WOJSKOWA AKADEMIA TECHNICZNA
im. Jarosława Dąbrowskiego
WYDZIAŁ CYBERNETYKI

PRACA DYPLOMOWA

STACJONARNE STUDIA I°

Temat: **SYSTEM REJESTRACJI UŻYTKOWNIKÓW W SERWISIE INTERNETOWYM**

Autor: **Piotr KWIATEK**

Kierownik pracy: **dr inż. Jan CHUDZIKIEWICZ**

Warszawa 2010

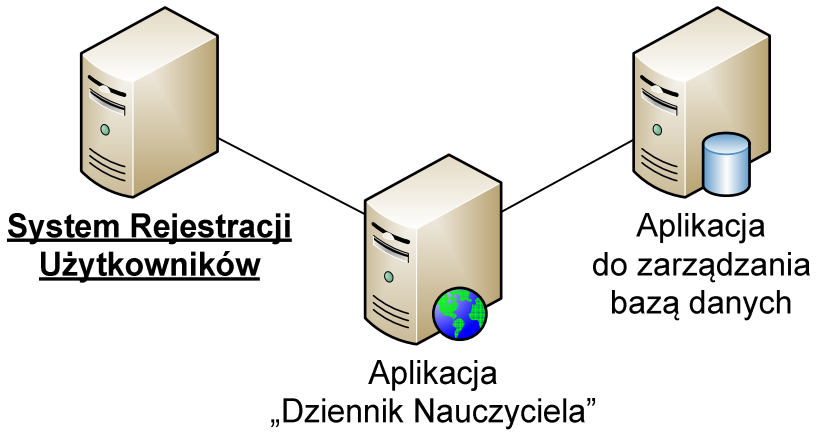
Obrona pracy odbyła się 8 lutego 2010r.

SPIS TREŚCI

WSTĘP

Systemy informatyczne posiadają w swoich zasobach dane o tożsamościach ich użytkowników. Na jednoznaczne odwzorowanie rzeczywistej tożsamości użytkownika w obiekt tożsamości w systemie pozwalają różne rodzaje mechanizmów uwierzytelniania. Dane zgromadzone w systemie informatycznym są niejednokrotnie bardziej cenne od wartości majątku firmy, dlatego też informatycy nieustannie pracują nad udoskonalaniem mechanizmów bezpieczeństwa uwierzytelniania, autoryzacji oraz zarządzania danymi o użytkownikach. W niniejszej pracy przeanalizowano oraz wskazano newralgiczne punkty systemów zajmujących się uwierzytelnianiem, autoryzacją użytkowników oraz rejestracją nowych kont.

Celem budowy Serwisu Rejestracji Użytkowników (SRU) jest dostarczenie bezpiecznych mechanizmów zarządzania użytkownikami dla aplikacji wspierającej pracę nauczyciela. Na rysunku 1 zaznaczono umiejscowienie SRU w relacji z pozostałymi aplikacjami, współpracującymi z projektowanym systemem.



Rys. 1. Ogólny schemat powiązań pomiędzy aplikacjami

Aby zapewnić wysoki poziom bezpieczeństwa oraz dobrą funkcjonalność systemu potrzebne jest odpowiednie rozpoznanie i analiza dziedziny problemu. W pierwszym rozdziale pracy scharakteryzowano koncepcje rozwiązań potwierdzania tożsamości użytkowników, z powodzeniem stosowane w Internecie. Zdefiniowano także ogół wymagań nakładanych na system, które mają znaczący

wpływ na wybór technologii implementacji, mechanizmów uwierzytelniania, autoryzacji oraz zabezpieczeń systemu.

Celem rozdziału drugiego jest wyspecyfikowanie oraz analiza możliwych sposobów przeciwdziałania zagrożeniom, na które narażone są systemy informatyczne. Wskazano newralgiczne punkty systemu oraz sposoby podnoszenia poziomu bezpieczeństwa. Przegląd aspektów bezpieczeństwa systemu poświęcono w szczególności bezpiecznemu przekazywaniu oraz przechowywaniu danych w systemie. Omówione zostały także znane rodzaje ataków, na które każda aplikacja internetowa, zapewniająca podstawową ochronę danych użytkowników, powinna być uodporniona.

Bazując na zdefiniowanych wymaganiach nakładanych na SRU oraz mając na uwadze omówione aspekty bezpieczeństwa, w rozdziale trzecim dokonano przeglądu dostępnych w ASP.NET oraz Windows® mechanizmów autoryzacji. Rozdział ten stanowi uzasadnienie wyboru przedstawionych mechanizmów.

Rozdział czwarty zawiera projekt systemu, stanowiący efekt kontaminacji zdefiniowanych wymagań funkcjonalnych, analizy bezpieczeństwa oraz wybranych mechanizmów i technologii.

W rozdziale piątym przedstawiono wybrane rozwiązania implementacyjne budowanego systemu, z wykorzystaniem odpowiednich narzędzi i technologii. W ramach podsumowania, w pracy zawarto opis przeprowadzonych testów działania aplikacji oraz propozycje zastosowania i dalszego rozwoju systemu.

I. DYSKUSJA WYMAGAŃ FUNKCJONALNYCH NAKŁADANYCH NA SYSTEM

Znaczna część wymagań funkcjonalnych nakładanych na omawiany system wynika z konieczności zintegrowania go z aplikacją internetową „Dziennik nauczyciela", wspomagającą pracę nauczyciela, zaimplementowaną w technologii ASP.NET. Początkową ideą była budowa SRU jako dedykowanego komponentu dla wyżej wymienionej aplikacji internetowej, jednak po dogłębnej analizie charakterystyki dziedziny oraz przeglądzie funkcji cechujących większość systemów uwierzytelniania, powstała myśl o stworzeniu niezależnego systemu uwierzytelniania i autoryzacji. Jego zadaniem byłoby spełnianie wszystkich wymagań stawianych przez aplikację dziennika nauczyciela, ale także byłaby możliwość wykorzystania go do uwierzytelniania użytkowników innych serwisów internetowych. Dzięki takiemu rozwiązaniu, SRU ze względu na uniwersalność wykonywanych usług mógłby je świadczyć dla wielu aplikacji internetowych jednocześnie. W poniższym rozdziale oprócz zasadniczej polemiki nad wymaganiami funkcjonalnymi wobec systemu, przestawiono funkcje, które posiadać powinien każdy system uwierzytelniania i zarządzania użytkownikami oraz te, które zasadniczo będą wspomagać aplikację internetowego dziennika nauczyciela.

1.1. Charakterystyka mechanizmów autentykacji w serwisach internetowych

„*Logować się to rozpocząć pracę z systemem komputerowym o kontrolowanym dostępie przez podanie identyfikatora użytkownika i hasła.*" [8] Na stronach internetowych jest to znany i powszechnie używany mechanizm potwierdzenia tożsamości internauty. W celu zalogowania się zazwyczaj potrzebne jest uprzednie zarejestrowanie się polegające na wypełnieniu odpowiedniego formularza lub uzyskanie danych potrzebnych do uwierzytelnienia od administratora witryny. Rejestrację oraz logowanie przeprowadza się w Internecie w różnych celach. Zanim jednak je wymienię pozwolę sobie przybliżyć pojęcia uwierzytelniania oraz autoryzacji.

Uwierzytelnianie jest procesem polegającym na zweryfikowaniu tożsamości osoby, urządzenia lub usługi biorącej udział w wymianie danych. Następuje po identyfikacji. Identyfikacją lub deklaracją tożsamości jest podanie nazwy użytkownika, natomiast samo uwierzytelnienie następuje w kolejnym kroku, czyli przy podaniu hasła, biletu, kodu itp. Uwierzytelnienie ma miejsce zawsze przed autoryzacją. [2 s.244; 3 s.16]

Autoryzacja jest procesem, w którym sprawdza się czy dany podmiot o ustalonej już w procesie uwierzytelniania tożsamości posiada prawo odczytu lub modyfikacji zasobów, do których żąda dostępu. Inaczej mówiąc proces uwierzytelniania sprawdza kim jest ta osoba, natomiast proces autoryzacji sprawdza do czego ma prawo. [3, s.27]

Głównym celem systemu jest zabezpieczenie wspieranej aplikacji przed nieautoryzowanym dostępem, podszywaniem się i pozyskiwaniem danych przez osoby postronne. Wspieraną aplikacją jest każdy serwis internetowy implementujący usługę uwierzytelnienia w SRU oraz posiadający konto „serwisu internetowego" w bazie danych SRU. Serwis internetowy może przekazać do realizacji w SRU wszystkie obowiązki związane z bezpiecznym przechowywaniem i przekazywaniem poświadczeń do systemu oraz bezpiecznym uwierzytelnieniem i autoryzacją użytkownika.

Zasadniczo podczas realizacji tego podstawowego celu systemu przewiduje się dwa scenariusze realizacji uwierzytelnienia. Pierwszym jest przypadek, kiedy system nie rozpozna poświadczeń podanych przez użytkownika lub mimo poprawnych danych logowania stwierdzi, że użytkownik nie ma wystarczających uprawnień do żądanych zasobów i odmówi dostępu. Ogranicza się to do wyświetlenia stosownego komunikatu informującego o nieprawidłowej nazwie użytkownika lub haśle, bądź poinformuje o błędzie autoryzacji dostępu do wybranego zasobu. Nieco obszerniejszym jest scenariusz, w którym użytkownik pomyślnie uwierzytelni się w SRU, a następnie uzyska dostęp do wybranego serwisu internetowego współpracującego z SRU. W tym przypadku system uwierzytelniania autoryzuje dostęp do wybranych zasobów w czasie trwania całej sesji użytkownika. To właśnie fakt kontroli dostępu jedynie do wybranych zasobów

i usług jest bardzo ważnym, a niejednokrotnie najważniejszym celem stosowania logowania.

Kolejnym z powodów, dla których wprowadza się mechanizmy logowania i rejestracji na stronach WWW jest personalizacja portalu. Osoba taka może mieć wpływ na wygląd, rozmieszczenie i zachowanie się wybranych komponentów wyświetlanych na stronie.

W sieci Internet oprócz strefy stron ogólnie dostępnych istnieją serwisy, które całą swoją zawartość przesłaniają formularzem logowania. Mowa tutaj o stronach zamkniętych dla użytkowników anonimowych, do których dostęp mają tylko zalogowani użytkownicy. Często witryny takie nie posiadają formularzy rejestracyjnych, a jeśli tak, to aktywacja konta po rejestracji następuje dopiero w skutek aprobaty administratora takiego serwisu. Rozwiązanie to stosowane jest najpowszechniej na stronach WWW zakładanych dla wybranej grupy osób skupiających się wokół konkretnej tematyki, instytucji, zainteresowań, prowadzonych projektów itp.

Każdy internauta, który choć raz korzystał z usług serwisów aukcyjnych z pewnością spotkał się z metodą sprawdzania wiarygodności użytkownika poprzez wysłanie do niego pocztą tradycyjną listów zawierających hasła potrzebne do pełnej aktywacji konta. Tym razem, jednym z ważniejszych celów jest sprawdzenie wiarygodności aukcjonera, wykluczenie zdublowanych kont użytkowników w systemie i ewentualnych prób podszywania się. Przykładem znanych polskich internetowych serwisów aukcyjnych, które praktykują wyżej wymieniony sposób sprawdzania wiarygodności danych wprowadzonych do formularza rejestracyjnego są Allegro i Świstak. Bardzo podobny przebieg zwiększenia wiarygodności danych podawanych w formularzu rejestracyjnym ma potwierdzenie adresu e-mail. W tym przypadku także wysyłany jest list, ale tym razem w postaci wiadomości elektronicznej z jednorazowym i unikalnym hasłem potrzebnym do pełnej aktywacji konta. Jest to dużo prostszy, tańszy, szybszy i wygodniejszy sposób na ograniczenie dublowania kont w systemie oraz zwiększenie wiarygodności użytkownika posiadającego owe nowo założone konto, jednak w tym przypadku nie potwierdza adresu zamieszkania. Podsumowując, wyżej opisane metody mogą dostarczyć nam

wysoce wiarygodnych danych o używanym adresie e-mail oraz adresie zamieszkania.

1.2. Wymagania niefunkcjonalne

Jak wspomniano wcześniej SRU z założenia miał być komponentem aplikacji ASP.NET wspomagającej pracę nauczyciela, dlatego też jednym z wymagań niefunkcjonalnych jest użycie odpowiedniej technologii przy budowie systemu. Warto nadmienić, że powodem wyboru tej samej technologii implementacji obu aplikacji wcale nie jest kwestia kompatybilności pomiędzy tymi aplikacjami internetowymi, lecz chęć zintegrowania wszystkich części aplikacji wspomagającej pracę nauczyciela w obrębie jednego systemu operacyjnego i serwera stron WWW. Niezależność naszej aplikacji od języka implementacji serwisów klienckich, czyli tych, dla których SRU świadczy usługi, jest na tyle duża, dzięki użyciu usług sieciowych ASP.NET. Wykorzystują one w swoim działaniu powszechnie znany standard SOAP[1], dzięki któremu możemy integrować aplikacje internetowe niezależnie od języka programowania oraz platformy deweloperskiej [9]. W celu wygodniejszej integracji tych aplikacji w obrębie jednego środowiska, SRU zostanie stworzony przy użyciu frameworka aplikacji internetowych ASP.NET. Jest to tworzona przez firmę Microsoft struktura do budowania dynamicznych stron internetowych. Aplikacja wspomagająca pracę nauczyciela oprócz konieczności korzystania z usług SRU współpracuje z aplikacją zapewniającą bezpieczny i wydajny dostęp do bazy danych, która także została napisana w ASP. Cały system składający się z trzech autonomicznych aplikacji zostanie więc osadzony na Windows 2008 Server z uruchomionymi usługami IIS[2] 7.0 oraz Microsoft SQL 2008.

---

[1] (ang. Simple Object Access Protocol), protokół umożliwiający wymianę ustrukturyzowanych wiadomości w rozproszonym środowisku. Często wykorzystywany do zdalnego dostępu do obiektów innego systemu. [1, s.564]

[2] (ang. Internet Information Services), serwer stron internetowych, działający w środowisku Microsoft Windows.

Kolejnym już bardzo oczywistym wymaganiem niefunkcjonalnym jest zachowanie możliwie jak największej kompatybilności z przeglądarkami internetowymi oraz z większością przeglądarek dla urządzeń mobilnych. Strona internetowa musi być więc zgodna ze standardem HTML 4.01 oraz CSS 2.

1.3. Wymagania funkcjonalne

1.3.1. Rejestracja nowego użytkownika

W aspekcie omawianego systemu, tworzenie nowych kont użytkowników możemy realizować na dwa sposoby. Pierwszym jest udostępnienie anonimowym użytkownikom możliwości samodzielnego utworzenia konta w systemie poprzez wypełnienie formularza rejestracyjnego. Drugim sposobem tworzenia kont użytkowników jest przerzucenie tego obowiązku na administratora aplikacji klienckiej, czyli np. nauczyciela korzystającego z aplikacji e-learningowej. Wybór scenariusza rejestracji nowego użytkownika nie jest trywialny, ponieważ każda z tych dróg przynosi inne korzyści, trudności i zagrożenia.

Zacznę od pierwszego sposobu, czyli udostępnienia formularza rejestracyjnego dla nieuwierzytelnionych internautów. Rozwiązanie to jest wygodniejsze, ponieważ umożliwia błyskawiczne założenie konta oraz dostęp do docelowego serwisu internetowego. W przeciwieństwie do zdalnego tworzenia kont, użytkownik nie musi kontaktować się innymi drogami z administratorem systemu w celu uzyskania danych potrzebnych do zalogowania się w systemie. Dzięki temu rozwiązaniu zyskujemy na czasie oraz odciążamy z często żmudnej pracy administratora kont użytkowników. Taki sposób realizacji tej funkcjonalności wydaje się być dość dobry, jednak nie zapewnia wiarygodnego powiązania danej osoby z kontem w systemie.

Rozwiązanie zdalnego tworzenia kont zapewnia wiarygodność danych podanych w profilu konta, ponieważ administrator po jego założeniu, w momencie przekazywania poświadczeń do zalogowania się w systemie, dokonuje potwierdzenia przynależności tej konkretnej osoby do nowo założonego konta. Warto jednak w tym przypadku zwrócić uwagę na kwestię bezpieczeństwa. Bardzo

często zdarza się, że hasła i loginy przekazywane są w niewystarczająco bezpieczny sposób, np. w wiadomościach e-mail zawierających hasła przesyłane w jawnej postaci lub po prostu na kartce papieru.

Chęć i konieczność spełnienia wszystkich wyżej wymienionych wymagań co do funkcjonalności tworzenia kont użytkowników skłoniła mnie do rozwiązania hybrydowego. Polega ono na udostępnieniu dla anonimowych użytkowników formularza rejestracyjnego, zapewniającego szybki dostęp tylko do SRU zaraz po utworzeniu konta. Autoryzacja dostępu do serwisu klienckiego natomiast wymaga aprobaty administratora wybranego serwisu internetowego, do którego użytkownik żąda dostępu. W tym momencie jednak pozostaje problem wiarygodności wprowadzonych przez internautę danych w formularzu rejestracyjnym. Administrator może opierać wiarygodność tych danych na wprowadzonym i potwierdzonym adresie e-mail lub musi skontaktować się inną drogą z użytkownikiem w celu potwierdzenia przynależności do konta. Bezpieczeństwo systemu zależy w największej mierze od człowieka i to właśnie on jest najsłabszym ogniwem w bezpieczeństwie systemu informatycznego.

Ostatecznie pozostaje tylko kwestia masowego zakładania kont, dlatego też dodatkowym zabezpieczeniem jest konieczność walidacji poprawności adresu email.

Realizacja tego zagadnienia w ten sposób sprawia, że SRU staje się jeszcze bardziej bezpieczniejszy ze względu na przekazywanie poświadczeń i wygodniejszy dla współpracujących z naszym systemem administratorów serwisów internetowych. Takie rozwiązanie wymaga jednak specjalnego rozwiązania kwestii autoryzacji dostępu użytkownika do wybranych serwisów internetowych implementujących usługę uwierzytelniania w naszym systemie, co bardziej szczegółowo opisano w kolejnym rozdziale.

Autonomiczność SRU implikuje także konieczność uwierzytelniania zarówno użytkowników jak i korzystających z naszych usług sieciowych serwisów internetowych. Ponieważ nie istnieją żadne różnice w uwierzytelnianiu użytkownika i zewnętrznej aplikacji internetowej, postanowiono wykorzystać ten

sam mechanizm uwierzytelniania i zarządzania użytkownikami w ASP.NET[3] na styku SRU z użytkownikami internetowymi oraz użytkownikami usług sieciowych. Konsekwencją tego jest konieczność wyboru rodzaju nowego konta podczas wypełniania formularza rejestracyjnego w celu nadania mu odpowiedniej roli w systemie.

Ostatnią, aczkolwiek jedną z bardziej istotnych kwestią poruszaną podczas tworzenia konta użytkownika jest polityka haseł w systemie. Sprowadza się to do odpowiedniej konfiguracji aplikacji ASP.NET, a dokładniej dostawcy mechanizmu Membership. Umożliwia on konfigurację takich wymagań co do złożoności hasła, aby było ono „silne", dzięki czemu konto jest odporne na ataki słownikowe typu „brute-force". [10]

1.3.2. Uwierzytelnienie i autoryzacja

Aplikacja internetowa wspomagająca pracę nauczyciela jest zgodna z charakterystyką stron dostępnych wyłącznie dla zamkniętej grupy użytkowników, dlatego też zadaniem budowanego systemu będzie kontrola dostępu do całego serwisu. Oznacza to, że strona internetowa serwisu klienckiego nie udostępni żadnych treści przed pomyślnym uwierzytelnieniem i autoryzacją użytkownika. Samo założenie konta w SRU pozwala nam tylko na natychmiastową możliwość uwierzytelnienia się w SRU. Mechanizm rejestracji wymaga tylko weryfikacji adresu e-mail oraz unikalnej nazwy użytkownika. Pozostałe dane nie mogą być wstępnie uznane za zgodne ze stanem faktycznym. O tym czy posiadacz konta w SRU może mieć dostęp do wybranego serwisu internetowego decyduje jego administrator. Dzięki udostępnionym usługom może zezwalać lub zabraniać dostępu do swojej witryny internetowej wybranym użytkownikom SRU. Decyzję o udzieleniu prawa do logowania się w danym serwisie internetowym ostatecznie podejmuje administrator tego serwisu. Reasumując, pomyślne zalogowanie się użytkownika w wybranym serwisie internetowym składa się z dwupoziomowego

[3] Framework dynamicznych aplikacji internetowych, zaprojektowany przez Microsoft. Umożliwia budowę stron, aplikacji internetowych oraz usług sieciowych.

uwierzytelnienia i autoryzacji. Najpierw użytkownik potwierdza swoją tożsamość w SRU, na podstawie której otrzymuje dostęp do jego zasobów. Następnie na podstawie tej uwierzytelnionej tożsamości, udzielany jest mu dostęp do wybranego serwisu internetowego. W serwisie partnerskim, czyli na przykład w aplikacji internetowej wspomagającej pracę nauczyciela następuje ponowne uwierzytelnienie i autoryzacja dostępu do jego zasobów.

Proces uwierzytelniania musi także występować na styku SRU i zewnętrznej aplikacji partnerskiej. Dzięki mechanizmowi ról, możliwe jest uwierzytelnienie takiej aplikacji internetowej bazując na poświadczeniach konta w SRU.

1.3.3. Role użytkowników w systemie

Uprawniony do korzystania z SRU użytkownik, czyli taki, który pomyślnie przechodzi proces logowania się do systemu podlega procesowi autoryzacji. System w zależności od rodzaju konta, a dokładniej w zależności od roli, do której należy dane konto udostępnia różne części serwisu. W systemie występować będą trzy role: rola administratora SRU, serwisu internetowego oraz rola zwykłego użytkownika.

Konto należące do roli administratora SRU posiada uprawnienia do zarządzania wszystkimi profilami użytkowników, założonych w systemie, czyli zarówno kontami użytkowników jak i kontami serwisów internetowych. Po zalogowaniu się na takie konto, administrator ma do dyspozycji panel służący podstawowej konfiguracji systemu. Rolę administratora można przydzielić jedynie podczas specjalnej konfiguracji aplikacji.

Kolejnymi dwiema rolami w systemie są: rola zwykłego użytkownika oraz rola serwisu internetowego. Konto może przynależeć do dokładnie jednej roli w danym momencie. Profile użytkowników należące do roli zwykłego użytkownika lub roli serwisu internetowego nie są tak krytyczne w kwestii bezpieczeństwa jak konta z rolą administratora, dlatego też typ konta wybierany jest samodzielnie przez internautę podczas wypełniania formularza rejestracyjnego.

Konto serwisu internetowego służy w systemie do identyfikacji i uwierzytelnienia zewnętrznej aplikacji internetowej, implementującej mechanizm

uwierzytelniania użytkowników. Administrator serwisu internetowego podczas korzystania z SRU używając tych samych poświadczeń może uwierzytelnić się w SRU przez stronę internetową systemu oraz w udostępnianych usługach sieciowych. Rola serwisu internetowego w SRU jest bardzo ważna, ponieważ na niej opiera się mechanizm autoryzacji dostępu wybranych użytkowników do serwisów internetowych. Dzieje się to na podstawie powiązań między kontami użytkowników i serwisów internetowych.

Konta zwykłych użytkowników pozwalają na uwierzytelnienie się w SRU w celu zarządzania kontem oraz uwierzytelnienia się w zewnętrznych, partnerskich serwisach internetowych. Pomiędzy kontami serwisów internetowych i kontami użytkowników istnieje powiązanie wiele do wielu. Umożliwia to serwisom internetowym na uwierzytelnianie i autoryzację wielu użytkowników, a także zwykłym użytkownikom logowanie się do wielu kooperujących z SRU serwisów internetowych.

1.3.4. Mechanizm grup

Mechanizm ról wbudowany w ASP.NET nie jest wystarczająco elastyczny, aby można było tworzyć drzewiaste struktury, umożliwiające na przykład dziedziczenie uprawnień. Aplikacja internetowa wspomagająca pracę nauczyciela, którą obsługuje w kwestii zarządzania użytkownikami SRU, rozgranicza dostęp do wybranych części i usług serwisu dla różnych rodzajów użytkowników. W systemie istnieją trzy podstawowe grupy użytkowników, które dzielą się na podgrupy: Administratorzy, Nauczyciele, Studenci oraz należące do grupy studentów grupy szkoleniowe.

Administratorzy mają dostęp do większości usług i danych zgromadzonych w systemie lub przynajmniej do ich administracyjnej części. Administrator jest osobą, która nie może zostać podpięta do grupy szkoleniowej lub nauczycielskiej. Jest też użytkownikiem niewidocznym dla użytkowników z innych grup. Nauczyciele to użytkownicy posiadający mniej uprawnień niż administratorzy. Z poziomu ich kont można zarządzać przydziałem studentów do przedmiotów. Ostatnią grupą użytkowników i zarazem najliczniejszą są studenci. Jest to grupa,

która posiada najmniej praw. O pozycji oraz prawach dostępu do zasobów zgromadzonych w systemie decydują nauczyciele i administratorzy. Warto zauważyć, że istnieje konieczność podziału grupy studentów na podgrupy. Wymaganie to wynika z potrzeby rozgraniczenia dostępu do zasobów szkoleniowych tylko dla studentów, którzy zostali zapisani na konkretny kurs.

Wymagania nakładane na sposób kontroli dostępu do wybranych zasobów serwisu, oparty na grupach, wymusza stworzenie specjalizowanego mechanizmu grup. Każdy serwis internetowy dzięki takiemu rozwiązaniu miałby możliwość tworzenia niezależnej struktury grup użytkowników, niewidocznej dla innych serwisów internetowych, współpracujących z SRU. Administrator serwisu internetowego, dzięki udostępnionym usługom sieciowym do tworzenia grup nadrzędnych, podrzędnych, usuwania ich oraz ich modyfikacji, a także przypisywania i usuwania użytkowników z grup, ma możliwość kontroli nad strefami, w których użytkownik może poruszać się w serwisie. Przykładowo student należący do grupy głównej „studenci” ma dostęp do części e-learningowej, ale ze względu na przynależność do podgrupy „I6G1S1” posiada dostęp do materiałów zgromadzonych tylko w obrębie tej grupy szkoleniowej. Problem implementacji mechanizmu autoryzacji dostępu do tych zasobów pozostaje w gestii aplikacji wspomagającej pracę nauczyciela.

1.3.5. Przypadki użycia dostępne dla różnych rodzajów kont w Systemie Rejestracji Użytkowników

Administrator SRU

Użytkownicy, których konta należą do roli Administratora SRU posiadają uprawnienia krytyczne dla bezpieczeństwa systemu. Administrator SRU posiada dostęp do podstawowej konfiguracji systemu, a także ma prawo do zarządzania wszystkimi użytkownikami systemu. Może usuwać, aktywować, dezaktywować oraz edytować dowolne konto zarejestrowane w systemie.

Serwis internetowy

Nieco mniej funkcji udostępnianych przez stronę internetową SRU posiadają serwisy internetowe. Po zalogowaniu mają one dostęp do panelu administracyjnego umożliwiającego zmianę adresu URL danego serwisu. SRU udostępnia dla tego rodzaju kont szereg usług sieciowych, dostępnych wyłącznie z poziomu aplikacji internetowej powiązanej z tym kontem. Udostępniane usługi realizują zadania uwierzytelniania, autoryzacji, a także operacje związane z zarządzaniem użytkownikami i grupami.

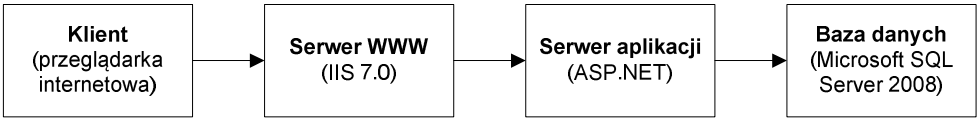
Użytkownik zwykły systemu

Trzecim rodzajem kont jest konto zwykłego użytkownika systemu. Oprócz podstawowych funkcji służących do zmiany hasła lub adresu e-mail osoba posługująca się takim kontem posiada możliwość zgłaszania prośby o autoryzację dostępu do wybranych serwisów internetowych. W panelu zarządzania kontem, użytkownik posiada listę serwisów internetowych, do których może wysłać zgłoszenie. Po pomyślnym zatwierdzeniu zgłoszenia użytkownika przez administratora serwisu internetowego, użytkownik zyskuje możliwość dostępu do zasobów wybranej strony WWW zaraz po wpisaniu jej adresu URL.

II. ANALIZA METOD ZABEZPIECZENIA DANYCH PRZESYŁANYCH ORAZ PRZECHOWYWANYCH POD KĄTEM WYKORZYSTANIA W SYSTEMIE

2.1. Analiza bezpieczeństwa przesyłanych danych

Internet jest siecią połączonych ze sobą komputerów, który powstał w wyniku zestawienia połączenia pomiędzy mniejszymi sieciami komputerowymi, lokalnymi, miejskimi, krajowymi itp. Zanim informacja od nadawcy trafi do odbiorcy, przechodzi przez pewną liczbę urządzeń sieciowych. Na każdym z tych urządzeń możliwe jest, jeśli informacja nie jest odpowiednio zabezpieczona, podejrzenie, skopiowanie lub zmiana przesyłanej informacji. Do informacji poufnych zaliczają się dane identyfikacyjne potrzebne do uwierzytelnienia w systemie. Wrażliwymi informacjami są także identyfikatory sesji zestawionych między użytkownikiem, a aplikacją. Niebezpieczeństwo wzrasta jeśli węzły aplikacji znajdują się na oddzielnych maszynach komunikujących się przez sieć.



Rys. 2. Model aplikacji System Rejestracji Użytkowników

Na rysunku 2 przedstawiono drogę przesyłania żądania wystosowanego przez klienta. Pomiędzy klientem a serwerem WWW najczęściej połączenie zestawiane jest przez Internet lub sieć lokalną. Aby zapewnić prywatność oraz integralność takiego połączenia, można użyć protokołu SSL. [1, s.14-15] W przypadku dwóch pozostałych połączeń, szyfrowanie nie jest wymagane, ponieważ rolę serwera WWW, aplikacji oraz serwera bazy danych pełni ten sam komputer. Wprowadzenie zabezpieczeń jest istotne, gdy rolę serwera WWW, aplikacji i bazy danych pełnią różne maszyny porozumiewające się przez sieć nienadzorowaną, np. Internet. W takim wypadku jednym z rozwiązań wzmacniających bezpieczeństwo połączenia

między tymi serwerami jest utworzenie wirtualnej sieci prywatnej VPN[4], do której należą wszystkie serwery pełniące różne role w systemie.

Zanim przejdę do omówienia sposobów bezpiecznego przesyłania informacji przybliżę atrybuty takiej transmisji, a są to [1, s.55]:

- **Prywatność** – dane nie zostały odczytane przez osoby postronne, np. za pomocą oprogramowania do monitorowania sieci,
- **Integralność** – dane są integralne, czyli zabezpieczone przed przypadkową lub celową modyfikacją transmitowanych danych.

SRU w celu zabezpieczenia połączenia z przeglądarką internetową klienta wykorzystuje protokół SSL[5]. Protokół ten pozwala na utworzenie zaszyfrowanego kanału komunikacji pomiędzy komputerem klienta, a serwerem WWW. Decydując się na zastosowanie tego rozwiązania powinniśmy odpowiedzieć na kilka ważnych pytań związanych z SSL, takich jak [1, s.57]:

- Czy klient korzysta z protokołu HTTPS[6] podczas komunikacji z witryną?
- Czy serwer odbiera komunikację za pomocą portu TCP[7] 443?
- Czy wydajność aplikacji nie spadła, utrudniając korzystanie z niej?
- Czy strona nie posiada zbyt wielu elementów graficznych, podlegających szyfrowaniu?
- Czy posiadamy uprawnienia do instalacji certyfikatu uwierzytelniania oraz czy posiadamy podpisany cyfrowo certyfikat?

Stosowane w aplikacji uwierzytelnienie w trybie Forms wymaga zabezpieczenia komunikacji za pomocą SSL. Bardzo ważnym jest, aby oprócz

---

[4] (ang. Virtual Private Network), *wirtualna prywatna sieć komputerowa. Polega na łączeniu ze sobą kilku sieci komputerowych bezpiecznym, szyfrowanym kanałem danych, za pośrednictwem sieci publicznej, np. Internetu.* [4]

[5] (ang. Secure Socket Layer), protokół zapewniający poufność i integralność danych, oparty na szyfrowaniu asymetrycznym.

[6] (ang. HyperText Transfer Protocol Secure), protokół przesyłania dokumentów hipertekstowych, szyfrujący przesyłane dane za pomocą protokołu SSL.

[7] (ang. Transmission Control Protocol), protokół wykorzystywany do komunikacji w sieci, działający w trybie klient-serwer. Serwer oczekuje połączeń na wybranym porcie. Klient ustanawia połączenie wysyłając pakiet na odpowiedni port serwera.

szyfrowania danych identyfikacyjnych przesyłanych do serwera WWW, zaszyfrować też wszystkie strony. Ma to na celu ochronę pliku cookie[8] wygenerowanego zaraz po uwierzytelnieniu. Posiadanie ważnego pliku cookie innego klienta daje możliwość bez znajomości jego nazwy użytkownika i hasła, odtworzenie pozostawionego przez klienta stanu aplikacji.

2.2. Analiza bezpieczeństwa przechowywanych danych.

W kontekście omawianego systemu za bezpieczne przechowywanie danych po stronie serwera można uznać ich postać, która po włamaniu na serwer będzie bezużyteczna dla intruza. Dane przechowywane w systemie wymagane do zalogowania się w serwisie internetowym to w większości przypadków ustalony przez użytkownika login oraz hasło do konta na podstawie, których system decyduje o uwierzytelnieniu logującego się. To właśnie te dane są głównym celem atakującego. Metody z dziedziny kryptografii pozwalają na umocnienie tej podatności. Dobrym przykładem jest kryptograficzny algorytm funkcji skrótu MD5, który z dowolnego ciągu tekstowego tworzy 128-bitowy ciąg znaków w systemie szesnastkowym. Przykładowo dla frazy „System Rejestracji Uzytkowników" skrót MD5 równa się:

889CA911BED442A7A5BF1B8B52D67EF2

Minimalna zmiana w tekście, powoduje wygenerowanie zupełnie odmiennego skrótu. Przykładem może być skrót dla frazy „Systemy Rejestracji Uzytkowników", który przybierze postać:

F272459A72C1CEA45FC1A75E82D9DD55

Kluczową własnością skrótów jest to, że nie jest możliwe ich odwrócenie w tekst, z którego zostały wygenerowane, dlatego często stosowanym zabiegiem jest przechowywanie w bazie danych haseł, wprowadzanych przez użytkowników w postaci skrótu MD5. Hasło wprowadzane podczas logowania do systemu

[8] Plik zawierający małą ilość danych, wysłanych przez serwer WWW w celu zapisania ich na dysku twardym użytkownika. Domyślnie, zapisane w tym pliku informacje mogą być odczytane tylko przez serwer, który je utworzył.

zamieniane jest za pomocą tego samego algorytmu na skrót i porównywane z przechowywanym w bazie, utworzonym podczas rejestracji skrótem MD5 prawidłowego hasła. Dzięki takiemu aspektowi polityki bezpieczeństwa, dane wykradzione z systemu są praktycznie bezużyteczne dla przestępcy internetowego.

Aktualnie algorytm MD5 uważany jest za niebezpieczny, ponieważ możliwe jest znalezienie kolizji funkcji haszującej, czyli dwóch zupełnie różnych wiadomości, których skróty będą sobie równe. Atakujący wykradając skróty haseł może za pomocą ataku urodzinowego znaleźć hasło, którego skrót będzie identyczny z przechowywanym w bazie danych.

Mechanizm ASP.NET Membership także wykorzystuje jednokierunkową funkcję haszującą w celu utworzenia skrótu haseł, wprowadzanych przez użytkowników. Funkcja mieszająca w ASP.NET oparta jest głównie na dużo bezpieczniejszym od MD5 algorytmie SHA1[9]. Dodatkowo hasło przed utworzeniem skrótu poddawane jest wielu operacjom konwersji, które jeszcze bardziej utrudniają odgadnięcie hasła, zgodnego z utworzonym w bazie danych skrótem.

2.3. Odporność na ataki

W informatyce, atak sieciowy jest umyślnym działaniem, którego celem jest zakłócenie prawidłowego działania systemu teleinformatycznego, ograniczenie dostępu, kradzież lub zniszczenie danych przetwarzanych przez system. W przypadku projektowanego przeze mnie systemu najsłabszym punktem jego bezpieczeństwa są pola logowania, w które użytkownik sam wprowadza dane. Celem uodpornienia systemu jest zabezpieczenie się przed wszelkiego rodzaju próbami przejęcia kontroli nad serwisem internetowym za pomocą wprowadzania odpowiednio spreparowanych danych.

[9] Kryptograficzna funkcja skrótu tworząca 160-bitowy skrót wiadomości, zaprojektowana przez NSA.

SQL Injection

W dosłownym znaczeniu jest to „zastrzyk SQL". Jest jedną z najczęściej spotykanych luk w zabezpieczeniach aplikacji internetowych. Jej wykorzystanie polega na wprowadzeniu w pola formularza odpowiednio przygotowanych fragmentów kodu SQL[10], które spowodują wykonanie nieoczekiwanych zapytań SQL do bazy danych. Dane wprowadzane do formularza w postaci nazwy użytkownika i hasła „wklejane" są do odpowiednich zapytań SQL w celu weryfikacji, czy w bazie danych istnieje użytkownik z takimi poświadczeniami. Podatność wynika z niewystarczającej filtracji danych pochodzących od użytkownika. Typ ataku opiera się na dobraniu znaków specjalnych umożliwiających wyjście poza obszar wprowadzania danych. Ma to na celu wykonanie w systemie własnego zapytania SQL. Przykładem znaków ucieczki z pola wprowadzania danych jest apostrof oraz średnik. Umożliwia to wprowadzenie własnego zapytania, na przykład tak jak w poniższym przykładzie, usuwającego tabelę „użytkownicy". [1, s.323-327]

```
X'; DROP TABLE użytkownicy
```

Całe zapytanie przyjmie postać:

```
SELECT * FROM użytkownicy WHERE user='x'; DROP TABLE użytkownicy
```

Najłatwiejszą metodą zabezpieczenia przed atakiem jest implementacja na poziomie aplikacji odpowiedniego filtru usuwającego niedozwolone znaki z wprowadzonych przez użytkownika ciągów tekstowych. Nie jest to jednak jedyna metoda.

Dobrą i sprawdzoną metodą ochrony aplikacji przed tego typu atakiem jest parametryzacja zapytań SQL. Polega ona na przekazywaniu do zapytania wartości zmiennych poprzez wypełnianie jego parametrów. W języku C# służy do tego metoda AddWithValue, która przyjmuje na swoim wejściu nazwy parametrów oraz

[10] (ang. Structured Query Language), język zapytań wykorzystywany głównie do zarządzania danymi w bazach danych.

ich wartości. Jej działanie polega na właściwym, w zależności od typu wprowadzanych danych, wypełnieniu parametrów zapytania SQL oraz bezpiecznej filtracji wprowadzanych danych. Przykładowe zastosowanie:

```
SqlCommand cmd = new SqlCommand("SELECT * FROM Uzytkownicy WHERE
  login=@nazwa_uzytkownika AND password=HASHBYTES('SHA1',
  @haslo)", DbConnector);

  cmd.Parameters.AddWithValue("@nazwa_uzytkownika", user);
cmd.Parameters.AddWithValue("@haslo", pass);
```

Atak słownikowy

Technika używana do siłowego odgadywania haseł lub kluczy. Polega na wprowadzaniu pozycji z określonego zbioru słownikowego, na przykład popularnych haseł lub listy słów występujących w danym języku. [1, s.554] Dobrą metodą zabezpieczenia przed takim atakiem jest wprowadzenie systemu anti-flood, który określa minimalny czas odstępu między kolejnymi próbami zalogowania. Przy nawet minimalnych interwałach czasowych odgadnięcie hasła wymaga dużej ilości czasu, a w większości przypadków atak jest już do tej pory namierzany przez administratora.

Cross-site forgery (CSRF lub XSRF)

Ta metoda ataku na stronę internetową polega na zmuszeniu nieświadomego użytkownika, posiadającego pożądane uprawnienia, do przesłania na serwer odpowiednio spreparowanych zapytań, które umożliwią wykonanie przez atakującego odpowiednich operacji wymagających uprawnień posiadanych przez wykorzystywanego użytkownika. Najczęściej spreparowaną daną jest odnośnik do atakowanej strony internetowej, który podsyłany jest nieświadomemu zagrożenia użytkownikowi. Atak CSRF wykorzystywany jest w serwisach wymagających logowania. Głównym celem jest wykorzystanie zaufania serwisu do tożsamości użytkownika. Na to, że użytkownik w końcu nauczy się spełniania warunków bezpiecznego korzystania z serwisu projektant systemu nie może liczyć, dlatego też twórcy stron wykorzystują szereg metod, które utrudniają atak CSRF. Z pośród

silniejszych zabezpieczeń wobec tego typu ataków są między innymi: hasła jednorazowe, skrócenie czasu ważności klucza sesyjnego, żądanie ponownego uwierzytelnienia podczas dostępu do newralgicznych elementów systemu, implementowanie dodatkowego, ukrytego pola tekstowego z losowym ciągiem znaków w celu utrzymywania kontekstu wywołań strony internetowej. [5, s.95, s.99]

Cross-site scripting (XSS)

Ten sposób ataku na serwis WWW podobnie jak SQL Injection polega na wprowadzeniu do systemu odpowiednio przygotowanych danych poprzez dostępne formularze. Najczęściej wykorzystanymi, w kontekście tego typu ataku, są skrypty napisane w języku Javascript. W tym przypadku także nieumiejętne filtrowanie danych może spowodować wykonanie przez przeglądarkę niepożądanych akcji. [5, s.95, s.99]

Przechwytywanie sesji

Sesja jest obiektem przechowującym przez pewien czas na serwerze szczegóły interakcji z klientem. Mechanizm sesji przypomina pamięć podręczną o charakterze nietrwałym i ulotnym. Sesja podtrzymywana jest podczas każdego zapytania ze strony klienta, czyli trwa tak długo, dopóki użytkownik uzyskuje dostęp do stron internetowych aplikacji. [1, s.564] Niebezpieczeństwo używania tego mechanizmu bez dodatkowych zabezpieczeń wiąże się z możliwością przechwycenia sesji, potocznie zwanym „Session hijacking”.

W przypadku, kiedy użytkownik uwierzytelnił się w systemie, przydzielany jest mu specjalny identyfikator pozwalający rozpoznać go przy następnym żądaniu, umożliwiający przywrócenie poprzedniego stanu aplikacji. Identyfikator ten przesyłany jest tak jak reszta danych w sposób jawny. Z racji tego, że protokół HTTP jest bezstanowy, dzięki sprawnej i szybkiej reakcji atakującego, identyfikator sesji może posłużyć atakującemu do przechwycenia stanu aplikacji uwierzytelnionego użytkownika oraz przejęcia kontroli nad jego kontem. Do odczytania takiego identyfikatora może posłużyć monitor sieci. Innym sposobem na

pozyskanie takiego klucza jest wykorzystanie omawianego przeze w tym rozdziale ataku XSS. Jedną z metod eliminowania takich podatności jest implementacja szyfrowania danych SSL oraz okresowa zmiana klucza sesyjnego. [3, s.24]

III. ANALIZA MECHANIZMÓW AUTORYZACJI DOSTĘPNYCH W ASP.NET ORAZ WINDOWS® POD KĄTEM WYKORZYSTANIA W SYSTEMIE

Możliwości konfiguracji sposobów uwierzytelniania i autoryzacji w .NET Framework w połączeniu z miejscami ich zastosowania w architekturze aplikacji jest tak dużo, że streszczenie ich wszystkich wykraczałoby poza ramy niniejszej pracy. W kontekście SRU, któremu w głównej mierze poświęcona jest niniejsza praca, przeanalizowano mechanizmy uwierzytelniania i autoryzacji pod kątem ich implementacji w SRU.

Zaprojektowanie odpowiedniej strategii uwierzytelniania i autoryzacji w kontekście rozproszonych aplikacji internetowych ma kluczowe znaczenie w kwestii bezpieczeństwa projektowanego systemu. Aby ułatwić sobie zadanie wyboru strategii autoryzacji, projektant aplikacji musi odpowiedzieć na kilka pytań, biorąc pod uwagę wymagania nakładane na budowany system:

- Gdzie zastosować autoryzację? [1, s.21]

  Kontrola dostępu może zachodzić przy dostępie do:

  - usługi IIS,
  - wybranych części aplikacji bazując na URL,
  - wybranych zasobów, np. plików,
  - metod, klas i poszczególnych bloków kodu programu.

- Jaki mechanizm uwierzytelniania wybrać? [1, s.30]

  W zależności od rodzaju aplikacji należy wybrać mechanizm uwierzytelniania zwracając uwagę na następujące kwestie:

  - Rodzaj tożsamości.

    Tryb uwierzytelniania Windows wymaga, aby użytkownicy aplikacji posiadali konta Windows,

  - Magazyn danych identyfikacyjnych.

    W przypadku uwierzytelniania Windows dane identyfikacyjne użytkowników zarządzane są przez system. Uwierzytelnianie Forms

wymaga natomiast zdefiniowania miejsca, w którym będą one przechowywane. Może to być baza danych SQL lub katalogu Active Directory,

- Sposób przekazywania informacji o tożsamości użytkowników do niższych warstw aplikacji.

  Zastosowanie kontroli dostępu na poziomie list ACL[11] wymaga przekazania (imitowania) tożsamości przez wątek bieżącego żądania, pochodzącego od użytkownika w celu prawidłowego zidentyfikowania użytkownika przez system operacyjny.

3.1. Wybór strategii uwierzytelniania i autoryzacji

Identyfikacja rodzaju udostępnianych przez aplikację zasobów

W pierwszym kroku należy ustalić jakie rodzaje zasobów udostępniać będzie projektowana przez nas aplikacja. Zasoby te można podzielić na trzy rodzaje:

- zasoby serwera WWW, tj: strony WWW, usługi sieciowe, strony HTML, obrazy,
- zasoby magazynu danych,
- zasoby sieciowe, tj: pliki, zasoby Active Directory.

W przypadku SRU udostępniane są zasoby serwera WWW oraz usługi sieciowe. Ustalenie rodzaju aplikacji jest pomocne przy wyborze strategii autoryzacji.

3.1.1. Wybór strategii autoryzacji

W ASP.NET dostępne są dwie strategie autoryzacji. Autoryzacja oparta na rolach oraz autoryzacja oparta na zasobach.

[11] (ang. Access Control List), lista kontroli dostępu definiująca uprawnienia do zasobów i usług.

Autoryzacja oparta na rolach

W tym przypadku użytkownicy wiązani są ze zdefiniowanymi w aplikacji logicznymi rolami. W zależności od przypisanej roli, system decyduje o dostępie do wybranej operacji. Wykonanie kodu metody poprzedzane jest sprawdzeniem przynależności podmiotu wywołującego do roli w systemie. Na tej podstawie system decyduje, czy podmiot ten posiada uprawnienia do wykonywanych przez tą metodę operacji. Role pełnią funkcję logicznych kontenerów, zawierających przynależnych do nich użytkowników. [1, s.30]

Autoryzacja oparta na zasobach

Użycie tego systemu autoryzacji pozwala na zabezpieczenie zasobów za pomocą list ACL. Listy te definiują dla każdego zasobu rodzaje operacji jakie mogą być wykonywane przez poszczególnych użytkowników. Autoryzacja oparta na zasobach deleguje proces kontroli dostępu systemowi operacyjnemu, który wykonuje zasadniczą część autoryzacji. Aplikacja dokonuje imitacji tożsamości użytkownika wywołującego, dzięki czemu system może wykorzystać własnych menadżerów zasobów. Rozwiązanie to sprawdza się w aplikacjach udostępniających pliki, np. aplikacjach do zarządzania FTP. [1, s.31]

Spośród tych dwóch strategii autoryzacji, pierwsza wykorzystywana jest w aplikacjach .NET zdecydowanie częściej ze względu na większą uniwersalność. Autoryzacja oparta na zasobach jest odpowiednia jedynie dla aplikacji pracujących z użytkownikami w obrębie jednej sieci Intranet[12], wykorzystującej autoryzację dostępu do zasobów za pomocą list ACL.

Biorąc pod uwagę to, że SRU nie jest aplikacją zorientowaną na dane, głównym celem systemu jest świadczenie usługi uwierzytelnienia dla anonimowych użytkowników oraz to, że aplikacja musi działać w Internecie, zastosowanie autoryzacji opartej na zasobach jest w tym przypadku nie możliwe.

[12] Sieć komputerowa obejmująca zasięgiem komputery jednej organizacji, w której w obrębie sieci LAN udostępnia się usługi WWW, FTP, e-mail.

3.1.2. Wybór tożsamości w celu autoryzacji dostępu do zasobów

W przypadku, kiedy autoryzacja dostępu do zasobów zachodzi na niższej warstwie aplikacji, istnieje potrzeba przekazania informacji o tożsamości do systemu operacyjnego lub bazy danych realizującej funkcję strażnika dostępu. W zależności od obranej strategii autoryzacji musimy wybrać tożsamość używaną do uzyskania dostępu w niższych warstwach aplikacji [1, s.22]:

- **Tożsamość użytkownika** – w tym przypadku należy przekazać informacje o tożsamości podmiotu wywołującego do niższej warstwy,
- **Tożsamość procesu** – dostęp do wybranych zasobów odbywa się za pośrednictwem tożsamości bieżącego procesu,
- **Konto usługowe** – jest to bardzo popularne rozwiązanie w połączeniu z autoryzacją opartą na rolach. Dostęp do zasobów zaplecza uzyskiwany jest za pomocą stałych tożsamości usługowych, np. dostęp do bazy danych realizowany jest dzięki uwierzytelnieniu z wykorzystaniem stałych poświadczeń do konta SQL,
- **Tożsamość niestandardowa** – pozwala na tworzenie własnych tożsamości zawierających kontekst zabezpieczeń. Wykorzystanie klas IPrincipal i IIdentity pozwala na wykorzystanie ról .NET oraz żądania uprawnień podmiotu.

Strategia autoryzacji oparta na rolach nie wymaga przekazywania tożsamości użytkownika do niższych warstw aplikacji. Wybór stałej tożsamości usługowej, w celu uzyskiwania dostępu do zasobów, czyli bazy danych SQL jest w tym przypadku najlepszym rozwiązaniem.

3.1.3. Wybór mechanizmu uwierzytelniania

Procesy uwierzytelniania i autoryzacji są sobą powiązane, ponieważ wszelkie zasady autoryzacji muszą opierać się na uwierzytelnionym użytkowniku. ASP.NET udostępnia cztery tryby uwierzytelniania.

Zintegrowane uwierzytelnianie Windows

Polega na wykorzystaniu istniejących w systemie Windows kont użytkowników. Stosowana metoda uwierzytelniania kontrolowana jest przez usługi IIS. Identyfikacja uwierzytelnionego użytkownika polega na asocjacji żądania HTTP z obiektem WindowsPrincipal, który zawiera szczegółowe informacje o tożsamości uwierzytelnionego klienta. Obiekt ten zawiera także listę ról, do których użytkownik należy. W przypadku tego trybu uwierzytelniania role są automatycznie grupami w systemie Windows. [1, s.16]

Uwierzytelnianie Forms

W aplikacji implementującej ten mechanizm, każda próba uzyskania dostępu do zasobów chronionych przez nieuwierzytelnionego użytkownika powoduje przeniesienie go do strony z formularzem logowania. Użytkownik kierowany jest do strony logowania określonej w pliku konfiguracyjnym aplikacji. Na stronie logowania podaje dane identyfikacyjne i przesyła je do serwera. W zależności od architektury naszej aplikacji, dane te weryfikowane są w bazie danych SQL Server lub katalogu Active Directory. Po pomyślnym uwierzytelnieniu tworzony jest bilet cookie zawierający obiekt FormsAuthenticationTicket, który przekazywany jest klientowi. Następnie klient przenoszony jest do żądanego zasobu. Użytkownik w wyniku nowego żądania do zasobu, po uwierzytelnieniu, przesyła utworzony plik cookie, na podstawie którego ASP.NET dokonuje autoryzacji. [1, s.16]

Uwierzytelnianie Passport

Na chwilę obecną rozwiązanie to używane jest jedynie przez samego twórcę tej metody uwierzytelniania, czyli firmę Microsoft. Usługa ta polega na korzystaniu z udostępnionego, scentralizowanego systemu uwierzytelniania Microsoft Passport (obecnie Windows Live ID). Dzięki tej usłudze użytkownik może uwierzytelnić się w dowolnej witrynie kompatybilnej z .NET Passport przy pomocy jednokrotnego logowania. Po pomyślnym uwierzytelnieniu w witrynie usługi, użytkownik jest przenoszony z powrotem do witryny docelowej. Warto zauważyć, że

uwierzytelnianie Passport zapewnia wysoki poziom bezpieczeństwa z racji wykorzystywanych technologii kryptograficznych oraz samego faktu przechowywania danych identyfikacyjnych poza aplikacją. Przy okazji omawiania tego trybu uwierzytelniania warto zwrócić uwagę na duże podobieństwo tego rozwiązania z rozwiązaniem zastosowanym w SRU, który także pełni rolę scentralizowanego systemu uwierzytelniania.

Uwierzytelnianie niestandardowe

Korzystanie z wbudowanych w .NET Framework modułów uwierzytelniania nie jest obowiązkowe. W przypadku gdy aplikacja posiada zaplanowaną, niestandardową strategię uwierzytelniania, na przykład wykorzystującą zewnętrzne istniejące już aplikacje, mamy możliwość takiej konfiguracji, w której żaden moduł uwierzytelniania nie zostanie uruchomiony. Taką konfigurację może z powodzeniem przyjąć aplikacja wspomagająca pracę nauczyciela, o której wielokrotnie wspominano już w niniejszej pracy. W tym przypadku wykorzystuje ona zewnętrzny system uwierzytelniania, czyli SRU.

Wybór mechanizmu uwierzytelniania nie jest łatwy i wymaga przeanalizowania kilku kwestii. Pierwszą z nich jest tożsamość użytkownika. Przekazywanie jej w oparciu o uwierzytelnianie Windows jest dostępne tylko dla aplikacji, których użytkownicy posiadają konta na serwerze Windows. Dyskwalifikuje więc to wybór tego trybu uwierzytelniania w przypadku budowanego SRU. Drugą kwestią jest wybór magazynu zarządzania danymi identyfikacyjnymi. W przypadku uwierzytelniania Windows, zarządzaniem danymi identyfikacyjnymi użytkowników zajmuje się system operacyjny. Tryb uwierzytelniania Forms wymaga ustanowienia dodatkowej przechowalni, np. bazy danych SQL lub katalogu Active Directory. W przypadku autoryzacji opartej na zasobach, dobór uwierzytelniania musi także uwzględniać przepływ tożsamości pomiędzy warstwami aplikacji.

3.2. Zastosowanie mechanizmów uwierzytelniania i autoryzacji w Systemie Rejestracji Użytkowników

Zgodnie z wymaganiami niefunkcjonalnymi opisanymi w rozdziale I, system musi być dostępny dla użytkowników niezależnie od ich przeglądarki internetowej oraz wykorzystywanego systemu operacyjnego.

Są to wymagania nakładane na większość internetowych aplikacji, dlatego też w SRU wykorzystano popularny model autoryzacji składający się z następujących etapów.

- Uwierzytelnianie użytkowników w trybie formularzy.

  Tryb ten nie wymaga posiadania konta w systemie Windows.
- Wykorzystanie mapowania użytkowników na role.

  Jest to najbardziej adekwatne rozwiązanie grupowania użytkowników według uprawnień dla tego typu aplikacji. System nie zajmuje się udostępnianiem zasobów, więc nie ma konieczności implementacji sztucznych mechanizmów kontroli dostępu do zasobów. Dodatkowo aplikacja zachowuje skalowalność.
- Autoryzacja dostępu w oparciu o powiązanie z rolami.

  Rozwiązanie to pozwala w łatwy sposób, polegający na sprawdzaniu przynależności użytkownika do roli, kontrolować dostęp do wybranych usług w systemie oraz wybranych jego części.
- Uzyskanie dostępu do zasobów bazy danych za pomocą stałego konta usługowego.

  W przypadku SRU autoryzacja dostępu do wybranych metod w systemie następuje już na poziomie aplikacji, dlatego też zastosowanie stałej tożsamości usługowej przy dostępie do zasobów jest bezpiecznym i łatwym rozwiązaniem.

W SRU zasoby systemu zostały podzielone na trzy części, do których dostęp posiadają użytkownicy należący do trzech różnych ról opisanych w rozdziale pierwszym. .NET Framework pracujący na platformie Windows 2008 udostępnia

cztery opcje autoryzacji ASP.NET: autoryzację opartą na URL, autoryzację opartą na pliku, żądanie uprawnień podmiotów oraz role .NET.

Do realizacji kontroli dostępu do wybranych stref SRU wykorzystano autoryzację opartą na URL oraz role .NET. Mechanizm ról pozwala podzielić użytkowników na grupy. Podmiotami wywołującymi usługi w SRU są: Użytkownicy, Serwisy internetowe oraz Administratorzy SRU. Każda z tych grup posiada uprawnienia dostępu do odrębnych dla siebie części systemu. W celu zastosowania opcji autoryzacji opartej na URL, aplikacja została zaprojektowana tak, aby zasoby dostępne dla tych grup zostały podzielone na foldery. Mechanizm ten zarządzany jest za pomocą pliku konfiguracyjnego aplikacji. Plik ten umożliwia ustawienie reguł kontroli dostępu dla wybranych użytkowników, do plików lub katalogów. Przykładowo:

- https://localhost/User/Accountdetails.aspx - lokalizacja dostępna dla Użytkowników,
- https://localhost/Admin/Accountdetails.aspx - lokalizacja dostępna dla Administratorów SRU,
- https://localhost/Referrer/Accountdetails.aspx - lokalizacja dostępna dla Serwisów Internetowych (administratorów tych serwisów).

IV. PROJEKT SYSTEMU

4.1. Cele

Celem projektowanego systemu jest oferowanie usług uwierzytelniania, autoryzacji i zarządzania użytkownikami na wysokim poziomie bezpieczeństwa. Gotowe rozwiązania SRU w wyżej wymienionych dziedzinach są przydatne dla serwisów nie posiadających mechanizmów zarządzania użytkownikami, autoryzacji i bezpiecznego uwierzytelniania. Podczas projektowania systemu skoncentrowano się na bezwzględnym spełnieniu wszystkich stawianych wymagań oraz na zapewnieniu jak największej skalowalności i elastyczności aplikacji.

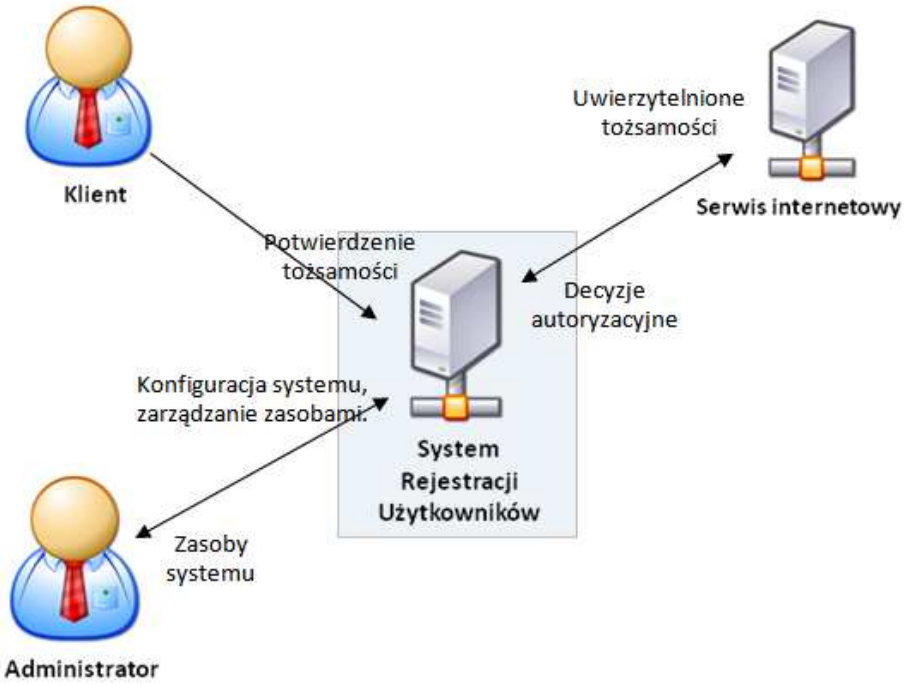
Wdrożenie usług SRU w serwisie internetowym powinno jak najmniej ingerować w jego strukturę oraz być proste w instalacji. System ten powinien przejąć obsługę uwierzytelnienia i autoryzacji użytkownika, a następnie udostępnić tożsamość uwierzytelnionego użytkownika po poprawnym sprawdzeniu poświadczeń. SRU musi także udostępnić szereg usług umożliwiających kontrolę dostępu użytkowników do serwisów internetowych. Z powodu całkowitego braku mechanizmu zarządzania użytkownikami system powinien dawać możliwość dowolnego grupowania użytkowników.

Użytkownik korzystający z aplikacji internetowej „Dziennik Nauczyciela” powinien w jak najbardziej przeźroczysty dla niego sposób zostać uwierzytelniony i autoryzowany do zasobów aplikacji docelowej, tak jakby poruszał się w ramach jednego spójnego systemu. SRU musi udostępniać także mechanizmy służące kontroli nad kontem użytkownika w systemie.

W celu wygodniejszej administracji SRU powinno rozróżniać się także użytkowników z uprawnieniami administratora SRU, dla których udostępniany będzie panel umożliwiający kontrolę nad kontami wszystkich użytkowników w systemie.

4.2. Model środowiskowy Systemu Rejestracji Użytkowników

Rozpoczynając projektowanie systemu dokonano jednoznacznego określenia obiektów zewnętrznych wchodzących w interakcję z SRU. Na diagramie kontekstowym z rysunku 3, SRU został przedstawiony jako pojedynczy proces, komunikujący się z obiektami zewnętrznymi.



Rys. 3. Diagram kontekstowy Systemu Rejestracji Użytkowników

Lista zdarzeń w kontekście SRU:

1) Klient – zwykły użytkownik:
    - rejestruje się,
    - loguje się,
    - uzyskuje dostęp do partnerskiego serwisu internetowego,
    - zmienia hasło,
    - zmienia e-mail.
2) Serwis internetowy – aplikacja internetowa:
    - otrzymuje informacje o tożsamości uwierzytelnionego użytkownika,
    - przydziela i zabiera prawo do autoryzacji w systemie wybranym użytkownikom,
    - tworzy oraz usuwa grupy użytkowników,
    - przyłącza i odłącza użytkowników od grup,

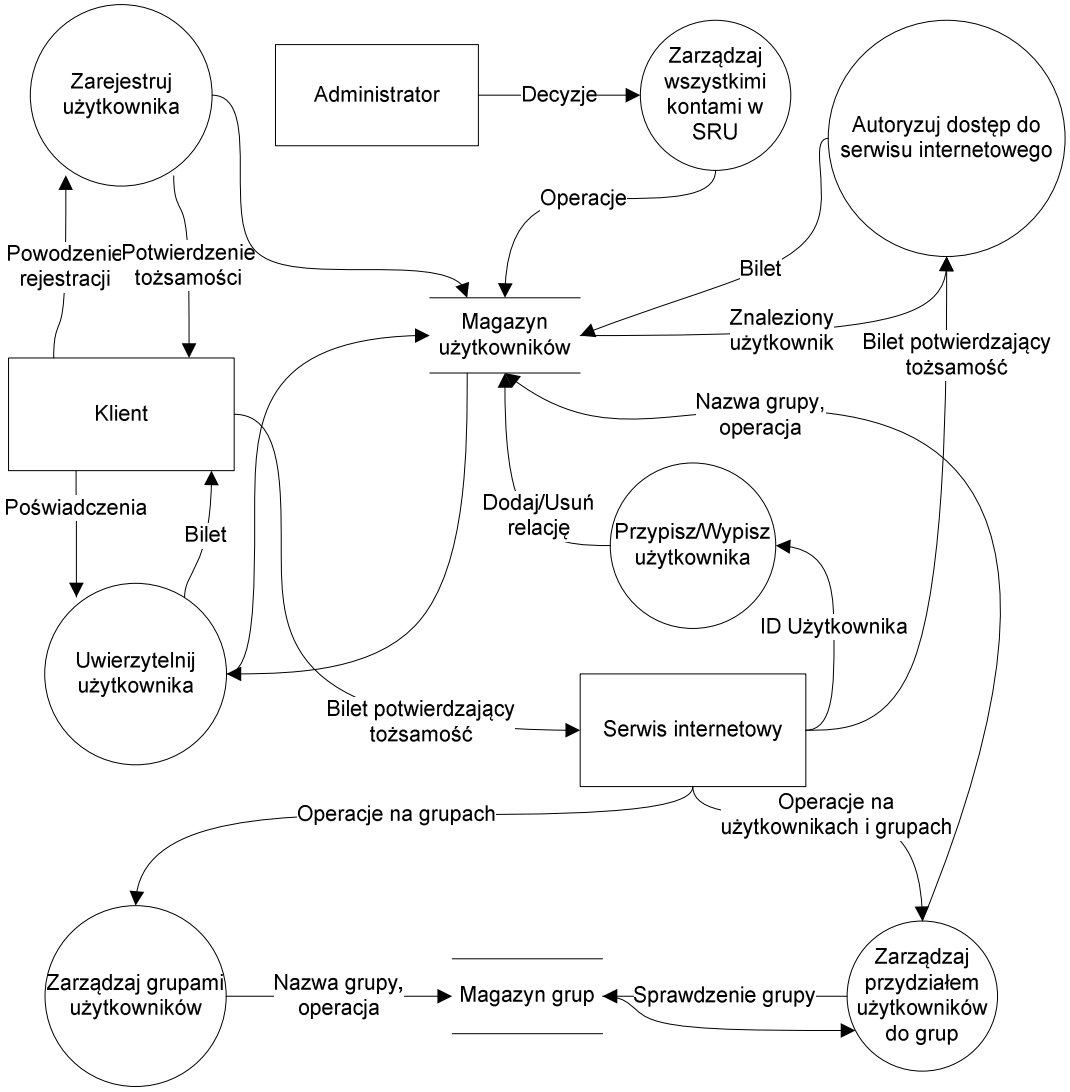
- wskazuje pod jaki adres uwierzytelniony użytkownik ma zostać przeniesiony.

3) Administrator SRU:

- aktywuje i dezaktywuje konta użytkowników i serwisów internetowych,
- usuwa konta z systemu,
- zmienia adres e-mail przypisany do konta.

4.3. Struktura funkcjonalna systemu

Diagram przepływu danych o zerowym (systemowym) stopniu szczegółowości, przedstawiony na rysunku 4, pozwala na zobrazowanie głównych funkcji systemu.

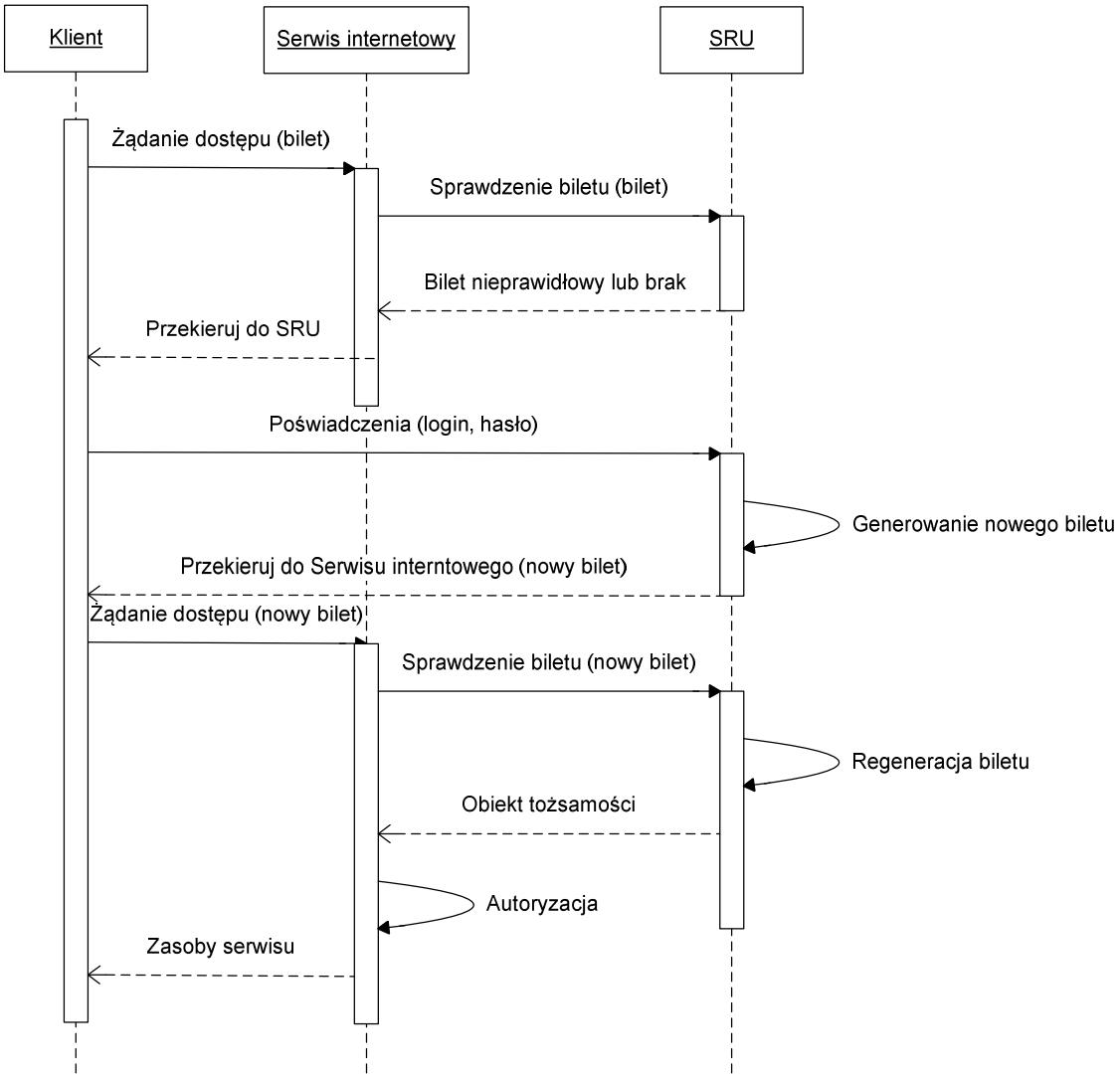


Rys. 4. Diagram przepływu danych poziomu zerowego

Rysunek 4 przedstawia procesy, będące najważniejszymi funkcjami systemu. Spełniają one cele postawione w wymaganiach funkcjonalnych nakładanych na system, dokładniej opisane na początku niniejszego rozdziału.

4.3.1. Proces „Uwierzytelnij użytkownika”

Podobny poziom abstrakcji prezentuje przedstawiony na rysunku 5 diagram sekwencji, obrazujący współpracę oraz kolejność wykonywanych czynności przez obiekty biorące udział w procesie uwierzytelniania użytkownika w serwisie internetowym.



Rys. 5. Proces uwierzytelniania użytkownika - Diagram sekwencji

Proponowanym rozwiązaniem zaspokajającym wymagania funkcjonalne nakładane na system jest stworzenie zautomatyzowanego mechanizmu delegującego proces uwierzytelniania z serwisu internetowego do SRU. Proces ten rozpoczyna się w momencie wysłania żądania dostępu do zasobów serwisu internetowego przez anonimowego internautę, oznaczonego na diagramie jako „Klient". Anonimowy jeszcze użytkownik nie posiada biletu pozwalającego na uwierzytelnienie, jest on nieprawidłowy lub nieważny. Serwis internetowy odbierając żądanie, wykonuje próbę uwierzytelnienia klienta komunikując się z SRU, który sprawdza przesłany bilet. Dla pełnego zobrazowania procesu uwierzytelniania założono, że użytkownik odwiedza serwis internetowy po raz pierwszy i nie posiada ważnego biletu. W takim wypadku serwis internetowy przenosi użytkownika do SRU. W czasie kiedy użytkownik pozostaje w interakcji z SRU, serwis internetowy nie posiada żadnej kontroli nad operacjami wykonywanymi w drugim systemie.

Klient za pomocą odpowiedniego formularza przekazuje poświadczenia, czyli nazwę użytkownika i hasło do SRU. Jeśli podane poświadczenia są prawidłowe, system uwierzytelnia klienta jedynie w obrębie własnej aplikacji. Po przekazaniu informacji do serwisu internetowego o pomyślnym uwierzytelnieniu, system generuje i przekazuje użytkownikowi nowo utworzony, unikalny bilet uwierzytelnienia, a następnie przenosi go z powrotem do partnerskiego serwisu internetowego.

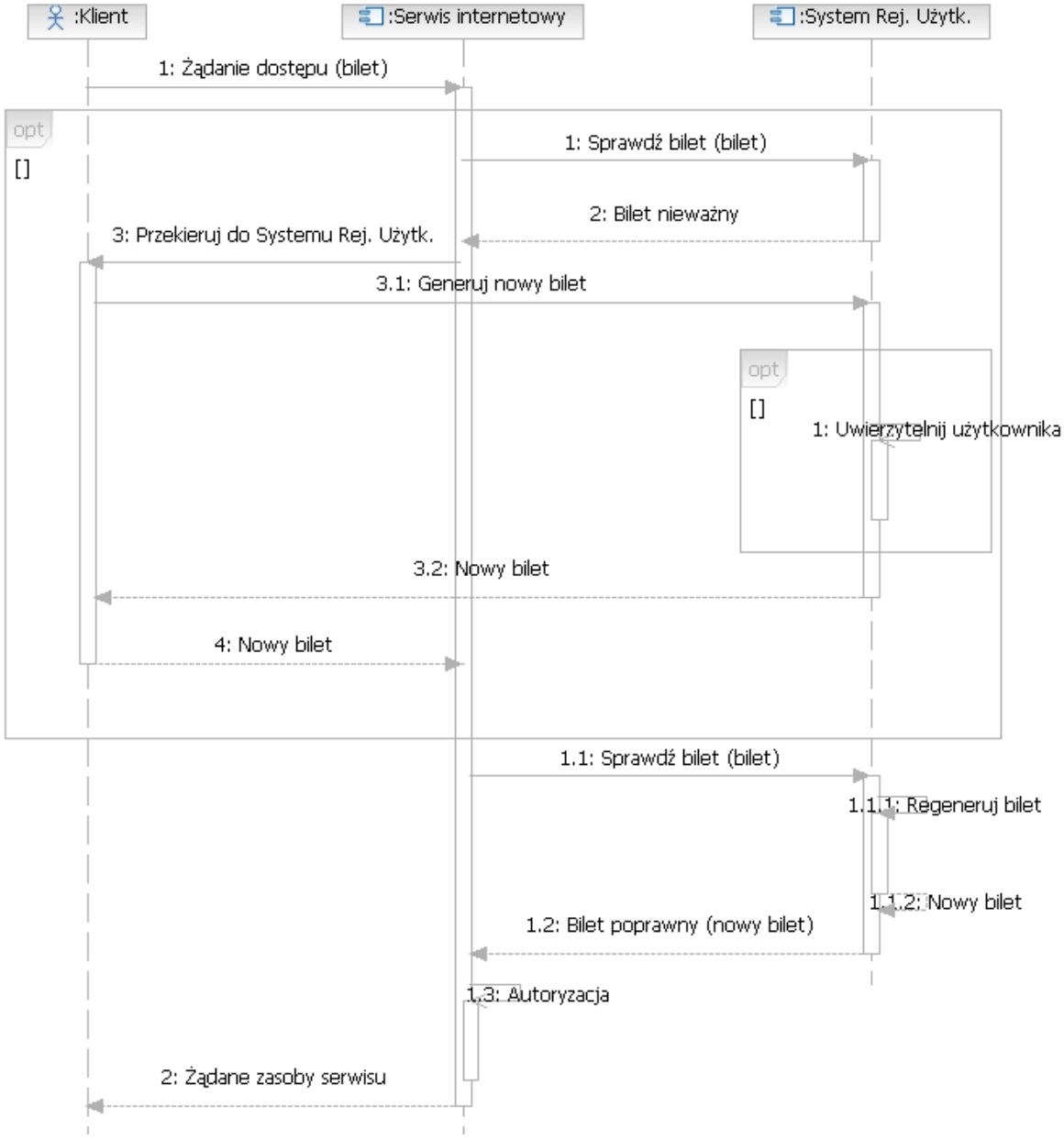
Użytkownik posiadający nowy bilet uwierzytelnienia, ponownie wykonuje operację żądania dostępu do zasobów serwisu internetowego, jednak tym razem z nowym biletem. Serwis internetowy tak jak poprzednio komunikuje się z SRU w celu sprawdzenia ważności przekazanego przez użytkownika biletu. SRU odnajduje w swojej bazie danych utworzony bilet, kojarzy go z użytkownikiem, a następnie potwierdza jego poprawność. Zanim jednak obiekt tożsamości użytkownika zostanie przesłany do serwisu internetowego, SRU regeneruje otrzymany bilet w celu podwyższenia poziomu bezpieczeństwa. W kolejnym kroku realizowany jest proces polegający na przesłaniu do serwisu internetowego specjalnego obiektu tożsamości. Informacje zawarte w obiekcie

tożsamości to: nazwa uwierzytelnionego użytkownika oraz nowy bilet, który jest ważny przez ustalony czas. Na tej podstawie serwis internetowy zezwala na dostęp do zasobów i przesyła żądane dane do przeglądarki internetowej klienta.

4.3.2. Proces „Autoryzuj dostęp do serwisu internetowego”

Uwierzytelniony klient posiada w systemie obiekt tożsamości, na który składa się nazwa użytkownika oraz bilet uwierzytelniający. Bilet jest losowym ciągiem znaków generowanym przez SRU po każdym pomyślnym uwierzytelnieniu. Klient logując się do SRU podtrzymuje sesję uwierzytelnienia jedynie z tym systemem. Bilet jest obiektem poświadczającym dla serwisu internetowego ważność i poprawność uwierzytelnienia użytkownika w SRU. W nawiązaniu do aplikacji dziennika nauczyciela, student będący w posiadaniu takiego biletu, przekazując go do tej aplikacji próbuje poświadczyć swoją tożsamość. „Dziennik nauczyciela” poprzez własny kanał komunikacyjny z SRU przekazuje do niego kod biletu przesłanego przez studenta i oczekuje odpowiedzi. Serwis internetowy na podstawie wyniku sprawdzania poprawności biletu dokonuje autoryzacji dostępu do całego serwisu.

Bilet jest jedną z bardziej wrażliwych danych, ponieważ posiadanie ważnego biletu wystarczy do uzyskania autoryzacji. Kod biletu przesyłany między klientem, a serwisem internetowym może zostać odczytany w sieci przez osobę postronną. To, czy połączenie między klientem, a serwisem internetowym będzie szyfrowane, leży już w gestii programisty serwisu internetowego. Aby zwiększyć wiarygodność tego, że przesyłany bilet pochodzi faktycznie od uwierzytelnionego w SRU klienta, stworzono mechanizm sprawdzania ważności oraz regeneracji biletu, który zmniejsza ryzyko podszycia się intruza pod uwierzytelnioną w danej chwili tożsamość. Sam mechanizm sprawdzania biletu bada, czy w kontekście danego użytkownika nie było konfliktów w użyciu biletów. Oznacza to, że jeśli dla danego użytkownika bilet został użyty podwójnie lub klient przesłał błędny kod biletu, aktualny także traci ważność. Dokładny przebieg autoryzacji przedstawia diagram sekwencji przedstawiony na rysunku 6.



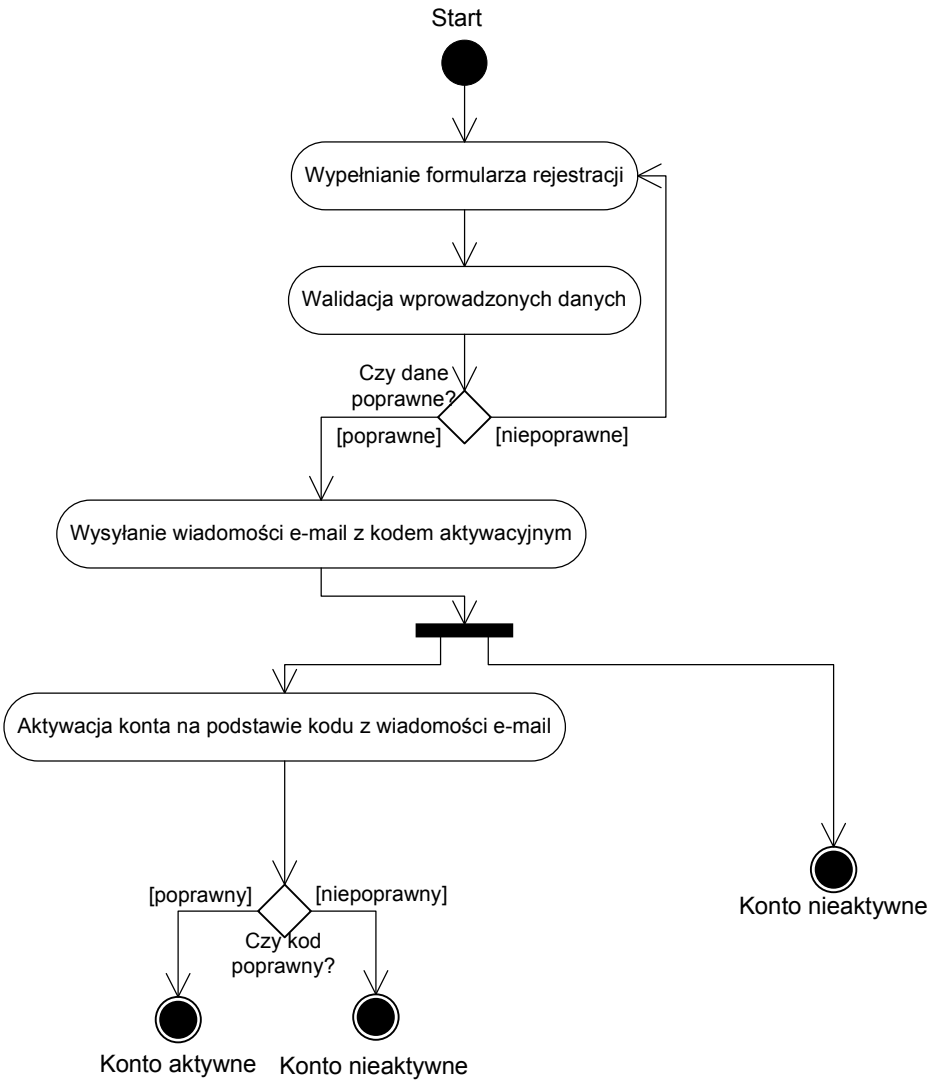
Rys. 6. Proces autoryzacji w serwisie internetowym – Diagram sekwencji

Proces autoryzacji w serwisie internetowym zaczyna się tak jak proces uwierzytelniania użytkownika, czyli od żądania dostępu. Tym razem jednak użytkownik posiada już bilet uwierzytelnienia, ponieważ pomyślnie przeszedł etap uwierzytelniania. Serwis internetowy odbiera przesłany wraz z żądaniem dostępu bilet, a następnie przesyła go w celu weryfikacji do SRU. Zanim jednak SRU zweryfikuje ważność biletu, sprawdza, czy klientowi zezwolono na autoryzację w serwisie, do którego żąda dostępu, a dopiero potem przechodzi do weryfikacji poprawności kodu biletu. Może jednak się zdarzyć, że bilet z różnych powodów jest już nie ważny. Na przykład wygasł z powodu zbyt długiego odstępu czasowego od

poprzedniego żądania, został już użyty lub jest błędny. Serwis internetowy w chwili otrzymania komunikatu o niepoprawnym bilecie przerywa proces autoryzacji i przenosi użytkownika do SRU. Jeśli klient posiada aktywną sesję uwierzytelnienia z SRU, automatycznie otrzyma nowo wygenerowany bilet, a następnie zostanie przekierowany z powrotem do serwisu internetowego, do serwisu docelowego. Cała operacja jest przeźroczysta dla klienta, czyli klient nie zauważa, że przechodził proces uwierzytelniania i generacji nowego biletu. Jeśli jednak, z jakiegoś powodu sesja z SRU wygasła, użytkownik zobligowany jest się zalogować. Po powrocie użytkownika do serwisu internetowego ponawiana jest próba uzyskania przez niego dostępu do zasobów. Jeśli tym razem bilet jest prawidłowy i ważny, serwis internetowy udostępnia zasoby użytkownikowi.

4.3.3. Proces „Zarejestruj użytkownika”

Bazując na wymaganiach funkcjonalnych w zakresie rejestracji nowych użytkowników, omówionych w rozdziale I, system musi umożliwić samodzielne założenie konta. W procesie rejestracji użytkownika bierze udział tylko internauta i SRU. Szczegółowym przedstawieniem procesu rejestracji użytkownika jest diagram aktywności z rysunku 7.



Rys. 7. Proces rejestracji nowego użytkownika – Diagram aktywności

Rejestracja nowego użytkownika rozpoczyna się od wypełnienia formularza rejestracyjnego. Analiza potrzeb informacyjnych do stworzenia konta obliguje do utworzenia odpowiednich pól w formularzu:

- Login – proponowana nazwa konta w systemie,
- Hasło,
- E-mail – wymagany w celu późniejszej aktywacji konta,
- Rodzaj konta – w zależności od tego wyboru, konto zostanie przypisane do roli zwykłego użytkownika lub serwisu internetowego.

Wprowadzone do formularza dane podlegają walidacji, zanim zostaną przesłane do funkcji tworzącej konto w systemie. Każde pole podlega różnym regułom walidacji:

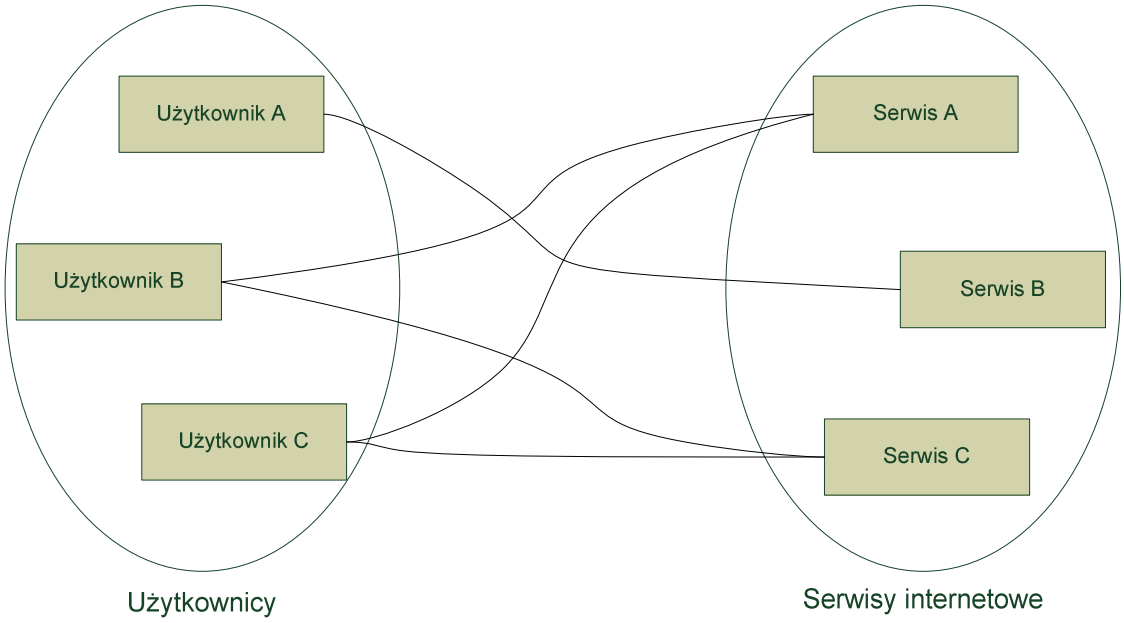
- Login.
  - Czy nazwa nie jest już zajęta?
  - Czy pasuje do wzorca prawidłowej nazwy użytkownika, tzn. czy nie zawiera niedozwolonych znaków, nie jest za krótki lub za długi itp.?
- Hasło.
  - Czy pasuje do wzorca prawidłowego hasła, tzn. zawiera odpowiednią ilość cyfr, znaków specjalnych oraz jest odpowiednio długie?
  - Czy jest identyczne z zawartością pola potwierdzenia hasła?
- E-mail.
  - Czy ma prawidłowy format?

W przypadku napotkania na jakikolwiek konflikt z regułami poprawności wprowadzanych danych, formularz musi zostać poprawiony. Po usunięciu ewentualnych błędów, konto w systemie zostaje utworzone, ale jest jeszcze nieaktywne. System w celu zabezpieczenia przed seryjnym zakładaniem kont oraz w celu potwierdzenia adresu e-mail podanego w formularzu może wysłać na podany adres e-mail wiadomość ze specjalnym kodem aktywacyjnym. Konto pozostaje nieaktywne do chwili wprowadzenia kodu aktywacyjnego na stronie SRU. Jeśli użytkownik odbierze wiadomość e-mail i zdecyduje się aktywować swoje konto, musi wykonać operacje zawarte w wiadomości e-mail. Po przesłaniu klucza do SRU, system sprawdzi go i jeśli jest poprawny, aktywuje powiązane z kluczem konto. Funkcja potwierdzania konta e-mail może zostać wyłączona przez administratora SRU. W takim wypadku konto aktywowane jest natychmiast po rejestracji. Użytkownik może zalogować się na swoje konto zaraz po jego założeniu.

4.3.4. Proces przypisywania użytkownika do serwisu internetowego

Teoretycznie dokładnie jedno konto w SRU wystarcza do tego, aby móc uzyskać autoryzację we wszystkich partnerskich serwisach internetowych. Rozwiązanie to jest wygodne, ponieważ użytkownik nie musi korzystać z wielu

kont, o różnych poświadczeniach do logowania się na wybranej stronie internetowej. SRU udostępnia odpowiednie usługi sieciowe dla serwisów internetowych, umożliwiające kontrolę nad tym, które konta mogą uzyskiwać autoryzację. Istnieje potrzeba utworzenia powiązania między użytkownikiem, a serwisem internetowym w relacji wiele do wielu. Rysunek 8 prezentuje przykładowe przyporządkowanie kont w roli użytkowników do kont w roli serwisów internetowych.



Rys. 8. Przyporządkowanie użytkowników do serwisów internetowych

Przykład z rysunku 8 pokazuje, że do Serwisu A mogą mieć dostęp tylko Użytkownik B i C, ale użytkownik B może mieć także dostęp do Serwisu C.

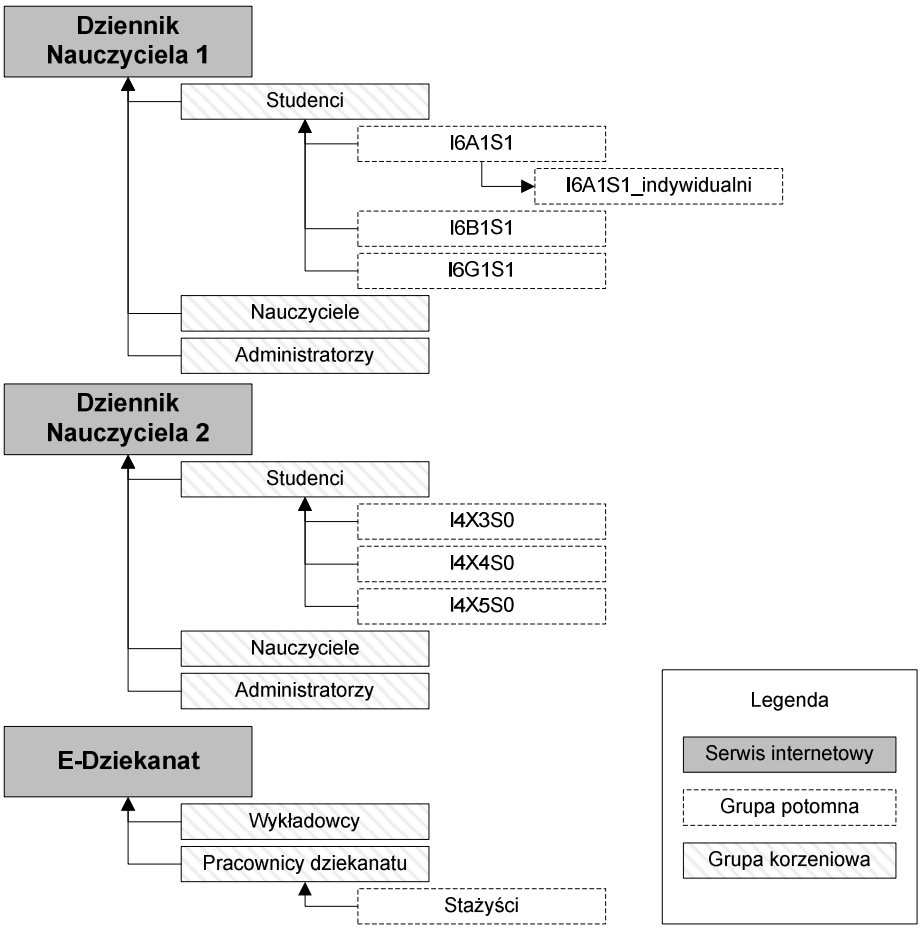
Proces przypisywania użytkownika do serwisu internetowego rozpoczyna się od zgłoszenia przez użytkownika chęci dostępu do wybranego serwisu internetowego. SRU powinien udostępniać narzędzie do przesyłania takich zgłoszeń. Użytkownik po wysłaniu zgłoszenia z prośbą o dostęp do konkretnej strony internetowej, musi czekać na reakcję ze strony jej administratora. Serwis internetowy za pomocą usługi sieciowej udostępnionej przez SRU akceptuje lub odrzuca nadesłane zgłoszenia.

Przebieg tego procesu przedstawia poniższy scenariusz:

1. SRU: Wyświetla panel zgłaszania wniosków o autoryzację w wybranym serwisie internetowym.
   Użytkownik: Wybiera serwis internetowy i wysyła wniosek.
2. SRU: Udostępnia listę zgłoszeń wysłanych do serwisu internetowego.
   Serwis internetowy: Pobiera listę.
3. Serwis internetowy: Akceptuje lub odrzuca zgłoszenie użytkownika.
   System: Tworzy powiązanie użytkownika z serwisem internetowym w bazie danych.

4.3.5. Proces zarządzania grupami użytkowników

Dla mechanizmu grup wymagane jest, aby każdy serwis internetowy mógł tworzyć niezależną, drzewiastą strukturę grup. Rysunek 9 przedstawia przykładowy wygląd struktury grup obsługiwanych przez SRU.



Rys. 9. Przykładowa struktura grup w SRU

Struktura i nazwy grup przedstawione na rysunku 9 są niezależne dla każdego serwisu internetowego. Jedynym ograniczeniem dla serwisu internetowego jest konieczność stosowania unikalnych nazw dla każdej z grup w swojej strukturze. Ułatwia to zarządzanie grupami przez serwis internetowy, ponieważ może on wskazywać na nie za pomocą ich nazw. SRU posiadając dostęp do całego repozytorium grup nie może rozróżniać ich po nazwach, ponieważ mogą istnieć grupy o takich samych nazwach, przypisane do różnych serwisów internetowych. SRU rozróżnia grupy za pomocą specjalnych identyfikatorów. Ograniczając się do struktury zależności grup w obrębie jednego serwisu internetowego można wyróżnić trzy rodzaje grup: grupę korzeniową, grupę nadrzędną oraz grupę potomną. Grupa korzeniowa nie posiada rodzica, natomiast grupa potomna powiązana jest grupą nadrzędną – rodzicem. Grupą nadrzędną dla grupy potomnej wcale nie musi być grupa korzeniowa, ponieważ nie istnieje ograniczenie co do ilości poziomów struktury grup. Oznacza to, że grupa potomna może być jednocześnie grupą nadrzędną, ale nie może być grupą korzeniową. Przykładem jest grupa I6A1S1 na rysunku 9.

Dokonując dekompozycji procesu zarządzania grupami wyróżniono przypadki użycia mechanizmu grup zaprezentowane w Tabeli 1.

Tabela 1. Przypadki użycia mechanizmu grup

Aktor	**Przypadek użycia**	**Parametry**	**Ograniczenia**
Serwis internetowy	Utwórz grupę	Nazwa grupy	Nazwa grupy nie może kolidować z już istniejącą w systemie.
	Utwórz grupę potomną	Nazwa grupy, nazwa grupy nadrzędnej	Nazwa grupy nie może kolidować z już istniejącą w systemie.
			Grupa nadrzędna musi istnieć.
	Usuń grupę	Nazwa grupy	Grupa nie może posiadać potomków.
	Pobierz wszystkie grupy	---	---
	Pobierz grupy korzeniowe	---	---
	Wskaż rodzica dla grupy	Nazwa grupy	Grupa musi być grupą podrzędną.
			Grupa musi istnieć.
	Sprawdź istnienie grupy	Nazwa grupy	---
	Zmień nazwę grupy	Nazwa grupy, nowa nazwa grupy	Nazwa grupy nie może kolidować z już istniejącą w systemie.

4.3.6. Proces zarządzania przydziałem użytkowników do grup

Użytkownik przypisany do serwisu internetowego może być przydzielany bez ograniczeń do różnych grup należących do tego serwisu. W nawiązaniu do rysunku 9 z rozdziału 4.3.5, użytkownik może znajdować się w grupie Studenci i grupie I6G1S1 jednocześnie. W kontekście całego SRU, użytkownik teoretycznie może być przypisany do wielu grup, należących do różnych serwisów internetowych bez ograniczeń. W praktyce, z faktu, że tylko serwis internetowy może przydzielać użytkowników do swoich grup wynika, że użytkownik nie może należeć do grup w obrębie serwisów internetowych, z którymi nie jest powiązany.

W tabeli 2 przedstawione są przypadki użycia, będące częścią procesu zarządzania przydziałem użytkowników do grup. Każdy z przypadków użycia podlega pewnym ograniczeniom, których mechanizm SRU musi przestrzegać.

Tabela 2. Przypadki użycia mechanizmu przydziału użytkowników do grup

Aktor	Przypadek użycia	Parametry	Ograniczenia
Serwis internetowy	Dodaj użytkownika do grupy	ID użytkownika, nazwa grupy	Nie może istnieć już takie powiązanie.
			Grupa musi istnieć.
			Użytkownik musi istnieć.
	Pobierz grupy podrzędne	Nazwa grupy nadrzędnej	Grupa musi istnieć.
	Pobierz grupy dla użytkownika	ID użytkownika	Użytkownik musi istnieć.
	Pobierz grupy korzeniowe dla użytkownika	ID użytkownika	Użytkownik musi istnieć.
	Pobierz grupy dla użytkownika	ID użytkownika	Użytkownik musi istnieć.
	Sprawdź przynależność użytkownika do grupy	ID użytkownika, nazwa grupy	Grupa musi istnieć.
			Użytkownik musi istnieć.
	Usuń użytkownika z grupy	ID użytkownika, nazwa grupy	Musi istnieć takie powiązanie.
			Grupa musi istnieć.
			Użytkownik musi istnieć.

4.3.7. Proces zarządzania kontami w systemie

Do tego procesu dostęp może mieć jedynie użytkownik pełniący rolę administratora w SRU. System musi udostępnić panel administracyjny, z poziomu którego, administrator będzie mógł zarządzać wszystkimi kontami w systemie.

Uprawnienia administratora ograniczają się tylko do zarządzania kontami użytkowników i serwisów internetowych. Administrator nie posiada uprawnień do zmian decyzji podejmowanych przez administratorów serwisów internetowych w kwestii przypisania użytkownika do serwisu internetowego oraz operacji związanych z mechanizmem grup.

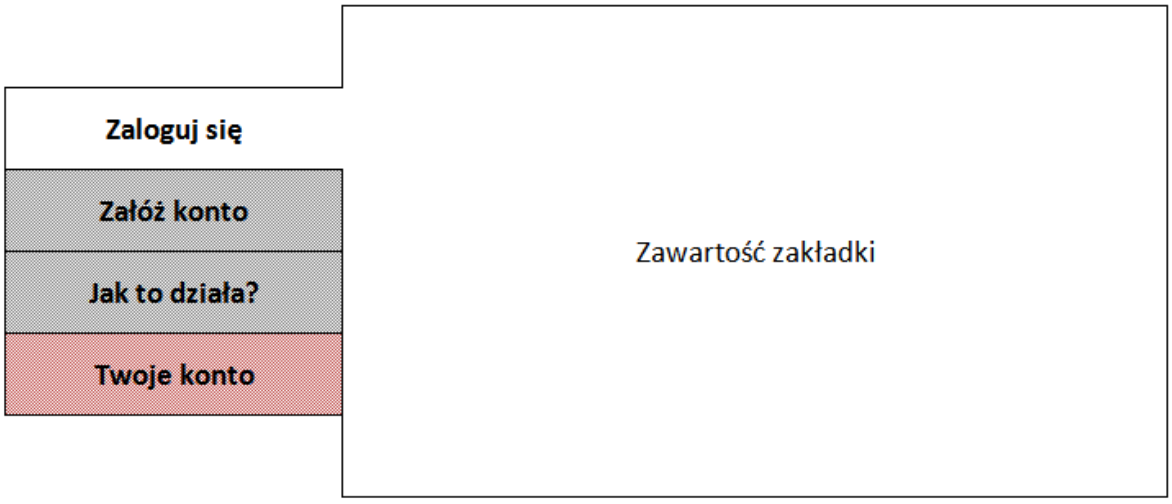
Przypadki użycia w procesie zarządzania kontami przedstawia poniższa tabela 3.

Tabela 3. Przypadki użycia panelu administratora SRU

Aktor	Przypadek użycia	Parametry	Ograniczenia
Administrator	Włącz konto	---	Konto musi być wyłączone.
	Wyłącz konto	---	Konto musi być włączone
	Zmień e-mail	Nowy e-mail	---
	Usuń konto	---	Nie można usunąć swojego konta.

4.4. Projekt interfejsu graficznego

Strona internetowa SRU nie jest witryną mającą na celu przekazywanie internaucie dużej ilości danych, dlatego większy nacisk położono na łatwą i intuicyjną nawigację.



Rys. 10. Szablon interfejsu graficznego

Rysunek 10 przedstawia projekt systemu nawigacji opartego na zakładkach. Każda z zakładek udostępnia inną funkcję systemu obsługiwaną przez stronę WWW.

- **Zaloguj się** – zakładka domyślna, dostępna dla użytkowników anonimowych, uruchamiana zaraz po wejściu na stronę główną SRU. Udostępnia formularz logowania,
- **Załóż konto** – udostępnia formularz rejestracyjny,

- **Jak to działa?** – zakładka informacyjna z podstawowymi informacjami o zasadzie działania systemu oraz polityce bezpieczeństwa,
- **Twoje konto** – zakładka jest niedostępna dla nieuwierzytelnionych użytkowników. W zależności od rodzaju konta, czyli roli, do której konto należy, udostępniana jest inna zawartość:
  - **Rola administratora** – udostępniany jest panel administracyjny, służący do zarządzania wszystkimi kontami w systemie.
  - **Rola serwisu internetowego** – zakładka udostępnia formularz umożliwiający zmianę adresu URL serwisu internetowego,
  - **Rola zwykłego użytkownika** – udostępniany jest mechanizm wysyłania wniosków o autoryzację do wybranych serwisów internetowych.

  Oprócz powyższych funkcji, w obrębie tej zakładki użytkownik ma dostęp do panelu zarządzania kontem, który umożliwia zmianę hasła, adresu email oraz usunięcie konta.

Aktywna zakładka wysuwana jest na pierwszy plan oraz zmieniany jest jej kolor. Próba wejścia w zakładkę, która jest niedostępna dla danego użytkownika kończy się wyświetleniem stosownego komunikatu o odmowie dostępu, po czym użytkownik przenoszony jest do strony logowania.

V. IMPLEMENTACJA, URUCHAMIANIE ORAZ TESTOWANIE SYSTEMU

5.1. Wykorzystane oprogramowanie

Programowanie

Do budowy SRU wykorzystano pakiet oprogramowania Microsoft Visual Studio 2008 Professional w wersji 9.0 SP1. Jest to zestaw narzędzi programistycznych do tworzenia aplikacji opartych na .NET Framework. Dla potrzeb niniejszej aplikacji wykorzystano narzędzie Microsoft Visual Web Developer 2008, należący do Visual Studio, który wspomaga tworzenie internetowych aplikacji ASP.NET. Do największych zalet usprawniających tworzenie aplikacji w Visual Studio można zaliczyć [6, s.29]:

- łatwy i wygodny dostęp do okien projektu wizualnego oraz kodu,
- technologia WYSIWYG[13] dla formularzy internetowych,
- mechanizm podpowiadania i uzupełniania kodu, co zmniejsza ilość popełnianych błędów,
- lista IntelliSense[14] wyświetlająca podpowiedzi dla metod i klas,
- natychmiastowa detekcja błędów,
- możliwość modyfikowania kontrolek.

Do projektowania i zarządzania bazą danych wykorzystano dołączone do pakietu Microsoft SQL Server 2008 narzędzie SQL Server Management Studio. Służy ono do konfiguracji, zarządzania i administrowania wszystkimi komponentami Microsoft SQL Server, a także do projektowania i edycji baz danych uruchomionych na serwerze SQL. Przydatną funkcją SQL Server Management

[13] (ang. What You See Is What You Get), akronim opisujący metodę prezentacji projektu na ekranie komputera w taki sposób, aby publikacja końcowa była jak najbardziej zbliżona do wyglądu projektu.

[14] Zaprojektowany przez Microsoft mechanizm automatycznego podpowiadania i uzupełniania kodu w Visual Studio.

Studio jest funkcja wizualnego tworzenia projektu bazy danych za pomocą fizycznego diagramu związków encji ERD[15].

Środowisko uruchomieniowe

Środowiskiem testowym do uruchamiania SRU jest zestaw oprogramowania składający się z: systemu operacyjnego Microsoft Windows 2008 Server, serwera stron internetowych IIS 7.0 oraz serwera baz danych Microsoft SQL 2008 Server. Razem z SRU w tym środowisku uruchamiana będzie aplikacja internetowa wspomagająca pracę nauczyciela „Dziennik nauczyciela" oraz system zarządzania bazą danych dedykowany dla tej aplikacji e-learningowej.

System Windows 2008 Server jest serwerowym systemem operacyjnym wydanym przez Microsoft. Głównym powodem wyboru tego systemu operacyjnego jest to, że posiada zintegrowany serwer Internet Information Services (IIS) w wersji 7.

Internet Information Services, w skrócie IIS, jest zbiorem usług internetowych, pracujących pod kontrolą systemów z rodziny Windows. Jest to jeden z najbardziej popularnych serwerów stron internetowych. IIS może służyć jako serwer FTP, HTTP, NNTP, SMTP. Na potrzeby SRU oraz wspieranych serwisów internetowych pracujących na tym samym serwerze, IIS pracuje w roli serwera HTTP i HTTPS.

Serwerem bazodanowym, z którego korzysta SRU jest Microsoft SQL Server 2008. Jest to serwer zarządzania bazą danych typu klient-serwer, charakteryzujący się wysoką wydajnością, niezawodnością i skalowalnością.

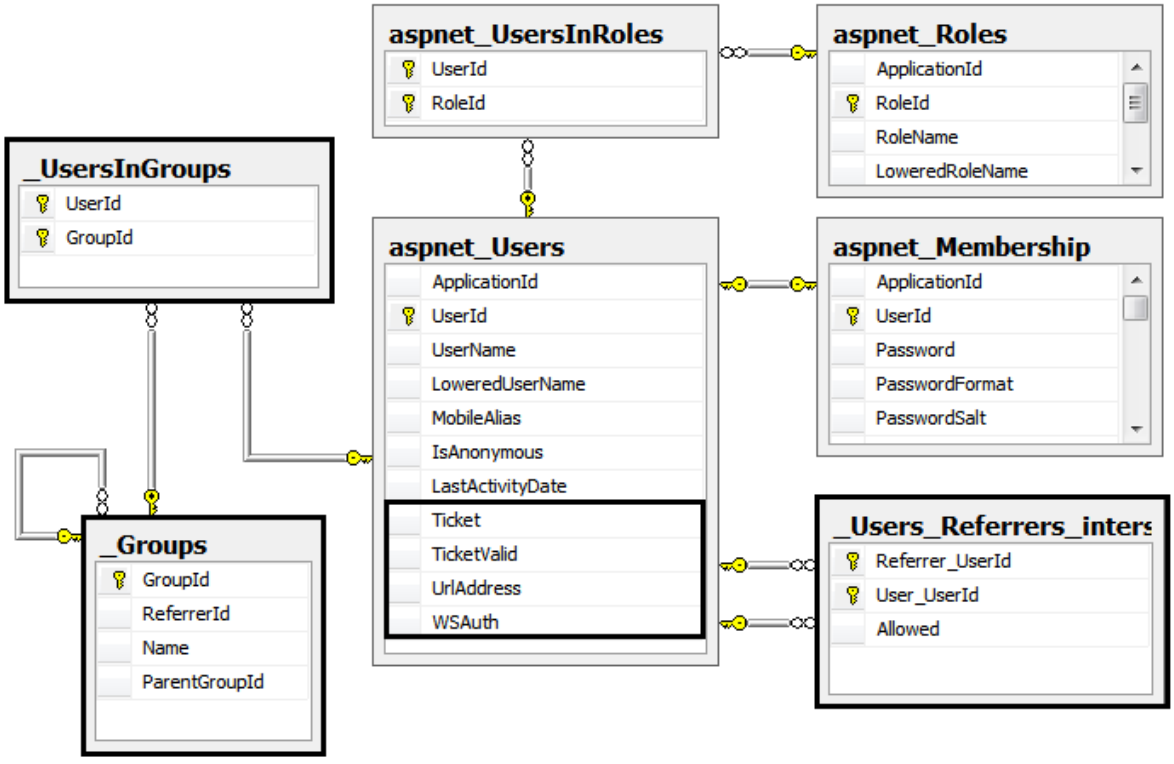
5.2. Baza danych

Struktura bazy danych w dużej mierze opiera się na strukturze utworzonej przez mechanizm ASP.NET Membership Provider,. Jest to wbudowany mechanizm zajmujący się zarządzaniem i przechowywaniem poświadczeń użytkowników

[15] (ang. Entity Relationship Diagram), diagram obrazujący relacje pomiędzy encjami, używany do opisu struktury relacyjnych baz danych.

w systemie, a także dostarczaniem rozwiązań uwierzytelniania i autoryzacji. Dodatkowo mechanizm Membership może zostać zintegrowany z mechanizmem ról ASP.NET Role Management. Za dostarczanie danych dla mechanizmów ASP.NET Membership oraz ASP.NET Role odpowiadają odpowiednio klasy SqlMembershipProvider oraz SqlRoleProvider. Każda z tych klas wymaga odpowiednio utworzonej struktury tabel, widoków i procedur w bazie danych. Aby wygenerować potrzebną strukturę poza lokalną bazą danych, na którą wskazuje SQL Server Provider, należy wykorzystać narzędzie aspnet_regsql.exe, znajdujące się w katalogu .NET Framework.

Z wygenerowaną przez aspnet_regsql.exe strukturą zintegrowano trzy dodatkowe tabele: *_Users_Referrers_intersection*, *_Groups* oraz *_UsersInGroups*. Modyfikacjom w postaci dodania czterech dodatkowych pól: *Ticket*, *TicketValid*, *UrlAddress* oraz *WSAuth*, poddana została także tabela *aspnet_Users*. Rysunek 11 przedstawia encje wykorzystywane w SRU. Pogrubioną linią oznaczone zostały nowo dodane encje i pola.



Rys. 11. Diagram ERD wybranych encji bazy danych SRU

Tabela aspnet_Users

W tej tabeli zapisane są podstawowe informacje o tożsamości użytkownika. Jest to główna tabela w bazie danych SRU, wiążąca ze sobą resztę wykorzystywanych tabel. Tabela posiada dodane cztery niestandardowe pola:

- *Ticket* – zawiera kod biletu uwierzytelniającego dla danego użytkownika,
- *TicketValid* – określa czas wygaśnięcia biletu zapisanego w polu Ticket,
- *UrlAddress* – pole wykorzystywane tylko w rekordach opisujących konto serwisu internetowego. Zawiera adres strony serwisu internetowego, do którego zwykły użytkownik musi być przeniesiony po poprawnym uwierzytelnieniu,
- *WSAuth* – pole informuje czy bilet nie został unieważniony.

Tabela _Users_Referrers_intersection

W tej tabeli zapisywane są powiązania użytkowników zwykłych kont z kontami serwisów internetowych. Pole Allowed informuje, czy zgłoszone powiązanie zostało zaakceptowane przez serwis internetowy. Domyślnie pole Allowed ustawiane jest na wartość False. Aktywne powiązania użytkowników z serwisami internetowymi posiadają w tym polu wartość True.

Tabela _Group

Tabela zawiera informacje o wszystkich grupach utworzonych w SRU niezależnie od przypisanych im serwisów internetowych.

- *GroupId* – identyfikator służący identyfikacji grupy przez SRU,
- *ReferrerId* – identyfikator serwisu internetowego, w obrębie którego grupa została utworzona,
- *Name* – nazwa grupy,
- *ParentGroupId* – rekordy grup podrzędnych wskazują w tym polu na grupę nadrzędną, do której należą.

Tabela aspnet_Roles

W tabeli zapisane są informacje o rolach występujących w SRU, czyli roli użytkownika, serwisu internetowego oraz administratora SRU.

Tabela aspnet_Membership

Encja tworzona automatycznie przez narzędzie *aspnet_regsql*. Zawiera dane poświadczeń użytkowników takie jak login, hasło, a także adresy e-mail oraz treści pytań do haseł. W celach bezpieczeństwa dane te są odpowiednio zaszyfrowane. Tabela występuje w relacji jeden do jednego z *aspnet_Users*.

Tabela aspnet_UsersInRoles

Służy do powiązania tabel *aspnet_Users* oraz *aspnet_Roles* w relacji wiele do wielu.

Tabela _UsersInGroups

Utworzona w celu powiązania tabel *aspnet_Users* oraz *_Groups* w relacji wiele do wielu.

5.3. Klasy

W ramach SRU zaimplementowano pięć klas: *Auth, Config, Group, RegisteredUser* oraz *SoapAuthTicket*, które zostały osadzone w folderze *App_Code* oraz jedną klasę o nazwie *Authorization* implementowaną poza systemem SRU.

Klasa *Config* jest klasą konfiguracyjną, zawierającą atrybuty, takie jak poświadczenia potrzebne do uwierzytelnienia w bazie danych SQL oraz adres URL systemu.

Klasa *SoapAuthTicket* jest szablonem obiektu tożsamości użytkownika i serwisu internetowego. Obiekt klasy *SoapAuthTicket* wykorzystywany jest podczas uwierzytelniania i autoryzacji w usługach sieciowych SRU. Wykorzystanie tej klasy zostało dokładnie omówione w rozdziale 5.4.

Jedyną klasą zaimplementowaną w ramach budowy SRU, jednak nie działającą fizycznie w ramach tego systemu jest klasa *Authorization*. Została ona

stworzona w celu uproszczenia instalacji usługi autoryzacji w serwisie internetowym. Utworzony za pomocą konstruktora domyślnego obiekt klasy *Auhtorization* przechowuje adres internetowy SRU oraz login i hasło do konta serwisu internetowego. Klasa posiada zaimplementowaną metodę *Authenticate()* pełniącą rolę strażnika dostępu do serwisu internetowego. Jej zadaniem jest przekazywanie biletu uwierzytelnienia użytkownika, uwierzytelnianie w usługach sieciowych SRU oraz autoryzowanie dostępu do zasobów serwisu internetowego.

Pozostałe trzy klasy definiujące podstawowe byty istniejące w SRU zostały omówione bardziej szczegółowo w dalszej części tego rozdziału.

5.3.1. Klasa użytkowników - RegisteredUser

Obiektami tej klasy są potencjalni użytkownicy SRU. Klasa przechowuje podstawowe informacje o koncie użytkownika oraz tym, czy konto istnieje w bazie danych. Zaimplementowane zostały metody do zarządzania kontem użytkownika. Jedne z ważniejszych to:

- *GenerateTicket()*

  Metoda służy do wygenerowania biletu uwierzytelnienia dla użytkownika. Losowo wygenerowany kod biletu zapisywany jest do pola *Ticket* w bazie danych oraz ustawiana jest jego ważność w polu *TicketValid*.

- *IsTicketValid(string ticket)*

  Funkcja sprawdza czy kod biletu wprowadzony w argumencie metody jest prawidłowy dla danego użytkownika, czy nie bilet nie został unieważniony oraz czy bilet nie jest starszy niż 300 sekund.

- *UnauthoriseUser()*

  Metoda wywoływana na przykład w momencie użycia błędnego biletu. Po jej wywołaniu wartość pola *WSAuth* dla danego użytkownika ustawiana jest na *False*, co powoduje unieważnienie aktualnego biletu użytkownika. Występuje także przeciążenie tej metody w postaci *UnauthoriseUser(string ReferrerId).* Służy ono do całkowitego usunięcia powiązania użytkownika z kontem serwisu internetowego.

- *DeleteUser()*

  Funkcja zanim usunie użytkownika, przygotowuje bazę danych do tej operacji. Usuwane są relacje użytkownika z serwisami internetowymi oraz grupami, a na końcu sam użytkownik.

- *SetNewMail(string newmail)*

  Metoda ustawia zadany w parametrze *newmail* adres e-mail wykorzystując atrybuty klasy *MembershipUser*.

- *AddUserToRole(string rolename)*

  Parametr przekazywany do tej metody może przyjąć dwie postaci: „*referrer*" określający konto serwisu internetowego oraz „*user*" określający konto użytkownika. W zależności od wartości zmiennej rolename konto przypisywane jest odpowiednio do roli użytkownika (*_system_U*) lub serwisu internetowego (*_system_RA*).

- *IsReferrersUser(string ReferrerId)*

  Metoda zwraca wartość *True* (prawda) lub *False* (fałsz). Funkcja zwraca wartość *True* w przypadku kiedy pomiędzy użytkownikiem, a zadanym w parametrze *ReferrerId* serwisem internetowym istnieje zaakceptowane powiązanie.

- *MemberOfGroups(string ReferrerId)*

  Wywołanie tej metody generuje listę grup, do których należy dany użytkownik w kontekście serwisu internetowego, określonego parametrem *ReferrerId*.

- *AuthoriseUser(string ReferrerId)*

  Metoda wywoływana przez administratorów serwisów internetowych, zajmujących się zarządzaniem zgłoszeniami dostępu od użytkowników SRU. Funkcja została zaimplementowana w taki sposób, aby przypisać konto użytkownika do serwisu internetowego zadanego parametrem ReferrerId niezależnie od tego, czy użytkownik wysyłał żądanie dostępu do serwisu internetowego czy nie. Wywołanie metody w kontekście istniejącego konta użytkownika oraz prawidłowego Id serwisu

internetowego spowoduje utworzenie powiązania między tymi bytami z atrybutem *Allowed* ustawionym na *True*.

- *RequestAuthorization(string ReferrerId)*

  Panel wysyłania zgłoszeń z prośbą dostępu do wybranego serwisu wykorzystuje tą metodę do tworzenia nieaktywnych powiązań pomiędzy użytkownikami i serwisami internetowymi, czyli powiązań z atrybutem *Allowed* ustawionym na *False*. Administrator serwisu internetowego za pomocą metody *AuthoriseUser(string ReferrerId)* może umożliwić dostęp ustawiając tym samym wartość *True* dla pola *Allowed* lub odrzucić zgłoszenie usuwając powiązanie za pomocą metody *UnathoriseUser(string ReferrerId)*.

Oprócz wymienionych metod klasa implementuje także metody typu Get-Set umożliwiające bezpieczne manipulowanie atrybutami obiektu.

5.3.2. Klasa usługi sieciowej - Auth

Metody tej klasy udostępniane są na zewnątrz SRU za pomocą usług sieciowych. Korzystają z nich serwisy internetowe. Funkcje tej klasy w większości opierają się na odpowiednim wywoływaniu i sterowaniu metodami klas *RegisteredUser* lub *Group*. Poniżej przedstawiono listę metod dostępnych dla nieuwierzytelnionych serwisów internetowych:

- *AuthenticateWithNoTicket(string login, string password)*

  Metoda pozwala na uwierzytelnienie serwisu internetowego w usługach sieciowych bez ważnego biletu uwierzytelnienia użytkownika, na podstawie przesłanych parametrów *login* oraz *password*.

- *Authenticate(string login, string password, string ticketcode)*

  Uwierzytelnienie serwisu internetowego z użyciem tej metody może nastąpić po spełnieniu szeregu warunków:

  - podane w parametrach login oraz password poświadczenia muszą być prawidłowe,

- serwis internetowy musi uwierzytelnić się w ramach konta o roli serwisu internetowego,
- konto użytkownika skojarzone z biletem musi posiadać prawa do autoryzacji w serwisie internetowym, wywołującym tą usługę,
- bilet podany w parametrze *ticketcode* musi być prawidłowy i aktualny.

Jeśli wszystkie z powyższych warunków zostaną spełnione serwis internetowy zostaje trwale uwierzytelniany w usłudze sieciowej, a metoda *Authenticate* zwraca nowy kod biletu.

W tabeli 4 przestawione zostały pokrótce metody klasy Auth dostępne dla aplikacji, które pomyślnie przeszły proces uwierzytelniania w usłudze sieciowej SRU. Metody przedstawione w tabeli 4 nie wymagają ponownego przekazywania poświadczeń, ponieważ dostęp do nich autoryzowany jest na podstawie specjalnego obiektu SoapTicket. Uwierzytelnianie, autoryzacja oraz bezpieczeństwo usług sieciowych SRU zostało bardziej szczegółowo omówione w rozdziale 5.4.

Tabela 4. Metody klasy Auth dostępne po uwierzytelnieniu

Nazwa metody (parametry)	**Opis**
AddCurrentUserToGroup (string GroupName)	Wykonuje metodę *AddUserToGroup* w kontekście bieżącego użytkownika.
AddUserToGroup (string UserId, string GroupName)	Dodaje użytkownika o podanym *UserId* do grupy o nazwie *GroupName*. Zapisuje zmiany w bazie danych.
AuthorisedUsers()	Metoda zwraca obiekt *DataSet* z listą użytkowników zaakceptowanych przez serwis internetowy.
AuthoriseUser (string UserId)	Metoda tworzy powiązanie bieżącego serwisu internetowego z użytkownikiem *UserId*.
CheckTicket (string ticket)	Metoda służy do sprawdzania poprawności i ważności biletu przekazanego w parametrze *ticket*. Analogicznie sprawdzane są warunki wyszczególnione w opisie metody *Authenticate*. Metoda dostępna jest dla serwisów internetowych uwierzytelnionych z usługą sieciową SRU.
CreateChildGroup (string Name, string ParentName)	Tworzy grupę *Name*, potomną do grupy *ParentName*. Zapisuje zmiany w bazie danych.

CreateGroup (string Name)	Tworzy grupę korzeniową o nazwie *Name*. Zapisuje zmiany w bazie danych.
DeleteGroup (string Name)	Usuwa grupę o nazwie *Name* z bazy danych.
GetAllGroups()	Zwracana jest lista wszystkich grup powiązanych z bieżącym serwisem internetowym, znajdujących się w bazie danych
GetAllUsers()	Metoda zwraca obiekt *DataSet* z identyfikatorami i nazwami wszystkich użytkowników zarejestrowanych w SRU.
GetChildrenForGroup (string GroupName)	Zwracana jest lista grup potomnych dla podanej w parametrze *GroupName* grupy nadrzędnej.
GetGroupsForCurrentUser()	Wykonuje metodę *GetGroupsForUser* w kontekście bieżącego użytkownika
GetGroupsForUser (string UserId)	Metoda sprawdza do jakich grup należy użytkownik o zadanym *UserId*, a następnie zwraca ich listę.
GetGroupsWithoutParentForCurrentUser()	Wykonuje metodę *GetGroupsWithoutParentForUser* w kontekście bieżącego użytkownika.
GetGroupsWithoutParentForUser(string UserId)	Metoda filtruje listę zwracaną przez metodę *GetGroupsForUser* odrzucając rekordy grup potomnych.
GetParentForGroup (string GroupName)	Zwracaną wartością jest nazwa grupy nadrzędnej do zadanej w parametrze *GroupName* grupy potomnej.
GetUserId()	Zwracaną wartością po wywołaniu tej metody jest identyfikator *Guid* bieżącego użytkownika.
GetUserName()	Zwracaną wartością po wywołaniu tej metody jest nazwa bieżącego użytkownika.
GetUsersForGroup (string GroupName)	Metoda zwraca obiekt *DataSet* zawierający listę użytkowników należących do grupy o nazwie *GroupName*.
IsGroupExists(string Name)	Sprawdza, czy grupa istnieje. Zwracana wartość: bool.
IsCurrentUserInGroup()	Wykonuje metodę *IsUserInGroup* w kontekście bieżącego użytkownika.
IsUserInGroup (string UserId, string GroupName)	Metoda szuka powiązania w bazie danych między użytkownikiem o podanym *UserId*, a zadaną przez *GroupName* grupą.
RemoveCurrentUserFrom Group ()	Wykonuje metodę *RemoveUserFromGroup* w kontekście bieżącego użytkownika.
RemoveUserFromGroup (string UserId, string GroupName)	Usuwa z bazy danych powiązanie użytkownika o podanym *UserId* z grupą o nazwie *GroupName*.
RenameGroup (string GroupName, string NameGroupName)	Metoda służy do zmiany nazwy grupy. Wymaga wprowadzenia nazwy istniejącej grupy oraz proponowanej nowej nazwy.

UnAuthoriseCurrentUser()	Wykonuje metodę *UnauthoriseUser* w kontekście bieżącego użytkownika.
UnauthorisedUsers()	Za pomocą tej metody serwis internetowy pobiera obiekt *DataSet* zawierający listę przesłanych przez użytkowników i oczekujących na akceptację wniosków o dostęp do serwisu internetowego.
UnauthoriseUser (string UserId)	Za pomocą tej funkcji serwis internetowy może trwale zablokować dostęp wybranemu użytkownikowi usuwając powiązanie konta użytkownika z kontem serwisu internetowego.

5.3.3. Klasa grup – Group

Klasa *Group* pozwala na tworzenie obiektów nowych grup lub grup już istniejących w bazie. Każdy obiekt klasy *Group* posiada atrybuty *_isgroupexists* oraz *_allowstoretodb*. Dla nowo tworzonych obiektów grup uruchamiane są, w zależności od podanych parametrów, konstruktory. Po utworzeniu obiektu, atrybut *_isgroupexists* posiada informację, czy grupa już istnieje w bazie danych, natomiast atrybut *_allowtostoredb* określa, czy istnieje możliwość zapisu obiektu w bazie danych. Klasa posiada zaimplementowane trzy rodzaje konstruktorów:

- *Group(int id)*

  Konstruktor tworzy obiekt na podstawie podanego ID grupy. Jeśli grupa o danym ID istnieje w bazie atrybuty *_isgroupexists* i *_allowtostoredb* przyjmują odpowiednio wartości *True* oraz *False*.

- *Group(string ReferrerId, string Name)*

  Konstruktor tworzy obiekt grupy korzeniowej na podstawie przekazanego Id serwisu internetowego oraz nazwy grupy. Jeśli grupa o podanej nazwie już istnieje w kontekście serwisu internetowego o podanym Id, atrybuty obiektu określają grupę jako istniejącą w bazie danych z brakiem możliwości jej zapisu. Jeśli grupa o podanej nazwie nie istnieje w strukturze grup dla danego serwisu internetowego, atrybuty *_isgroupexists* i *_allowtostoredb* ustawiają wartości odpowiednio *False* oraz *True*.

- *Group(string ReferrerId, string Name, string ParentName)*

  Konstruktor służy do tworzenia obiektów grup potomnych. Oprócz

warunków sprawdzanych w przypadku grupy korzeniowej, weryfikowane jest czy odwołanie do grupy nadrzędnej *ParentName* wskazuje na istniejącą grupę.

Klasa *Group* posiada komplet metod służących do zarządzania grupami oraz przypisywania użytkowników do grup. Wykonanie każdej z poniżej przedstawionych metod warunkowane jest flagami *_isgroupexists* oraz *_allowtostoredb*. Poniżej przedstawiono metody zaimplementowane w klasie *Group*:

- *AddUserToGroup(string UserId)*

  Metoda tworzy bezpośrednio w bazie danych rekordy w tabeli *_UsersInGroups* będące powiązaniem użytkownika o zadanym *UserId* z Id grupy, w kontekście której wykonywana jest niniejsza metoda.

- *DeleteGroupFromDb()*

  Grupa może zostać usunięta jeśli nie posiada grup potomnych, czyli nie istnieją grupy z atrybutem *ParentGroupId* wskazującym na bieżącą grupę. Metoda zajmuje się także usunięciem z tabeli *_UsersInGroups* wszystkich rekordów wiążących użytkowników z bieżącą grupą. Spełnienie tych warunków pozwala na wykonanie polecenia *DELETE* na rekordzie bieżącej grupy.

- *GetChildren()*

  Metoda zwraca listę grup posiadających w polu *ParentGroupId* identyfikator bieżącej grupy.

- *GetUsers()*

  Metoda wybiera z tabeli *_UsersInGroups* rekordy wiążące bieżącą grupę z użytkownikami. Zwracaną wartością jest obiekt *DataSet* zawierający identyfikatory oraz nazwy użytkowników przypisanych do danej grupy.

- *HaveUser(string UserId)*

  W zależności od tego, czy w tabeli *_UsersInGroups* zostanie znaleziony rekord wiążący bieżącą grupę z użytkownikiem o zadanym w parametrze *UserId* identyfikatorze, metoda zwraca wartość *True* lub *False*.

- *IsGroupExists()*

  Zwracana jest wartość flagi *_isgroupexists*.

- *RemoveUserFromGroup(string UserId)*

  Metoda usuwa z tabeli *_UsersInGroups* rekord wiążący użytkownika o identyfikatorze *UserId* z bieżącą grupą.

- *RenameGroup(string NewName)*

  Metoda zmienia nazwę grupy na nową podaną w parametrze *NewName*. Relacje między grupami pozostają zachowane, ponieważ opierają się na identyfikatorach grup, a nie na nazwach.

- *StoreGroupToDb()*

  Użycie tej metody powoduje zapisanie obiektu grupy w bazie danych. Metoda działa tylko w obiektach grup, które nie istnieją jeszcze w bazie danych. Jej pomyślne działanie warunkuje flaga *_allowtostoredb* obiektu grupy.

5.4. Usługi sieciowe Systemu Rejestracji Użytkowników

Usługi sieciowe SRU publikowane są pod adresem https://adres_sru/Webservice/Auth.asmx. Polityka autoryzacji dostępu do zasobu Auth.asmx zezwala na wywoływanie listowanie usług wszystkim użytkownikom. Zapewnia to odpowiedni wpis konfiguracyjny w pliku Web.config. Plik Auth.asmx służy do zdalnego wywoływania metod klasy *Auth*, znajdującej się w folderze *App_code*. Usługi dostępne są wyłącznie dla użytkowników w roli serwisu internetowego, dlatego też korzystanie z metod wymaga na początku uwierzytelnienia.

Protokół SOAP, wykorzystywany do komunikacji z usługą sieciową SRU, zawiera oprócz wymaganego elementu *Body*, opcjonalny nagłówek Header. Specyfikacja SOAP nie definiuje jednoznacznie co musi zawierać element Header, co pozwala na przekazywanie dodatkowych informacji, nie koniecznie związanych z przesyłaną wiadomością. [11] Na modyfikację przesyłanych elementów SOAP Header pozwala metoda *SoapHeader* klasy *SoapHeaderAttribute*. Serwis internetowy może uwierzytelnić się w usługach sieciowych SRU wykorzystując

metody *Authenticate* lub *AuthenticateWithNoTicket*. Dla obu z tych metod za pomocą poniższego kodu został skonfigurowany protokół SOAP w taki sposób, aby wraz z wysyłanym dokumentem SOAP, wysyłany był obiekt *soapticket*.

[SoapHeader("soapticket",Direction=SoapHeaderDirection.Out)]

Pomyślne wykonanie metody *Authenticate* skutkuje utworzeniem obiektu o nazwie *soapticket* klasy *SoapAuthTicket* oraz wysłanie go zgodnie z dyrektywą *Direction* do klienta usługi. Obiekt *soapticket* zawiera potwierdzoną przy uwierzytelnieniu nazwę konta serwisu internetowego oraz nazwę uwierzytelnionego użytkownika.

Klient przechowując w stanie aplikacji obiekt klasy *Auth*, wraz z którym przesłany został obiekt *soapticket* może być, bez ponownego podawania poświadczeń, autoryzowanym we wszystkich usługach sieciowych. Metody wymagające uwierzytelnionej tożsamości skonfigurowane są za pomocą poniższego kodu na odbiór obiektu soapticket podczas jej wywołania.

[SoapHeader("soapticket",Direction=SoapHeaderDirection.In)]

W przypadku braku obiektu *soapticket* generowany jest wyjątek *SoapException* informujący o błędzie uwierzytelnienia.

Posiadanie obiektu tożsamości *soapticket* pozwala na dostęp do usług bez podawania poświadczeń. Istnieje ryzyko, że osoba postronna może odczytać zawartość obiektu *soapticket* podczas przesyłania go przez sieć. Aby temu zapobiec dostęp do usług może następować wyłącznie z wykorzystaniem protokołu HTTPS.

5.5. Implementacja autoryzacji dostępu do serwisu internetowego

Zgodnie z diagramem sekwencji z rysunku 6 zamieszczonego w rozdziale 4.3.2, autoryzacja dostępu do serwisu internetowego rozpoczyna się od przesłania biletu. Kod biletu jest losowo generowanym skrótem MD5. Użytkownik może przekazać go do systemu na dwa sposoby: za pomocą metody GET lub poprzez stan sesji z serwisem internetowym. W przypadku, gdy użytkownik po raz pierwszy wywołuje stronę serwisu internetowego, w stanie sesji nie ma zapisanego obiektu usługi sieciowej SRU. Brak tego obiektu w stanie sesji powoduje, że metoda *Authenticate*, uruchamiana przy każdym wywołaniu strony serwisu internetowego,

oczekuje biletu uwierzytelnienia przesyłanego metodą GET. Następnie kod biletu przesyłany jest do SRU. Jeśli kod biletu nie istnieje lub jest nieprawidłowy, usługa SRU zwraca wartość „*0*”, co oznacza, że użytkownik nie może zostać autoryzowany. Procedura odmowy dostępu do serwisu internetowego kończy się przeniesieniem użytkownika na stronę SRU *https://adres_SRU/Default.aspx?ref=login*, gdzie *login* jest nazwą konta serwisu internetowego.

Po przeniesieniu użytkownika do SRU, za pomocą mechanizmu Membership sprawdzany jest stan uwierzytelnienia użytkownika na stronie SRU. Jeśli użytkownik jest uwierzytelniony, kod metody *Page_Load* strony Default.aspx wykonuje operację wygenerowania biletu i przeniesienia użytkownika z powrotem do serwisu internetowego. Adres serwisu zdalnego ustalany jest na podstawie loginu przekazanego za pomocą metody GET pod indeksem „*ref*”. SRU odnajduje w bazie danych URL serwisu, a następnie uzupełnia go o nowo wygenerowany bilet, tak jak w poniższym przykładzie:

*https://dziennik.wat.edu.pl/Default.aspx?**ticket=nazwa_uzytkownika+kod_biletu***

Serwis internetowy wywołany z biletem, przekazanym metodą GET rozpoczyna procedurę autoryzacji od początku. Ponieważ w stanie sesji nic nie istnieje, wywoływana jest usługa autoryzacji w SRU z przekazanym metodą GET kodem biletu. Jeżeli tym razem procedura uwierzytelnienia w usłudze sieciowej powiedzie się oraz kod biletu będzie prawidłowy i aktualny, SRU przekaże do serwisu internetowego nowy bilet. Metoda *Authenticate* klasy *Authorization* w serwisie internetowym zapisuje do stanu sesji dwa obiekty: obiekt usługi sieciowej *Auth* oraz nowy bilet ważny do kolejnej procedury autoryzacji, a następnie udostępnia żądane zasoby.

Kolejne wywołania stron serwisu internetowego także podlegają procedurze autoryzacji. Uruchamiana podczas ładowania strony metoda *Authenticate* pobiera ze stanu sesji obiekty usługi sieciowej *Auth* oraz bilet, a następnie wywołuje metodę *CheckTicket* udostępnioną przez SRU. Metoda ta nie wymaga ponownego uwierzytelniania w usłudze sieciowej, ponieważ bazuje na obiekcie tożsamości

przesyłanym w nagłówku protokołu SOAP. W metodzie *CheckTicket* zaimplementowano także mechanizm sprawdzania bezpieczeństwa biletu. Jeśli w kontekście użytkownika wprowadzono błędny bilet lub podany bilet został już wykorzystany, uruchamiana jest metoda *UnauthoriseUser* z klasy *RegisteredUser*. Powoduje to unieważnienie biletu uwierzytelnienia użytkownika, nawet gdy jest aktualny, co skutkuje koniecznością wygenerowania nowego biletu i rozpoczęcia procedury autoryzacji od nowa.

5.6. Implementacja rejestracji użytkownika

Formularz rejestracji nowego użytkownika został zaimplementowany z wykorzystaniem kontrolki *asp:CreateUserWizard*. Dostarcza ona interfejs służący do tworzenia nowych kont użytkowników w systemie. Zgodnie z projektem, formularz musi posiadać pole wyboru typu konta. ASP.NET umożliwia modyfikację szablonu kontrolki *asp:CreateUserWizard*. Dodatkowe pole zaimplementowano z wykorzystaniem kontrolki *asp:DropDownList*, posiadającej dwie opcje: „Użytkownik” oraz „Serwis internetowy”.

Walidacja wprowadzonych do formularza danych jest dwupoziomowa. Zanim dane zostaną przesłane na serwer, pola formularza przechodzą wstępną walidację z użyciem języka JavaScript[16]. Kontrolka *asp:RequiredFieldValidator* sprawdza czy wskazane pole zostało wypełnione. Kontrolka *asp:CompareFieldValidator* weryfikuje, czy zawartości pól hasła oraz powtórzenia hasła są identyczne. Jeśli wstępna walidacja nie wykryje błędów, dane przesyłane są do serwera, gdzie mechanizm MembershipProvider weryfikuje zgodność wprowadzonych danych z wymaganiami zdefiniowanymi w konfiguracji zawartej w pliku Web.config.

[16] Obiektowy skryptowy język programowania. Implementacje JavaScript w przeglądarkach internetowych pozwalają na uruchamianie skryptów, wykonywanych na stronach internetowych po stronie klienta.

Poniżej przedstawiono konfigurację dostawcy *AspNetSqlMembershipProvider*:

```
<add name="AspNetSqlMembershipProvider"
        type="System.Web.Security.SqlMembershipProvider, System.Web,
        Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
        connectionStringName="ConnectionString"
        enablePasswordRetrieval="false"
        enablePasswordReset="true"
        requiresQuestionAndAnswer="false"
        applicationName="/"
        requiresUniqueEmail="true"
        passwordFormat="Hashed"
        maxInvalidPasswordAttempts="5"
        minRequiredPasswordLength="8"
        minRequiredNonalphanumericCharacters="2"
        passwordAttemptWindow="10"
        passwordStrengthRegularExpression=""/>
```

Zgodnie z powyższą konfiguracją hasło powinno składać się minimum z ośmiu znaków i minimum dwóch znaków specjalnych. Mechanizm sprawdza też, czy nazwa użytkownika nie została już zajęta.

Pomyślne utworzenie konta uruchamia dwa zaimplementowane zdarzenia: *CreateUserWizard1_SendingMail* oraz *CreateUserWizard1_CreatedUser*.

Wysyłanie wiadomości e-mail

Kod obsługujący wysyłanie wiadomości e-mail z kluczem aktywującym konto zawiera metoda zdarzenia *CreateUserWizard1_SendingMail*. Treść wiadomości pobierana jest z sekcji *MailDefinition* znajdującej się w ramach kontrolki *asp:CreateUserWizard*. Operacja polega na pobraniu szablonu wiadomości e-mail oraz modyfikacji pól *<%ActivationCode%>* i *<%UrlAddress%>* w zależności od kodu aktywacyjnego i adresu SRU. Po przygotowaniu treści wiadomości, e-mail wysyłany jest z wykorzystaniem zewnętrznego konta pocztowego. Dostęp do zewnętrznego konta pocztowego zapewnia klient SMTP skonfigurowany w ramach tagów *mailSettings* lokalnego pliku Web.config.

Utworzone konto pozostaje nieaktywne do chwili uaktywnienia go specjalnym kluczem aktywacyjnym, przesłanym w wiadomości e-mail. Proces aktywacji polega na wywołaniu formularza Confirm.aspx, znajdującego się w katalogu *Mail* SRU, z podanym metodą GET kluczem. Przykładowy link aktywujący:

*https://adres_SRU/Mail/Confirm.aspx?username=nazwa_uzytkownika&**code=kod***

Plik Confirm.aspx sprawdza podany czy podany *username* oraz *code* zgadza się z wygenerowanym wcześniej kluczem aktywacyjnym dla użytkownika o nazwie *username*. Po pozytywnej weryfikacji konto jest aktywowane.

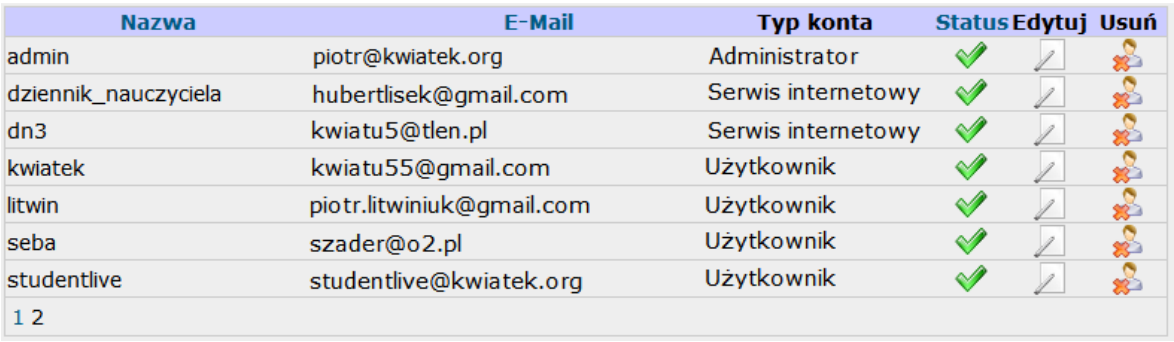
5.7. Panele administracyjne

Jeśli wizyta użytkownika w SRU jest bezpośrednia, tzn. wprowadzony adres nie zawiera loginu serwisu internetowego w pasku adresu po atrybucie „*ref*”, użytkownik po pomyślnym uwierzytelnieniu pozostaje w SRU. Kod formularza Default.aspx, w zależności od typu konta, przenosi użytkownika po zalogowaniu do odpowiedniego panelu administracyjnego:

- https://adres_SRU/User/Accountdetails.aspx - panel dla użytkowników zwykłych,
- https://adres_SRU /Admin/Accountdetails.aspx – panel administratorów SRU,
- https://adres_SRU /Referrer/Accountdetails.aspx – panel dla serwisów internetowych.

Panel zarządzania użytkownikami administratora SRU

Panel administracyjny do zarządzania kontami użytkowników w SRU został oparty na kontrolce *asp:GridView* przedstawionej na rysunku 12.

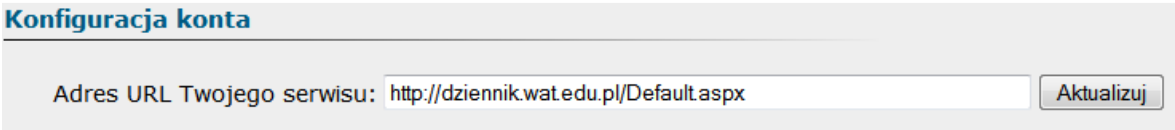


Rys. 12. Panel zarządzania użytkownikami administratora SRU

Tabela zawiera podstawowe informacje o użytkowniku takie jak login oraz adres email. W kolumnie Typ konta w widnieje informacja o roli, do jakiej konto należy. W kolumnie status, ikona informuje czy użytkownik może zalogować się na swoje konto. Status nieaktywny występuje na kontach, które nie dokonały pełnej aktywacji poprzez kliknięcie linku aktywacyjnego z wiadomości e-mail lub na kontach, które zostały wyłączone przez administratora. Zarządzanie kontami umożliwiają dwa graficzne przyciski: edytuj oraz usuń. Naciśnięcie przycisku edytuj wyświetla formularz edycji konta. Istnieje w nim możliwość zmiany statusu konta oraz adresu e-mail. Przycisk usuń powoduje wywołanie metody *DeleteUser* z klasy *RegisteredUser*, która przeprowadza operację bezpiecznego usunięcia użytkownika.

Panel zarządzania kontem serwisu internetowego

Użytkownik po zalogowaniu na konto serwisu internetowego w SRU posiada dostęp do formularza zmiany adresu URL swojej strony internetowej. Wygląd formularza przedstawionego na rysunku 13. Wykonanie polecenia aktualizacji adresu URL powoduje zmianę pola *UrlAddress* w rekordzie tabeli *aspnet_Users* dla danego konta.



Rys. 13. Formularz zmiany adresu URL serwisu internetowego

Panel zarządzania kontem dla użytkowników zwykłych

W panelu zarządzania kontem użytkownika SRU udostępniono narzędzie do wysyłania wniosków z prośbą o dostęp do wybranego serwisu internetowego. Kontrolka *asp:GridView* wyświetla wszystkie konta serwisów internetowych zarejestrowane w SRU. Przy każdym z nich istnieje przycisk. wywołujący metodę *RequestAuthorization* z klasy *RegisteredUser*. Metoda ta tworzy rekord w tabeli *_Users_Referrers_intersection* za pomocą poniższego zapytania SQL:

INSERT INTO _Users_Referrers_intersection (Referrer_UserId, User_UserId, Allowed) VALUES (IdSerwisuInternetowego, IdUzytkownika, 0);

5.8. Interfejs graficzny

System Rejestracji Użytkowników wykorzystuje mechanizm ASP.NET Master Pages. Mechanizm Master Pages pozwala zaprojektować szablon strony, a następnie utworzyć strony z zawartością dodawaną w czasie ładowania strony wzorcowej. W nawiązaniu do rysunku 10 z rozdziału 4.4, szablonem jest cała tabela, natomiast miejscem zmiennej zawartości obszar okna „Zawartość zakładki”.

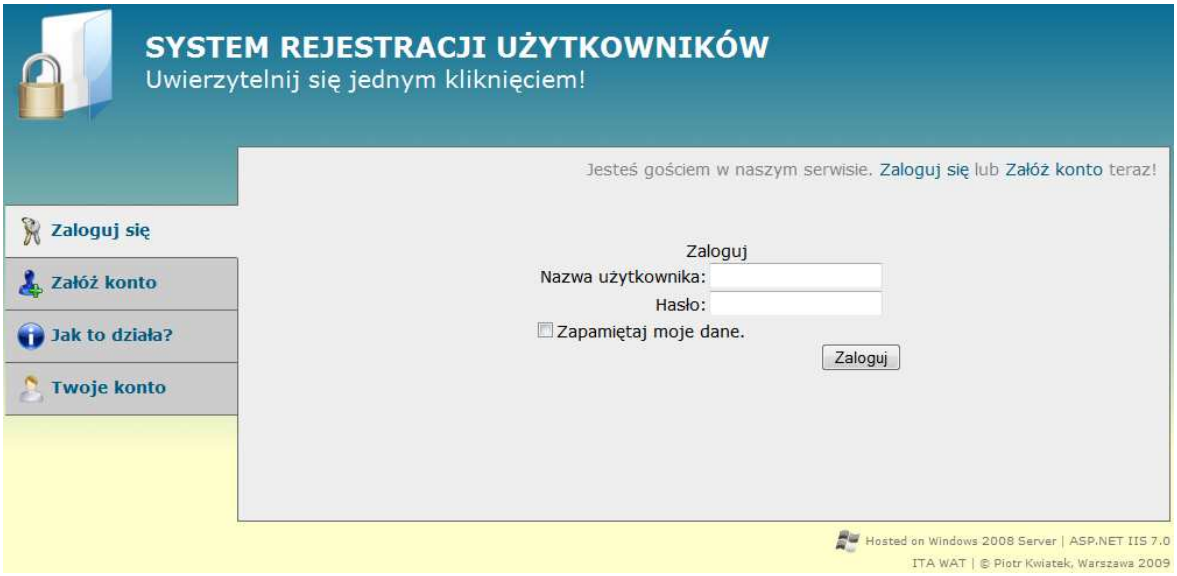
Plik DefaultLayout.master zawiera szablon szaty graficznej SRU. W miejscu prezentowania treści zakładek umieszczono kontrolkę *asp:ContentPlaceHolder*. Każda ze stron zawartości wskazuje na plik wzorcowy z szablonem za pomocą kodu:

<%@ Page Title="" Language="C#"
MasterPageFile="~/DefaultLayout.master"
AutoEventWireup="true"
CodeFile="Accountdetails.aspx.cs"
Inherits="Admin_Accountdetails" %>

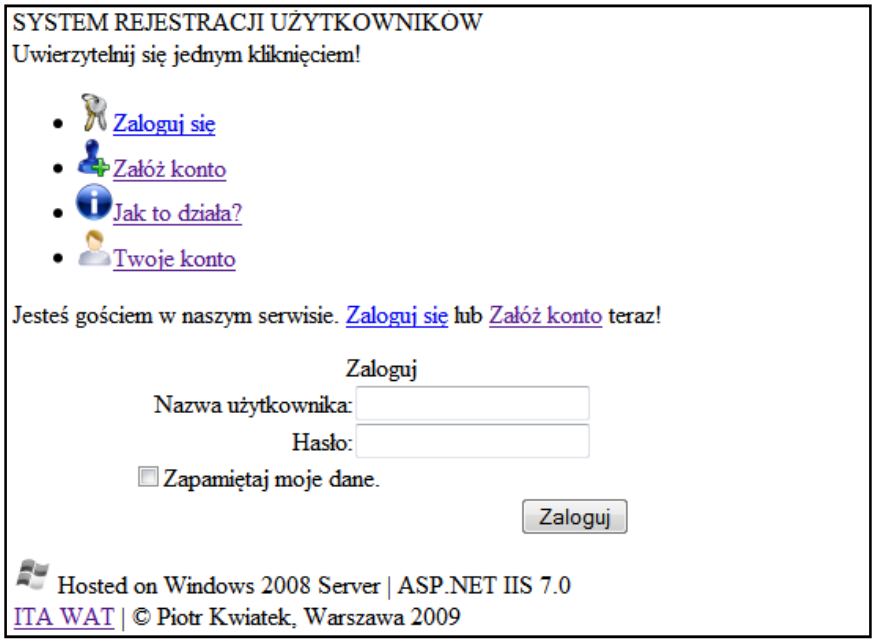
Strona zawartości wypełnia kontrolkę *asp:ContentPlaceHolder* zawartą w szablonie strony DefaultLayout.master wykorzystując kontrolkę *asp:Content*.

Szata graficzna strony internetowej SRU została zaprojektowana zgodnie ze standardem HTML 4.01 oraz CSS 2 z zachowaniem rozdzielności kodu HTML i CSS. Dokument HTML szablonu strony zawiera tylko znaczniki określające

znaczenie poszczególnych części strony, bez informacji na temat ich sposobu prezentacji. Oznacza to, że całość kodu odpowiedzialnego za prezentację zawiera plik z arkuszem kaskadowym stylów CSS. W pliku z arkuszem stylów StyleSheet.css zamieszczona jest lista dyrektyw ustalających w jaki sposób przeglądarka internetowa ma wyświetlić wybrane elementy HTML, takie jak czcionki, kolory tła, tekstu, marginesy, rozmieszczenie elementów DIV. Na rysunku 14 przedstawiony został wygląd strony internetowej SRU z załadowanym arkuszem stylów. W celu przetestowania działania CSS usunięto powiązanie dokumentu HTML z arkuszem stylów. Efekt błędu załadowania pliku StyleSheet.css przedstawia rysunek 15.



Rys. 14. Strona logowania Systemu Rejestracji Użytkowników

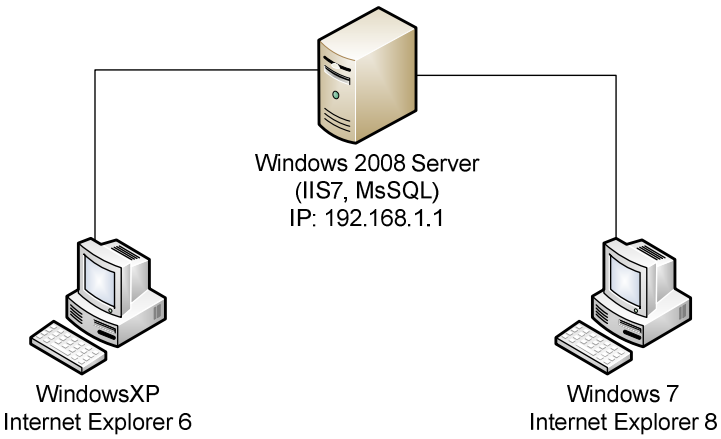


Rys. 15. Błąd ładowania kaskadowego arkusza stylów dla strony SRU

5.9. Testowanie aplikacji

System Rejestracji Użytkowników uruchamiano w różnych konfiguracjach testowych. Podczas testów wykonywano także testy penetracyjne[17], mające na celu ocenę stanu bezpieczeństwa systemu, wykrycie podatności na znane ataki.

Środowisko testowe 1 – pojedynczy serwer



Rys. 16. Schemat środowiska testowego 1

[17] Zabieg wykonywany w celu oszacowania poziomu bezpieczeństwa systemu.

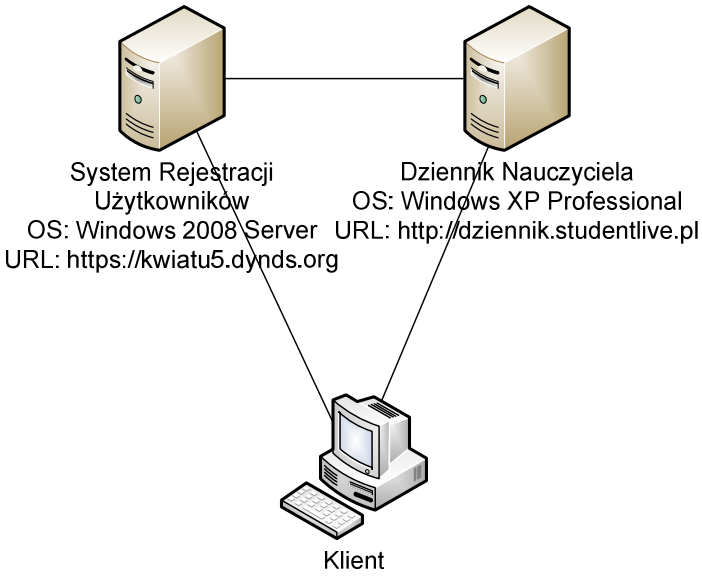
W zaprezentowanej na rysunku 16 konfiguracji testowej komputer z systemem operacyjnym Windows 2008 Server pełni rolę serwera dla SRU, aplikacji „Dziennik Nauczyciela” oraz dla aplikacji zarządzania bazą danych. Poniższy scenariusz testowy ma na celu sprawdzenie wydajności pracy aplikacji oraz przeprowadzenie testu penetracyjnego na użycie nieprawidłowego biletu uwierzytelniania.

1) wywołano adres „Dziennika nauczyciela” *http://192.168.1.1 /dziennik/* na komputerze z systemem Windows XP,
2) przekierowano na stronę *https://192.168.1.1/sru/Default.aspx?ref=dn*,
3) podano poświadczenia w formularzu logowania,
4) przekierowano na stronę: *http://192.168.1.1/dziennik/Default.aspx?ticket=kwiatekEBBF8CF932D41D0A1EEEDF8EDDE2A24F*,
5) zasoby aplikacji „Dziennik nauczyciela” zostały udostępnione,
6) na komputerze z systemem Windows 7 wywołano adres: *http://192.168.1.1/dziennik/Default.aspx?ticket=kwiatekEBBF8CF932D41D0A1EEEDF8EDDE2A24F*,
7) przeglądarka przekierowała użytkownika na stronę: *https://192.168.1.1/sru/Default.aspx?ref=dn*
8) na komputerze z systemem Windows XP uruchomiono podstronę aplikacji „Dziennik nauczyciela”,
9) przekierowano na stronę: *http://192.168.1.1/dziennik/Default.aspx?ticket=kwiatekA3D6A38A5404B22770268E6A098AB6E4.*

Z wyników przeprowadzonego testu można wywnioskować, że SRU pomyślnie autoryzował dostęp użytkownikowi do serwisu internetowego „Dziennik nauczyciela”. W celu testu bezpieczeństwa aplikacji na wypadek odczytania biletu uwierzytelnienia przez osobę postronną, zasymulowano użycie już nieaktualnego biletu na komputerze intruza z systemem Windows 7. Użytkownik próbujący się podszyć nie został autoryzowany. Wywołanie strony aplikacji „Dziennik nauczyciela” na komputerze prawidłowo uwierzytelnionego użytkownika nie powiodło się, ponieważ aktualny bilet został unieważniony. Wnioskując po nowo wygenerowanym bilecie uwierzytelnienia, przekazanego metodą GET, użytkownik

został przeniesiony na stronę SRU, z powodu błędu autoryzacji. Próba podszycia się przez intruza została udaremniona.

Środowisko testowe 2 – SRU osadzony na serwerze internetowym



Rys. 17. Schemat środowiska testowego 2

1) uruchomiono przechwytywanie pakietów w narzędziu Wireshark[18] 1.2.5 na serwerze SRU,
2) wywołano adres „Dziennika nauczyciela” *http://dziennik.studentlive.pl/*
3) przekierowano na stronę *https://kwiatu5.dyndns.org/ Default.aspx?ref=dn3*,
4) podano poświadczenia w formularzu logowania,
5) przekierowano na stronę: *http://dziennik.studentlive.pl?ticket =kwiatekAB28DEC72A1FFE283B9C7CA871003A92*
6) zasoby aplikacji „Dziennik nauczyciela” zostały udostępnione,
7) wywołano dowolną stronę aplikacji „Dziennik nauczyciela”,
8) strona została wyświetlona,
9) wyłączono przechwytywanie pakietów w Wireshark.

[18] Program rejestrujący wszystkie dane przesyłane przez sieć.

Powyższy test miał na celu sprawdzenie działania SRU na oddzielnym, niezależnym serwerze uruchomionym w Internecie. Wyniki testów wskazują na identyczne działanie SRU w obrębie jednego serwera, jak i na niezależnych. Odnotowano jedynie spadek wydajności aplikacji, związany z wolniejszą wymianą danych w sieci Internet. W trakcie działania aplikacji Wireshark uruchomionej na serwerze SRU, odnotowano pakiety protokołu HTTPS wymieniane z aplikacją „Dziennik nauczyciela". Ich zawartość była zaszyfrowana co uniemożliwiało odczytanie jakichkolwiek danych.

Walidacja zgodności ze standardem HTML 4.01 oraz CSS 2

Celem sprawdzenia, czy nie występują błędy w dokumentach HTML oraz CSS wykorzystano udostępnione przez konsorcjum W3C[19] narzędzia weryfikujące dokumenty (X)HTML oraz arkuszy stylów CSS. Oba narzędzia dostępne są w Internecie pod adresem: http://validator.w3.org/. Narzędzia nie znalazły błędów podczas weryfikacji poprawności kodu stron SRU.

[19] Organizacja regulująca standardy kodowania oraz przesyłu stron WWW.

PODSUMOWANIE

Głównym celem pracy była budowa systemu przejmującego rolę uwierzytelniania, autoryzacji oraz zarządzania użytkownikami we wspieranej aplikacji „Dziennik nauczyciela”. W trakcie pracy, podczas analizy dostępnych technologii i narzędzi, system będący początkowo nieodłącznym komponentem wspieranego serwisu internetowego, przerodził się w centralny system bezpiecznego uwierzytelniania i autoryzacji. Dzięki takiej ewolucji Systemu Rejestracji Użytkowników, niniejsza praca wzbogaciła się o wiele analiz bezpieczeństwa, wynikających z aspektu zdalnej autoryzacji. W pracy poruszono kwestię bezpieczeństwa przesyłanych danych pomiędzy serwerami WWW oraz serwerem WWW a klientem.

Dzięki specjalnie utworzonemu środowisku testowemu, udało się przetestować działanie wszystkich usług oferowanych przez SRU we współpracy z aplikacją wspomagającą pracę nauczyciela. Pozytywne wyniki pracy nad SRU pozwalają na stwierdzenie, że cel niniejszej pracy został osiągnięty.

Wdrożenie SRU w Internecie jest możliwe, aczkolwiek ilość wykonanych testów penetracyjnych nie pozwala na uznanie SRU w pełni bezpiecznym. Aplikację należałoby również wyposażyć w podpisany cyfrowo certyfikat SSL. Na tym etapie rozwoju systemu z powodzeniem można używać go w wewnętrznych sieciach firmowych lub uczelnianych, w których zagrożenia są mniejsze, ze względu na bardziej zaufanych użytkowników sieci.

BIBLIOGRAFIA

[1]. Meier J.D.: Mackman A., Dunner M, Vaasireddy S.: *Building Secure ASP.NET Applications. Authentication, Authorization, and Secure Communication*, Microsoft Corporation, 2002.

[2]. Zabir O.: *ASP.NET 3.5. Tworzenie portal internetowych w nurcie Web 2.0*, Wydawnictwo Helion 2008.

[3]. OWASP Team: *A Guide to Building Secure Web Applications*, The Open Web Application Security Project, Giugno 2002.

[4]. Leniek A.: *Słownik slangu informatycznego*, http://www.i-slownik.pl/1,1534,vpn.html, 2006. [stan na: 16.01.2010r.]

[5]. Gajda W.: *Ataki XSS oraz CSRF na aplikacje internetowe,* Internet nr 7/2005.

[6]. Liberty J., Hurwitz D.: *ASP.NET Programowanie*, Helion, Warszawa 2007.

[7]. Matulewski J, Orłowski S.: *Uwierzytelnianie i autoryzacja w ASP.NET 2.0*, artykuł hakin9 Nr 6/2007.

[8]. Dubisz S.: *Uniwersalny słownik języka polskiego PWN*, Wydawnictwo Naukowe PWN, 2006.

[9]. Howard R.: *Web Services with ASP.NET*, Microsoft Corporation 2001, http://msdn.microsoft.com/en-us/library/ms972326.aspx. [stan na: 16.01.2010r.]

[10]. MSDN Library: *Securing Membership*, Microsoft Corporation 2010, http://msdn.microsoft.com/en-us/library/ms178398.aspx. [stan na: 16.01.2010r.]

[11]. Skonnard A.: *Understanding SOAP*, Microsoft Corporation 2003, http://msdn.microsoft.com/en-us/library/ms995800.aspx. [stan na: 16.01.2010r.]

WYKAZ TABEL

WYKAZ RYSUNKÓW

Wyrażam zgodę na udostępnienie mojej pracy przez Bibliotekę Główną WAT w czytelni oraz w ramach wypożyczeń międzybibliotecznych.

Data	*Podpis*
18.01.2010r.	*……..……*