INFORMATYKA EKONOMICZNA BUSINESS INFORMATICS 2(40) • 2016

ISSN 1507-3858
e-ISSN 2450-0003

**Robert Kutera, Wiesława Gryncewicz**
Uniwersytet Ekonomiczny we Wrocławiu
e-mails: robert.kutera@ue.wroc.pl; wieslawa.gryncewicz@ue.wroc.pl

# SINGLE SIGN ON AS AN EFFECTIVE WAY OF MANAGING USER IDENTITY IN DISTRIBUTED WEB SYSTEMS. THE ACTGO-GATE PROJECT CASE STUDY

# SINGLE SIGN ON JAKO EFEKTYWNY SPOSÓB ZARZĄDZANIA TOŻSAMOŚCIĄ UŻYTKOWNIKÓW W ROZPROSZONYCH SYSTEMACH INTERNETOWYCH NA PRZYKŁADZIE PROJEKTU ACTGO-GATE

DOI: 10.15611/ie.2016.2.02
JEL Classification: O32, O33

**Summary:** The article deals with the issues of user identity management in the context of distributed web systems. The basic concepts and presented possible solutions related to the property of Single sign on (SSO) which can provide an access to multiple connected systems after a single login have been identified. The method of selecting a specific solution that can provide SSO has been exemplified by the decision process in the ActGo-Gate project. Taking the above information into consideration, OpenID Connect (OIDC) should better meet the project's expectations as it satisfies all the mentioned requirements to an acceptable extent, and therefore should ensure full SSO property in cooperation with the common user directory. Then the particular solution chosen for implementing OIDC is Connect2id (as Identity Provider) with the OpenDJ LDAP user directory. The most important arguments in favor of that decision were presented in the paper.

**Keywords:** Single sign on, authentication, distributed web systems, user identity, user management.

**Streszczenie:** W artykule podjęto problematykę zarządzania tożsamością użytkowników w kontekście rozproszonych systemów internetowych. Zidentyfikowano podstawowe pojęcia oraz zaprezentowano możliwe rozwiązania związane z właściwością Single sign on (SSO), która może zapewnić dostęp do wielu powiązanych systemów po jednokrotnym zalogowaniu. Sposób doboru konkretnego rozwiązania zapewniającego SSO przedstawiono na przykładzie procesu decyzyjnego w projekcie ActGo-Gate. W rezultacie wybrano protokół OpenID Connect oraz rozwiązanie Connect2Id, które w największym stopniu odpowiadają przedstawionym wymaganiom. W artykule zostały przedstawione najważniejsze argumenty przemawiające za tą decyzją.

**Słowa kluczowe:** Single sign on, autentykacja, rozproszone systemy internetowe, tożsamość użytkownika, zarządzanie użytkownikami.

## 1. Introduction

Nowadays people use many different independent software systems and accumulate more and more identities. User identity is considered as a set of permanent or long-lived temporal attributes associated with a user entity [Ceccarelli et al. 2015]. In order to access the service most of systems require the user to be registered and logged in. Quite often a user is registered in many web sites under the same user name and with the same or closely related passwords, which is not the best security practice. What is more they often forget their credentials, and the user management system have to send an unencrypted e-mail with these confidential data [Singer, Friedman 2013]. The number of password-protected accounts that an average user has is big and is still increasing. According to different studies, 44% of users have more than 20 password-protected accounts [Password Statistics 2015; Keszthelyi 2013]. "The average user has 6,5 passwords, each of which is shared across 3,9 different sites. Each user has about 25 accounts (...) and types an average of 8 passwords per day" [Florencio, Herley 2007]. Dashlane – a company that is specialized in password management applications – has published on their blog a very surprising prognostication: "The number of accounts we use is growing at a 14% rate, meaning it doubles every 5 years. In 2020, the average number of accounts per Internet user will be 207" [Online … 2015].

That is why the management of multiple user names and passwords is such an annoying aspect of the current Internet. But it is also one of the most serious security weaknesses. According to the survey which was conducted by the mobile identity company TeleSign [Password Statistics 2015], 73% of online accounts are guarded by duplicated passwords, many of which have not been changed in five years or more. 21% of people use passwords that are over 10 years old, 47% of internet users use passwords that are at least 5 years old. People should search for better ways to secure their online accounts. 80% of people are worried about their online security, but – on the other hand – they are still using weak, old or repeatable passwords that are easily hacked [Internet users 2016]. Every internet service provider is responsible for the safety of authentication credentials and for private data management.

Taking into account all those security issues and the lack of the required competencies needed to fulfil them – more and more administrators of websites decide to outsource those management and authentication services to third parties, external entities specialized in that kind of activity. Such entities can solve those problems using different techniques. They can host, store, manage and secure user data on behalf of a particular service provider. They offer application programming interfaces (API), which can provide an access to user data for external applications. Such solutions enable to share user data among more than one web system, thus providing Single sign on (SSO) property for third party applications. The aim of the paper is to present an example of such a solution based on an internet platform for gathering services dedicated to elderly people – ActGo-Gate (AGG).

## 2. Background

Current information systems are more and more frequently built in a distributed architecture. A distributed system consists of a collection of autonomous computers, connected through a network and distribution middleware, which enables computers to coordinate their activities and to share the resources of the system so that users perceive the system as a single, integrated computing facility [Coulouris et al. 2012]. SSO solutions can be implemented widely for any web systems that have the nature of a distributed system. They enable companies to create a multiple related environment to log in once and to gain access to services offered by different companies [Single sign-on 2016; Singer, Friedman 2013]. Their popularity is still growing, but the main factor of their current market position is the fact that most of the leaders of the global internet market make it available to use their accounts to logging in to third party applications. The authors of these applications have only to call the particular provider's API and configure some basic parameters on the client's side.

The ease of implementation for authors as well as the convenience and usability for users make it a very common solution for authenticating users on many web portals and services or authorising their access to resources. Authentication is the kind of security mechanism whereby systems may securely identify their users [Osmanoglu 2014]. Authentication systems provide answers to the questions about identity: Who is the user? Is the user really who he/she presents himself/herself to be? Authorization, on the other hand, is the security mechanism by which a system determines what level of access a particular authenticated user should have to secured resources controlled by the system [Osmanoglu 2014]. Such a mechanism provides answers to questions about the system privileges of a particular user: Is user X authorized to access resource R? Is user X authorized to perform operation P? Is user X authorized to perform operation P on resource R?

In other words, when users try to log in to any system, they are usually first requested to identify themselves with a user name and a password. Afterwards, the data input is checked against an existing user record to verify if the given combination is authentic. If so, the user becomes authenticated (i.e. the identification data he/she supplied previously is valid). Finally, a set of pre-defined permissions and restrictions for that particular login name is assigned to this user, which completes the final step, authorization. Usually authorization cannot be performed without any kind of authentication and sometimes the second term is used to mean the combination of both.

According to the University of Guelph the benefits of SSO protocols apply to many areas [SSO Benefits 2016]:

1. User experience: The most apparent benefit is that users can move between services securely and uninterruptedly without specifying their credentials each time. SSO effectively joins these individual services into portals and removes the service boundaries – switching from one application to the next appears seamless to the user.

2. Security: The users' credentials are provided directly to the central SSO server, not the actual service that the user is trying to access, and therefore the credentials cannot be cached by the service. The central authentication point – the SSO service – limits the possibility of phishing.

3. Resource savings: IT administrators can save their time and resources by utilizing the central web access management service. The application and the web developers receive a complete authentication and authorization framework that they can use to build secure, user customized services.

## 3. Single sign on solutions – review and critical analysis

The SSO solution is meant in the article as the complete software infrastructure that enables the usage of a particular SSO protocol, including a particular user directory, the server software playing the role of Identity Provider, the communication channel and the client implementations of Service Providers. The user directory is any repository that contains the user identity details. Identity Provider (IdP) is a trusted server-side application that creates, maintains and manages identity information and provides authentication to other service providers [Denis et al. 2015]. Its main role is to provide identifiers for users willing to interact with the system, to assert to such a system that the user's identifier is recognized by the provider, and possibly to provide other requested details about the user identity. This could be achieved thanks to the in-built authentication module which is able to decode and verify a security token transported commonly via the basic web communication channel – Hypertext Transfer Protocol (HTTP), with using the GET or POST method. In turn, Service Provider (SP) requests and obtains an identity assertion from the identity provider [Bower 2016]. On that basis, SP can grant or refuse an access for a specific resource. Often there are dedicated solutions for service providers that could be installed and implemented to the resource server to enable the quick configuration of an authentication solution. The third actor that is involved in the authentication/authorization process, is the client – the web browser operated by the user is playing that role in most cases.

The most popular protocols used for enabling the property of single sign on are currently:

- Security Assertion Markup Language (SAML),
- OAuth 2.0,
- OpenID Connect.

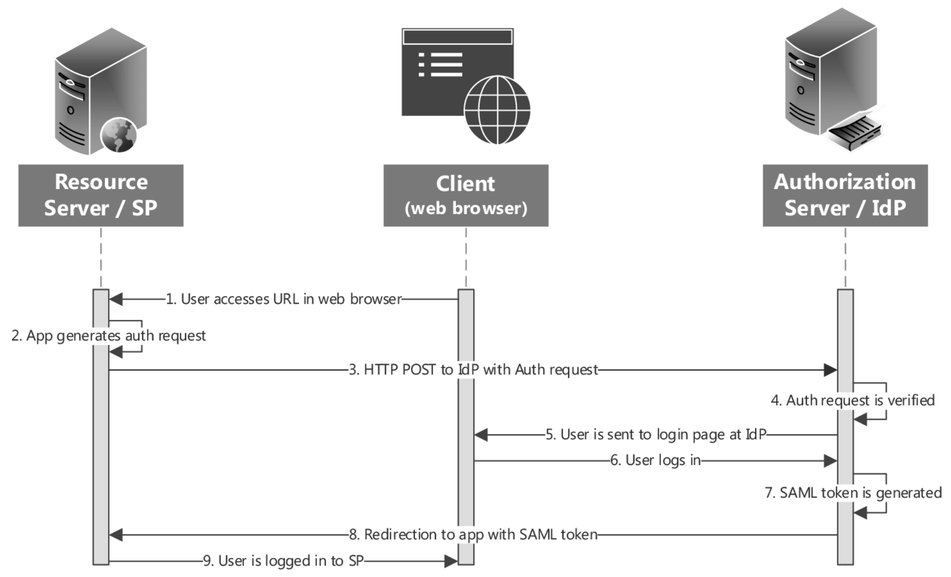
Resource Server / SP
Client (web browser)
Authorization Server / IdP
1. User accesses URL in web browser
2. App generates auth request
3. HTTP POST to IdP with Auth request
4. Auth request is verified
5. User is sent to login page at IdP
6. User logs in
7. SAML token is generated
8. Redirection to app with SAML token
9. User is logged in to SP

**Figure 1.** SAML-based authentication scenario

Source: own preparation on the basis of: http://www.mutuallyhuman.com/blog/2013/05/09/choosing-an-sso-strategy-saml-vs-oauth2/.

SAML 2.0 is an XML-based open standard data format, based on XML, for exchanging authentication and authorization data between parties, in particular between an identity provider and a service provider.

The scenario of authenticating the user with SAML (presented in Figure 1) begins with opening the web-browser by the user and entering the address of a particular web application providing any content accessible for the user after he/she is recognized by the system. this plays a role of SP in the schema above. That page does not handle authentication itself so it builds a SAML request, adds a signature, encrypts it if necessary, and encodes it. This request can take the form of Authnrequest or AuthNRequest, depending on the HTTP method chosen (GET or POST respectively). Later the user is redirected to the IdP in order to authenticate it. The IdP receives the request, decodes it, optionally decrypts it, and verifies the signature. If the request is positively verified, the IdP will display to the user a login form where he/she can enter his/her credentials. After the successful logging in, the IdP generates a SAML token with the included user identity details (name, email, etc.). The SAML token is sent to the SP and similarly the user is redirected back to the SP. There, the SP verifies the SAML token, performs an optional decryption, and extracts the user identity details to find out who he/she is and what his/her permissions are. The web application mentioned at the beginning now allows the user to get into the system, usually with creating a cookie and session. At the end of the whole process the user

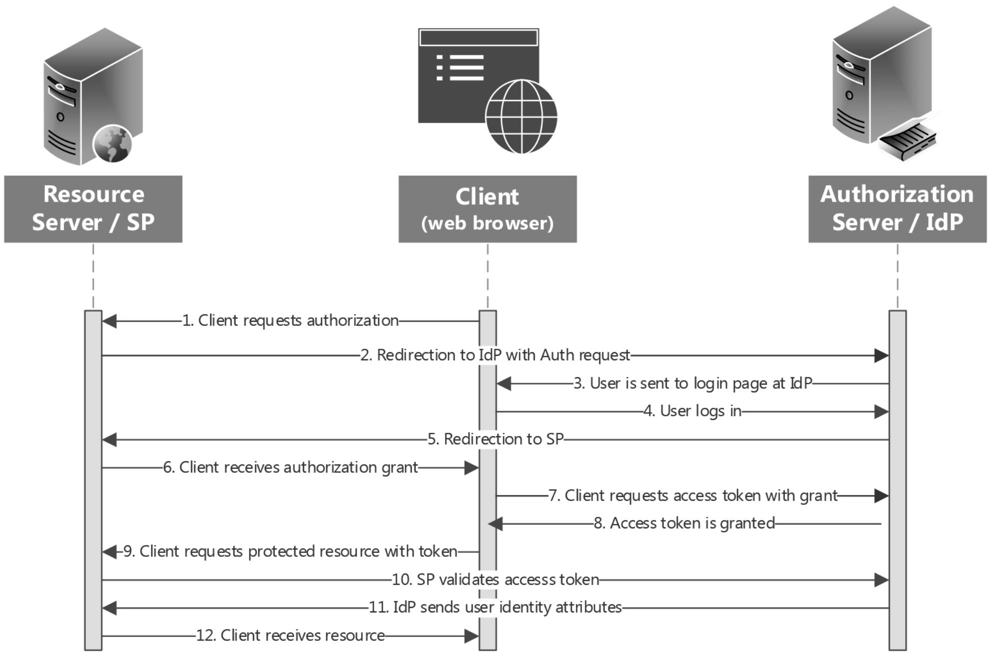
Resource Server / SP
Client (web browser)
Authorization Server / IdP
1. Client requests authorization
2. Redirection to IdP with Auth request
3. User is sent to login page at IdP
4. User logs in
5. Redirection to SP
6. Client receives authorization grant
7. Client requests access token with grant
8. Access token is granted
9. Client requests protected resource with token
10. SP validates accesss token
11. IdP sends user identity attributes
12. Client receives resource

**Figure 2.** OAuth 2.0 – based authorization scenario

Source: own preparation on the basis of: http://www.mutuallyhuman.com/blog/2013/05/09/choosing-an-sso-strategy-saml-vs-oauth2/.

can access all the allowed features and content of the web application as an authenticated user. What is worth mention, the user's credentials are never passed through the web application, only through the IdP [Dennis 2013].

The next protocol, OAuth is an open protocol to allow secure authorization in a simple and standard method from the web, mobile and desktop applications [Saldanha 2013]. So OAuth can be rather considered as an authorization system, not an authentication system. Of course a user might actually be "authenticating" himself/herself to a specific website using OAuth, but this relies on allowing this website to use the information stored by the OAuth provider. The authorization process with OAuth 2.0 is presented in Figure 2. Using the same scenario as in the case of the SAML presentation, the relevant steps are specified below. Similarly as above the user opens the web browser and enters the address of the web application, which does not provide authentication mechanisms itself. That is why the user is redirected to the IdP with a request for authorization. A login form is displayed to the user, who is asked to approve the SP to act on his/her behalf. After this approval the user is logged in and redirected back to the requesting web application. The SP receives an authorization grant code as a part of the redirect and then passes this along to the application. It then uses that code to request an access token from the IdP. The IdP validates that code and, if the result is positive, grants an access token. The access token is then used by the web application to request resources from the SP. The SP receives the request for a specific resource with the access token as a query parameter in URL. In order to confirm the validity of the access token it sends the token directly to the IdP. If it is valid, the IdP sends back the basic user identity attributes. After that the web application sends the requested resource back to the user.

A third protocol worth considering, is OpenID Connect (v 1.0) – an authentication protocol based on OAuth 2.0 that provides a RESTful HTTP API and uses JSON as data format [Denis et al. 2015]. It is a simplified format that has gained large traction and is supported by many vendors, e.g. Google, IBM, Microsoft, Ping Identity etc. It extends pure OAuth 2.0 by providing user identity details in an efficient way so the requesting application knows not only the user's access rights to a particular asset, but also has a deep knowledge (to the given extent) about user identity. It also supports native apps like OAuth 2.0. What is worth mentioning, OpenID Connect uses terminology of OAuth 2.0 but provides a few technology-specific terms [Sakimura 2014]:

- Relying Party (RP) – (instead of the commonly used Service Provider) OAuth 2.0 client application requiring end user authentication and claims from an OpenID Provider,
- OpenID Provider – (OP) OAuth 2.0 authorization server that is capable of authenticating the end user and providing claims to a Relying Party about the authentication event and the end user,
- Claim – a piece of information asserted about an entity,
- UserInfo Endpoint – protected resource that, when presented with an access token by the client, returns authorized information about the end user represented by the corresponding authorization grant.

The OpenID Connect protocol, as it is based on OAuth 2.0, follows the same steps as OAuth (see again Figure 2), but the main difference is that OpenID connect provides an additional step to obtain information about user identity. Conceptually OAuth is developed for granting access to resources, not for authenticating the user. OpenID Connect provides an additional flow for providing id_token with some information about the user. If the complete set of information is required, the access_token is needed to request the OpenID provider for getting the UserInfo. The UserInfo Endpoint then returns the claims about the user.

Summarizing the three described protocols, there are advantages and disadvantages for all solutions. The most important ones are collected in Table 1.

There are not only these three protocols connected with identity management. Many further examples like Higgins, U-Prove, MicroID, and Liberty Alliance can be mentioned [Koussa 2013]. All of them have their niches but they are not so popular among customers, and consequently not supported by most of the SSO solutions.

**Table 1.** The comparison of SSO protocols

| Protocol | Advantages | Disadvantages |
|---|---|---|
| SAML 2.0 | SAML 2.0 is still widely used, but mainly only in the enterprise segment. It also supports a large variety of connectors.<br>It has been the main Web SSO standard in the last couple of years in the enterprise segment.<br>It operates on a ‚more' centralized user repository / base. | The standard is from 2005 and is slowly being phased out by most of the software vendors. It will still be supported, but it will not be enhanced or extended.<br>It does not support native mobile applications, as this was not foreseen at the inception of the standard. Any application which potentially could be wrapped in a native app now or in the future will require complex work-arounds.<br>Enterprise clients are slowly moving away from SAML for new projects and also focusing on more open standards. |
| OAuth 2.0 | OAuth 2.0 is still widely used mainly in front-facing consumer segments.<br>It supports a large variety of connectors<br>It has been the main ‚Web' SSO standard in the consumer based segment so far.<br>Native mobile applications are supported.<br>It operates on a ‚decentralized' user base. | Architecture based security gaps have/had to be solved in the implementations of it.<br>It does not directly contain user information (name, address, etc.).<br>Clients are moving away from OAuth 2.0 for new projects. |
| OpenID Connect 1.0 | It is a WebSSO standard which has gained much traction since 2014 and is slowly superseding OAuth 2.0 by the vast majority of IT players.<br>Native mobile apps are supported<br>It operates on a ‚decentralized' user base.<br>It resolves the potential security gaps of OAuth 2.0 and enhances the data by including user data (name, address etc.)<br>Management & Configuration is simple. | It is a newer standard, thus not all current frameworks in the SSO area officially support it via connectors. E.g. Shibboleth does not support it with an official connector yet.<br>Facebook does not support it with their Facebook Connect yet. |

Source: own preparation.

Alternatively there is also the possibility to use user directories and their protocols such as LDAP protocol (Lightweight Directory Access Protocol) with Microsoft Active Directory (AD). LDAP is a directory publishing service and specially designed for directory service providers [Liferay 2016]. On the other hand AD is a directory service provider, where user data can be stored and managed (add new users, remove or modify their records, specify privileges, assign policy etc.). AD then is a kind of directory-based database, and LDAP is one of the most popular protocols used to communicate with it. One of the SSO solutions that is native for Active Directory is Active Directory Federation Services (ADFS), developed by Microsoft and run on their server operating systems to provide users with SSO property to access the systems/applications located across organizational boundaries. That solution is not so popular in the web environment because most of the web systems are run by Linux-based operating systems. That is why this solution will not be taken into consideration any longer.

There is a lot of potential solutions that could be considered to provide the SSO property to a distributed web system. A few of them are presented in Table 2.

**Table 2.** Selected complex web SSO solutions for distributed web systems

| Name | Description |
|---|---|
| Thinktecture Identity Server | Framework and hostable component that allows implementing single sign-on and access control for modern web applications and APIs using protocols like OpenID Connect and OAuth2. Supports a wide range of clients like mobile, web, SPAs and desktop applications and is extendible to allow integration in new and existing architectures. |
| Connect2ID | Identity management software for enterprises, based on OpenID Connect and OAuth 2.0, web and service-oriented, implemented in Java and extendable with other libraries like JWT, LDAPAuth or JSON2LDAP, offering a wide choice of databases for persisting its data like LDAP, MySQL, PostgreSQL, H2 and Redis |
| Keycloak | Integrated SSO and IDM for browser apps and RESTful web services. Built on top of the OAuth 2.0, Open ID Connect, JSON Web Token (JWT) and SAML 2.0 specifications. |
| Shibboleth | An open-source implementation for identity management and federated identity-based authentication and authorization (or access control) infrastructure based on SAML. It allows for cross-domain single sign-on and removes the need for content providers to maintain user names and passwords. It is one of the most dependable open source SAML single sign-on servers available and is in production at more than 5,000 organizations worldwide. |
| Gluu Server | Free open source access management suite with support for SAML and OpenID Connect SSO, and OAuth2 based web and API access management. The Gluu Server can include multiple components. Each one fulfills a different requirement, and can be included or excluded in individual deployments based on an organization's unique requirements (eg. Shibboleth, Asimba SAML Proxy, Gluu OpenDJ, oxTrust). |
| CAS / Central Authentication Service | Protocol and SSO server/client implementation, being Java (Spring Webflow/ MVC servlet) server component, offers pluggable authentication support (LDAP, database, X.509, 2-factor), support for multiple protocols (CAS, SAML, OAuth, OpenID), cross-platform client support (Java, .Net, PHP, Perl, Apache, etc.), integration with uPortal, Liferay, BlueSocket, Moodle, and Google Apps etc. |

Source: own preparation on the basis on the product's web pages.

Summarizing the review, there are pros and cons (both infrastructural as well as functional) for all the mentioned solutions. All of them have their weaknesses mostly concerning the security. For example all of them have difficulties in handling the full log out functionality, including the propagation of session expiration among all the connected clients and require additional programming work to cover it. Usually a choice of a one particular solution should be made on the basis of the analysis of the functional and non-functional requirements gathered at the initial stage of the software development process. The first group will give the picture of the functionality needed by the users, the second one will show the technical limitations and values of the quality indicators that need to be fulfilled.

## 4. ActGo-Gate – project scope and ICT platform overview

Further considerations on choosing SSO solutions will be based on the findings gathered during the planning of the technical implementation of the ActGo-Gate research project. Its utilitarian aim is to create an ICT based marketplace supporting entrepreneurship, self-fulfillment and the social participation for golden workers and active retirees. Moreover its cognitive aim is to create a transferable model as gate for different occupation modules. The model builds on local social marketplaces that play the role of the starting point for developing three occupational modules, starting with an established user base and along existing structures:

- The "Serve the community" module (Weil der Stadt, Germany) enables customers to participate by offering their informal support to other community members (informal volunteering work).
- The "Flexible occupation" module (Hamburg, Germany) brings together local service providers with golden workers and active retirees, who want to engage in part-time jobs and occupations.
- The "Get involved with organizations" module aims to bring together people for social projects, e.g. as part of corporate volunteering programs (St. Gallen, Switzerland).

The fully developed AGG will be the first of its kind to provide a gate for a comprehensive range of occupational possibilities (voluntary, paid, task-based, project-based, etc.). It will provide end users with easy access to this integrated gate and enable them to offer their skills, abilities and experiences to other community members. For the integration of the occupational modules with the existing local marketplaces, a modular approach is considered as the most suitable for ensuring the right extent of flexibility and scalability. It will further offer tools for efficient transactional occupational operations (appointments coordination, quality assurance, payment handling, reporting etc.), both for professional as well as informal activities.

From the utilitarian point of view the AGG platform is a gate web application, whose main functional feature is to gather occupational opportunities coming from different service e-marketplaces, and offer them to older adults. Both the supply and demand side is supported by the platform that means the users can look for one-time formal and informal services or long-term, recurrent jobs. They can also post a one-time demand of something that has to be done for them.

Technically the gate application is an offer aggregator that integrates some data flows (offer feeds, notifications). The information structure of offer is standardized, offers are grouped into categories and can be browsed in different views of catalog: the list, the grid and the map. The catalog is equipped with basic or advanced sorting and filtering options, as well as the location-oriented full-text search engine. It also enables a single sign on to all the connected web services (therefore with using an AGG account users can log into each connected service). Users from the AGG level can edit their user data (which will be synchronized with all the connected services), browse and manage all their own offers and view all the notifications coming from the connected services. Particular connected services (respectively Benevol-jobs.ch and Amiona.ch in AGG pilot implementations) are still responsible for the core logic of the specific processes (like recruitment and appointment coordination). The interface of the application is designed with the principles of a responsive web design paradigm. That is why it can easily adapt its content to the different screen sizes of the end devices.

## 5. Choosing an SSO solution for the distributed web system – case study

### 5.1. Status Quo

The ActGo-Gate project consists of three different functional modules which will be prepared and run as separate pilot implementations. They will have some common functionality, but will also introduce some new unique functions dedicated to their business model. They should have their own interfaces, business logic and data sources.

Taking into consideration the scope of functionality of each module a decision has been made that two modules' implementations: "Serve the Community" and "Flexible Occupation" will be built on top of the "Amiona" Appointment Coordination System (ACS) provided by BEI AG from Switzerland, while the third one – on top of the "Benevol- Jobs.ch" Recruiting Service System (RS), provided by Clavis IT AG from Switzerland. On the very top of them a gate application will be built to provide functionality of all the modules via one single point of access to the AGG end users. This application is implemented using the PHP Laravel framework and provides three levels of integration: integration on user interface level, data/backend level and security level.

In the current case study only the third level of integration will be taken into consideration with the short list of requirements presented in the next section.

### 5.2. Description of the requirements

During the analysis of the presented above status quo and user/partner needs, the most important functional requirements were identified for the AGG platform. The crucial functionality in that point is to provide SSO functionality and integrate systems in a way that users can have access to all the features of the module applications with one pair of credentials. All systems should be able to recognize their users and present personalized interfaces to them. It is very important to distribute all the necessary credentials and authentication information from a centralized user directory to the AGG gate application, ACS and RS. There should be an intuitive mechanism for user authentication, where a single user's action of logging into AGG causes that he or she is already authenticated in ACS and RS during the current browser/server session. A third party system/component providing identity management is required for identifying users and allowing them access to the requested resources. This system needs to use secure connections between the authentication services on the server and client side to minimize the risk of security violation and provide a high level of data protection.

Furthermore, for the AGG project the most crucial non-functional requirements are:

- compatibility with contemporary web development standards and the ease of integrating with them,
- support of actual authentication protocols for SSO,
- openness to different user directories (import),
- appropriately high security level for critical data,
- acceptance and trust within the user community.

The first concern that appears in the analysis is the choice of the authentication protocol that will be most suitable for the project's needs. As there will be three separate applications: AGG Gate App, ACS and RS based on different Web technologies, there is a need to find a solution that offers out-of-the-box connectors (or in the worst case such features that can be easily developed and integrated with the existing code) for technologies used in the AGG platform (PHP – Laravel, Python – Pyramid, Java – Liferay Portal). Usually, with the greater popularity of the SSO solution, the possibility of getting support from the developer's community of Web technologies is rising.

The chosen solution should support actual SSO protocols. Looking at Table 2 could help in establishing that some identity management solutions could handle such popular protocols as OAuth, OpenID Connect and SAML and also has in-built functionality of connecting directly with different user directories. Others are dedicated for one protocol. However, from the project's perspective, the most crucial is authentication with providing identity details rather than pure authorization to resources, so the best choice for ensuring SSO to the project's platform from all the analyzed protocols could be SAML or OpenID Connect.

Business partners use different user directories or databases and there is a need to have the possibility of integrating them into the identity provider (authentication server). At least there should be the possibility of importing user data from the external user directories such as MySQL databases, Active Directory or other LDAP user directories. The ability to use already stored user data is important especially for more complex systems like the project's applications: ACS and RS have already gathered a significant number of records of users who should be allowed to use all functionalities of the platform, not only the ones served by the source application.

The SSO solution should also allow the platform to securely store and own all of the user credentials. It has to be able to establish a trust relationship with the authentication server. All client applications (AGG Gate application, ACS, RS) should be able to utilize this trusted authentication server to authenticate users in a secure SSL channel. Choosing the SSO solution must be made in considering the fact that if the gate application is compromised, then all private personal or commercial data contained in all the other connected systems are jeopardized. That is why the solution should be able to handle advanced methods of authentication and enforce a strong password policy. The code of the server and client applications that enable SSO has to be free of any security vulnerabilities, backdoors or security bugs so it has to be a solution with a high maturity level and continuous support.

The last requirement concerns the acceptance and trust within the user community. First of all the acceptance level indicates that the solution meets typical the expectations of a large group of users that concerns ease of use, lack of problems with implementation and usage as well as safety and reliability. Furthermore the larger the community of users, the greater the possibility of getting help in cases of any strategic (at decision stage) or technical (at operational stage) problems.

### 5.3. Selection and explanation of the choice of the SSO solution

Analysis of the requirements should lead to the decision about the most suitable SSO solution. First of all the choice has to be made from among three of the protocols: OAuth 2.0, OpenID Connect and SAML. The choice is difficult as they are in some way complementary but different in their basis. As indicated in professional knowledge resources [Saldanha 2013], SAML should be used when the use case involves SSO when at least one actor or participant is an enterprise. SAML or OpenID Connect could be used when there is the need to provide access to a partner or customer application to the base portal and the use case requires a centralized identity source (Identity Provider). On the other hand OAuth and OpenID Connect can be used if the use case involves providing access (temporarily or permanent) to resources (such as accounts, pictures, files etc.) and handling native applications on mobile devices (bearer tokens). What is more, they use RESTful web paradigm, so they enable communication with web services using REST and JSON.

Taking the above information into consideration, OpenID Connect (OIDC) should better meet the project's expectations as it satisfies all the mentioned requirements to an acceptable extent, and therefore should ensure full SSO property by cooperating with the common user directory. Then the particular solution chosen for implementing OIDC is Connect2id (as Identity Provider) with an OpenDJ LDAP user directory – it is a solution natively prepared for OIDC handling as well as ready for communication with AGG, ACS and RS APIs thanks to additional extensions written by the author of the core server application.

First of all, OIDC is built on top of successful open identity and security standards like OAuth 2.0 and TLS (also known as SSL or "https"). It utilizes the good parts from OAuth 2.0 and solves the majority of its problems noticed in the past. As OAuth 2.0 is a very popular and commonly accepted authorization solution, it is often natively supported by popular web frameworks and systems. Unfortunately these solutions usually do not provide the functionality of handling OIDC, too. Nevertheless there is still a wide offer of implementations of OIDC (http://openid.net/developers/libraries/) written in the most popular programming languages like PHP, Python, Ruby, JavaScript and Java. Some of them are complex solutions providing both IdP and RP, while others provide only one kind of functionality – IdP or RP respectively. However there is only a limited selection of dedicated plugins for the most popular web applications and frameworks which could help to easily implement a Relying Party to such an application or framework (for RS there is a connector: https://github.com/csgf/OpenIdConnectLiferay). This is needed to implement one of the standard language-specific solutions and adjust it to enable the usage by the particular web application or framework. On the other hand, it has the advantage that it is substantially easier for developers to implement and deploy than other identity protocols, enabling simpler deployments without sacrificing security even if they need to write some custom code. It uses straightforward REST/JSON message flows with the goal of simplifying the whole process as much as possible [OpenID 2016]. Thereby OIDC can be easily understandable for most modern web applications developed in compliance with service-oriented architecture (SOA) standards. What is additionally worth mentioning is that most major web market players have or are moving to either directly integrate this standard or implement the specification and thus support the standard due to its appeal and simplicity.

OpenIDConnect implementations can be divided into two main groups:

- complex applications, that provide handling multiple SSO standards that offers a wide range of connectors for the most popular web applications, web frameworks or programming languages, but also have higher performance requirements,
- dedicated OIDC applications, simple and lightweight, written in a particular programming language.

The Connect2id application chosen for the project purposes, is an IdP implementation for the OIDC protocol with integration and customisation web APIs (based on REST and JSON). It is written in Java and based on such open source products such as Infinispan datagrid and the Nimbus JOSE+JWT library. It enables out-of-the-box integration with the LDAP OpenDJ solution. It offers additional tools for communication with LDAP user directory which give the developers additional possibilities of making operations on user data using built-in REST web services. Data are transmitted in JSON-based standard, JSON Web Token (JWT is an open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object [JWT 2016]. The choice of that solution can be circumstantiated with its lightweight nature and native usage of REST standard which will make the implementation easier and be compliant with SOA principles applied for the rest of the AGG components.

Connect2id is a dedicated IT solution for OIDC and that is why it supports natively this SSO protocol. It provides a full range of OIDC functions on the server side and acts as identity provider (IdP) in the SSO scenario. Client (Relying Party, RP) functionality requires writing a custom code or using third party connectors for specific programming languages, web frameworks or web systems (there is a native RP only for Java). They are all compatible as they use the same standard – OIDC, so there should not be any problems with proper "understanding" between RPs and IdP. OIDC IdPs usually are able to deliver the actual user details from user directories: LDAP, Active Direcotry, or MySQL databases. Connect2id offers dedicated utilities and web services for cooperation with LDAP user directories as well as relational databases MySQL, PostreSQL and H2, and supports Redis, too. Connect2id, as a native OIDC application, provides a simple JSON schema for releasing consented user information (claims), such as name, profile and contact details, to client applications. The claims can be included in the ID token or returned at the UserInfo endpoint (requires an access token). It additionally supports the aggregation of UserInfo claims from one or more data sources (LDAP directory, HR database, etc.).

The chosen IT solution should provide an appropriately high security level for critical data. Public-key-encryption-based authentication frameworks like OIDC (and its predecessors) globally increase the security of the whole Internet by putting the responsibility for user identity verification in the hands of the most expert service providers [OpenID 2016]. Never having to share passwords with the OIDC server is good for security, according to the key principles of "need to know" and minimizing credential access. This reduces the potential ways of attack and simplifies the protection of the assets. In OIDC a set of exchangeable user identity details (claims) is firstly defined, then at the approval stage the users can consent or deny the sharing of these details. Connect2id can provide a full range of that functionality as well as additional features (most of them in development phase) like two-factor authentication and supporting the use of hardware authentication devices, biometric and embedded cryptography methods of secure authentication. LDAP-based user directories provides their own security mechanisms and policies of access rights to ensure the access for assets only for the authorized users.

The last requirement concerns the acceptance and trust within the user community. OIDC is a fairly new web protocol for SSO. The final OIDC specifications were launched on February 26, 2014. The certification program for OIDC was launched on April 22, 2015. Google, Microsoft, Ping Identity, ForgeRock, Nomura Research Institute, and PayPal OIDC deployments were the first to self-certify conformance. The OpenID Foundation is a non-profit international standardization organization of individuals and companies committed to enabling, promoting and protecting OpenID technologies. Formed in June 2007, the foundation serves as a public trust organization representing the open community of developers, vendors, and users [OpenID Foundation 2016]. Connect2id is a rather niche product, but is made by a professional company (founded in 2010 in Bulgaria), experienced in the SSO area, and is still supported and developed by them. What is important is that the application is built on top of solid IT standards and proven open-source technology and getting support from professional development communities like Stackoverflow.com should not be a problem.

## 5.4. Validation of choice of SSO solution – proof of concept

The chosen solution implementing the OIDC protocol should be able to provide the functionality for running such proof of concept (PoCs) as:

- **Single sign-on** – once logged in with AGG credentials, users can switch between ACS and RS without having to login again.
- **Registration** – new users can register to AGG with username (or e-mail) and password of choice. Confirmation is sent/registration is finalized via e-mail. Accounts already existing in ACS or RS can be migrated by the special migration process invoked from the AGG login & registration middleware interface.
- **Forgot/change password functionality** – users can change a forgotten password online from the AGG login & registration middleware interface.

These PoCs require one to have an IdP, dedicated login & registration point, as well as RPs functionality for each system implemented. Some of the OIDC implementations provide functionality of both IdP and RPs, as well as customizable login pages, other ones concentrate on providing one functionality of IdP or RP respectively.

As previously mentioned, after a series of tests the choice of using the Connect2id solution as IdP was made. It will be equipped with the custom login page that will handle all the logic of registering, logging into the platform or managing credentials. It will be connected with each main component of the system which should be provided with the RP code responsible for communication with IdP (making requests, providing responses). Other PoCs require the implementation of a custom application that will connect directly with the LDAP user directory and perform needed operations on its assets.

Regarding the first PoC, the SSO property will be ensured by using the OIDC protocol. The communication between the components engaged in the user authentication process takes the same shape as OIDC. Once logged in with AGG credentials, users can have access to all the resources and web services of ACS and RS without having to log in again. This is possible by providing a secure communication channel between IdP and any SPs and an intuitive authentication mechanism with the use of security tokens. The login process scenario is presented in Figure 3. It shows that any attempt of logging in from ACS or RS will be started with redirecting to the login & registration point.

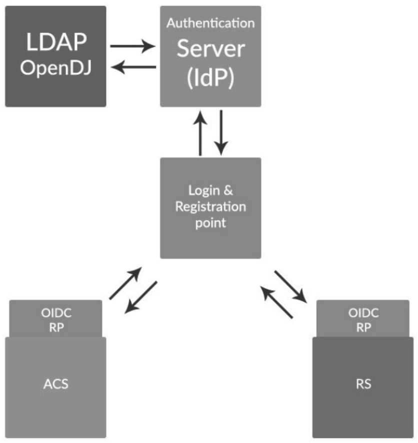

**Figure 3.** Login process scenario

Source: own preparation.

The next PoC for the implemented authentication system is to register new users with the username and password of choice. This will be realized using the custom login page. The code will be written in PHP and will use the native PHP's LDAP functions to connect with the user directory and perform the needed CRUD operations. The registration process should handle these two scenarios:

- User is not registered in any of systems (AGG, ACS, RS),
- User is registered at least in one of the existing ACS or RS systems, not in AGG.

Users have to fill in the registration form, provide basic data such as valid email address, password and password confirmation. The e-mail is checked for existence in LDAP directory and (if not in existence, add user to LDAP directory, generate a unique ID and send information about the new user to ACS and RS, which should have the proper methods for receiving and handling this response). The whole process is illustrated in Figure 4.

In the second case the process looks the same as above. If a user has already an account in one of existing systems (ACS, RS) an action is performed of extending the user account with a new field: agg_id. This value will allow for integrating previous transactions, appointments etc. with the new AGG account.

The AGG registration process must send information about the new user to other systems and that information is then processed in the proper way. Any errors on the ACS or RS side should be notified with error information to AGG and notification sent to the platform administrator about it so he/she will be able to react and fix it using his/her administration tools.

The login & registration point will also be responsible for handling any requests for restoring/changing the password. The user has to identify him/herself by clicking an activation hyperlink in the email message sent to the valid email address

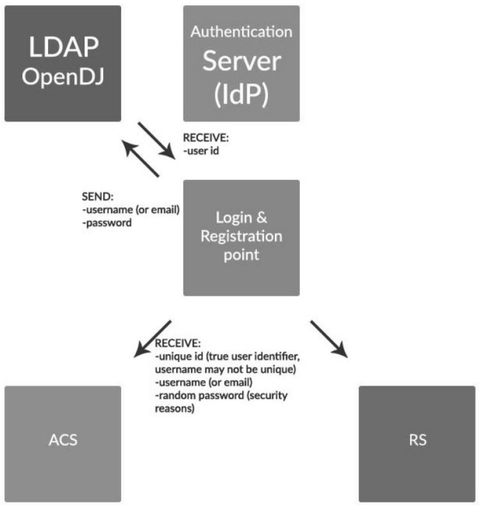

**Figure 4.** Registration process scenario

Source: own preparation.

previously added to the user account (the existence of email in LDAP is verified). The code responsible for handling the logic of these functions will connect directly to the user directory, perform the required operations on the data and provide the user with the expected results. The user will finally receive an email notification with a generated token (a token is stored in database with timestamp). The notification contains the hyperlink to click for confirmation of password change. After getting the URL token back it is verified in the database (its existence and generation time, it cannot be older than 12 hours). If everything is valid, the submitted new password is updated in the user directory.

## 6. Conclusions

The AGG project is dedicated to elderly people. Due to demographic changes and the increasingly ageing population an information and communication platform has been developed. Its vision is to create an ICT based marketplace supporting entre-preneurship, self-fulfilment and social participation for golden workers and active retirees. For such a group it is highly important to know that their passwords and other private data are safe and well managed. Taking into consideration their age, it is also not easy to remember many web addresses, many user names and passwords. That is why in the AGG project the SSO solution was chosen for implementation instead of the local authentication system. SSO is a crucial property for ensuring the integration of separate systems covering the functionality of the project. When looking for services, users do not have to log into many systems, but they need to log in only once and have access to all of them which is secure for credentials and convenient for users. All SSO benefits can increase the competitiveness of AGG application as a tool which can support the daily activity and independent life of elderly people.

## Bibliography

Bower S., 2016, *Using Cornell Shibboleth for Authentication in your Custom Application*, http://blogs.cornell.edu/cloudification/2016/07/11/using-cornell-shibboleth-for-authentication-in-your-custom-application/.

Ceccarelli A. et al., 2015, *Continuous and Transparent User Identity Verification for Secure Internet Services*, IEEE Transactions on Dependable and Secure Computing, vol. 12.

Coulouris G., Dollimore J., Kindberg T., Blair G, 2012, *Distributed systems. Concepts and design*, Pearson Education.

Denis M., Leon J. C., Ormancey E., Tedesco P., 2015, *Identity federation in OpenStack-an introduction to hybrid clouds*, Journal of Physics: Conference Series, vol. 664, no. 2, IOP Publishing.

Dennis Z., 2013, *Choosing an SSO Strategy: SAML vs OAuth2,* https://www.mutuallyhuman.com/blog/2013/05/09/choosing-an-sso-strategy-saml-vs-oauth2/.

Florencio D., Herley C., 2007, *A Large-Scale Study of Web Password Habits*, Security, Privacy, Reliability, and Ethics, ACM, pp. 657-665.

Internet users, 2016, http://www.internetlivestats.com/internet-users/ (9.11.2016).

JWT, 2016, https://jwt.io/introduction/.

Keszthelyi A., 2013, *About passwords*, Acta Polytechnica Hungarica, vol. 10, no. 6.

Koussa S., 2013, *Federated Identities: OpenID vs SAML vs OAuth*, http://www.softwaresecured.com/2013/07/16/federated-identities-openid-vs-saml-vs-oauth

Liferay, 2016, https://web.liferay.com/community/wiki/-/wiki/Main/LDAP+with+AD+in+Liferay+6.0.5/ pop_up.

Online Overload – It's Worse Than You Thought, 2015, https://blog.dashlane.com/infographic-online-overload-its-worse-than-you-thought/ (9.11.2016).

OpenID Foundation, 2016 http://openid.net/foundation/.

OpenID, 2016 http://openid.net/connect/faq/.

Osmanoglu E., 2014, *Identity and Access Management: Business Performance Through Connected Intelligence*, Elsevier.

Password Statistics, *The Bad, the Worse and the Ugly (Infographic)*, 2015, https://www.entrepreneur.com/article/246902 (8.11.2016)

Sakimura N., 2014, *Open ID Connect*, http://openid.net/specs/openid-connect-core-1_0.html.

Saldanha A., 2013, *SAML vs. OAuth: Which One Should I Use?*, https://dzone.com/articles/saml-versus-oauth-which-one.

Singer P., Friedman A., 2013, *Cybersecurity and Cyberwar: What Everyone Needs to Know?*, Oxford University Press.

Single sign-on, 2016, https://en.wikipedia.org/wiki/Single_sign-on (9.11.2016).

SSO Benefits 2016, https://www.uoguelph.ca/ccs/security/internet/single-sign-sso/benefits (9.11.2016)

Surya M., Anithadevi N., 2015, *Single Sign on Mechanism Using Attribute Based Encryption in Distributed Computer Network*, Graph Algorithms, High Performance Implementations and Its Applications (ICGHIA 2014), Procedia Computer Science, 47, pp. 441-451.