа дестинация България. Алманах научни изследвания, том 27, 2019, с. 7-22

[2] Kanwel, S. Lingqiang, Z. Asif, M. Hwang, J. Hussain, A. Jameel, A. The Influence of Destination Image on Tourist Loyalty and Intention to Visit: Testing a Multiple Mediation Approach. Sustainability 2019, 11, 6401. https://doi.org/10.3390/su11226401

[3] Szostak, D. Spatial and Infrastructural Conditions of Functioning of the Contemporary Tourism Market – Analysis of Selected Issues. European Research Studies Journal. Volume XXVI, Issue 2, 2023, pp. 337-347

[4] Pai, C. Y. Liu, S. Kang, A. Dai. The role of perceived smart tourism technology experience for tourist satisfaction, happiness and revisit intention. Sustainability, 12 (16) (2020), p. 6592, 10.3390/su12166592

[5] Радулов, Н. Сигурност 4.0. София, Научнотехнически съюз по машиностроене, 2019, с. 72

[6] Антонова, К. Иванова, П. Роботизацията и изкуственият интелект – нов модел на сътрудничество и/или взаимодействие между машините и хората на работното място. Човешки ресурси & Технологии. Бр. 1, 2021, с. 65

[7] Alberto M. Flutter Complete Reference: Create beautiful, fast and native apps for any device. Independently published, 2020. p. 5-10

[8] Dino E. Programming ASP.NET Core, First edition, fast and native apps for any device. Microsoft Press, 2018. p. 15-16

[9] Matthias B. RESTful API Design: 3 (API-University Series). CreateSpace Independent Publishing Platform, 2016. p. 50-53

[10] Nigel P. Docker Deep Dive: Zero to Docker in a single book. Independently published, 2020. p. 20-21

[11] Hans-Jurgen S. Mastering PostgreSQL 13: Build, administer, and maintain database applications efficiently with PostgreSQL 13, 4th Edition. Packt Publishing, 4th ed. Edition, 2020. p. 10-13

[12] I. Nedelchev, V. Tabakova-Komsalova, I. Stoyanov, S. Stoyanov, V. Ivanova and T. Kazashka, "Supporting Digitization of a Cultural and Historical Heritage Platform," 2024 International Conference Automatics and Informatics (ICAI), Varna, Bulgaria, 2024, pp. 490-493, doi: 10.1109/ICAI63388.2024.10851643.

[13] Rafalimanana, H. J. L. Razafindramintsa1, S. Cherrier, T. Mahatody, L. Geogrge, V. Manantsoa, Jason-RS, a Collaboration between Agents and an IoT Platform, International Workshop

[14] Soliman, M. Guetl, V. Evaluation of intelligent agent frameworks for human learning. 14th International Conference on Interactive Collaborative Learning, Piestany, Slovakia, 2011, pp. 191-194, doi: 10.1109/ICL.2011.6059574.
[15] Stoyanov, S.; Doychev, E.; Stoyanova-Doycheva, A.; Tabakova-Komsalova, V.; Stoyanov, I.; Nedelchev, I. A Regional Multi-Agent Air Monitoring Platform. Future Internet 2025, 17, 112. https://doi.org/10.3390/fi17030112