System centralnego uwierzytelniania z pojedynczym logowaniem (CAS — Central Authentication Service)

informacje ogólne, instalacja serwera CAS,
integracja usług USOS z CAS-em

Maja Górecka-Wolniewicz

20 maja 2007

Spis treści

1 Wprowadzenie

Opracowanie jest przeznaczone dla osób zainteresowanych wdrożeniem systemu centralnego uwierzytelniania z pojedynczym logowaniem oraz administratorów instalujących aplikacje USOS zintegrowane z systemem CAS. Omówiono cel stosowania systemów centralnego uwierzytelniania oraz zasady ich działania. Opisano proces instalacji oprogramowania serwera CAS oraz wymaganych przez CAS komponentów programowych. Przedstawiono przebieg konfiguracji aplikacji USOS (USOSweb, UL i APD) w celu włączenia możliwości uwierzytelniania za pomocą systemu CAS. Omówiono również problemy obsługi przeterminowania sesji i wylogowania w systemach pojedynczego logowania oraz zagadnienie rejestracji usług uprawnionych do korzystania z systemu CAS.

2 Historia CAS i rola systemów centralnego uwierzytelniania z pojedynczym logowaniem

System CAS (Central Authentication Service) jest jednym z popularniejszych systemów pojedynczego logowania (ang. *Single Sign-On System*, SSO). Projekt i pierwsza implementacja powstały na uniwersytecie Yale. Od grudnia 2004 r. CAS jest rozwijany w ramach programu JA-SIG (http://www.ja-sig.org), nakierowanego na rozwój nowoczesnych technologii wspierających systemy informatyczne na uczelniach wyższych. Oprogramowanie jest napisane w języku Java i bazuje na mechanizmie serwletów.

Rosnąca liczba aplikacji wymagających uwierzytelniania sprawia, że centralne uwierzytelnianie z pojedynczym logowaniem staje się coraz bardziej potrzebną usługą w sieciach akademickich. Dzięki niej użytkownik posługuje się tymi samymi danymi uwierzytelniania w celu dostępu do różnych usług, a dodatkowo zawsze wprowadza dane uwierzytelniania do tego samego formularza (wspólna strona logowania, jeden adres URL tej strony), tym samym pojawienie się nieznanej postaci strony logowania, czy przejście pod obcy adres w celu zalogowania sygnalizuje jednoznacznie zagrożenie bezpieczeństwa wprowadzanych danych. Systemy SSO umożliwiają automatyczne zalogowanie do aplikacji, jeżeli wcześniej użytkownik pomyślnie zakończył proces uwierzytelnienia, czyli użytkownik uwierzytelnia się raz, a następnie są ustanawiane nowe, uwierzytelnione sesje w kolejnych aplikacjach otwieranych w danej przeglądarce. Większość dostępnych systemów pojedynczego logowania, również CAS, działa w ramach środowiska przeglądarki internetowej i wykorzystuje mechanizm ciasteczek.

CAS jest już obecnie systemem bardzo dojrzałym i szeroko rozpowszechnionym na uczelniach w Stanach Zjednoczonych, Francji, Wielkiej Brytanii, Hiszpanii, Szwecji. Ogromną zaletą jest dokładnie zdefiniowany protokół oraz bogata oferta bibliotek CAS, dzięki czemu integracja dowolnej aplikacji z CAS-em jest bardzo prosta. Istnieją interfejsy API dla różnych języków programowania: Perl, PHP, Python, Java, ASP.NET, Prado, Ruby on Rails oraz moduł AuthCAS do Apache'a i klient uPortal. Instalacja serwera CAS nie jest skomplikowana. Autorzy oprogramowania przygotowali bardzo wygodne mechanizmy dostosowywania CAS-a do lokalnych potrzeb.

Nie stanowi problemu dodanie własnych metod uwierzytelniania, np. w oparciu o bazę LDAP, albo przy wykorzystaniu certyfikatu klienta. Bardzo przydatną własnością jest możliwość użycia CAS-a w roli pośrednika (ang. *proxy*) — w tym modelu jedna aplikacja może reprezentować użytkownika wobec innej aplikacji.

3 Dostęp do oprogramowania

Aktualne oprogramowanie serwera CAS, rozwijane w projekcie JA-SIG, można pobrać ze strony: http://www.ja-sig.org/products/cas/downloads/index.html.

Najnowszą stabilną wersją jest obecnie (koniec kwietnia 2007) wersja 3.0.7.

Na lipiec 2007 jest planowana przełomowa wersja 3.1, w której zapowiadane jest znaczące rozbudowanie funkcjonalności, przede wszystkim w zakresie obsługi wylogowania oraz zarządzania rejestracją uprawnionych usług (rozdz. 8).

Z tej samej strony można pobrać oprogramowanie klienckie dla różnych języków programowania, m.in. Java, Python, Perl, a także moduł CAS dla serwera Apache, moduł dla oprogramowania uPortal itp.

4 Zasada działania CAS-a

Funkcjonowanie systemu CAS bazuje na mechaniźmie przekierowania. Współpraca z systemem CAS odbywa się za pośrednictwem przeglądarki internetowej. Załóżmy, że po otwarciu przeglądarki użytkownik wpisze adres aplikacji wymagającej uwierzytelnienia i zintegrowanej z systemem CAS. Aplikacja automatycznie przekieruje użytkownika na stronę związaną z serwletem logowania CAS. W tej sytuacji użytkownik nie miał dotychczas ustanowionej sesji uwierzytelniania w ramach danej sesji przeglądarki, dlatego serwlet logowania wymusi przejście na stronę z formularzem logowania, na której użytkownik musi podać swoje podstawowe dane uwierzytelniania, np. nazwę użytkownika i hasło. CAS sprawdza dane uwierzytelnienia i jeżeli nie są one poprawne, to ponownie pokazuje stronę logowania. Gdy dane są poprawne i uwierzytelnienie powiodło się, serwer CAS przekierowuje użytkownika do aplikacji klienckiej, przekazując jednocześnie tzw. bilet usługi (ang. *service ticket*). Aplikacja transparentnie w odniesieniu do użytkownika sprawdza poprawność biletu — jest to realizowane za pomocą usługi walidacji CAS (ang. *CAS validation*). W pozytywnej odpowiedzi na zlecenie walidacji serwer CAS przekazuje nazwę użytkownika. Przeglądarka ustanawia wówczas uwierzytelnioną sesję. Jeżeli użytkownik przejdzie następnie do innej aplikacji używającej CAS-a (zakładamy, że jest nadal aktywna sesja danej przeglądarki), to aplikacja ponownie przekieruje go na stronę związaną z serwletem logowania CAS. Tam użytkownik od razu uzyska tzw. wtórne dane uwierzytelniania (bilet usługi) i natychmiast zostanie przekierowany do aplikacji (bez wezwania do podawania nazwy użytkownika i hasła). Aplikacja kliencka wykorzystująca mechanizm pojedynczego logowania może honorować wtórne dane uwierzytelniania, a jeżeli jest to uzasadnione, to może wymuszać ponowne zalogowanie (strona logowania musi być wówczas wywołana z parametrem `renew=true`).

Opisany mechanizm działania CAS-a przedstawiono dla przypadku, gdy wybrana technika uwierzytelniania bazuje na wezwaniu użytkownika do wpisania danych uwierzytelniania do formularza. CAS może współpracować z dowolnym typem uwierzytelniania, np. można do tego celu wykorzystać środowisko PKI i klucze prywatny oraz publiczny użytkownika. CAS umożliwia również zdefiniowanie alternatywnych metod uwierzytelniania, np. realizujących model: użyj danych PKI, jeżeli uwierzytelnienie nie powiedzie się, to przedstaw formularz logowania.

Jeżeli dana aplikacja korzysta z innej usługi wymagającej uwierzytelnienia (dobrym przykładem jest klient pocztowy WWW, np. IMP, który współpracuje z usługą IMAP), to jest możliwe uwierzytelnienie wykonane przez aplikację w imieniu użytkownika — służy do tego protokół biletów pośrednich CAS (ang. *proxy ticket CAS protocol*). Aplikacja występująca jako pośrednik, otrzymuje od serwera CAS bilet (ang. *proxy ticket granting ticket*) umożliwiający ubieganie się o bilety pośrednika (ang. *proxy ticket*) dla konkretnej usługi.

Powyższy opis przebiegu pracy aplikacji zintegrowanych z CAS-em pokazuje, że dzięki współpracy z CAS-em aplikacja jest bezpieczniejsza, gdyż nie można za jej pomocą wykraść danych uwierzy-

telniania. Aplikacje takie domyślnie nie wymagają rejestracji w systemie CAS, ale dla podniesienia bezpieczeństwa systemu jest możliwe wymuszenie rejestracji uprawnionych aplikacji i odrzucanie prób uwierzytelnienia kierowanych z obcych adresów (rozdz. 8). Ważną własnością CAS-a jest używanie dwóch typów danych uwierzytelniania:

- danych podstawowych (ang. *primary credentials*), czyli tych, które są wprowadzane na stronie logowania CAS;
- danych wtórnych (ang. *secondary credentials*), których istnienie oznacza, że uprzednio z powodzeniem dokonano uwierzytelnienia na podstawie danych podstawowych, dane te są zapamiętywane w przeglądarce poprzez mechanizm ciasteczek (ang. *cookies*) i noszą nazwę TGC (ang. *Ticket Granting Cookie*).

Serwer CAS do współpracy z aplikacją posługuje się biletami, mającymi postać napisów o określonej składni (charakteryzuje je prefiks ST- lub PT-). Bilet usługi jest wydawany przeglądarce po udanym uwierzytelnieniu w celu otwarcie dostępu do konkretnej usługi. Przeglądarka dostarcza bilet, a usługa waliduje go, by mieć gwarancję poprawnego uwierzytelnienia. Analogiczną funkcję spełnia bilet pośrednika.

Protokół CAS jest dokładnie opisany na stronie projektu JA-SIG:
http://www.ja-sig.org/products/cas/overview/protocol/.

Jest to protokół oparty na HTTP (ang. *Hypertext Transfer Protocol*). Zapytania przekazywane do serwera CAS to wywołania przy użyciu metody GET, dotyczące konkretnych komponentów usługi CAS, adresowanych za pomocą określonych identyfikatorów URI. Odpowiedzi przekazywane przez serwer CAS zależą od rodzaju zapytania. W przypadku wywołania komponentu logowania na ogół odpowiedzią jest prezentacja ekranu logowania. Z kolei po zaakceptowaniu danych uwierzytelniania serwer przekierowuje do aplikacji. Odpowiedzią na zlecenie wylogowania jest strona informująca o wylogowaniu z systemu. Jeżeli zlecenie wiąże się z wywołaniem komponentu walidacji biletu, to odpowiedzią jest
`yes/no<LF>nazwa użytkownika<LF>`
lub — w nowszej wersji serwera — odpowiedź ma postać zgodną z protokołem XML.

Użytkownik może zakończyć sesję uwierzytelniania poprzez dostęp do strony wylogowania CAS. Najbardziej zalecaną formą jest zamknięcie przeglądarki po zakończeniu pracy z aplikacjami CAS (rozdz. 6).

5 Obsługa przeterminowania sesji w aplikacjach CAS

CAS nie posiada mechanizmów zarządzania sesją, nie oferuje narzędzi do śledzenia, co użytkownik robi po uwierzytelnieniu. Typowo aplikacje korzystające z CAS-a same ustanawiają własne sesje, poprzez ciasteczka umieszczane w pamięci lub stosują inne metody.

Wygodnym mechanizmem jest ustawianie przez aplikacje, po skutecznym zalogowaniu CAS, specjalnego ciasteczka, wskazującego, że przeglądarka po swoim starcie wykonała udane uwierzytelnienie w CAS-ie. Jeżeli to ciasteczko byłoby zdefiniowane w takiej domenie, do której będą miały dostęp inne aplikacje danej instytucji (np. w domenie `.umk.pl`, a aplikacje posługują się adresami w poddomenie `umk.pl`, np. `nazwa.umk.pl`), to aplikacje te mogą wykorzystać informację o wcześniejszym zalogowaniu CAS na etapie realizacji uwierzytelnienia w bieżącej aplikacji. Takie rozwiązanie bywa przydatne np. gdy korzystamy z biblioteki phpCAS do obsługi klienta CAS. Biblioteka phpCAS bazuje na mechanizmie schowka (ang. *cache*) w celu ograniczenia częstotliwości przekierowywania zapytań do serwera CAS. Ciasteczko we wspólnej domenie mówiące, czy użytkownik jest zalogowany

w CAS-ie, jest pomocne jako dodatkowy element kontrolny: gdy ciasteczko nie istnieje, to nawet w przypadku, gdy bieżąca aplikacja uważa, na podstawie swoich ustawień sesyjnych, że użytkownik jest zalogowany, należy przekierować zapytanie do serwera CAS, bo najprawdopodobniej inna aplikacja poprzez wylogowanie skasowała sesję CAS.

Jeżeli aplikacja używająca CAS-a obsługuje przeterminowanie sesji aplikacji, to w procedurze obsługi przeterminowania należy uwzględnić współpracę z CAS-em. Jeżeli sesja CAS jest nadal aktywna (nie przedawniły się dane wtórne CAS, o których jest mowa w rozdz. 4), to proces ponownego uwierzytelnienia użytkownika odbywa się automatycznie, poza użytkownikiem. Jeżeli w chwili stwierdzenia przedawnienia sesji aplikacji okaże się, że również przedawniła się sesja CAS, to użytkownik zostanie przekierowany na stronę uwierzytelniania CAS.

6 Problem obsługi wylogowania z aplikacji CAS

Obsługa wylogowania z aplikacji zintegrowanej z CAS-em wymaga dobrego zrozumienia mechanizmów współpracy wszystkich aplikacji tego typu udostępnianych w danej przeglądarce. Głównym powodem powstania technik pojedynczego logowania było ułatwienie współpracy użytkownika z chronionymi aplikacjami udostępnianymi za pomocą przeglądarek. Dlatego zakładano, że użytkownik po pierwszym zalogowaniu będzie korzystał z różnych usług zintegrowanych z CAS-em i nie będzie wylogowywał się z poszczególnych aplikacji. Okazuje się, że tego typu podejście jest trudne do wdrożenia wśród użytkowników, którzy przywykli do wylogowywania się po zakończeniu pracy z aplikacją. Przyciski „Wyloguj", obecne w aplikacjach nie posługujących się CAS-em, muszą zostać w jakiś sposób obsłużone w aplikacji CAS. Jednym z podejść do obsługi wylogowania z aplikacji zintegrowanej z CAS-em jest wykonywanie w czasie wylogowania jedynie tych czynności, które są potrzebne do zmiany statusu użytkownika w aplikacji na anonimowego, jak wyczyszczenie sesji. Takie rozwiązanie nie sprawdza się w sytuacji, gdy aplikacja próbuje zawsze zalogować użytkownika — jeżeli są dostępne dane wtórnego uwierzytelnienia CAS, to efekt wylogowania nie zostałby zauważony, gdyż aplikacja natychmiast zalogowałaby ponownie użytkownika, jako że nadal jest aktywna sesja usługi CAS. Inne rozwiązanie to wbudowanie w aplikację możliwości wylogowania z CAS-a i akceptacja konsekwencji wylogowania.

W obecnej wersji oprogramowania serwera o wylogowaniu użytkownika z CAS nie są powiadamiane aplikacje, które wcześniej korzystały z usługi CAS. Ponieważ zazwyczaj aplikacje działają w oparciu o utrzymywane wewnętrznie informacje dotyczące statusu użytkownika, w tej sytuacji pojawi się rozbieżność — w rzeczywistości użytkownik utracił już uprawnienia związane z logowaniem CAS, ale tylko aplikacja, z której nastąpiło wylogowanie wie o zakończeniu pracy. Podobny efekt pojawi się, gdy użytkownik po zalogowaniu w CAS-ie pracuje w różnych aplikacjach, a następnie w jednym z okien wywoła stronę wylogowania CAS. Wówczas żadna z aplikacji klienckich CAS nie dowie się, że nastąpiło wylogowanie.

Opisane wyżej problemy nie pojawiłyby się, gdyby aplikacje zrezygnowały z lokalnego przechowywania informacji o uwierzytelnieniu i przy każdym zleceniu związanym z dostępem do chronionych zasobów byłoby realizowane odwołanie do serwera CAS, ale efektem takiego rozwiązania mogłoby być przeciążenie serwera CAS i znaczące pogorszenie efektywności pracy aplikacji.

Podobnie jak w przypadku problemu przeterminowania, w tej sytuacji pomocne byłoby również korzystanie z ciasteczka we wspólnej domenie — zniknięcie takiego ciasteczko, lub zmiana wartości, świadczyłaby o wylogowaniu z CAS-a.

W rozdz. 9 opisano zmiany planowane w nowej wersji oprogramowania serwera, które mają m.in. wyjść naprzeciw przedstawionym problemom.

7 Instalacja systemu CAS

System CAS to aplikacja zaimplementowana w języku Java i korzystająca z serwletów Javy oraz Java Server Pages.

Przed rozpoczęciem instalacji CAS-a muszą zostać spełnione następujące wymagania dotyczące pakietów zainstalowanych w systemie:

- Java 1.5 lub wyższa wersja (obecnie jest dostępna wersja 1.6)
  http://java.sun.com/javase/downloads/index.jsp,
- Apache Tomcat 5.x lub wyższa wersja http://tomcat.apache.org.

W celu dostosowania aplikacji do lokalnych potrzeb niezbędna jest instalacja pakietu

- Apache Ant http://ant.apache.org

który jest używany przy kompilacji pakietu CAS.

Wersję źródłową serwera CAS pobieramy ze strony (sekcja „CAS Server Releases"):
http://www.ja-sig.org/products/cas/downloads/index.html.
Bardzo dobry opis instalacji i dostosowania CAS-a znajduje się na stronie
http://www.ja-sig.org/products/cas/server/index.html.

Serwer CAS wymaga zastosowania SSL-a. Sekcja „Solving SSL Issues" zawiera informacje pomocne podczas przygotowania bezpiecznego serwisu CAS. W przypadku pierwszej instalacji bezpiecznej usługi opartej na serwerze Tomcat przydatna może być instrukcja „SSL Configuration How-To" dostępna pod adresem:
http://tomcat.apache.org/tomcat-5.0-doc/ssl-howto.html.

Ważną rzeczą przy konfiguracji współpracy Tomcata z SSL-em jest dodanie do magazynu kluczy zaufanych urzędów publicznych środowiska Java klucza urzędu nadrzędnego związanego z certyfikatem serwera.

Jeżeli w pliku `/tmp/ca.cer` jest wersja PEM certyfikatu głównego urzędu certyfikacyjnego, to dodanie go do magazynu zaufanych kluczy środowiska Java jest realizowane za pomocą polecenia:

```
keytool -import -keystore $JAVA_HOME/jre/lib/security/cacerts
      -trustcacerts -file /tmp/ca.cer -alias uniCA
      -storepass haslo_do_magazynu
```

Zazwyczaj CAS wymaga dostosowania do lokalnych potrzeb w dwóch aspektach:

- realizacja uwierzytelniania zgodnie z wybranym lokalnie mechanizmem (lub kilkoma różnymi mechanizmami),
- ustalenie wyglądu i treści strony logowania/wylogowania.

7.1 Przygotowanie programu obsługi uwierzytelniania

W sekcji „How to Write an Authentication Handler" opisu instalacji serwera, na stronie projektu JA-SIG, zostały przedstawione przykłady, jak należy budować programu obsługi (ang. *handler*) uwierzytelniania. Najprostszym rozwiązaniem jest przygotowanie modułu obsługi, który rozszerza

klasę `AbstractUsernamePasswordAuthenticationHandler`. Inne podejście może bazować na implementacji interfejsu `AuthenticationHandler`. Kod programu do uwierzytelniania należy umieścić w katalogu `localPlugins/src`.

W przypadku zastosowania LDAP-a do uwierzytelniania autorzy instrukcji instalacyjnej zalecają korzystanie z bibliotek Spring LDAP i Ldap Template. Od wersji 3.0.5 serwera CAS biblioteki związane z uwierzytelnianiem LDAP są włączone do dystrybucji (znajdują się w katalogu `target`). Aby potrzebne biblioteki zostały włączone do przygotowywanej dystrybucji `cas.war`, należy je skopiować do katalogu `localPlugins/lib`.

CAS może korzystać z certyfikatów X.509 do uwierzytelniania klientów. Sposób realizacji takiego rozwiązania został przedstawiony w sekcji „Configuring CAS to use X.509 Certificates".

7.2 Dostosowanie wyglądu

W dystrybucji CAS-a dostarczane są dwa wyglądy stron usługi: domyślny (katalog `default`) korzystający z CSS oraz prosty (katalog `simple`) pokazujący rozwiązanie wymagające najmniej przygotowania.

Implementacje wyglądu stron znajdują się w katalogu
`webapp/WEB-INF/view/jsp`. W celu wprowadzenia nowego wyglądu należy utworzyć katalog np.
`webapp/WEB-INF/view/js/nazwa instytucji/ui`. W tym katalogu należy umieścić własne implementacje następujących stron:

- `casConfirmView.jsp` — strona potwierdzenia, gdy użytkownik wybrał opcję „ostrzegaj przed zalogowaniem",
- `casGenericSuccess.jsp` — strona prezentowana użytkownikowi, gdy zaloguje się bez pośrednictwa innej aplikacji (bezpośrednie logowanie CAS),
- `casLoginView.jsp` — strona prezentowana użytkownikowi w celu pobrania danych uwierzytelninania,
- `casLogoutView.jsp` — strona pokazywana po zakończeniu sesji CAS.

Wymienione strony muszą być przygotowane w technologii JSP, w prezentowanych w dokumentacji przykładach są również używane biblioteki (ang. *taglib*): standardowa (JSTL) oraz Spring.

Informacja o obowiązującym dla danego serwera CAS wyglądzie jest umieszczona w pliku
`webapp/WEB-INF/classes/default_views.properties`
Tam należy wprowadzić odpowiednie zmiany, by zaczął obowiązywać nowy wygląd stron.

W pliku `webapp/WEB-INF/deployerConfigContext.xml` należy wskazać zaimplementowany program obsługi uwierzytelniania. Najlepiej wiersz

```
<bean
class="org.jasig.cas.authentication.handler.
 support.SimpleTestUsernamePasswordAuthenticationHandler" />
```

zastąpić wpisem wskazującym nowy program obsługi, np.:

```
<bean
class="org.jasig.cas.authentication.handler.
 support.LDAPAuthenticationHandler." />
```

Po dostosowaniu aplikacji CAS należy przejść do katalogu `localPlugins/src` i wywołać polecenie

```
ant war
```

Jeżeli kompilacja zakończy się poprawnie, to w katalogu `localPlugins/target` powstanie plik `cas.war`, zawierający aplikację serwera CAS. Ten plik należy umieścić w katalogu `webapps` instalacji Tomcat, lub, jeżeli instalacja Tomcat nie dopuszcza automatycznego rozpakowania pliku war, to należy go rozpakować w lokalizacji wybranej jako korzeń drzewa dokumentów aplikacji CAS (`docBase`).

Tak przygotowany serwer CAS jest gotowy do pracy.

8 Rejestracja uprawnionych serwisów

Domyślnie CAS pozwala, by dowolna usługa korzystała z CAS-a jako dostawcy uwierzytelniania. W CAS3 dodano możliwość zdefiniowania klientów, którzy mogą zgłaszać się do CAS-a w celu uwierzytelnienia. Rejestrowanie usług uprawnionych do korzystania z serwera CAS podnosi bezpieczeństwo. Nie jest wówczas możliwe stworzenie usługi korzystającej z danego serwera CAS, która przechwyci dane o użytkowniku po przekierowaniu do serwera CAS (serwer CAS w odpowiedzi na pozytywną walidację biletu usługi odsyła identyfikator zalogowanego użytkownika). Jeżeli publikacja nazwy logowania użytkownika nie stanowi zagrożenia bezpieczeństwa danych, to korzystanie z funkcjonalności rejestracji uprawnionych serwisów nie jest potrzebne.

W celu włączenia możliwości definiowania usług uprawnionych do korzystania z CAS-a należy zmodyfikować plik `webapp/WEB-INF/web.xml`.
W definicji parametru `contextConfigLocation` w sekcji ustawienia wartości należy dodać `/WEB-INF/approvedServ`
tak by blok wyglądał następująco:

```
<context-param>
             <param-name>contextConfigLocation</param-name>
             <param-value>
                  /WEB-INF/applicationContext.xml,
                  /WEB-INF/deployerConfigContext.xml
             </param-value>
</context-param>
```

Następnie należy zmodyfikować plik `WEB-INF/classes/services.xml` poprzez dodanie bloku `<bean ..>..</bean>` dla każdej zarejestrowanej usługi.

Blok ma następującą postać:

```
<bean id="appsN" class="org.jasig.cas.services.RegisteredService">

<constructor-arg index="0"><value>https://localhost:8443/sN</value>
</constructor-arg>
<constructor-arg index="1"><value>true</value>
</constructor-arg>
<constructor-arg index="2"><value>true</value>
</constructor-arg>
<constructor-arg index="3"><value>test</value>
</constructor-arg>
<constructor-arg index="4"><value>https://localhost:8443/sN</value>
</constructor-arg>
 </bean>
```

gdzie:
`constructor-arg index="0"` jest identyfikatorem usługi,

`constructor-arg index="1"` wskazuje, czy usługa ma prawo występowania jako pośrednik,
`constructor-arg index="2"` wskazuje, czy jest wymuszane uwierzytelnienie,
`constructor-arg index="3"` jest nazwą wzorca związanego z usługą (ang. *theme*),
`constructor-arg index="4"` wskazuje adres pośrednika usługi, jeżeli jest wykorzystywana ta funkcjonalność.

Usługi nieuprawnione nie zostaną zaakceptowane. Pojawi się wówczas wyjątek `UnauthorizedServiceException` Wersja 3.1 oprogramowania serwera CAS będzie istotnie usprawniona pod kątem rejestracji usług, bieżące rozwiązanie jest traktowane jako przejściowe.

9 Planowane poszerzenie funkcjonalności serwera CAS

9.1 Moduł rejestracji uprawnionych serwisów

W wersji 3.1 są zapowiadane znaczące zmiany w ramach modułu zarządzania usługami. Najważniejsze nowe własności to:

- możliwość łatwego włączenia i wyłączenia obowiązku rejestracji,
- możliwość rejestrowania grup usług, m.in. za pomocą wzorców i wyrażeń regularnych,
- możliwość definicji atrybutów przekazywanych do konkretnych usług (w wersji 3.1 poza identyfikatorem użytkownika będzie można uzyskać z serwera CAS inne atrybuty dotyczące zalogowanego użytkownika),
- przechowywanie usług uprawnionych w dedykowanej bazie,
- niezależny interfejs do administrowania zarejestrowanymi usługami.

9.2 Obsługa pojedynczego wylogowania

W wersji 3.1 pojawią się mechanizmy związane z obsługą pojedynczego wylogowania z CAS-a. Nie jest jeszcze wiadomo, jaki model zostanie wybrany. Najprawdopodobniej CAS podczas realizacji zlecenia wylogowania danego użytkownika wyśle informację o wylogowaniu do wszystkich usług, które w czasie aktywności sesji użytkownika zgłosiły się po bilet dla usługi. Informacja będzie wysyłana na specjalny URL usługi zarejestrowany po otrzymaniu zlecenia z danej usługi (ang. *parameter callback*). W ten sposób klient CAS dowie się, że nastąpiła zmiana statusu użytkownika.

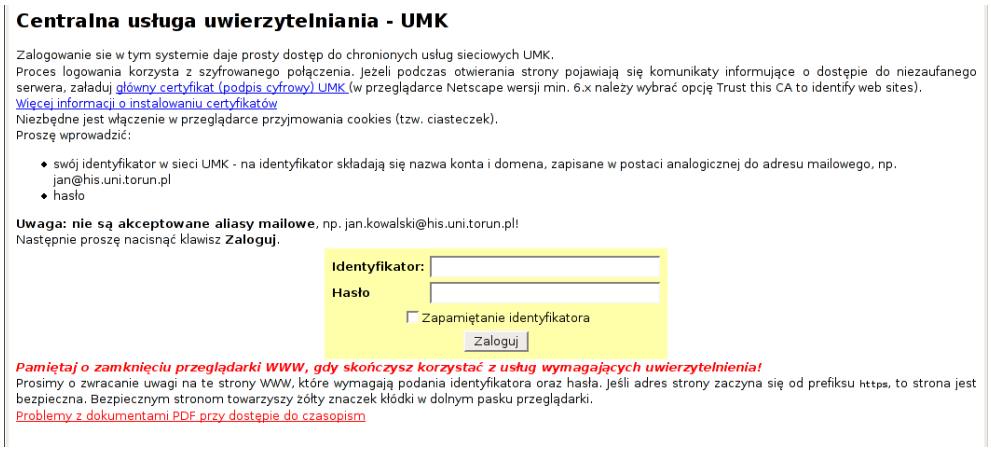
10 Integracja aplikacji USOS z systemem CAS

Trzy aplikacje USOS, USOSweb, UL i APD zostały w sposób jednolity zintegrowane z CAS-em. Celem integracji było umożliwienie takiej konfiguracji aplikacji, by w zakresie logowania był używany system CAS. Jeżeli na etapie konfiguracji (rozdz. 11) włączono korzystanie z CAS-a, to wymienione aplikacje USOS bazują na systemie CAS przy logowaniu i wylogowaniu użytkownika.

10.1 Logowanie za pośrednictwem usługi CAS

Jeżeli sesja aplikacji jest otwierana po uprzednim zalogowaniu w systemie CAS, czyli przeglądarka ma dostępne w postaci ciasteczka wtórne dane uwierzytelniania CAS (rozdz. 4), to zalogowanie w

aplikacji następuje automatycznie. W przeciwnej sytuacji aplikacja uaktywnia sesję anonimową, a po naciśnięciu przycisku związanego z logowaniem ('ZALOGUJ' w USOSweb i APD, 'Logowanie' w UL) prezentuje ekran logowania systemu CAS. Ekran logowania CAS jest specyficzny dla danej instytucji, nie wiąże się z aplikacją. Zawiera formularz logowania, za pośrednictwem którego użytkownik wprowadza swoje dane uwierzytelniania. Strona powinna być podpisana certyfikatem instytucji, a grafika powinna nawiązywać do danej uczelni.



Ilustracja 1: Przykładowy ekran logowania — Uniwersytet Mikołaja Kopernika w Toruniu

Po udanym zalogowaniu do systemu CAS w górnej części strony aplikacji USOS pojawia się typowy dla pracy w tym trybie tekst informujący o zalogowaniu, podawane jest również imię i nazwisko użytkownika. W przypadku stosowania systemu CAS do uwierzytelniania użytkowników w aplikacjach USOS może się zdarzyć, że po udanym uwierzytelnieniu CAS pojawi się komunikat o braku uprawnień w danej aplikacji. Oznacza to, że użytkownik nie ma żadnych uprawnień do pracy w aplikacji, może tylko pracować w trybie anonimowym. W tej sytuacji przy próbie dostępu do chronionych sekcji aplikacji pojawia się komunikat o braku uprawnień. Na przykład, w aplikacji USOSweb pojawi się wówczas na stronie następująca informacja:

Ilustracja 2: Brak uprawnień – użytkownik nie jest zarejestrowany w aplikacji

Jeżeli jest możliwe zalogowanie z innym identyfikatorem, który ma właściwe uprawnienia, to należy najpierw wylogować się (p. 10.3).

10.2 Obsługa przeterminowania sesji

Jeżeli nastąpi przeterminowanie sesji w aplikacji USOS, to dokonywane jest sprawdzenie, czy użytkownik jest nadal zalogowany w systemie CAS. Jeżeli tak, to sesja aplikacji jest odtwarzana automatycznie, bez potrzeby logowania. Jeżeli nie są dostępne dane uwierzytelniania CAS, to wymuszane jest zalogowanie w systemie.

10.3 Wylogowanie z aplikacji

Wylogowanie z aplikacji następuje po naciśnięciu przycisku związanego z wylogowanieniem ('WYLOGUJ' w USOSweb i APD, 'Wylogowanie' w UL).

11 Konfiguracja aplikacji USOS zintegrowanych z CAS-em

W rozdziale przedstawiono wskazówki dotyczące konfiguracji aplikacji USOS w przypadku, gdy planowane jest korzystanie z usługi CAS do realizacji zadań logowania i wylogowania. W podanych przykładach zakładamy, że serwer CAS jest gotowy do pracy i działa pod adresem
https://login.domena.pl/cas
na standardowym porcie 443.

11.1 USOSweb

W celu instalacji USOSweba zintegrowanego z systemem CAS należy podczas procesu konfiguracji (polecenie `configure`) ustawić odpowiednie zmienne systemowe.
Oto fragment konfiguracji dotyczący współpracy z CAS-em:

```
ZMIENNA: CAS (text)
Opis: Usługa CAS (0 - brak, 1 - obowiązująca)

CAS [0]: 1
```

wpisujemy wartość 1 - aplikacja współpracuje z CAS-em

```
ZMIENNA: CAS_HOST (text)
Opis: Serwer CAS

CAS_HOST []: login.domena.pl
```

Podajemy nazwę komputera, na którym działa CAS

```
ZMIENNA: CAS_PORT (text)
Opis: Port serwera CAS

CAS_PORT []: 443
```

Podajemy numer portu, na którym działa CAS

```
ZMIENNA: CAS_URI (text)
Opis: Ścieżka do serwera CAS

CAS_URI []: /cas
```

Podajemy ścieżkę serwera CAS
Jeśli proces konfiguracji zakończył się prawidłowo, to wykonujemy standardowe czynności

```
make
make install
```

Tak skonfigurowany USOSweb będzie korzystał z systemu CAS w celu realizacji zadań logowania i wylogowania.

11.2 UL

W celu instalacji UL-a zintegrowanego z systemem CAS należy podczas procesu konfiguracji (polecenie `configure`) ustawić odpowiednie zmienne systemowe.
Oto fragment konfiguracji dotyczący współpracy z CAS-em:

```
Usługa CAS (0 - brak, 1 - włączona) [] 1
```

wpisujemy wartość 1, gdy aplikacja ma współpracować z CAS-em

```
Serwer CAS []: login.domena.pl
```

podajemy nazwę komputera, na którym działa CAS

```
Port serwera CAS []: 443
```

podajemy numer portu, na którym działa CAS

```
Ścieżka do serwera CAS []: /cas
```

podajemy ścieżkę serwera CAS
Jeśli proces konfiguracji zakończył się prawidłowo, to wykonujemy standardowe czynności

```
make
make install
```

Tak skonfigurowany UL będzie korzystał z systemu CAS w celu realizacji zadań logowania i wylogowania.

11.3 APD

Konfiguracja integracji APD z CAS-em następuje na etapie działania instalatora webowego. Na ekranie instalatora, w sekcji Konfiguracja CAS realizujemy poniższe czynności:

- zaznaczamy `Obsługa CAS włączona` — oznacza to, że aplikacja ma współpracować z CAS-em,
- w polu `Host CAS` wpisujemy `login.domena.pl` — podajemy nazwę komputera, na którym działa CAS,
- w polu `Port CAS` wpisujemy `443` — podajemy numer portu, na którym działa CAS,
- w polu `Ścieżka CAS` wpisujemy `/cas` — podajemy ścieżkę do serwera.

Tak skonfigurowany APD będzie korzystał z systemu CAS w celu realizacji zadań logowania i wylogowania.