Web-based Business Process Management System

Milena Karova[1,*],Iliyan Boychev[2]
Technical University of Varna, Varna, Bulgaria[1]
Technical University of Varna, Varna, Bulgaria[2]
m.karova@tu-varna.bg

Abstract: *The report presents a developed Web-based information system for managing business processes. The system includes system design and development, database design and development, and Web interface design. A special encryption algorithm was used to generate the key. The main purpose is to provide protection, both at the system's entrance and during data transfer and storage. For the web platform, the main protections are 1) Control of access to the system by filling in a username and password, individual for each external user; 2) Ensuring protection of input data by irreversible hashing of the password with a randomly generated key from the system; 3) Using resource protection i.e. without authorization it is not possible to review and change the information in the database 4) Using of special unique identifiers to access the database information; 5) Using a token to send requests to the API.*
Keywords*: web-based information system, business process, password protection, authorization, encryption algorithm, tasks, secure communication*

1. Introduction

Creating a web platform for a business, regardless of industry, can have a huge impact on its success and greatly improve productivity. Allowing third parties to monitor and even participate to some degree in a company's work process would attract more customers. In the development of a web platform, an important role is played by the programming interface. Such a well-defined interface enables a secure and easy exchange of data streams with customers and partner systems. This, in turn, makes it possible to provide different functions and services in different forms - desktop applications, mobile applications, websites, and others.

Nowadays, having a web platform and API plays an important role in business development. They provide the opportunity to integrate modern functionalities and services to attract more customers and partners. An important part of creating an API, in addition to selecting and implementing an architectural model, is good documentation and building strong system defenses against various types of cyber attacks.

There are a lot of software products [1][2][3] that offer services related to personnel management. Research has been done on what some of the better-known software has to offer and based on the research a comparative feature has been made with the current development.

"BambooHR" is a well-designed software solution that helps automate a company's operational tasks. The software collects and provides useful information for employees to streamline work processes and improve efficiency. “Workday” provides a modern and easy-to-use interface similar to BambooHR[4]. The software offers a wide variety of robust features that help companies with financial management, human capital management, service automation, talent management, recruiting, compensation management, and analytical reporting. "IceHRM" is one of the most simplified and effective information systems due to its security and accessibility [5]. Most companies prefer the product because of its centralized support, which makes managing the company and its resources easy. The app has a rich user interface that makes it extensible and customizable.

All three products have unique functionalities such as employee time tracking, automatic payroll process, insurance addition, email system integration, the ability to integrate third-party software products via API, and many more operations. The products have various implemented protections like, hashing, encryption, and tokens on APIs. By integrating APIs, applications easily manage to modernize their content while also managing to implement new up-to-date functionalities. In this way, they remain competitive in the market and manage to develop cross-platform. Despite their modern look, this software is dependent on cloud space and requires a constant internet connection, or their offline version is limited to functionalities.

2. Design of Web System

A) Employee tasks management

The main functionality supporting the management of the work of employees is the creation of tasks. To create a task, at least two users are needed - one who assigned the task and one to whom it was assigned. It is necessary to determine the priority of the task so that the manager can plan the tasks properly and the more urgent ones are done first. Status is also needed, which can be changed during the execution of the task. All this data should be stored in one centralized database. Thus, managers and directors will have access to information based on which they can properly plan the work process. This process is shown in Fig. 1

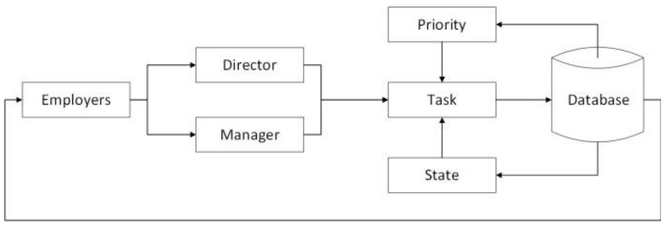

Fig. 1. Create task process

To integrate communication with third parties, having a separate table with authorized users is good. An API compliant with modern architectural requirements is also needed. RESTful APIs stand out as having the most advantages. They, in turn, require additional security measures such as a sufficiently secure authorization token for each request, limiting the number of requests received per unit of time, encrypting communication, creating an external representation of identifiers so that foreign resources cannot be accessed, requiring certificates, and others. A well-secured web platform should have protection on every page, whether the user trying to access the resource is authorized. Accordingly, when logging in, it is good to keep this authorization, whether it is in the form of an API key or the protected version - a token.

Privileges need to have a name and a number, an index that shows the priority. For the states, in addition to the name, it is necessary to note whether each is finite. If it is finite, backups will need to be provided (See Fig. 2), since running tasks carry more useful information than closed ones, and the total number of tasks can become too large and difficult to load for a simple machine. Archived tasks must have their table in the database, analogous to a current tasks table.

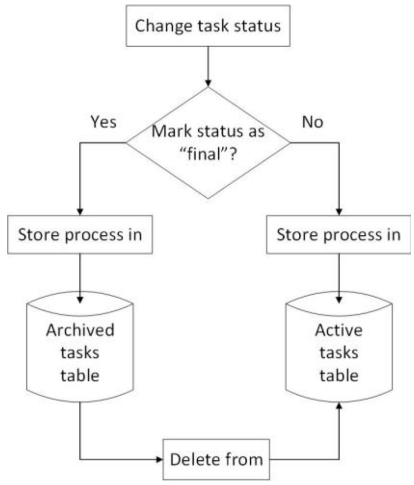

Fig. 2. *Archive complete task steps*

B) Web system

Fig. 3 is shown a UML-case diagram of the development Web system

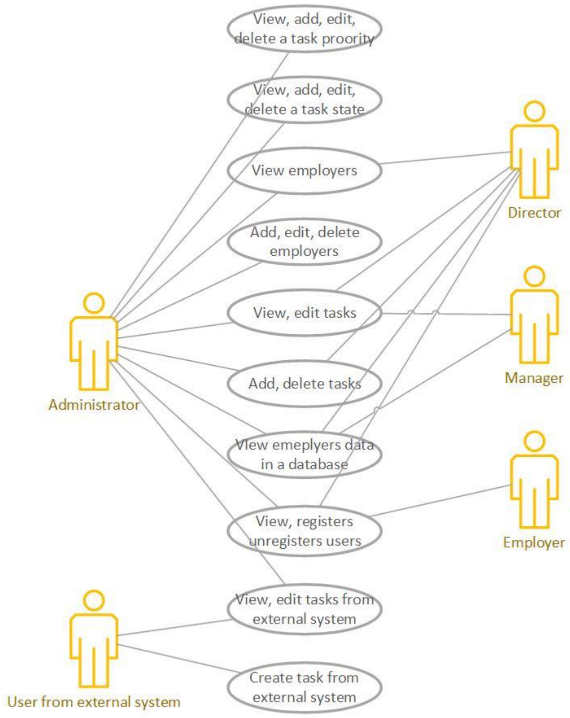

Fig. 3. *UML-case diagram of Web system*

Four different levels of access are available in the system (Fig. 3):

- administrators
- directors
- managers
- employees

The administrator is responsible for the various settings in the system as well as servicing new or existing employees. He has the highest level of privileges and for him, all the functionalities of the system are unlocked. The administrator can view, add, edit, and delete the states and priorities that can be selected when creating a task. When setting states, a state can be marked as final by marking the backup option. This way, changing the status to final for the task will remove it from the current task table and move it to the archived task table. Administrators can view, add, edit, and delete tasks and employees, register and delete external users, and view, delete, and change only the status and status of tasks entered by external systems. To be able to assign tasks in the system, there must be at least two employees, at least one entered priority and at least one entered status. Admin can also change the status of registered employees and view their profile in the system. Only the administrator cannot delete his profile. In this way, at least one administrator is always provided.

The director can view information about registered employees and current tasks. It can monitor the status of tasks, assign tasks to managers, register and delete external users, and view, delete, and change only the status and status of tasks entered by external systems. It can also edit and remove tasks entered by the main system, reassign them to another employee, and change their status. To be able to assign tasks in the system there must be at least two employees, at least one priority entered and at least one status entered by an administrator.

Managers, in turn, can only view tasks assigned to them and change their status. Based on this information, they can make decisions about the work process, such as what will be the tasks of the day that the employees will have to complete. The system clearly shows by priority which tasks are more urgent than others and should be completed first. With statuses that indicate a task as completed, it is archived. This backup is deleting the task from the current task table and moving it to the archive task table as well. The task is removed from the manager's task list and is no longer loaded. Managers can also review the information entered for them in the system.
Employees can review their data entered into the system

C) Optimistic approach

The system loads data from the database into the memory of the machine on which it is launched. This gives rise to a certain problem related to the actuality of the data being worked with. If a director-level employee edits the title of a task and at the same time a manager-level employee has already changed the status of the task and saves it in the database, the director would be working with out-of-date data. If attempted to write, this would compromise the already existing information on the server, as the director would unknowingly change the title and status without intending to. This kind of problem is easily solved with an optimistic approach. It is implemented by adding column in the system tables. This column indicates the number of changes made so far. In the example described above, if the director tries to save the task by changing the title only, the system will go to the database, and read the counter recorded there. Then the values of the newly read counter and already read and stored in memory will be compared. If the values do not differ, the change will be written to the database and the counter will be incremented by one. But if the values don't match, it means that the record was changed prematurely, as, in the example above, the manager changed the status of the task before the director saved his change. In such a case, any actions with the record are terminated until it is reloaded and updated in memory (See Fig. 4).

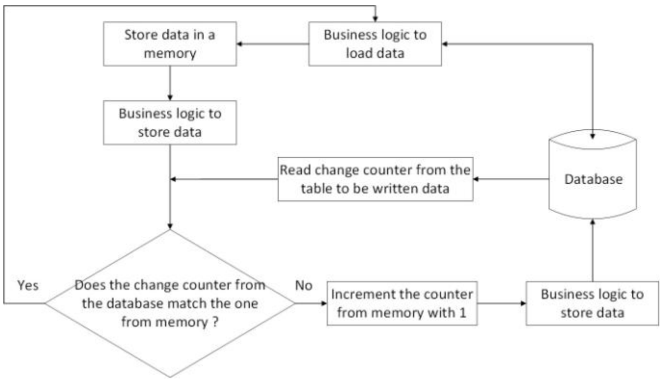

Fig. 4. *A summary of the optimistic approach.*

D) Web System Architecture

The system architecture is based on the Document-View model. The document stores the data, manages its printing, and coordinates the updating of multiple data views. The view displays the data and manages the employee's interaction with it, including selecting, adding, editing, and deleting.

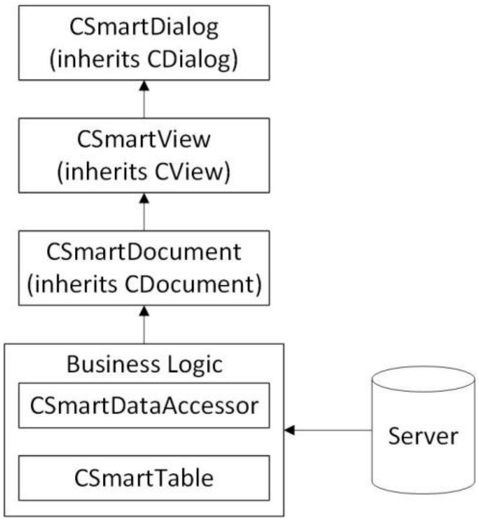

Fig. 5. *Web System Architecture*

The architecture is based on the following classes:

- CDocument – class for supporting objects used to store or control program data and provides the basic functionality of documents defined by the programmer.

- CView – provides the basic functionality of views. A view is attached to a document and acts as an intermediary between the document and the user. The view renders the document data on the screen and interprets user input as operations on the document. In the current development, the CListView class, a descendant of CCtrlView, which in turn is a descendant of CView, is used for data visualization. CListView simplifies the use of the list control and CListCtrl, the class that encapsulates the list control functionality, with the MFC document view architecture

- CFrameWnd – supports objects that provide the frame around one or more document views. In current development, it is inherited from CMDIFrameWnd, which in turn is inherited from CMDIFrameWndEx. CFrameWnd's successors build on its base functionality. - CDocTemplate – maintains an object that coordinates one or more existing documents of a given type and manages the creation of the correct document, view, and frame window objects for that type. It can be either CSingleDocTemplate or CMultiDocTemplate, depending on whether the system uses one or several different documents for different data.

In the current system, "Smart" classes have been developed, inheriting the base classes and upgrading them according to the needs of the application (Fig. 5). Additional classes added:

- CSmartTable to work with the data (read, add, edit, and delete) directly from a specific table in the database.

- CSmartDataAccessor takes care of the conditions under which it is read, written, and delete, and takes care if changes are required on the other tables, except the one that is the main one for the class

An advantage of this type of architecture is that in the presence of multiple views, the data used is kept in the same document. This allows individual views to work with the same code common to all views. This code is in the document. It, in addition to storing and processing the view data, also takes on the task of updating all views when the data changes from any of the views using the document.

The implementation of the Document-View architecture separates the data itself from its display and user operations on it. All data changes are managed through the document class. The view calls this interface to access and update the data. Documents, their associated views, and the frame windows that frame the views are created from the document template. The document template is responsible for creating and managing all documents of a single data type.

3. Tests

According to the set goals and objectives, the following functional tests were performed on the desktop and the web system:

- User registration by an employee with the role of administrator or director through the desktop system (Fig. 6). Registration is easy by entering a username.

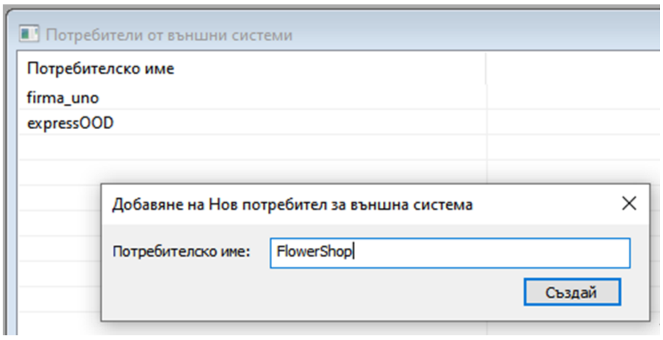

Fig. 6. *Registration screen*

After authorization, the web platform client receives a JWT with its ID. The token is stored in the browser's Local Storage. After registration, the user can change his password. Warning messages appear when passwords do not match.

- After logging in, all entered tasks of the user are loaded. In Fig. 7 demonstrates how, of all tasks visible through the desktop system, only those registered by "FlowerShop" are loaded.

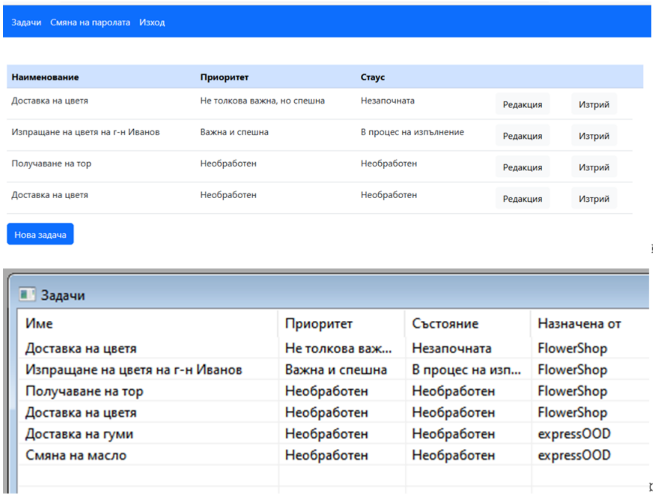

Fig. 7. *All table tasks entered by the logged-in user.*

The User can create and edit tasks. Every single task can be deleted. When logging out of the web system, the token is deleted.

The Administrator and Director role employees can view, edit and delete tasks entered from an external system (Fig. 8). Editing can only be done on the status and status of a task.

4. Conclusion

Based on the tests performed, the development fulfils all the set goals and tasks. The web system allows viewing, creating, editing, and deleting tasks, with various protections applied to the available resources. The API provides security by encrypting the access password with a key, limiting the number of requests that can be received per unit of time, and requiring authorization with each request. The desktop system has four different levels of access, each of which is tailored to the functionality it needs to perform. Additional services related to creating and deleting users to access the web interface and viewing, editing, and deleting tasks entered from an external system have been implemented. Through the web system, transparency is achieved in the work of a company. Its customers are allowed to monitor and even participate in the work process by managing tasks entered by them and monitoring their completion. The programming interface allows flexibility in developing different client applications with which to exchange data securely. Despite the available functionalities, the development could be further developed, such as the implementation of an option to send leave requests from employees to management or send emails.

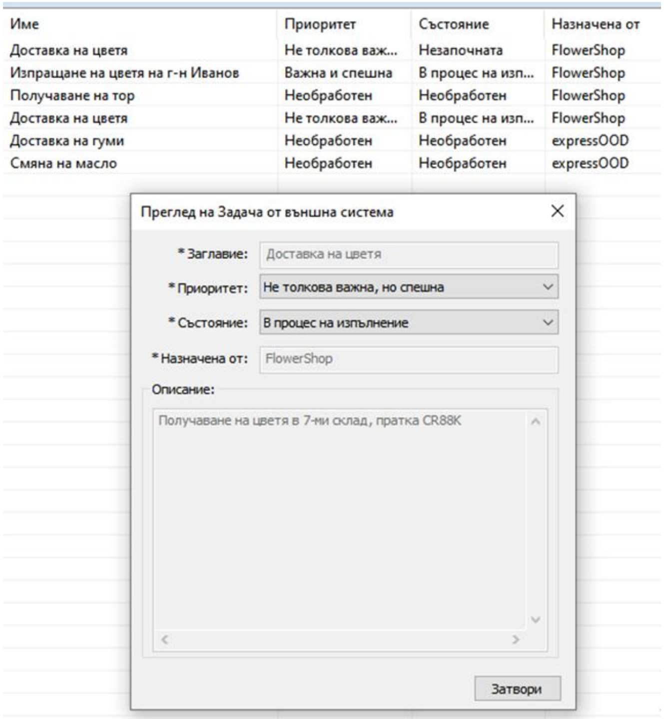

Fig. 8. *Register of tasks entered from an external system.*

5. References

1. E. Medvedeva, S.Kroshilin, Sergey. Transformation of Personnel Management Process: Reshaping Responsibilities of Non-Medical Personnel. City Healthcare. 4. 60-72. 10.47619/2713-2617.zm.2023.v.4i2;60-72. (2023)

2. Y. Tokareva, E. Gasparovich, E. Uskova, A. Zotov, (2023). Personnel Development Management at the Enterprise With Mixed Labor Organization. Management of the Personnel and Intellectual Resources in Russia. 12. 5-12. 10.12737/2305-7807-2023-12-2-5-12,(2023)

3. A. Litvinyuk, L. Babynina, L. Kartashova, L. Ivanova-Shvec, E. Novikova, Z. Troska, E. Kuzub, S. Kartashov, (2023). Modern technologies of personnel management. 10.12737/1863127. (2023)

4. K. E. Pearlson, C. Saunders, D. F. Galletta, Managing and Using Information Systems: A Strategic Approach, 7th Edition, Wiley, ISBN: 978-1-119-56115-6, (2019)

5. K. Sousa, Effy Oz, Management Information Systems, Seventh Edition, Cengage Learning EMEA, ISBN: 9781305172180, (2014)