UDC 004.9
DOI: 10.15827/2311-6749.21.3.1

Developing a single sign-on for information systems

I.I. Kholod [1], *Dr.Sc. (Engineering), Professor, Dean of the Faculty of Faculty of Computer Science and Technology, iiholod@mail.ru*
M.V. Kovynev [1], *Student, kovinevmv@gmail.com*
I.S. Grigoryev [1], *Student, grigorievivan.1@mail.ru*
P.V. Korytov [1], *Student, thexcloud@gmail.co*

[1] *Saint Petersburg Electrotechnical University "LETI", Saint Petersburg, 197376, Russian Federation*

The paper discusses the problem of creating a single authorization system for university information systems. The number of information systems is constantly growing, therefore, there is a question of providing the user with a centralized single sign-on to such systems. The authors studied the existing solutions. After analyzing the solutions, they have selected the fastest growing systems, such as the Blitz Identity Provider and IBM Security Access Manager for Enterprise Single Sign-On, and considered their key advantages and basic restrictions. The solutions were compared according to the following criteria: cost, availability of an open source code, restrictions on connected systems.

The authors review the existing university subsidiary systems and propose the implementation of their own single sign-on system. The paper describes certain aspects of the system implementation in detail, including the authorization process in subsidiary systems and displaying of subsidiary system pages; they consider the system architecture and the authorization mechanism. The security issue and developed solution testing is studied, the main advantages of the resulting solution are revealed: cross-platform, centralized display of the subsidiary system menu in one place, cost. During testing of the developed system, no XSS vulnerabilities, SQL injections, etc. are identified. The system supports authorization through VKontakte, other subsidiary systems and the basic version by name and password. The developed information system is used at ETU "LETI".

Keywords: *single sign-on, web application, authorization security, unified information space.*

Nowadays, there are more than a billion sites on the global Internet [1]. Most sites have their own authorized access system. However, such approach forces us to store usernames and passwords for each system. And the more users have to remember them, the higher the chance of data loss or theft. Moreover, a user must remember the link to the system, follow the link, enter the authorization data and only after that he gets the appropriate rights that will allow using the system. If a user needs to work with two or more systems at the same time, the login script is repeated the corresponding number of times.

The solution to the problem of large number of user credentials is single sign-on. *Single Sign-On* (SSO) is a technology that allows a user to switch between systems without re-authentication.

There are two types of SSO:

– Enterprise, which implies the installation of an agent on user workstations for automatic substitution of the user's login and password in the application authentication windows [2];

– Web Single Sign-on, which provides authentication functionality in web applications. This technology provides a single authentication service (provider) for an organization's connected applications supporting such protocols as SAML, OAuth, and OpenID. In this case, one account is used to access all applications [2].

This approach allows a user to reduce many of his credentials in different systems to a single profile and saves time by eliminating the need for repeated authorizations [3].

The SSO disadvantages include the growing importance of a single profile in the context of information security, since with a successful attack on a profile, an attacker immediately gains access to all data associated with SSO and all resources that are accessed using this profile. Thus, when creating SSO, it is nesessary to pay extra attention to the protection of credentials [4].

For big organizations that have many subsystems in their management, the task is to create a compromise solution that will create comfortable working conditions for users and at the same time maintain the proper level of protection [5].

The overview of the existing solutions

Let us study the market for existing SSO offers. The sought solution must have the following properties:

– an open source code. Providing full access to the source code provides extensive opportunities for modification and subsequent improvement of the single sign-on system, as well as tight integration with the subsidiary systems;

– the ability to connect an unlimited number of subsidiary systems. It is planned to use ETU “LETI” as an experimental site, so the problem of the number of “subsidiary systems” that can be connected to SSO is important since the number of the university information systems is growing.

According to MarketsandMarkets’ forecasts, which is a global analytics agency, the total Single Sign-On market volume will grow to $1,599.8 million by 2021 compared to $846.6 million in 2016. Let us consider Blitz Identity Provider and IBM Security Access Manager for Enterprise Single Sign-On as the fastest growing systems [6].

Blitz Identity Provider

Blitz Identity Provider authentication server is software for managing user login to applications. It provides the ability to equip company's websites and mobile apps with user account security features that contain the latest security practices.

Key features [7]:

– single sign-on. Common methods of connecting applications are supported: SAML, WS-Federation, OpenID Connect, OAuth 2.0, connection via web-proxy;

– flexible authentication. There are more than 10 ways to authenticate users: password; using an electronic signature; using social networks and USIA; two-factor authentication using hardware key fobs, software generators, mobile applications, SMS codes. Different authentication rules can be applied to different groups of users and applications;

– access control. Applications access is regulated depending on the attributes of users, their access groups, login context parameters and the authentication methods used;

– logging of security events. Centralized registration of all successful and unsuccessful access events, changes in account security settings.

In addition, the system capabilities also include:

– customizable user self-service services, i.e. automation of the processes of registering accounts, maintaining profiles, recovering a forgotten password;

– web console for managing all settings of the authentication system, automating account management processes, auditing access events;

– customizable interface appearance, i.e. it is possible to customize an individual design for accessing each connected application;

– high performance (more than 1000 requests per second with response time less than 200 ms) with low demand on hardware resources for system deployment;

– Blitz Identity Provider is included in the Unified Register of Russian Programs for Electronic Computers and Databases.

IBM Security Access Manager for Enterprise Single Sign-On

IBM Security Access Manager for Enterprise Single Sign-On is an IBM solution that automates access to all applications in an organization using access profiles and corporate policies. The technology provides a single registration in Microsoft Windows, Mainframe Web, Java using any gadgets that are end network devices, including PCs, laptops, tablets, terminal servers.

The solution supports a wide range of authentication tools: RFID tags, smart cards including hybrid, biometrics. Users can take advantage of the security policy configuration features: timeout blocking, automatic termination of inactive sessions. The solution supports the multiplayer mode.

The system interacts directly with external users and systems; provides unified access control, reduces the number of errors in account management, reduces the time to make changes to accounts by increasing automation, provides data about user accounts across operating systems, monitors and audits user accounts to verify compliance with corporate policy.

Key features [8]:

– support for a wide range of smart cards, RFID tags, biometrics, hybrid smart cards (when organizing multi-factor authentication);

– the ability to flexibly configure security policies, such as locking the screen and closing applications by timeout, automatic ending of inactive sessions, etc.;

– support for multiplayer mode;

– support for terminal mode work.

The existing solutions were considered taking into account the following criteria: cost, open source, restrictions on connected systems, Web version, security of solutions, display in one tab.

The Blitz Identity Provide authentication server is a commercial project of the Softline. Only the partners of the REAK SOFT company can tell the cost of the Enterprise version. The Standard version is free, but has limited functionality.

IBM Security Access Manager for Enterprise Single Sign-On costs $144 for 12 months.

The Blitz Identity Provide authentication server and the IBM Security Access Manager for Enterprise Single Sign-On are commercial projects, therefore the source code access is not provided.

To connect this technology to your systems, it is nesessary to purchase a license, and depending on the cost, the number of connected subsidiary systems changes, for example, the Standard Edition from Blitz Identity Provide makes it possible to connect up to 20 applications, therefore, as in the case of an IBM product, it is nesessary to have a license for a bigger number of subsidiary systems.

The current technology of corporate single sign-on (Enterprise Single Sign-on, ESSO) implies the installation of an agent on user workstations, which provides automatic substitution of the user's login and password. However, this client is not suitable for all devices. As for the need for an interaction web interface, it should be noted that the browser is installed on most modern devices.

IBM's solution is ESSO, while Blitz Identity Provide has a web interface.

Solution security

The solutions under consideration, in particular the Blitz Identity Provide, provide a server that will perform all user' authorization actions. Therefore, all authorization data will be stored not inside the network, but on the company's server. So, companies are trying to secure their client data as much as possible, in particular, this solution provides two-factor authorization [5].

However, it should be noted that moving such server to the internal network reduces the risk of being compromised. Therefore, this reduces the need to develop additional security systems.

To work in several systems simultaneously, a user needs to authorize several times; he will have as many tabs (applications) launched as the number of applications he interacts with. At the same time, the considered solutions perform authorization for the user; the interface provides an opportunity to go to another application for a couple of clicks, having already automatically logged in on the corresponding page.

After reviewing and comparing existing solutions, we have identified their key features. Despite the fact that these solutions are in high demand, they have some drawbacks. It is proposed to develop a solution devoid of the indicated drawbacks and test it at ETU “LETI”. This solution will be placed on the university servers, so third parties will not participate in the implementation, thereby increasing the safety of work.

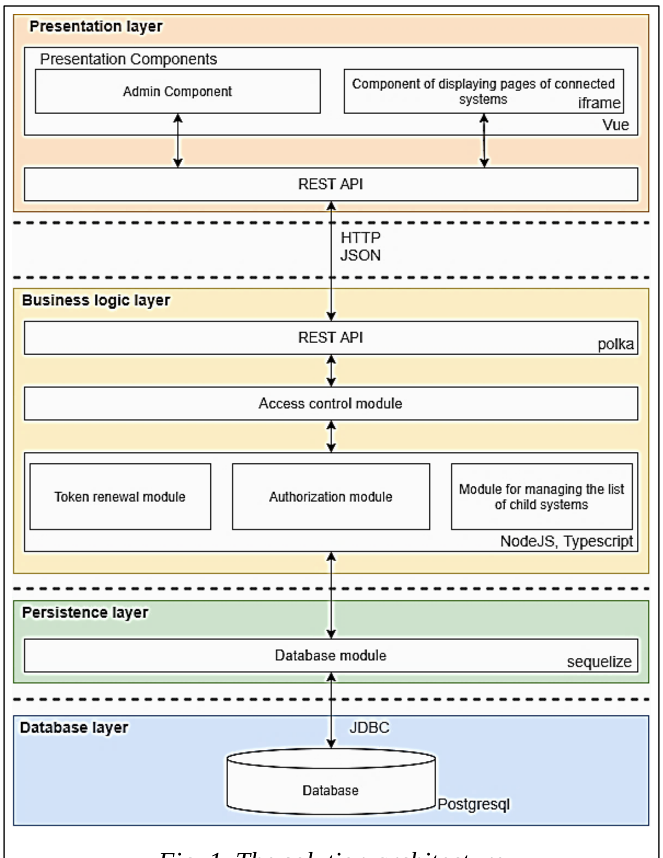

Fig. 1. The solution architecture

The description of the developed solution

The Universal Personal Account information system (UPA IS) is designed to provide a single entry into the university systems.

The main tasks to be solved:

– providing the user with a centralized access to the subsidiary systems to work with them;

– providing the ability to customize menu for connected systems;

– using pluggable subsidiary systems as an additional authorization method and external systems such as VKontakte.

The developed solution architecture is shown in Figure 1.

The solution is designed using a multilayer architecture [9] described using the architectural pattern “Layers” [10, 11].

PostgreSQL is used as a DBMS at the data storage level [12]. The ORM framework Sequelize (sequelize/sequelize) is used to access the data. Hereinafter, links to open source software on GitHub are provided in this format.

Authorization in the system is possible through the VKontakte social network using the OAuth 2.0 protocol, as well as through other subsidiary systems. For this purpose, the necessary endpoints are set when adding a subsidiary system, then when entering the UPA IS system, the user is immediately loaded with rights in this system.

At the level of business logic, modules in TypeScript are implemented to perform the tasks described in the corresponding layer.

The interaction between the client and server parts built using the polka web server (lukeed/polka) and the Node.js platform (nodejs/node) is carried out through the REST API [13].

Unlike the considered analogs, the UPA client is “thin”; it is a web application written in TypeScript using the Vue.js framework (vuejs/vue). This choice is due to the ease of scaling and high adaptability of these technologies in comparison, for example, with Angular and React [14, 15].

The system has an authority separation. The administrator is given the ability to add/edit/delete subsidiary systems. The users are able to authorize in systems added by the administrator. In case of successful authorization, tabs of a subsidiary system appear in the side menu, so the user can start actively working with it.

Let us consider the most significant features of the components and processes of the resulting information system.

Authorization

Upon entering the IS, the user is given the opportunity to link the current profile in the UPA system to any subsidiary IS. The general scheme of the authorization mechanism is shown in Figure 2.

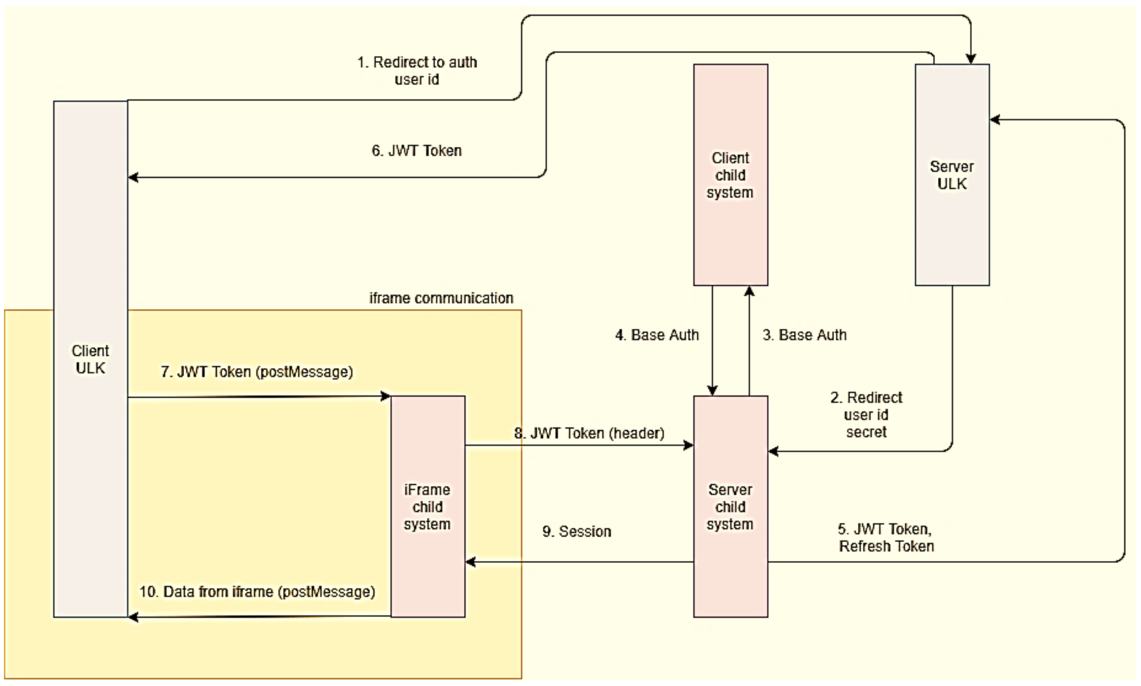

Fig. 2. The process of authorizing a user in the system for the first time

The user logs into the system through the VKontakte network or another subsidiary system, thus, information about him appears in the session.

The user clicks the button “link the account to the subsidiary IS” which enables a request to the UPA server, and the user ID is retrieved from the session.

Further, there is an appeal to the endpoint of the subsidiary IS. The link to the reverse redirect, the user ID implemented by OAuth 2.0, is passed.

If the current user does not have a session, then we perform basic authorization in the system.

The user enters authorization data in the subsidiary system, then redirects with the authorization code. This code is used by the UPA server to obtain a JWT token.

The subsidiary system server reports the JWT token to the UPA server, the token to update. The JWT token contains information about the user and the menu that is available to him in the subsidiary IS. If the JWT token lifetime expires, a request is made to receive a new one using a refresh token.

This JWT token is reported to the UPA client in order to pass authorization in the iFrame.

Using the postMessage method, this token is sent to the client side of the subsidiary IS. It is set that all subsequent client’s requests to the server will be executed with a header containing the given token.

Further, the subsidiary IS client sends a request for authorization in the system with this token.

The server checks the received token and gives the session, for example, in the Cookie form.

The subsidiary system can inform the client about the authorization success/failure in the system via postMessage.

After going through all the stages described above, the user enters the UPA IS. This mechanism allows linking the current profile in the UPA system to any subsidiary IS.

Displaying pages of subsidiary systems

Displaying pages of subsidiary systems is implemented through iFrame. iFrame or floating frame is a separate window, HTML document that is displayed along with other page content in the browser window. The usage example:

```
      <iframe
src="http://child.etu.ru/iframe/page/example?hideSidebar=true"
width="1920" height="1080" >
          Your browser does not support iframes!
      </iframe>
```

To display the page, it is necessary to provide a link to it. With this capability, it is possible to pass additional information through Query parameters, for example, whether to display the navigation and sidebars or not.

For client parts to interact, it is necessary to transfer messages between them, for example, JWT tokens. This message is implemented via postMessage. Below are the sending from the UPA to the iFrame and the message handler in the subsidiary IS.

```
const data = {'anyDataKey': 'anyDataValue'}
f.postMessage(JSON.stringify(data), 'http://child.etu.ru/page/example')

function handlerMessage(e) {
    var origin = e.origin
    if (origin !== 'http://ulk.etu.ru'){
        return false;
    }
    const data = JSON.parse(e.data)
    // work with data
}
window.addEventListener('message', handlerMessage);
```

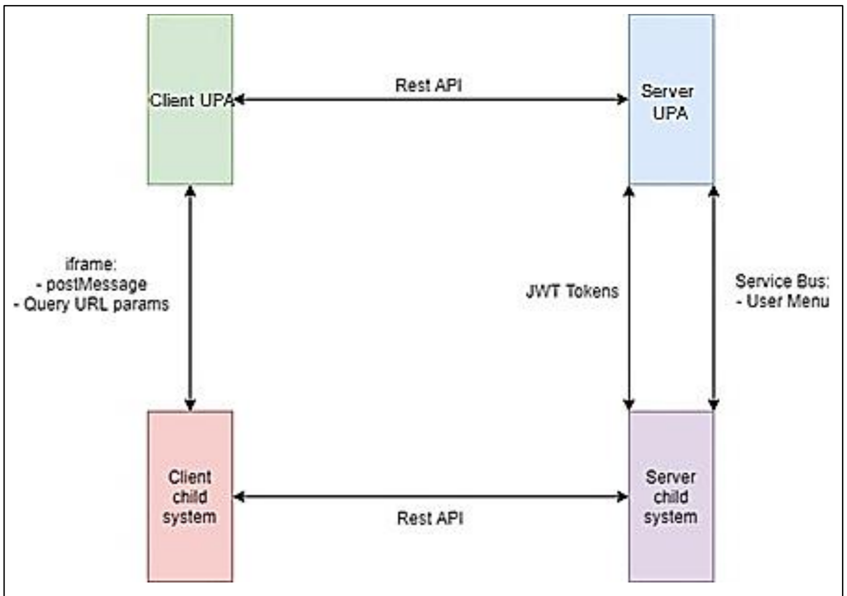

Fig. 3. The scheme of interaction between the UPA client/server parts and the subsidiary IS

Considering the fact that user's rights, profile or menu in the subsidiary system may change, the UPA IS will contain irrelevant information. To update it, the UPA implements a bus that runs daily and makes requests to the subsidiary system for each user and saves his menu on the UPA server. Menu information is stored in the token, which allows immediate menu update by updating the token.

The general bus layout is shown in Figure 3.

Menu display

In this implementation, UPA IS displays the entire content of the subsidiary system page. The problem of a double menu arises: both UPA and the subsidiary IS have own site navigation. The user might get lost and not understand which menu is responsible for what. This is especially inconvenient if he wants to work in two or more subsidiary systems at the same time, so it was decided to combine the navigation menu into one common one located in the UPA. For this purpose, in each JWT token, next to the user ID JSON is put in the following format responsible for the menu:

```
    {
      "menu":[
         {
            "name":"Partners",
            "child":[
               {
                  "name":" List of partners",
                  "url":"https://dev.digital.etu.ru/trajectories/iframe/part-
ners/list",
                  "icon":{
                     "type":"font-awesome",
                     "content":["fa", "list-ul" ]
                  }
               },
            ]
         },
         {
            "name":"Library",
            "child":[
               {
                  "url":"https://dev.digital.etu.ru/trajectories/iframe/library",
                  "name":" Library browsing",
                  "icon":{
                     "type":"font-awesome",
                     "content":[ "fa", "book"]
                  }
               }
            ]
         }
      ]
```

Testing

In order to check the developed IS performance, as well as to identify and correct errors, we have tested security to prevent unauthorized access and the interface to identify errors that arise during user's interaction with the UPA.

To identify problems related to the developed IS security, the security testing included the following tests:

– nginx configuration;
– the presence of SQL injections, XSS;
– authorization in the system;
– security authorization.

The security testing of nginx configuration was based on the gixy program, which is used to detect security problems and searches for other nginx configuration errors. No vulnerabilities were found during testing.

SQL injection testing was done manually using the sqlmap utility. During manual testing, we detected and corrected 1 injection. Automatic testing involved SQL substituting injections from the PostgreSQL database set into the given IS URLs. In total, more than 500,000 requests were sent to the server. No SQL injections were detected.

Testing for XSS was based on the Burp Suite web application security audit platform [16]. A sitemap was built. For every request sent to the server, an XSS substitution attack from a given set was performed. In total, more than 200,000 requests were sent to the server. XSS were not found. To prevent the XSS attacks, special headers were installed on the site.

The exchange of authorization data in the system is based on JWT tokens. In order for an attacker to narrow the circle of searching for token options, it was decided to insert a check for the token encryption algorithm. If an attacker manages to pick up an alternative encryption token CVE-2015-9235, then his token will be incorrect. Additionally, it was found that in order to increase the of the token signature searching time, the length of the secret-part must be at least six. Then, to find it at a speed of 300 requests/sec, it will take more than five years to find a valid token, which is much longer than its lifetime.

User interface testing

The testing involved creating test scenarios for using the developed system for each process [17]. An example of a test scenario for creating an audience booking request by an instructor is presented in Table 1.

Let us imagine positive and negative scenarios for creating an audience booking application by a teacher. Positive ones are: a user enters a username and password and enters the UPA; a user clicks to log in through VKontakte or "subsidiary system name" and enters the UPA. The negative ones are: a user did not fill in all the fields and clicks to enter using a username/password; a user has not filled in at least one field and clicks enter via login/password; a user clicks to enter through VKontakte, but does not confirm the requested permissions inside the redirect window; a user clicks to login via “subsidiary system name”, but the system is not active at the moment.

During testing, about 30 system errors were found and fixed.

After reviewing and comparing ready-made solutions, the main UPA advantages are highlighted:

– The system is designed as a web application, which enables simultaneous remotely operation in several systems via the Internet.

– Storing a database with user authorization data on a separate server increases the system security, since it is necessary to ensure only server security, in contrast to distributed systems.

– Centralized display of subsidiary system menu in one place.

– The subsidiary system interface is moved to the UPA side menu in terms of navigation. This improves usability, as all necessary links will be stored in one place, and it will be easier for a user to find them. Thus, it is possible to work in several systems at the same time in just one browser.

Table. 1

A test scenario for user's authorization in the system

Column name	Action / Result
Scenario name	System authorization
Preconditions	The user was logged into the system earlier The user has set a password for login (optional for login via login/password) The user is not currently authorized in the system
Input values	User goes to login page The user enters the login/password from the UPA system The user clicks the “Login” button
Expected result	User successfully logs in
Postconditions	The user login is displayed in the administrator audit log

Considering that it is necessary to purchase an annual (or monthly) license to use commercial project SSO, the developing your own solution looks more profitable in the long term. In addition, UPA IS bypasses restrictions on the number of connected systems, in contrast to commercial projects.

Conclusion

The identified functional requirements have become the base for the developed UPA, which automates the process of entering the university systems by providing a single entry point. Unlike the considered analogs, the system is cross-platform, since it is a full-fledged web application and does not require the installation of additional clients. The system supports authorization via VKontakte, other subsidiary systems and the basic version by login and password.

During testing of the developed system, it was not possible to identify XSS vulnerabilities, SQL injections, etc. The authorization security audit also did not reveal any vulnerabilities.

The proposed solution has the following disadvantages: subsidiary systems must support the implemented communication protocol. This is necessary to provide basic security for user authorization. Another point of system expansion is the introduction of two-factor authentication. Since the loss/theft of the username/ULC system password entails a corresponding loss of access to subsidiary systems, the importance of one password increases.

The developed UPA IS is used in the processes of ETU “LETI”.

Acknowledgements: The authors are grateful to Ivanov V.S., Timofeev A.A. and Petrunenko D.A. for their participation implementing the UPA IS individual components.

References

1. *Internet Statistics 2020: Websites, Domains, Hosting, Traffic*. Available at: https://sdvv.ru/articles/elektronnaya-kommertsiya/statistika-interneta-2020-sayty-domeny-khosting-trafik/ (accessed June 01, 2021) (in Russ.).

2. Bellamy-McIntyre J., Lutteroth C., Weber G. OpenID and the Enterprise: a model-based analysis of single sign-on authentication. *Proc. XV IEEE Intern. EDOC*, 2011, pp. 129–138. DOI: 10.1109/EDOC.2011.26.

3. Tarasenko I.D., Dudarev V.A. Single Sign-On development for information resources on materials science. *Bulletin of the Technological University*, 2014, vol. 17, no. 19, pp. 381–382 (in Russ.).

4. Sun S.T., Beznosov K. The devil is in the (implementation) details: an empirical analysis of OAuth SSO systems. *Proc. ACM Conf. CCS '12*, 2012, pp. 378–390. DOI: 10.1145/2382196.2382238.

5. Fett D., Küsters R., Schmitz G. Analyzing the BrowserID SSO system with primary identity providers using an expressive model of the web. *Computer Security – ESORICS 2015. LNCS*, vol. 9326, pp. 43–65.DOI: 10.1007/978-3-319-24174-6_3.

6. *Markets and Markets. Single Sign-on Market*. Available at: https://www.marketsandmarkets.com/Market-Reports/single-sign-on-market-83280444.html (accessed June 01, 2021) (in Russ.).

7. *Identity Blitz. Blitz Identity Provider*. Available at: https://identityblitz.ru/products/blitz-identity-provider/ (accessed June 01, 2021) (in Russ.).

8. *Oracle. Oracle Identity Federation*. Available at: https://www.oracle.com/middleware/technologies/oracle-identity-federation.html (accessed June 01, 2021) (in Russ.).

9. Henninger S., Corrêa V. Software pattern communities: Current practices and challenges. *Proc. XIV Conf. PLOP*, 2007, pp. 1–19. DOI: 10.1145/1772070.1772087.

10. 10. Belyaev S.A., Vasilev A.V., Kudryakov S.A. The monitoring of information trends system’s architecture based on the free software. *Software & Systems*, 2016, vol. 29, no. 4, pp. 85–88. DOI: 10.15827/0236-235X.114.085-088 (in Russ.).

11. Belyaev S.A., Cherepkova Y.S. Architecture of the simulation environment for conducting experiments with intelligent agents. *Software Journal: Theory and Applications*, 2017, vol. 3. DOI: 10.15827/2311-6749.17.3.4. Available at: http://swsys-web.ru/ru/architecture-for-conducting-experiments-with-intelligent-agents.html (accessed June 01, 2021) (in Russ.).

12. Riggs S., Ciolli G. *PostgreSQL 10 Administration Cookbook*. UK, Birmingham, Packt Publ., 2018, 550 p.

13. Belyaev S.A. *Games Development in Javascript*. St. Petersburg, 2016, 128 p. (in Russ.).

14. *Why GitHub? Js-framework-benchmark*. Available at: https://github.com/krausest/js-framework-benchmark (accessed June 01, 2021) (in Russ.).

15. Belyaev S.A. *Web Technologies: Laboratory Workshop*. St. Petersburg, 2019, 76 p. (in Russ.).

16. Rahalkar S. Extending burp suite. *A Complete Guide to Burp Suite*, 2021, pp. 131–145. DOI: 10.1007/978-1-4842-6402-7_9.

17. Chernaya O.S., Fedorova Y.Y., Belyaev S.A. Applying of methods and tools to automated testing of telemetric data processing software. *Proc. of Saint Petersburg Electrotech. Univ. J.*, 2013, vol. 9, pp. 55–58 (in Russ.).

УДК 004.9
DOI: 10.15827/2311-6749.21.3.1

Разработка системы единой авторизации для информационных систем

И.И. Холод [1], *доктор технических наук, профессор, декан факультета компьютерных технологий и информатики, iiholod@mail.ru*
М.В. Ковынев [1], *студент, kovinevmv@gmail.com*
И.С. Григорьев [1], *студент, grigorievivan.1@mail.ru*
П.В. Корытов [1], *студент, thexcloud@gmail.com*

[1] *Санкт-Петербургский государственный электротехнический университет «ЛЭТИ», Санкт-Петербург, 197376, Россия*

В статье рассматривается проблема создания системы единой авторизации для информационных систем вуза.

Количество информационных систем постоянно растет, в виду этого стоит вопрос о предоставлении пользователю централизованного единого входа в такие системы. Авторы изучили рынок существующих решений. В ходе анализа были выбраны самые быстрорастущие системы, такие как Blitz Identity Provider и IBM Security Access Manager for Enterprise Single Sign-On, рассмотрены их ключевые достоинства и базовые ограничения. Проведено сравнение данных решений по следующим критериям: стоимость, наличие открытого исходного кода, ограничения на подключаемые системы. Авторы рассмотрели существующие дочерние системы вуза и предложили реализацию собственной системы единого входа.

В работе подробно описаны отдельные аспекты реализации системы, в том числе процесс авторизации в дочерних системах и отображения страниц дочерних систем, рассмотрена архитектура системы и механизм авторизации. Изучен вопрос безопасности и тестирования разработанного решения, выявлены основные преимущества полученного решения: кроссплатформенность, централизованное отображение меню «дочерних» систем в одном месте, стоимость.

В ходе тестирования разработанной системы не выявлены уязвимости XSS, SQL-инъекции и прочее. Система поддерживает авторизацию через сеть «ВКонтакте», другие дочерние системы и базовый вариант по имени и паролю. Разработанная информационная система используется в СПбГЭТУ «ЛЭТИ».

Ключевые слова: *единая авторизация, web-приложение, безопасность авторизации, единое информационное пространство.*

Благодарности. *Авторы выражают благодарность Иванову В.С., Тимофееву А.А. и Петруненко Д.А. за участие в реализации отдельных компонентов информационной системы «Универсальный личный кабинет».*

Литература

1. Статистика Интернета 2020: сайты, домены, хостинг, трафик. URL: https://sdvv.ru/articles/elektronnaya-kommertsiya/statistika-interneta-2020-sayty-domeny-khosting-trafik/ (дата обращения: 01.06.2021).
2. Bellamy-McIntyre J., Lutteroth C., Weber G. OpenID and the Enterprise: a model-based analysis of single sign-on authentication. Proc. XV IEEE Intern. EDOC, 2011, pp. 129–138. DOI: 10.1109/EDOC.2011.26.
3. Тарасенко И.Д., Дударев В.А. Разработка системы единой авторизации для информационных ресурсов по материаловедению // Вестн. технологич. ун-та. 2014. Т. 17. № 19. С. 381–382.
4. Sun S.T., Beznosov K. The devil is in the (implementation) details: an empirical analysis of OAuth SSO systems. Proc. ACM Conf. CCS '12, 2012, pp. 378–390. DOI: 10.1145/2382196.2382238.
5. Fett D., Küsters R., Schmitz G. Analyzing the BrowserID SSO system with primary identity providers using an expressive model of the web. In: Computer Security – ESORICS 2015. LNCS, vol. 9326, pp. 43–65. DOI: 10.1007/978-3-319-24174-6_3.
6. Markets and Markets. Single Sign-on Market. URL: https://www.marketsandmarkets.com/Market-Reports/single-sign-on-market-83280444.html (дата обращения: 01.06.2021).
7. Identity Blitz. Blitz Identity Provider. URL: https://identityblitz.ru/products/blitz-identity-provider/ (дата обращения: 01.06.2021).
8. Oracle. Oracle Identity Federation. URL: https://www.oracle.com/middleware/technologies/oracle-identity-federation.html (дата обращения: 01.06.2021).
9. Henninger S., Corrêa V. Software pattern communities: Current practices and challenges. Proc. XIV Conf. PLOP, 2007, pp. 1–19. DOI: 10.1145/1772070.1772087.
10. Беляев С.А., Васильев А.В., Кудряков С.А. Архитектура системы мониторинга информационных трендов на основе свободного программного обеспечения // Программные продукты и системы. 2016. Т. 29. № 4. С. 85–88. DOI: 10.15827/0236-235X.114.085-088.
11. Беляев С.А., Черепкова Ю.С. Архитектура среды моделирования для проведения экспериментов с интеллектуальными агентами // Программные продукты, системы и алгоритмы. 2017. № 3. DOI: 10.15827/2311-6749.17.3.4. URL: http://swsys-web.ru/ru/architecture-for-conducting-experiments-with-intelligent-agents.html (дата обращения: 01.06.2021).
12. Riggs S., Ciolli G. PostgreSQL 10 Administration Cookbook. UK, Birmingham: Packt Publishing, 2018, 550 p.
13. Беляев С.А. Разработка игр на языке JavaScript. СПб: Лань, 2016. 128 с.
14. Why GitHub? Js-framework-benchmark. URL: https://github.com/krausest/js-framework-benchmark (дата обращения: 01.06.2021).
15. Беляев С.А. Web-технологии: лабораторный практикум. СПб: Изд-во СПбГЭТУ «ЛЭТИ», 2019. 76 с.
16. Rahalkar S. Extending burp suite. In: A Complete Guide to Burp Suite, 2021, pp. 131–145. DOI: 10.1007/978-1-4842-6402-7_9.
17. Черная О.С., Федорова Ю.Ю., Беляев С.А. Применение методов и средств автоматизированного тестирования для проверки качества программных комплексов обработки измерительной информации // Изв. СПбГЭТУ «ЛЭТИ». 2013. Т. 9. С. 55–58.