NICOLAUS COPERNICUS UNIVERSITY IN TORUŃ

Pedagogy and Psychology of Sport. eISSN 2450-6605.
Journal Home Page
https://apcz.umk.pl/PPS/index

DOBROVOLSKYI, Yurii and TANASHCHYSHENA, Inessa. Information system for dynamic monitoring of physical factors of the natural environment. Pedagogy and Psychology of Sport. 2026;30:71298. eISSN 2450-6605. https://doi.org/10.12775/PPS.2026.30.71298

The authors declare that there is no conflict of interests regarding the publication of this paper.
Received: 15.01.2026. Revised: 15.04.2026. Accepted: 15.04.2026. Published: 27.04.2026.

INFORMATION SYSTEM FOR DYNAMIC MONITORING OF PHYSICAL FACTORS OF THE NATURAL ENVIRONMENT

Yurii Dobrovolskyi, Inessa Tanashchyshena

Yuriy Fedkovych Chernivtsi National University, Kotsyubynsky 2, Chernivtsi, 58012, Ukraine

y.dobrovolsky@chnu.edu.ua (Yurii Dobrovolskyi),
i.tanashchyshena.y@chnu.edu.ua (Inessa Tanashchyshena)

Abstract

Physical environmental factors, namely ecological, astronomical, tectonic, significantly affect it and create a burden on both nature in general and human communities in particular. Therefore, monitoring such impacts is critically important for their prediction in order to preserve human health and economic infrastructure. Due to this, the need for integrated information systems capable of appropriate monitoring and forecasting of critical events that affect the environment is growing. Combining physical, astronomical and tectonic parameters in a single information platform makes it possible to create a comprehensive approach to data analysis and visualization. The work is devoted to the development of an information system for dynamic monitoring of physical environmental factors, as well as factors of astronomical and tectonic origin. The main innovation of the work is the combination of these factors for display on web resources and the use of the haversine formula for analyzing geographic data, which increases the accuracy of calculations. The system integrates data from APIs (OpenUV, OpenWeatherMap, NASA, IRIS-WS, AirVisual), providing convenient access to information on ultraviolet radiation, temperature, humidity, atmospheric pressure, air quality, magnetic storms and earthquake amplitude. The developed web application is based on a modern technology stack (Java, Spring, Angular, PrimeNG), which ensures high performance and ease of use. The work analyzed existing information systems, justified the choice of technologies, implemented the system architecture, and tested its performance and reliability in a test environment. The results obtained indicate the prospects of combining physical, astronomical, and tectonic factors to create integrated information systems. This opens up new opportunities for improving environmental monitoring and preventing potential threats.

Keywords: information system, monitoring, environment, API, haversines, data integration, software.

Introduction

The modern world faces numerous environmental challenges that require constant monitoring and analysis of physical, astronomical, and tectonic factors. Such data are of critical importance for predicting natural phenomena, assessing environmental impacts on human health, and preventing emergencies. Existing monitoring information systems mostly focus only on individual aspects, such as ecological parameters or seismic activity, and do not provide an integrated approach to the analysis of these factors. The goal of this work is to develop an information system for the dynamic monitoring of physical, astronomical, and tectonic environmental factors by integrating data from various API sources, as well as implementing visualization and calculation algorithms to increase data analysis accuracy.

Research Methods

Systems analysis to evaluate existing information systems and justify the novelty of the development; geographic calculation methods, including the use of the haversine formula to increase the accuracy of geographical parameter analysis; instrumental development methods, such as the use of Java, Spring, Angular, and PrimeNG technologies to create a productive and user-friendly information system; data integration methods for combining information from various APIs (OpenUV, OpenWeatherMap, NASA, IRIS-WS, AirVisual); testing methods to verify the functionality and reliability of the system.

Review of Analogues

In the modern world, the development of information systems for environmental monitoring has become widespread. Monitoring systems based on internet technologies and artificial intelligence allow for obtaining data from numerous sensors, processing it, and providing accurate forecasts and information visualization. Below, we consider four key systems that play a significant role in the field of environmental monitoring: NOAA Weather Radar [1], Smart City Platforms [2], and URAD Monitor [3].

NOAA Weather Radar is a recognized global-level system that provides accurate and timely data on atmospheric conditions. However, the system's primary focus is solely on atmospheric data, without considering astronomical and tectonic factors. Furthermore, maintaining this infrastructure is high-cost due to the use of satellites and complex equipment. Effective access to data requires a stable high-speed internet connection, which can be an issue in some regions.

AirVisual is a system oriented towards monitoring air quality on a global scale. Its advantages include ease of use, data accessibility from many points worldwide, and the ability to forecast pollution levels. However, AirVisual focuses only on air quality and does not account for other environmental or tectonic indicators. Additionally, data accuracy depends on local monitoring stations, which may vary in reliability.

URAD Monitor specializes in the local monitoring of radiation levels, air quality, and other environmental parameters. It allows for real-time data acquisition, and its modularity enables solutions to be adapted to specific needs. The disadvantages of URAD Monitor lie in the narrow range of parameters, as the system does not support the monitoring of weather or tectonic factors. Furthermore, some of its modules require special skills for setup and use. In many cases, data collection depends on private users, which can affect its reliability and

completeness.

The provided analysis shows that the development of a comprehensive information system combining data of different origins is capable of filling the gap in existing solutions, providing an effective tool for environmental monitoring and risk forecasting.

Research Results

Key Parameters for Assessing Location Livability

The following parameters were selected as the most important for assessing the suitability of a location for living at a specific time [4-6].

Ultraviolet Index (UV Index) – an indicator for determining the level of solar ultraviolet (UV) radiation reaching the Earth's surface. Air temperature – directly affects human activity, specifically comfort, health, and safety. Air humidity – directly affects human comfort, health status, and even environmental conditions. Air quality – determines air pollution and the content of harmful substances within it. Atmospheric pressure – affects numerous natural and physiological processes; it is a primary parameter used by meteorologists to predict the weather. Seismic activity – capable of causing serious consequences for human life, infrastructure, and the environment. Magnetic storms – can have varying degrees of intensity, from weak to very powerful, which can be dangerous for modern technological society.

Justification for API Selection for Data Integration

In modern information systems, choosing appropriate APIs for data integration is one of the key stages of development. APIs allow for connecting to external information sources, providing access to large arrays of real-time data. Criteria for API selection include data accuracy and reliability, availability, level of detail, and ease of integration. Consequently, after the analysis, the best APIs were selected for the implementation of the web application. Each selected API provides access to a specific parameter important for environmental assessment. Let's look at each in detail:

- OpenWeatherMap API [7] is one of the most popular and reliable tools for obtaining meteorological data.
- OpenUV API [8] specializes in providing data on ultraviolet radiation levels (UV index), which is a vital factor for assessing risks associated with prolonged outdoor exposure.
- AirVisual API provides air quality data, including particulate matter content (PM2.5, PM10), ozone levels, nitrogen dioxide, carbon monoxide, and other pollutants.
- NASA API offers a wide range of scientific data, particularly information on magnetic storms and solar activity.
- IRIS-WS Library is a library that provides access to seismic activity data from a global network of seismographic stations.

The choice of each API is dictated by the need for accurate data concerning individual physical environmental factors. Integrating them into a single system will allow for comprehensive environmental parameter analysis necessary for assessing regional viability and making informed decisions.

For project development, the Java programming language [9] was chosen, which is one of the most popular languages providing stability, scalability, and security for information system development. For projects requiring complex integration with various APIs, work with geographic data, and implementation of mathematical models, Java demonstrates significant advantages. Specifically, its rich ecosystem of libraries and tools allows for the efficient resolution of developer tasks. In the context of my project – creating a dynamic information system for monitoring physical, astronomical, and tectonic factors – Java stands as the

optimal choice.

Creating modern web applications that effectively combine performance, intuitiveness, and flexibility is an important aspect of any monitoring system. Angular, as one of the leading frameworks for frontend development, is an ideal tool for implementing such requirements. This framework was developed by Google and is widely used for creating dynamic web interfaces capable of handling large volumes of real-time data.

To create modern, dynamic, and user-friendly web applications, the use of high-quality tools for interface building is critically important. One such tool is the PrimeNG component library [10], which is based on Angular and provides a wide set of ready-to-use UI elements. Its use significantly reduces development time, optimizes user interaction, and increases productivity.

The choice of technologies for this project, from APIs for data integration to frameworks for interface creation, is justified both from a technical standpoint and with consideration for user convenience and efficiency. All selected tools allow for the creation of a powerful, fast, and interactive system for monitoring environmental factors necessary for ensuring comfortable and safe living conditions.

Investigation of Possible Use Cases

Assessing possible use cases is a key process aimed at defining how users interact with the system. Within this analysis, functional and non-functional requirements are considered, potential risks are assessed, different implementation approaches are compared according to defined criteria, and test scenarios are developed.

Functional Requirements

- Location suitability determination: The system must provide functionality to calculate indicators determining the suitability of a place for living, including temperature, humidity, air quality, UV index, seismic activity, magnetic storms, and atmospheric pressure.
- Connection to API sources: Ensuring communication with several APIs to obtain up-to-date data, including OpenWeatherMap, OpenUV, NASA API, IRIS-WS, and AirVisual. The system must be able to automatically check source availability, handle errors, and provide verified links to data sources.
- Data analysis and result presentation: The interface should display calculation results in the form of interactive graphs or tables. Information about each factor should be accompanied by a detailed explanation of where the data was obtained and its relevance.

Non-Functional Requirements

- Performance and scalability: The system must support fast access to large volumes of data, even under high load. This is important when processing requests to several APIs simultaneously.
- User-friendly interface: The interface must be intuitive, allowing users with different levels of technical knowledge to navigate the application easily. PrimeNG is used to achieve high-quality user experience.
- Minimization of resource consumption: Unlike AI-based systems, the application should use minimum computing resources, making it accessible to a wider range of users.
- Accessibility and responsiveness: The application must be accessible via a web browser with support for responsive design, allowing it to work on various devices, including mobile phones, tablets, and computers.
- Openness to integration: The possibility for further functional expansion by adding

new APIs or other data sources to increase analysis accuracy.

Development of Information System Architecture and Structure

The system consists of the following modules:

- Data Collection Module – responsible for interacting with APIs and obtaining data from external sources. This module handles raw data acquisition, converting API responses into convenient objects and passing them to subsequent analysis stages.
- Data Processing Module – performs analysis and calculations based on the collected information.
- Configuration and Management Module – organizes application architecture, configures connections to external APIs, and ensures the operation of the entire system.
- User Interaction Module – provides interface operation and information transfer between the user and the system.

The hardware component for ensuring application operation is a computer. It is the primary device on which the program is launched and functions, ensuring performance and stability. This modular division ensures logical and technical isolation of different system parts, facilitating ease of scaling, testing, and further project improvement. All modules interact through clearly defined interfaces, increasing application stability and reliability.

It is also worth considering the haversine formula method and its use in the code:

```
private static Event findClosestEvent(List<Event> events, double targetLatitude, double targetLongitude) {
    return events.stream()
        .min((event1, event2) -> {
            double distance1 = calculateHaversineDistance(
                targetLatitude, targetLongitude,
                event1.getPreferredOrigin().getLatitude(),
                event1.getPreferredOrigin().getLongitude()
            );
            double distance2 = calculateHaversineDistance(
                targetLatitude, targetLongitude,
                event2.getPreferredOrigin().getLatitude(),
                event2.getPreferredOrigin().getLongitude()
            );
            return Double.compare(distance1, distance2);
        })
        .orElse(null);
}

private static double calculateHaversineDistance(double firstLatitude, double firstLongitude, double secondLatitude, double secondLongitude) {
    double deltaLat = Math.toRadians(secondLatitude - firstLatitude);
    double deltaLon = Math.toRadians(secondLongitude - firstLongitude);
    double haversineValue = Math.sin(deltaLat / 2) * Math.sin(deltaLat / 2) +
        Math.cos(Math.toRadians(firstLatitude)) * Math.cos(Math.toRadians(secondLatitude)) *
            Math.sin(deltaLon / 2) * Math.sin(deltaLon / 2);
    double centralAngle = 2 * Math.atan2(Math.sqrt(haversineValue), Math.sqrt(1 - haversineValue));
    return EARTH_RADIUS * centralAngle;
}
```

The code provides scalability as it works with arbitrary sets of events and allows for the quick determination of the nearest event. By using the haversine formula, the distance accounts for the Earth's curvature, making results accurate for medium and long distances. The method uses stream processing and minimizes the number of operations, performing only those necessary to identify the nearest element.

Code Snippets of the Interface Development

The system interface was implemented utilizing the Spring Framework [11]:

```
<div class="button-container">

<button pButton label="Перейти на сторінку форми" icon="pi pi-arrow-right" [routerLink]="['/form']" class="styled-button"></button>

</div>
```

This control element facilitates a seamless transition to the form page following the review of preliminary parameters or datasets. It constitutes a critical component of the interactive interface, particularly in scenarios where the form represents a core functional module, such as data entry or system configuration.

```
div class="form-group">

<label for="lat">Широта</label>

<input id="lat" type="text" pInputText placeholder="Введіть широту" formControlName="lat" class="styled-input" />

<small class="p-error" *ngIf="indicatorForm.get('lat')?.invalid && indicatorForm.get('lat')?.dirty">

Широта необхідне поле

</small>

</div>
```

The presented code fragment implements a specific form component designed for latitude coordinate input. The input field is integrated with a descriptive label to define the expected data format and utilizes the PrimeNG library's pInputText directive to enhance UI consistency and interactivity. Within the Angular framework, the input value is bound to the lat key via the formControlName attribute, enabling reactive synchronization and automated state tracking.

A robust validation mechanism is implemented to ensure data integrity: conditional error messages are rendered if the field remains null or contains invalid data upon user interaction (touched state). This approach enhances the system's responsiveness and facilitates real-time error correction, maintaining a high standard of usability without requiring additional navigational overhead.

```
<div class="indicator-wrapper">
    <div *ngFor="let indicator of results.indicators" class="indicator" [ngClass]="{'danger': indicator.status.includes('Небезпеч'), 'safe': indicator.status.includes('Хорошиий')}">
      <p-card [header]="indicator.name">
       <div class="card-emoji">{{ getEmoji(indicator.name) }}</div>
       <p>{{ indicator.description }}</p>
       <p>{{ indicator.recommendation }}</p>
      </p-card>
      </div>
    </div>
```

The following implementation facilitates the dynamic rendering of result indicators based on the results.indicators data model. Each indicator is encapsulated within a PrimeNG card component (p-card), comprising a header, a functionally mapped icon (retrieved via the getEmoji method), a detailed description, and a corresponding recommendation. The system applies conditional CSS class binding (e.g., danger or safe) to the card components based on the indicator's state, enabling immediate visual differentiation between critical and nominal operational conditions.

The significance of this implementation lies in the development of a high-usability interface that minimizes cognitive load. By streamlining the presentation of indicators and recommendations, the system ensures rapid situational awareness. The UI architecture prioritizes information hierarchy, utilizing visual cues to emphasize critical parameters and facilitate efficient data interpretation by the end-user.

System Operational Algorithm

Upon initializing the web application, the system executes the primary routing sequence to render the main interface.

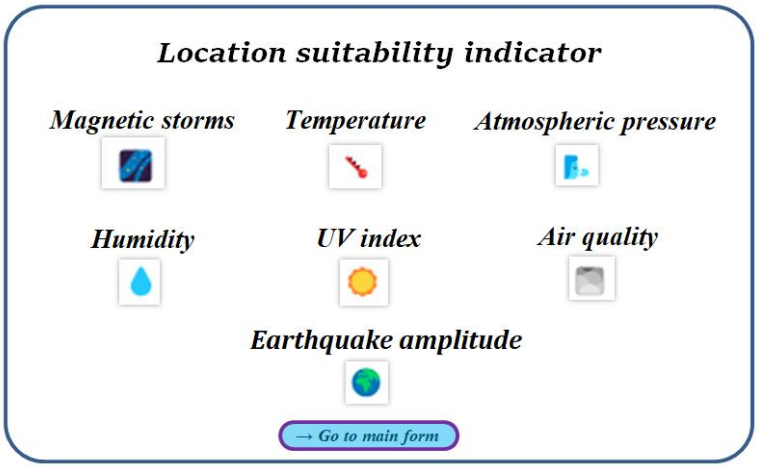

Fig. 1. Main menu interface

The main menu aggregates all essential parameters utilized for the site suitability assessment. This comprehensive dataset encompasses seven key indicators: geomagnetic activity (magnetic storms), ambient temperature, atmospheric pressure, humidity, UV index, air quality, and seismic amplitude.

Upon reviewing these environmental metrics, the user may proceed to the specialized interface for further evaluation, specifically the 'Location Suitability Form' (Fig. 2), which facilitates an assessment of the site's habitability and compatibility with human settlement requirements.

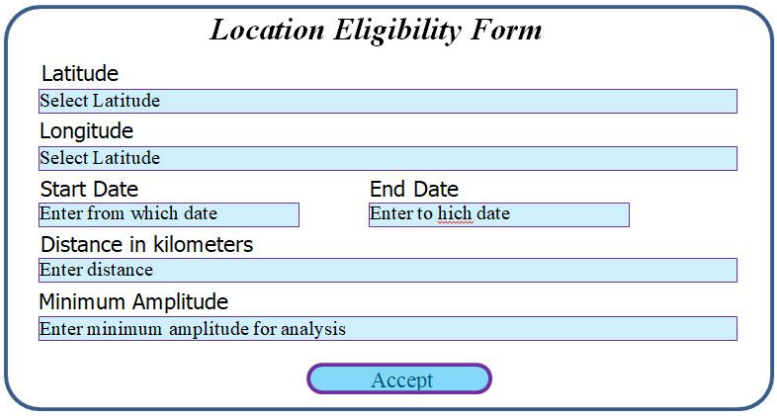

Fig. 2. Location suitability assessment form.

Subsequently, the user is required to input the necessary data to facilitate the location suitability computation. Upon successful data submission, the information undergoes an analytical processing stage, and the resulting output is generated as illustrated in Fig. 3:

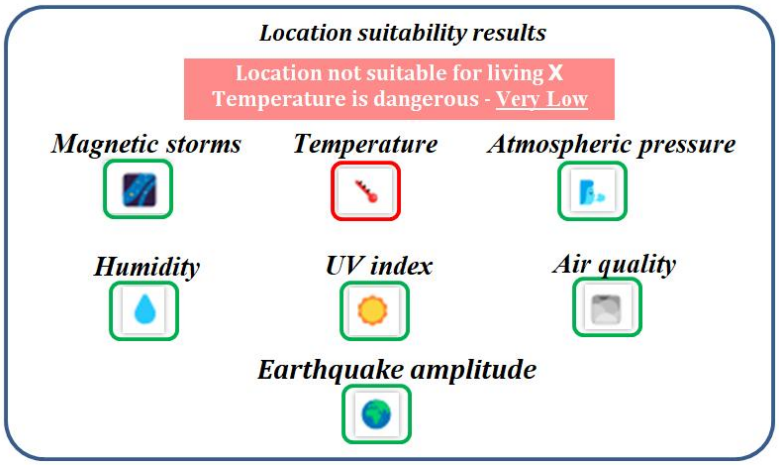

Fig. 3. Analysis Results

Testing

Performance testing of the web application was conducted; specifically, to ensure accuracy and reliability, comparisons were made between the created application and other data sources providing these indicators. The goal of this testing was to show the effectiveness of simultaneous data processing and acquisition using the created information system versus manual data collection. The test results are shown in Table 1.

Table 1. Data processing speed testing in seconds.

Developed Information System (s)	Manual Testing (s)
4.3	12.5
5.1	11.2
3.8	12.0
4.2	11.5
5.5	12.8
6.1	13.2
4.4	12.7
4.6	11.6
5.2	11.9
4.3	13.1

As seen from the test results, the created system is significantly faster compared to manual data collection. In just a few seconds, the system can retrieve and process information for all 7 indicators, whereas in manual mode, a user requires significantly more time to navigate between different sites and enter each parameter manually. This comparison clearly shows that automating the process of data processing from APIs significantly reduces time costs, which is especially important for real users who need timely information.

Also, testing of the created information system and the popular service Gemmo.AI [12]

for the use of computer resources was carried out, the results of which are given in Table 2.

Table 2. Testing of the use of computer resources

System	CPU Usage	Memory Usage	API Call Count	Response Time	Purpose	Efficiency
Developed Info System	10-15%	50-150 MB	7 (API)	4.5 sec	API-based monitoring of 7 environmental parameters	High operational throughput via automation and resource optimization through API integration
Gemmo.AI	20-30%	200-500 MB	5 (Sensors + API)	12.5 sec	Flood forecasting, water quality degradation monitoring, and environmental management	System performance is contingent upon sensor data availability; real-time processing requirements necessitate higher computational resource allocation

Conclusions

A dynamic information system has been created that allows users to quickly and conveniently obtain comprehensive information about environmental factors with high accuracy and minimal resource expenditure. The result is achieved through the integration of several types of data: physical, astronomical, and seismic, within a single interface. Testing demonstrated high efficiency. On average, the time to collect data for all seven parameters in the system was 4.5 seconds. In comparison, manual data collection through open sources requires approximately 15 seconds. This means the system provides roughly a threefold time saving for the user, significantly simplifying the information access process. The application also stands out for its low level of resource consumption, using 10-15% CPU and 50-150 MB of RAM, which are optimal for a web application of this type. Thus, the project fully meets the set goals and tasks, offering perspectives for further development.

References

[1] National Oceanic and Atmospheric Administration. URL: https://www.nhc.noaa.gov/ (Accessed on April 23, 2026).

[2] The Power of Smart City Platforms URL: https://www.smartcityss.com/resources/the-power-of-smart-city-platforms-integrating-data-and-services], AirVisual [AirVisual. URL: https://www.saveecobot.com/platform/airvisual (Accessed on April 23, 2026).

[3] URad Monitor. Real-time Air Quality Sensor Network. URL: https://aqicn.org/network/urad/ (Accessed on April 23, 2026).
[4] Environmental Health: From Global to Local, 3rd Edition. 2016. URL: https://www.wiley.com/en-fr/Environmental+Health%3A+From+Global+to+Local%2C+3rd+Edition-p-9781118984765#download-product-flyer (Accessed on April 23, 2026).
[5] Environmental Hazards: Assessing Risk and Reducing Disaster. 2023. URL: https://www.yakaboo.ua/environmental-hazards-assessing-risk-and-reducing-disaster.html?srsltid=AfmBOoq-k18J1LUxF466VMOi5fFw5lXTDh1QCEZN4f7Mn77NtgQ0pxOK (Accessed on April 23, 2026).
[6] Raymond Poon. Harmful Natural Chemicals and Radiation in the Environment. URL: https://www.worldscientific.com/worldscibooks/10.1142/8535?srsltid=AfmBOookg3GRBBBQMmA8RySLcg4wYzDv1MdWEV4zX3uMdNiJMSZRM5yt#t=aboutBook (Accessed on April 23, 2026).
[7] OpenWeatherMap API Documentation. URL: https://openweathermap.org/api (Accessed on April 23, 2026)..
[8] OpenUV API Documentation. URL: https://www.openuv.io/ (Accessed on April 23, 2026).
[9] Java Platform Documentation (Oracle). URL: https://docs.oracle.com/javase/ (Accessed on April 23, 2026).
[10] PrimeNG Documentation. URL: https://primefaces.org/primeng/ (Accessed on April 23, 2026).
[11] Spring Framework Documentation. URL: https://spring.io/projects/spring-framework (Accessed on April 23, 2026).
[12] Gemmo.AI Environmental Monitoring Platform. URL: https://gemmo.ai/ (Accessed on April 23, 2026).

About the authors

Yuriy Dobrovolsky. Graduated from the Faculty of Physics and Mathematics in 1984. Received the degree of Doctor of Technical Sciences. Currently, he is a professor at the department of software engineering at Yuri Fedkovich Chernivtsi National University. Scientific interests include research and development of mathematical models and algorithms for creating reliable software, cryptographic algorithms, as well as reliable measuring, medical, and environmental technology.

ORCID ID:0000-0003-0626-0594.

Inessa Tanashchyshena. Graduated from Yuri Fedkovich Chernivtsi National University in 2011. Major: Informatics, Qualification: University Lecturer, Researcher (Computer Systems). Assistant at the department of software engineering at Yuri Fedkovich Chernivtsi National University. Area of scientific interests - cryptographic algorithms and computer modeling.

ORCID ID:0009-0008-5119-8420.