Secure Systems Design and Technology Development. Veliko Tarnovo, Bulgaria
Proceedings of the International Scientific Conference, 2026. Volume III, 209-214

Intelligent Information System for Technical Supervision of High-Risk Facilities

Nikolay Nikolov
Faculty of Mechanical Engineering
Technical University of Sofia
Sofia, Bulgaria
nikolay.d.nikolov@tu-sofia.bg

Georgi Stanchev
Faculty of Mechanical Engineering
Technical University of Sofia
Sofia, Bulgaria
gstanchev@tu-sofia.bg

Boryana Ilieva - Mihaylova
Faculty of Mechanical Engineering
Technical University of Sofia
Sofia, Bulgaria
bilieva@tu-sofia.bg

***Abstract*-High-risk facilities (HRF) are subject to strict regulatory oversight, encompassing mandatory registration, periodic technical inspections, and systematic maintenance of technical documentation. In practice, the management of these processes is hindered by a range of organizational shortcomings: facility data are scattered across heterogeneous registers and files, there are no automated mechanisms for tracking inspection deadlines, and the document workflow between technical supervision bodies (TSB) and facility operators relies predominantly on paper-based and traditional communication channels. This paper presents the design and implementation of an intelligent information system (IIS) aimed at digitalizing and optimizing these processes. The applicable regulatory framework - the Law on Technical Requirements for Products and the relevant subordinate legislation and the identified practical challenges are examined. Based on interviews with representatives of TSBs and facility operators, system requirements were defined and a three-tier web architecture was proposed, implemented with React (presentation layer), Ruby on Rails (business logic), and PostgreSQL (data management), supplemented by Redis for session management and background job processing, and Active Storage for file handling. Access control is secured through JSON Web Tokens (JWT) and a role-based access control model (RBAC) with three levels - administrator, TSB and operator. The system centralizes the electronic technical record of each HRF, automatically calculates and monitors periodic inspection deadlines, generates mandatory documents and application forms, and provides a traceable document workflow with version control. Achieved results are presented alongside identified limitations - the absence of a mobile client, lack of integration with external national registries, and no support for qualified electronic signatures and directions for future development of the system are outlined.**

Keywords— high-risk facilities, technical supervision, information system, digitalization, document workflow, JWT, RBAC, Ruby on Rails, React, PostgreSQL.

I. INTRODUCTION

High-risk facilities (HRF) are an integral part of modern industrial and public infrastructure. Boilers, pressure vessels, lifting cranes, gas equipment and installations, and others [3], [6], [7] – every type of them carries the potential for accidents and incidents.

The legal framework requires their registration, their safe and incident-free operation, and their compliance with the technical requirements, standards, and safety rules. In Bulgaria, this framework is set out by the Technical Requirements for Products Act (TRPA) [1] and the ordinances implementing it [2]–[5]. They introduce mandatory registration of the facilities and various technical inspections carried out by Technical Supervision Bodies (TSB) [4]. Each of these activities is accompanied by the drafting and submission of written applications, the preparation of inspection reports and protocols, and other documentation, the greater part of which continues to be kept and required on paper.

The data on each individual HRF is typically scattered across different people, files, and folders, with no one responsible for keeping it up to date as a whole. There is no mechanism to provide reminders of when a specific administratively defined deadline is approaching. Correspondence between TSBs and HRF operators is conducted through traditional channels, while the preparation of documents, reference notes, protocols, and reports is time-consuming and labour-intensive. Missed administrative deadlines are a systemic problem.

The present work proceeds from these findings and puts forward a concrete solution – an information system intended to replace existing practices with a unified digital environment for managing the technical file of an HRF, carrying out its technical supervision, and maintaining it in sound operating condition.

The aim is for the system to create and store an electronic technical file for each HRF, to track various

Online ISSN 3044-7771
https://doi.org/10.68302/std2026.vol3.103

deadlines, to send notifications for approaching or overdue ones, to automatically generate the required applications, and to provide a shared working environment for TSBs, for HRF operators, and for the persons who maintain and repair them. The development was carried out within project No. 231ИП0035-06 and is structured in the following steps: analysis of the regulatory framework and existing practices; definition of the architecture; design of the logical model and the user interface; implementation and testing; evaluation of the results and outlining directions for further development.

II. DESIGN METHODOLOGY

The development follows a structured engineering approach [8], organised into eight sequential stages. The starting point is the pre-design study – an analysis of the current regulatory framework, observation of existing practices, and semi-structured interviews with representatives of TSBs and HRF operators. The aim of this stage is to define the problems and to verify whether the proposed solution is organisationally and technically feasible.

In the analytical phase, a formal logical model of the technical supervision processes has been built [9] – the main entities, their attributes, and the relationships between them have been identified. The model serves both as a specification for the developer and as a basis for validation with end users. The design is iterative: each functional core is followed by a short feedback cycle before moving on to the next. This approach allows for the early detection of discrepancies between regulatory requirements and technical implementation – a problem typical of regulation-bound systems.

The architectural decision was taken at the pre-design stage: a three-tier web architecture with a clear separation between the presentation layer, the business logic, and data management. The eight stages of the development are summarised in Table 1. The architecture of the system is shown in Figure 1.

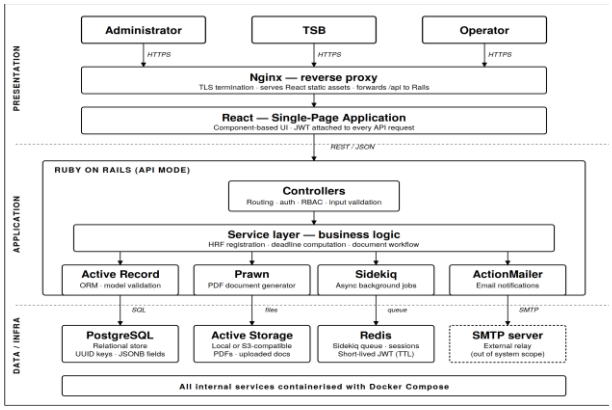

Fig. 1 Architecture of the information system

The client layer includes the three categories of system users: administrator, TSB, and HRF operator, each of whom interacts with the application through a web browser. Communication is encrypted, and every request passes through Nginx, configured as a reverse proxy. Nginx performs a dual role: it serves the static files of the React application and forwards API requests to the server layer, thereby isolating the internal infrastructure from direct external access.

The server layer is implemented in Ruby on Rails in API mode and contains five logical modules. The controllers receive incoming requests, validate the data, and delegate the processing to the service layer. The business logic covers the rules for HRF registration, the calculation of deadlines, and document management. The automated tasks module (Sidekiq) operates asynchronously, periodically checking upcoming deadlines and triggering notifications without blocking the main application. The PDF generator (Prawn) produces the applications and lists required by the regulations. ActionMailer is responsible for sending email notifications to the TSBs that have registered a given HRF.

The data management layer is divided into three components with distinct purposes. PostgreSQL stores all structured data – the HRF registers, the technical file, the users, and the history of performed technical inspections, functional checks, and repairs. Active Storage manages the files associated with the technical files – inspection reports, protocols, instructions, declarations of conformity – and abstracts them from the specific storage medium, which allows a future transition to cloud storage without changes to the business logic. Redis is used for three separate purposes: storage of short-lived JWT tokens for password recovery, session management, and the queue of Sidekiq background jobs. The entire infrastructure is containerised with Docker Compose, which makes the configuration reproducible and simplifies the transition between test and production environments.

№	Stage	Outcome
1	Definition of scope and objectives	Scope description. Technical specification.
2	Study of the information system	Feasibility analysis. Key processes and entities.
3	Analysis of practices. Logical modelling	Formal logical model of the processes.
4	Planning. Conceptual design	General project description.
5	Detailed design	Structure. Functionalities. User interface.
6	Implementation	Development. Formal testing.
7	Deployment	System testing in a real environment.
8	Operation. Evaluation. Maintenance	Assessment of functioning and efficiency. Adjustments.

Table. 1 Stages of system design

III. DESCRIPTION OF THE INFORMATION SYSTEM

The information system is designed as a unified digital environment that brings together all processes related to the technical supervision of HRFs – registration of the facilities, management of documentation, tracking of deadlines for various types of technical inspections, and communication between the participants. The following

subsections describe, in turn, the architectural solutions, the logical data model, the mechanisms for authentication and access control, the management of documentation, and the module for automated notifications and document generation. Each of these components is designed to operate independently while also integrating seamlessly with the others – a principle embedded in the three-tier architecture of the system.

A. Architecture

The three-tier architecture [8] was chosen not as a formal convention, but as a practical response to the specific application context. In an environment where regulatory requirements change frequently, the ability to adapt only the business logic, without rewriting the interface or the database schema, is a real operational need rather than architectural overhead. A microservice decomposition [10], [11] was considered but rejected at this stage, as the operational complexity it would introduce is not justified by the scale of the domain.

The presentation layer is implemented as a Single Page Application with React [13]. The choice of the SPA model stems from the nature of the work with the data. System users deal with long forms, dynamic lists, and document previews, where a full page reload would significantly degrade usability. The React component model allows only the affected parts of the interface to be updated independently. Communication with the backend takes place entirely through a REST API, which makes the frontend technologically independent and, if necessary, replaceable without changes to the server-side logic.

The server layer is built with Ruby on Rails [12], configured in API mode, without the Rails templating mechanism, since the entire view is handled by React. Rails was chosen for its mature ecosystem for working with relational data, its built-in validation mechanisms, and its well-documented conventions for structuring business logic. Rails' Active Record allows the regulatory rules to be expressed directly as code – for example, the mandatory presence of a TSB for every registered HRF is enforced at the model level, not only on user input.

The database is PostgreSQL [14]. The choice over alternatives such as MySQL is determined by two specific requirements: support for UUIDs as primary keys without additional extensions, and JSON columns for storing the technical characteristics of the facilities, whose structure varies depending on the type of HRF. PostgreSQL supports indexing of JSON fields, which enables efficient filtering.

Redis [15] is used for three separate purposes: a queue for background jobs (Sidekiq), storage of password-recovery tokens with TTL, and session caching. Separating these three responsibilities into distinct Redis databases (by number) is deliberate and allows the cache to be flushed independently without the risk of deleting active jobs. Active Storage abstracts file storage: in the test environment, files are written locally, while the transition to cloud object storage requires a change only in the configuration, with no code modifications.

The entire stack is containerised with Docker Compose [16]. In the development environment, each service – Rails, the React dev server, PostgreSQL, Redis – runs in a separate container with defined dependencies. In the production environment, Nginx is added as a reverse proxy in front of the Rails application: it handles SSL termination, serves the static files of the React application directly from disk, and forwards only dynamic requests to the Rails process. This reduces the load on the Ruby process and allows the individual components to be scaled independently.

B. Logical model

The proposed model is built around the HRF as the central entity (HighRiskFacility), linked simultaneously to the TSB, the operator, and its technical documentation. The User can take on different roles depending on the organisational context. The model also includes standalone entities for notifications and reminders, since deadline management is a functional core of the system rather than a secondary add-on. Table 2 summarises the main entities and attributes of the model.

Entity	**Main attributes**	**Description**
User	id, email, role	System user
HighRiskFacility	id, reg_number, type, params	High-risk facility
Document	id, version, revision_date, file	Document from the technical file
Notification	id, type, status	Automated notification
Reminder	id, due_date, severity	Deadline reminder

Table. 2 Main entities in the system

C. Authorisation and access control

For identity authentication, the JSON Web Token (JWT) standard [17] is proposed. Upon successful login, the system is to issue a token with which all subsequent API requests are authenticated. Temporary password-recovery tokens are stored in Redis with a short validity period – an approach that does not require an additional database and is sufficiently secure for the intended use case.

Access rights are to be managed through a role-based access control model (RBAC) [18] with three roles: Administrator, TSB, and Operator. The distinction is not merely formal – TSBs register HRFs and add to their technical file, whereas operators can only access them for reading and attach their own documentation. The system also supports two-factor authentication with backup codes [20]. Table 3 shows the rights by role.

Functionality	**Administrator**	**TSB**	**Operator**
User management	Yes	No	No
Facility register	Yes	Yes	Own only
Adding and editing HRFs	Yes	Yes	No
Uploading documents	Yes	Yes	No

Viewing documents	Yes	Yes	Yes
Notifications and reminders	Yes	Yes	Yes
Generating reports and reference notes	Yes	Yes	Yes

Table. 3 Role model and main access rights

D. HRF and documentation management

The core of the proposed model is the electronic technical file of each HRF. When a new facility is entered, its registration and factory numbers, its kind, type, and technical characteristics and parameters are validated. The system must not allow the entry of data that is incompatible with the selected kind of facility or with the TSB exercising supervision over it. Documents are stored with support for versioning and revision dates; if an overwrite is attempted, a warning is displayed so that information is not lost by accident.

The system must automatically calculate the date of the next periodic inspection, including inspections involving different kinds of testing depending on the type of the facility and its date of manufacture. Inspection reports, protocols from functional checks, and other documents may be associated with each HRF.

E. Automated notifications and document generation

The notifications module is to operate on the basis of periodic background jobs that scan upcoming deadlines and prepare an email containing an application to the corresponding TSB, to be approved and sent by the HRF operator. The logic is deliberately simple, without complex infrastructure, since the events are predictable and calendar-determined.

For PDF document generation, the use of an appropriate library is envisaged – in the proposed implementation, Prawn. The system supports several types of reports, from lists of upcoming technical inspections to registers of lifting accessories, with the option of sorting by key attributes. The supported types are listed in Table 4.

Report type	**Content**	**Format**
Upcoming technical inspections	List and deadline	PDF
Overdue technical inspections	Status and dates	PDF
On-demand / extraordinary technical inspections	Statuses and deadline	PDF
Lifting accessories (LA)	Register (monthly)	PDF
All lifting equipment	Register (quarterly)	PDF
Written application for technical inspection	HRF and TSB data	PDF + email

Table. 4 Generated reports and documents

IV. FUNCTIONAL MODEL OF THE SYSTEM

The proposed functional model defines six main modules, whose interaction ensures full coverage of the processes involved in exercising technical supervision over HRFs. Each module is described through the requirements for its functionality, without being tied to a specific implementation. The aim is to set out a clear specification that can serve as a basis for subsequent development and validation [19]. Table 5 summarises the modules and the requirements for each.

Module	**Functional requirements**
User management	Registration with verification, role profiles (Administrator / TSB / Operator), two-factor authentication, session management, and access recovery
HRF register	Creation, updating, and archiving of HRF records with validated technical characteristics; filtering by kind, TSB, operator, and operational status; automatic determination of deadlines for various types of technical inspections
Technical documentation	Electronic technical file for each HRF; inspection reports with revision date/year; overwrite control; role-based rights for uploading and viewing
Notifications and reminders	Automatic generation of notifications for approaching and overdue deadlines; delivery by email to TSBs; traceability of the status of each notification
Document workflow and reports	Automatic generation of written applications for technical inspections; PDF lists of lifting accessories and lifting equipment; sorting and filtering by key attributes; delivery by email
Administration	Management of users and facilities; system statistics and audit trail; protection against accidental deletion of critical records

Table. 5 Functional model - modules and requirements

A central principle of the model is the separation of responsibilities: TSBs register and maintain the information on HRFs. Operators have access only to their own facilities and may only attach documents such as protocols from functional checks. The administrator manages the system but does not replace the role of the TSB. This distinction is dictated not by technical convenience, but by the logic of the TRPA and the various ordinances. The system must reflect actual powers, not blur them.

The notifications module is a functional core of the system, since it addresses directly the main problem identified – missed administratively defined deadlines. The requirement is that notifications be generated automatically, without any action on the part of the user, and reach the TSB and the operator simultaneously. In this way, the responsibility for tracking deadlines is not imposed on one side alone, but is shared and transparent.

The model is technologically neutral. The requirements described can be implemented using different technology stacks. A subsequent work will present a concrete implementation based on the proposed model, including a description of the chosen technologies, the testing results, and an assessment of the system's fitness for use in a real environment.

V. COMPARISON WITH GENERAL-PURPOSE SCHEDULING SYSTEMS

A reasonable question is whether widely used general-purpose scheduling tools, such as Microsoft Outlook or Google Calendar, could replace the proposed system.

These tools are mature and overlap with part of the functionality described above: they support deadline reminders, recurring events, shared visibility, and basic collaboration. For an organisation that tracks only a small number of inspection dates, a shared calendar may even be sufficient.

The difference lies in scope. General scheduling systems treat every entry as a generic, unstructured event and have no model of the regulated domain. The proposed system, by contrast, is organised around the HRF as a typed entity, together with the obligations attached to it. A calendar entry can remind a user that an inspection is due, but it cannot derive that date from the applicable ordinance, link it to the technical file of the facility, enforce which role may act on it, generate the required statutory application, or retain a traceable record of what was done and when. Table 6 summarises this functional difference.

Capability	**Outlook / Google Calendar**	**Proposed system**
Reminders, recurrence, shared visibility	Yes	Yes
Structured HRF records (typed entities, technical attributes, history)	No	Yes
Automatic regulatory deadline calculation from rule sets	No	Yes
Document versioning and revision control	No	Yes
Role-specific workflows (TSB / operator / administrator)	Limited / manual	Yes
Automated generation of applications and PDF documents	No	Yes
Audit trail and compliance traceability	No	Yes

Table. 6 Comparison with general-purpose scheduling systems

In short, a general scheduler answers only the question of when, whereas regulatory technical supervision additionally requires what, who, under which rule, and with what evidence – none of which such tools provide on their own. This is precisely what motivates a domain-specific compliance-management system rather than the adaptation of a general scheduling tool.

VI. CONCLUSION

The proposed model is built around the provisions of the applicable regulatory acts, not against them. Particularly important is the principle of automated notification. The problem of missed deadlines is systemic and cannot be solved through organisational measures alone without a supporting technological infrastructure. The role-based distinction between TSBs and operators reflects the actual powers under the TRPA and its implementing ordinances, while the technological neutrality of the specification allows implementation on different stacks depending on the organisational context.

Three areas have been deliberately left for the next phase of development: mobile access with offline mode and synchronisation, integration with public registers, and support for a qualified electronic signature (QES). Including them in the main scope would complicate the model to the point where its validation in a real environment would be difficult. The stepwise approach – first a model, then an implementation, then an extension – is justified by the complexity of the subject area. In the longer term, the accumulated history lays a foundation for predictive models that optimise the schedules of the stakeholders.

The present paper proposes a functional model of an information system for the technical supervision of HRFs, built on an analysis of the applicable regulatory framework and of the practical problems established in the sector. The model defines six functional modules, the principles of role-based access differentiation, and the architectural framework within which they are to interact. The main contribution lies not in the technological implementation, but in the systematisation of the requirements – a step which, in systems of this kind, is often omitted or carried out implicitly.

The three functional priorities of the model – a centralised electronic file, automated deadline tracking, and the generation of regulatorily required documents and applications – have been determined on the basis of a study conducted with representatives of TSBs and HRF operators. It is precisely at these three points that existing practices are labour-intensive, unsustainable, and difficult to trace. The model does not claim to solve all problems in the sector, but rather to have begun solving the problems identified.

The architectural framework proposed in the paper is three-tier and technology-independent. Beyond being good practice, the separation between the presentation layer, the business logic, and data management is a precondition for resilience in a strict regulatory environment subject to constant change. A concrete implementation based on the proposed model – including the choice of technologies, a detailed description of the implementation, and results from testing in a real environment – is the subject of a subsequent publication.

The defined limitations of the model – mobile access with offline mode, integration with external registers, and QES support – are not omissions but deliberately deferred requirements. Their inclusion in the main scope would complicate the model to the point where its validation in a real environment would be impeded.

In a broader context, the proposed approach – systematisation of regulatory requirements, definition of a functional model, and subsequent implementation – is applicable in adjacent regulatory areas as well [21]–[23]: fire safety, construction supervision, and licensing regimes – which is also the starting point of the present work. The authors regard this paper as the first in a series aimed at systematically addressing these problems.

REFERENCES

[1] Law on the Technical Requirements for Products (ZTIP), State Gazette of the Republic of Bulgaria, no. 86/1999, last amended SG no. 21/2021. (in Bulgarian)

[2] Law on Health and Safety at Work (ZBUT), State Gazette of the Republic of Bulgaria, no. 124/1997, last amended SG no. 9/2023. (in Bulgarian)

[3] Ordinance on the safe operation and technical supervision of lifting equipment (NBETNPS), State Gazette of the Republic of Bulgaria, no. 78/2004. (in Bulgarian)

[4] Ordinance on the conditions and procedure for issuing licences for technical supervision of high-risk facilities and for maintaining a register of the facilities (NURILOTNSPORVRS), State Gazette of the Republic of Bulgaria, no. 79/2003. (in Bulgarian)

[5] Ordinance No. 1 of 2002 on the conditions and procedure for acquiring and recognising qualifications for practising professions in the operation of lifting cranes and mobile work platforms, State Gazette of the Republic of Bulgaria, no. 51/2002. (in Bulgarian)

[6] Regulation (EU) 2023/1230 of the European Parliament and of the Council of 14 June 2023 on machinery, Official Journal of the European Union, L 165, 29 June 2023.

[7] Directive 2014/68/EU of the European Parliament and of the Council of 15 May 2014 on the harmonisation of the laws of the Member States relating to the making available on the market of pressure equipment, Official Journal of the European Union, L 189, 27 June 2014.

[8] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, 3rd ed. Upper Saddle River, NJ: Addison-Wesley Professional, 2012.

[9] M. Fowler, Patterns of Enterprise Application Architecture. Boston, MA: Addison-Wesley, 2002.

[10] C. Richardson, Microservices Patterns: With Examples in Java. Shelter Island, NY: Manning Publications, 2018.

[11] S. Newman, Building Microservices: Designing Fine-Grained Systems, 2nd ed. Sebastopol, CA: O'Reilly Media, 2021.

[12] D. H. Hansson, "Ruby on Rails: The Guide," Rails Core Team, 2024. [Online]. Available: https://guides.rubyonrails.org. [Accessed: Mar. 2025].

[13] Meta Open Source, "React — A JavaScript library for building user interfaces," 2023. [Online]. Available: https://react.dev. [Accessed: Mar. 2025].

[14] PostgreSQL Global Development Group, "PostgreSQL 16 Documentation," 2023. [Online]. Available: https://www.postgresql.org/docs/16/. [Accessed: Mar. 2025].

[15] Redis Ltd., "Redis Documentation," 2023. [Online]. Available: https://redis.io/docs. [Accessed: Mar. 2025].

[16] Docker Inc., "Docker Compose Documentation," 2023. [Online]. Available: https://docs.docker.com/compose/. [Accessed: Mar. 2025].

[17] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," Internet Engineering Task Force (IETF), RFC 7519, May 2015. doi: https://doi.org/10.17487/RFC7519

[18] D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli, Role-Based Access Control, 2nd ed. Norwood, MA: Artech House, 2007.

[19] Systems and software engineering — Systems and software quality requirements and evaluation (SQuaRE) — System and software quality models, ISO/IEC Standard 25010:2011, 2011.

[20] Information security, cybersecurity and privacy protection — Information security management systems, ISO/IEC Standard 27001:2022, 2022.

[21] M. Janssen and N. Helbig, "Innovating and changing the policy-cycle: Policy-makers be prepared!," Government Information Quarterly, vol. 35, no. 4, pp. S99–S105, 2018. doi: https://doi.org/10.1016/j.giq.2015.11.009

[22] I. Mergel, N. Edelmann, and N. Haug, "Defining digital transformation: Results from expert interviews," Government Information Quarterly, vol. 36, no. 4, p. 101385, 2019. doi: https://doi.org/10.1016/j.giq.2019.06.002

[23] I. Lindgren, C. Ø. Madsen, S. Hofmann, and U. Melin, "Close encounters of the digital kind: A research agenda for the digitalization of public services," Government Information Quarterly, vol. 36, no. 3, pp. 427–436, 2019. doi: https://doi.org/10.1016/j.giq.2019.03.002