Development of an Information System for the Distance Learning Process Organization

A Obukhov¹, A Volkov¹, N Maistrenko¹
¹Tambov State Technical University, 392000 Tambov, Russian Federation

E-mail: Obukhov.art@gmail.com

Abstract. The article describes the process of creating a personal account system for correspondence students, starting with the study of the issue as a whole and ending with the process of developing one of the main components of the information system. The process of creating an IS includes such stages as: determining the required functionality, designing the overall architecture of the system and each of its components. The technologies and software solutions used to system implementation are described, as well as the method of integrating the created solution with the existing information infrastructure of the educational institution. A brief mathematical description of one of the tasks solved by the information system, automated schedule generation, is also given.

1. Introduction
For the successful functioning of any organization that consists of more than one person, it is necessary to ensure information interaction between employees and customers of this organization. Such interaction includes both the direct exchange of information between the participants and the ability to obtain information about the current state of a particular process in the organization.

One of the striking examples of a geographically distributed organization is the correspondence department of any higher educational institution. If, in the case of full-time study, the interaction of students with teachers and the administration (dean's office) occurs mainly in person, then part-time students can stay outside the region in which the university is located for a long time and come only during the certification events. To maintain communication with students during their absence from the university, various technologies are used, from traditional phone calls and sending paper documents by mail to modern methods of communication via the Internet [1-3].

The most convenient modern solution for organizing the interaction "student - dean's office", both in terms of use and in terms of support of working capacity, is the "personal cabinet" system, which can include all the necessary components for both the dean's office staff and students. Such system allows you to integrate everything you need in one place, allowing you to provide global management and monitoring of each component's state [4, 5].

Existing ready-made solutions may not take into account all the features of the educational process in this particular institution. Therefore, it is advisable to develop such system directly within the specific university, since this will allow taking into personal account all the wishes of employees and students, as well as integrating it with the existing components of the digital infrastructure of the university [6,7].
This article will consider the task of designing and developing a comprehensive “personal cabinet” solution for organizing interaction between part-time students and the dean’s office for the Distance Learning Institute of Tambov State Technical University.

2. Main stages of design workflow
The first stage includes the definition of a specific set of functions required in this particular case, coordination of implementation possibilities with the technical service of the university and drawing up a technical task. At the same time, it is necessary to take into personal account the possibility of integrating the developed solution into the existing digital infrastructure of the university: for example, a general authorization and authentication system, a database of students, progress, etc.

At the next stage, the target platform for which the software package will be developed is selected, and the necessary computing resources for its operation are determined. In our case, the system is divided into two functional parts: a client module (for students) and a server module (for administration). The client module is executed in the form of a mobile application, the administration module is in the form of a web service combined with the server part of the mobile application. [8] Having chosen a platform, we need to decide on the technology stack used to develop each part of the system [9]. Based on the planned functionality and capabilities provided by the existing infrastructure, information and datalogical models are built, as well as a functional diagram of the solution being developed [10]. The implementation is carried out in two stages: first, an independent server part is developed, then a dependent client part.

The first step in creating the backend is preparing and setting up the environment in which the application will run. After the deployment of the required software stack, a database of the corresponding structure is created and all the necessary software modules (their functional and, if necessary, visual components) are developed.

The creation of the client part of the system is a classic cycle of mobile application development, which includes: design and development of the general architecture, creation of the user interface, and implementation of all functional modules ("business logic") of the application. [11-13]

After the development of all the components of the system, it is necessary to debug, eliminate all errors that have arisen and test the application with the participation of real users. The last stage, preceding the service commissioning, is the creation of reference documentation for working with the system and its support.

3. System architecture and main functional principles
The system "Personal cabinet of the correspondence student" consists of two main parts: client side and server side. The server side implements the backend of the system: interaction with the infrastructure of the university and the main logic of the system, and also contains the frontend of the administrative part of the personal cabinet. The main database is also located here. The client side implements a frontend for students and is a mobile application for the Android platform.

Server side contains a five subsystems [14]:
- visualization subsystem - designed to display information to the user;
- input-output subsystem - responsible for processing information entered by the user;
- network subsystem - responsible for interaction with the university IS, and also manages the data exchange with the client part of the system;
- database communication subsystem - is responsible for extracting information from the database and storing it in the database;
- scheduling subsystem - implements automated filling of the schedule for extramural students based on the full-time schedule.

Client side contains a two general subsystems:
- UI subsystem - responsible for displaying the user interface, processing it, as well as receiving and checking information entered by the user;
- subsystem for working with data - implements the logic for receiving information from the internal (local database) or remote (server part) storage.

Next, we will consider the scheme of functioning of each part of the system.

3.1. Administrative web-interface

The server side of the system is a website based on a PHP framework. This layer implements the concept of a single main script that runs when any request to a website is made. This script contains the logic of the system kernel. The kernel performs additional initialization, full verification of the user's access rights to the resource, and loading the module containing the business logic. The kernel is also responsible for rendering the main web page frame, which includes common elements used across all pages [15, 16].

The functionality of the administrator interface is grouped into several main sections: schedule, notifications, documents. Each of the sections is an interface for interacting with the database and includes pages for displaying information from the database, as well as pages for editing it.

3.2. Mobile application backend

The second task, which the server part of the system solves, is to service requests from the mobile application and interact with the general IS of the university. The functioning of this component of the system is fully automatic, without any user intervention. The API concept assumes a request-response interaction. When an incoming request is received from a client application, the system first of all checks whether this request is authorized.

For all cases, except for a request for a student to log into the system, authentication is carried out using the token issued to the user when logging in. A login request is authorized using a unique application key, which is recorded in the source code of the client side. In case of successful authorization, the system determines which request came from the user. To do this, each request is executed with the indication of the so-called. method, that is, the action to be performed. The system reads the method name and checks if a matching handler module exists. If a handler module is found, the system loads it and redirects the execution to this module. After the module finishes its work, the result is packed into a server response and sent to the client.

In case of receiving an unauthorized request, authorization error, the absence of a method processing module, or an error occurs during its operation, information about the problem is also packed into a server response and sent to the client.

To process some requests, such as, for example, a request for authorization, it is necessary to interact the university IS. The work with the IS is performed in the same way as the "client-server" interaction within the personal cabinet - through a set of APIs (different APIs can be used for different components of the external IS). In this case, the software of the personal cabinet itself acts as a client: it generates requests, sends them to external servers and processes responses.

3.3. Client mobile application

The client side of the personal account is a classic mobile application for the Android platform. Interaction with it occurs according to the standard Android scheme: the functionality of the application is divided into separate screens, where each screen provides a logically complete set of functions. The client application is exclusively a frontend and does not perform any complex data processing, all information displayed to the user is obtained from the server backend. This approach requires a continuous internet connection of the client's mobile device. To organize the possibility of autonomous work, the application implements a data caching strategy: if there is an Internet connection, the information received from the server is not only displayed on the screen, but at the same time is saved to the local database. If there is no Internet connection, the user will be shown this saved data.
Operations that require a sufficiently long time for their execution (for example, work with a local database or network requests) are performed asynchronously with respect to the processing of the user interface. The business logic layer is also responsible for managing asynchrony.

4. Formalization of the schedule generation process

One of the tasks in the development of a personal account is the automated generation of classes schedule for part-time students based on the existing full-time schedule [17, 18].

Compared to the traditional scheduling problem, there are a number of distinctive features to consider. Firstly, in the timetable for extramural education there is no such thing as a "calendar grid", that is, the timetable is not a weekly or bi-weekly schedule, but simply a list of pairs arranged by certain dates within a specified period. Secondly, in addition to all the standard restrictions that arise when scheduling from scratch, one additional restriction is introduced - the absence of intersections with the schedule of full-time students.

To formalize the task at hand, it is first necessary to introduce a number of conventions [19]:

- \( P \) — set of teachers;
- \( G \) — set of student groups;
- \( Z \) — set of disciplines;
- \( V \) — set of occupation types (lecture, practice, etc);
- \( A \) — set of audiences;
- \( K \) — set of buildings;
- \( D \) — set of days in a given interval;
- \( L \) — set of classes;
- \( b_{zv} \) — the number of hours for each type of occupation for each discipline, \( z = 1, 2, \ldots, Z \); \( v = 1, 2, \ldots, V \);

Next, consider the matrices:

- \( X_{pd} \) — three-dimensional Boolean matrix of teacher employment, in which \( l \) means that teacher \( p \) on day \( d \) conducts a lesson the \( l \)-th class;
- \( X^*_{pd} \) — three-dimensional Boolean matrix of full-time teacher employment, in which \( l \) means that teacher \( p \) on day \( d \) conducts a lesson with the \( l \)-th class;
- \( X_{gd} \) — three-dimensional Boolean matrix of study group occupancy, in which \( l \) means that group \( g \) on day \( d \) has an occupation of the \( l \)-th class;
- \( X_{ad} \) — three-dimensional Boolean matrix of audience occupancy, in which \( l \) means that audience \( a \) on day \( d \) is used on the \( l \)-th class;
- \( X^*_{ad} \) — three-dimensional Boolean matrix of audience employment for full-time education, in which \( l \) means that audience \( a \) on day \( d \) is used on the \( l \)-th class;

To draw up a lesson schedule, you need to find a three-dimensional matrix \( T_{adl} \) in which \( t_{adl} = (p, g, z, v) \) is a tuple corresponding to a lesson in audience \( a \) on day \( d \) on the \( l \)-th class, where \( p \) is the teacher, \( g \) is the study group, \( z \) - discipline and \( v \) - type of activity. In this case, the following restrictions must be met:

- non-intersection with the full-time schedule for the employment of teachers: \( x_{pd} \land x^*_{pd} = O \) (where \( O \) — null matrix);
- non-intersection with the full-time schedule for the employment of audiences: \( x_{ad} \land x^*_{ad} = O \);
- the ability to move students from audience to audience between classes: \( t_i \left( a_i, a_{i+1} \right) \leq t_g \left( l_i \right) \), where \( t(a_i, a_j) \) — a function that returns the average travel time (in minutes) between audiences \( a_i \) and \( a_j \) (if the audiences are located in the same building, the function returns 0), \( t_g(l_i) \) — function that returns the length of the break (in minutes) between classes \( l_i \) and \( l_{i+1} \);
appropriateness of the audience and the type of activity: \( v_{i,j} \in f_r(a_{i,j}) \), where \( f_r(a_i) \) — a function that returns possible types of activities for the audience \( a_i \).

At the same time, several target criteria for optimization can be distinguished:

- minimum number of crossings per day;
- even distribution of the load during the training period;
- the smallest number of “windows” between classes per day;

etc. The formal statement of the optimization problem for the criterion “minimum number of crossings per day” will have the form:

\[
\sigma^2(f_i(a_{i,j})) \rightarrow \min, i = 1, 2, ..., L,
\]

when \( f_i(a) \) — function that returns the building number by audience number.

5. Practical realization

Since the server side of the system is a web application written in the PHP programming language, it requires an appropriate runtime environment. This environment is a full-fledged web stack that supports technologies such as HTTP, HTTPS, PHP scripting, and communication with MySQL databases. The specific software products required to deploy this stack may vary depending on the technical capabilities provided by the university infrastructure. However, the recommended configuration for the platform to work is a web environment collectively called LAMP.

The backend is based on the Evening Sphere Portal web framework, which is a set of tools that simplify the creation of web applications in PHP. This framework implements all the basic features necessary to create web applications of any type, such as authorization, communication with a database, automatic construction of a web page layout based on templates (Fig. 1). Also, this framework implements a flexible modular architecture that allows you to structure the code of a web application. All the functionality of the application is divided into separate, logically complete areas of responsibility, which are implemented by separate modules. Each module, in turn, is divided into a number of independent, logically complete scenarios - actions, each of which is responsible for processing one web page or its fragment.

The client side of the personal account is a full-fledged Android application that requires the Android Runtime. An application based on the Evening Sphere Portal is a module, or a set of several modules that directly implement the logic of the application. All typical tasks, such as initialization, connecting to the database, authorizing users, checking access rights and launching a specific module, are performed by the system kernel. This allows you to focus on solving applied problems. Each module consists of two parts: the logic layer (directly the module code) and the presentation layer (templates). The module is divided into separate functional blocks - the so-called. actions. In most cases, a specific action is responsible for processing a specific web page and implements all the functionality of a given web page. However, the architecture of the system also provides for embedded actions, or even purely functional actions that do not participate in page rendering. In addition to files that implement specific actions, each module includes several system files, such as a file with access settings, a module initialization file, a general module settings file, a file with common functions for a given module, etc. For initial deployment of a module on a freshly installed framework the configuration file of the database tables is also required, if it is used by this module. The functionality of the server side of the personal account is assembled into a single module – the “izo” module.

The development of any module for the platform begins with creating a set consisting of several service files: a module initialization file, a module configuration file, a file with common functions for a given module, a rights management file, and an optional main action file. After creating the skeleton and checking the functionality of the module by accessing its main action through a web browser, you can start implementing specific actions that are responsible for certain functional components of the module. The actions required for the server part of the personal account can be divided into several groups according to their purpose:

- izzo_api* actions, which are responsible for the operation of the client-server API;
izo_*_form actions, which are responsible for organizing data entry forms;
other izo_* actions responsible for displaying data.

Let's take a closer look at creating an action for each of these groups using specific examples. The rest of the actions in each group are created in the same way.

![Figure 1. Fragment of the information system interface.](image)

5.1. Client-server API related actions

Since the set of methods of the client-server API is quite extensive, it is advisable to divide the implementation of these methods into several separate actions, each of which will contain the code of several methods responsible for similar operations [20]. The main logic of the API is implemented in the izo_api action. This action reads the method name from the HTTP request parameters, checks for authentication credentials, and loads the appropriate handler action for this method. Also, upon completion of the handler, the main API action generates and sends a JSON response containing the result of the method or an error message.

Handlers for specific API methods are implemented in the izo_api_* actions and are grouped by area of responsibility: authorization, working with a schedule, working with notifications, working with documents, etc. The code inside each handler action contains one main switch-case construct that calls the code of a specific method by its name. The essence of most of the methods is to obtain a specific set of data from database tables, and pack this data into the format described by the API specification. The process consists of three stages: generating an SQL query, reading information from the database, and repackage the received data into the required format. In some cases (for example, when reading study progress), the university IS acts as a data source instead of a local database. Then, instead of generating SQL queries to the local database, corresponding queries to external APIs are executed. Actions in this group do not have template files.

5.2. Data input forms related actions

The second group of actions is designed to display and process input forms for various data. In the action code of this group, two main operations are performed: checking the correctness of the entered data and writing information to the database tables. The validation criteria may vary depending on the input data. For example, for text values, it checks to see if an empty string has been passed. For numeric values (as well as for values in the form of date and time), the minimum and maximum allowed values are checked (for example, a class number cannot be negative). For values of the "enumeration" type, the correspondence of the passed value to one of the possible options is checked.
If the data is entered correctly, an SQL query to the database is generated and executed, and the fact of creating or editing data is recorded into the system log. If the data has not been verified, the request is not generated, and the user is notified of an error.

Action templates in this group contain the HTML code for displaying input forms, as well as the minimum necessary auxiliary binding, such as a block for displaying informational messages and links for exiting edit mode.

5.3. Data view related actions

The last group, which includes actions that are not included in the previous two, contains pages designed to display various data sets, such as a schedule, a list of notifications, a queue of requests for documents, and so on. The algorithm for this group of actions is quite simple: based on the HTTP request parameters, such as the way data is sorted or the criteria for filtering them, one or more SQL queries are formed to one or more tables. The result of executing queries is packed into the $result array and passed to the template file for rendering.

The template implements the necessary loops of iterated data, which form a visual representation in HTML elements: tables, lists, blocks. In addition, templates for this group of actions include navigation elements (for example, links for organizing page navigation), as well as elements for choosing how to sort data and a form for entering filtering parameters. Also, the actions of this group implement transitions to edit pages and the function of deleting data from the database, for which special links are formed [21].

Delete requests are processed at the start of the action so that the data is deleted from the database before the main execution of the fetch request. The fact of data deletion is recorded in the system log.

6. Conclusion

This article discusses the task of developing an information system "personal account" for organizing interaction between students and the dean's office of the Institute of Distance Learning. The results obtained can be used to implement various information systems and automate the educational process.

The technologies and software solutions used to system implementation are described, as well as the method of integrating the created solution with the existing information infrastructure of the educational institution.

The paper also sets the task of optimizing the schedule for the Institute of distance learning. The solution of this problem will be the subject of further research, since at present the problem of finding the optimal schedule is still relevant.

7. References

[1] Zelinskiy S 2020 Analysis of the possibilities of the MOODLE learning management system for organization of distance learning in the conditions of the university ScienceRise: Pedagogical Education 5(38) 33–6
[2] Franchuk N P and Prydacha T V 2021 Organization and conduct of classes in educational institutions during distance learning Journal of Physics: Conference Series 1840(1) 012054
[3] Vakaliuk T A, Spirin O M, Lobanchykova N M, Martseva L A, Novitska I V and Kontsedailo V V 2021 Features of distance learning of cloud technologies for the organization educational process in quarantine Journal of Physics: Conference Series 1840(1) 012051
[4] Fazliakhmetov T R, Salekhova L L, Muhutdinova G I and Aslanov R M 2021 Using the messenger software opportunities in distance education Revista on line de Política e Gestão Educacional 378–86
[5] Salama R, Uzunboylu H and Alkaddah B 2020 Distance learning system, learning programming languages by using mobile applications New Trends and Issues Proceedings on Humanities and Social Sciences 7(2) 23–47
[6] Shakhovska N, Vysotska V and Chyrun L 2016 Intelligent Systems Design of Distance Learning Realization for Modern Youth Promotion and Involvement in Independent Scientific Researches *Advances in Intelligent Systems and Computing* 175–98

[7] Bozkurt A 2019 From Distance Education to Open and Distance Learning *Advances in Educational Technologies and Instructional Design* 252–73

[8] Mupfiga P, Mupfiga M C and Zhou T G 2017 Enhancing teaching and learning through the use of mobile technologies in Zimbabwean universities *Journal of Systems Integration* 8 43–51

[9] Krasnyansky M, Ostroukh A, Karpushkin S and Obukhov A 2016 Algorithm for Structural and Parametric Synthesis of Electronic Document Management System of Research and Education Institution *Journal of Applied Sciences* 16(7) 332–7

[10] Snopko S L, Gmar D V, Rylov A V, Artamonov O A and Shahgeldyan K I 2019 Developing X-Ray Simulator for Luggage Screening 2019 *International Science and Technology Conference “EastConf”* (Vladivostok: IEEE) pp 64-66.

[11] Kang B, Song B, Yang S and Lee J 2017 A Development Technique for Mobile Applications Program *Studies in Computational Intelligence* 47–62

[12] Brügmann S, et al 2015 Towards content-oriented patent document processing: Intelligent patent analysis and summarization *World Patent Information* 40 30–42

[13] Kirikova M and Pudane M 2014 Viable Systems Model Based Information Flows *New Trends in Databases and Information Systems* 97–104

[14] Anvari A, Alipourian G A and Taleb-Beidokht R M A 2012 An Investigation of Innovation in Higher Educational Environments-A Consideration of Five Substructures (Technical, Administrative, Information Systems, Information Technology and Knowledge Management) *Middle-East Journal of Scientific Research* 11(9) 1278-1285

[15] Krasnyanskiy M et al. 2018 Formalization of document management using multilevel graph model of information processing *International Multidisciplinary Scientific GeoConference: SGEM* (Albena: Stef92 Technology) pp 413-420

[16] Belyanina N V and Lyovin S A 2014 *Subsystems of the electronic document management system and their functions* (Moscow: Works of SGA) p 110

[17] Rustauletov B 2020 The Automation of Course Scheduling in Higher Education Institutions: Mathematical Methods and Software Products *Proceedings of International Young Scholars Workshop* (Kaskelen: Suleyman Demirel University) pp 196-212

[18] Shi D, Fan W, Xiao Y, Lin T and Xing C 2020 Intelligent scheduling of discrete automated production line via deep reinforcement learning *International Journal of Production Research* 58(11) 3362–80

[19] Zhdanov I R, Bubareva O A 2018 Mathematical Model and Heuristic Methods for Solving the Multi-Criteria Scheduling Problems 2018 19th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices (EDM) (Erlagol: IEEE) pp 207-211

[20] Romanov E L, Troshina G V, Menzhulin S A 2020 Client-server application framework based on an object-oriented network model *IOP Conference Series: Materials Science and Engineering* 919 052014

[21] Obukhov A D and Krasnyanskiy M N 2020 Automated organization of interaction between modules of information systems based on neural network data channels *Neural Computing and Applications* 33(12) 7249–69

**Acknowledgments**

This work was carried out as part of a grant from the President for young candidates of sciences MK-74.2020.9. The work performed by the Laboratory of medical VR simulation systems for training, diagnosis and rehabilitation using the computing equipment of the Center for Collective Use “Digital Engineering”.

---

This page contains references to various scholarly works, including articles and conference proceedings. The content is formatted as a list of citations, each entry providing details such as the authors, title, journal or conference, volume, issue, and page numbers. The references cover a range of topics within the fields of education, technology, and research methodologies, highlighting advancements in intelligent systems design, educational technologies, and innovative approaches to teaching and learning. The acknowledgments section indicates the support for this research through a grant from the President for young candidates of sciences.