

# Uwierzytelnianie do aplikacji webowych za pomocą certyfikatu klienta

Wojciech Sabat 13k5

---

## 1. Wstęp teoretyczny

Współcześnie, sieci komputerowe odgrywają ogromną rolę w funkcjonowaniu całych społeczeństw. Są nie tylko źródłem rozrywki, ale pełnią również kluczową rolę w finansach, biznesie, a nawet bezpieczeństwie publicznym na każdym szczeblu. Dlatego tak ważne jest zapewnienie integralności i poufności ruchu w Internecie, czy intranecie. Obecne rozwiązania w tym zakresie bazują na szyfrowaniu symetrycznym i asymetrycznym, oraz certyfikatach klucza publicznego.

### 1.1 Szyfrowanie

Szyfrowanie to proces przekształcenia informacji dającej się odczytać na niezrozumiałą ciąg w celu jej utajnienia. Informację zrozumiałą nazywamy jawną, niezrozumiałą zaś szyfrogramem lub kryptogramem. Szyfry dzielimy na szyfry ograniczone oraz szyfry z kluczem. Pierwszy rodzaj nie będzie wykorzystywany w projekcie będącym przedmiotem tego dokumentu. Spośród drugiego rodzaju wyróżniamy zaś szyfry z algorytmem symetrycznym i asymetrycznym. Algorytmy symetryczne to takie, w których klucz szyfrujący może być wyznaczony z klucza deszyfrującego i klucz deszyfrujący z klucza szyfrującego. W algorytmach asymetrycznych klucz deszyfrujący(prywatny) jest inny od klucza szyfrującego(publicznego) i nie można wyznaczyć go z klucza szyfrującego. Szyfrowanie asymetryczne wykorzystuje operacje jednokierunkowe, czyli takie, które da się łatwo przeprowadzić w jedną stronę, ale wyjątkowo trudno w drugą. Algorytmy asymetryczne są bardzo bezpieczne jednak ich wykonanie trwa znacznie dłużej. Dlatego też, obecne rozwiązania polegają na ustanowieniu połączenia szyfrowanego asymetrycznie, jedynie do ustalenia klucza symetrycznego, którym szyfrowana będzie dalsza część komunikacji.

## 1.2 Certyfikaty

Certyfikat klucza publicznego to informacja o kluczu publicznym podmiotu, podpisana(uwierzytelniona) przez zaufaną третią stronę. Certyfikat musi składać się z: klucza publicznego podmiotu uwierzytelnianego, opisu jego tożsamości oraz cyfrowego podpisu zaufanej organizacji. Owe zaufane trzecie strony, to instytucje certyfikujące (Certificate Authority), które są powszechnie uważane za wiarygodne i po sprawdzeniu wiarygodności podmiotu oferują usługę podpisania jego certyfikatu. W utworzonym projekcie nie będzie wykorzystany jednak podpis żadnej zaufanej instytucji. Zamiast tego utworzony zostanie certyfikat samo podpisany (self-signed certificate). Podstawowym standardem certyfikatów, który jest obecnie wykorzystywany jest standard X.509, który został również użyty w omawianym projekcie. Certyfikaty w przeważającej większości przypadków wykorzystywane są do potwierdzenia wiarygodności serwera. Czasami jednak wykorzystuje się je do uwierzytelnienia klienta w wypadkach, gdy konieczny jest najwyższy poziom bezpieczeństwa. W projekcie przedstawiona została implementacja tego drugiego przypadku.

## 1.3 SSL i TLS

Transport Layer Security to przyjęte jako standard w Internecie rozwinięcie protokołu kryptograficznego SSL (Secure Sockets Layer). Zapewnia on bezpieczną komunikację w sieciach komputerowych poprzez poufność i integralność transmisji danych. Opierają się na opisanych wcześniej szyfrowaniu asymetrycznym i certyfikatach X.509.

# 2. Implementacja

## 2.1 Utworzenie certyfikatów

Do utworzenia certyfikatów wykorzystana została implementacja protokołów SSL i TLS o nazwie OpenSSL. Jest ona darmowa, a dodatkowo wspiera wszystkie popularne systemy operacyjne. Certyfikaty wygenerowane były za pomocą interfejsu konsolowego. Cały proces przebiegał następująco:

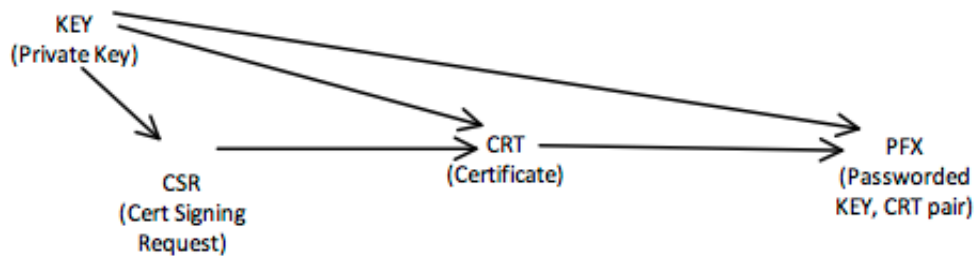
### **Stworzenie certyfikatu głównego:**

1. Wygenerowanie klucza prywatnego Instytucji uwierzytelniającej  
`openssl genrsa -out SabatKeyCA.key 2048`
2. Stworzenie prośby o podpisanie certyfikatu  
`openssl req -new -nodes -key SabatKeyCA.key -out SabatRequestCA.csr`
3. Samopodpisanie certyfikatu  
`openssl x509 -req -trustout -days 365 -in SabatRequestCA.csr -signkey SabatKeyCA.key -out SabatCertCA.crt`
4. Stworzenie pary certyfikatu i klucza publicznego  
`openssl pkcs12 -export -in SabatCertCA.crt -inkey SabatKeyCA.key -out SabatPairCA.pfx`

### **Stworzenie certyfikatu klienta:**

5. Wygenerowanie klucza prywatnego klienta  
`openssl genrsa -out sabattestclientcert.key 2048`
6. Stworzenie prośby o podpisanie certyfikatu  
`openssl req -new -nodes -key sabattestclientcert.key -out sabattestclientcert.csr`
7. Podpisanie certyfikatu klienta za pomocą certyfikatu głównego I jego klucza prywatnego  
`openssl x509 -req -days 365 -in sabattestclientcert.csr -CA SabatCertCA.crt -CAkey SabatKeyCA.key -set_serial 01 -out sabattestclientcert.crt`
8. Stworzenie pary certyfikatu i klucza publicznego klienta  
`openssl pkcs12 -export -in sabattestclientcert.crt -inkey sabattestclientcert.key -out sabattestclientcert.pfx`

Prosty schemat tworzenia plików z odpowiadającymi rozszerzeniami wygląda następująco:



## 2.2 Import certyfikatów

System Windows posiada Magazyn certyfikatów, czyli miejsce, w którym przechowywane są wszystkie certyfikaty zarówno Instytucji zaufany, jak i prywatne certyfikaty użytkownika, które służą do jego uwierzytelniania. Do zarządzania nimi wykorzystywany jest program mmc.exe, znajdujący się w folderze `$\Windows\System32`. Certyfikaty pogrupowane są w trzech przestrzeniach: certyfikaty użytkownika, certyfikaty usług oraz certyfikaty maszyny. W przestrzeni certyfikatów maszyny, do podkatalogu „Trusted Root Certification Authorities” zaimportowany został plik certyfikatu głównego. Z kolei do podfolderu „Personal” zaimportowana została para certyfikatu z kluczem prywatnym klienta.

## 2.3 Utworzenie projektu aplikacji

Tworzenie aplikacji webowych to obecnie bardzo szeroko rozwinięta gałąź programowania. Ilość dostępnych frameworków do budowy tego typu systemów jest bardzo duża i daje programistom szeroki wachlarz możliwości. Do implementacji projektu wykorzystana została platforma ASP.NET MVC, rozwijana przez firmę Microsoft i oparta na technologii ASP.NET. Głównym powodem dokonania takiego wyboru była znajomość tego środowiska oraz dostępność gotowej, rozbudowanej infrastruktury.

## 2.4 Struktura aplikacji

Aplikacja jest jedynie szkieletem, służącym do zaprezentowania implementacji uwierzytelniania przy pomocy certyfikatów klienta i nie dostarcza ona żadnej rozbudowanej funkcjonalności. Nie jest też szczególnie użyteczna. Spełnia jednak swoją rolę przykładu działającego, zabezpieczonego walidacją certyfikatów oprogramowania. Jej najważniejsze części to model studenta „Student.cs”, repozytorium przechowujące dane studentów „StudentRepository.cs”, dwa kontrolery „HomeController.cs” i

„StudentGradesController.cs”, oraz odpowiadające im widoki. Koncept zakłada dostępność strony głównej dla każdego użytkownika witryny. Oceny studentów zaś jako dane tajne, powinny być zabezpieczone i dostępne jedynie uwierzytelnionym klientom (w domyśle studentom).

[Admibesiko](#) [Strona główna](#) [API](#) [Dane ściśle tajne](#)

# Administracja i Bezpieczeństwo Systemów Komputerowych

Projekt przedstawiający wykorzystanie certyfikatu klienta do uwierzytelnienia w aplikacji webowej.

Te dane są jawne i mogą być widziane przez każdego

Politechnika Krakowska

Politechnika Krakowska »

Certyfikaty

Certyfikaty SSL/TLS według Wikipedii

Wikipedia »

Szyfrowanie asymetryczne według Wikipedii

Szyfrowanie klucza publicznego.

Wikipedia »

© 2017 - Wojciech Sabat Grupa 13k5

*Strona główna*

sabattestsite.local/StudentGrades

[Admibesiko](#) [Strona główna](#) [API](#) [Dane ściśle tajne](#)

## Ściśle tajne oceny końcowe

Name	Surname	University	Group	Final Grade
Jan	Kowalski	Politechnika Krakowska	13k5	4
Maciej	Nowak	Uniwersytet Jagielloński	12k8	3
Zbigniew	Zawisza	Politechnika Krakowska	13k4	2,5
Wioletta	Kwiecień	Uniwersytet Jagielloński	13k2	4,75
Zygmunt	Noszczyński	Akademia Górniczo Hutnicza	11k1	4
Monika	Kubowicz	Politechnika Krakowska	11k4	3
Karolina	Curuś	Uniwersytet Jagielloński	13k5	2
Marcin	Miśkowiec	Politechnika Krakowska	11k3	3,5
Abelard	Stokłosa	Akademia Górniczo Hutnicza	13k4	4
Anastazja	Kamiński	Politechnika Krakowska	13k4	4,98
Jarosław	Lem	Politechnika Krakowska	12k5	3
Eugeniusz	Szymański	Akademia Górniczo Hutnicza	13k1	4,4

© 2017 - Wojciech Sabat Grupa 13k5

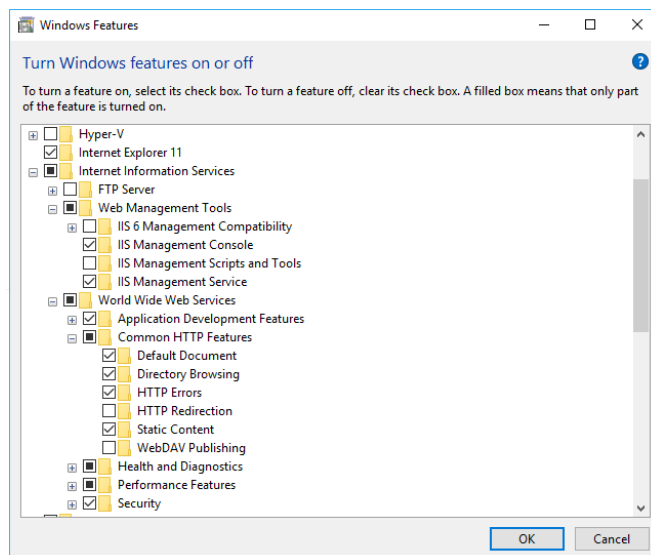
*Oceny studentów*

## 2.5 Problem z serwerem IIS

IIS (Internet Information Services) to zbiór usług internetowych, wykorzystywanych między innymi do wdrażania aplikacji webowych i uruchamiania ich na serwerze. Pozwala on na rozbudowaną konfigurację aplikacji. Razem z zintegrowanym środowiskiem programistycznym Microsoft Visual Studio instalowana jest uproszczona wersja serwera o nazwie IIS Express. Niestety nie pozwala ona na uruchomienie aplikacji z wykorzystaniem protokołu TLS, co jest kluczowe dla omawianego przykładu.

## 2.6 Konfiguracja IIS

Aby włączyć serwer IIS w systemie Windows należy w panelu sterowania odnaleźć opcje wyłączenia/włączenia dodatkowych funkcjonalności systemu i uaktywnić Internet Information Services jak poniżej:

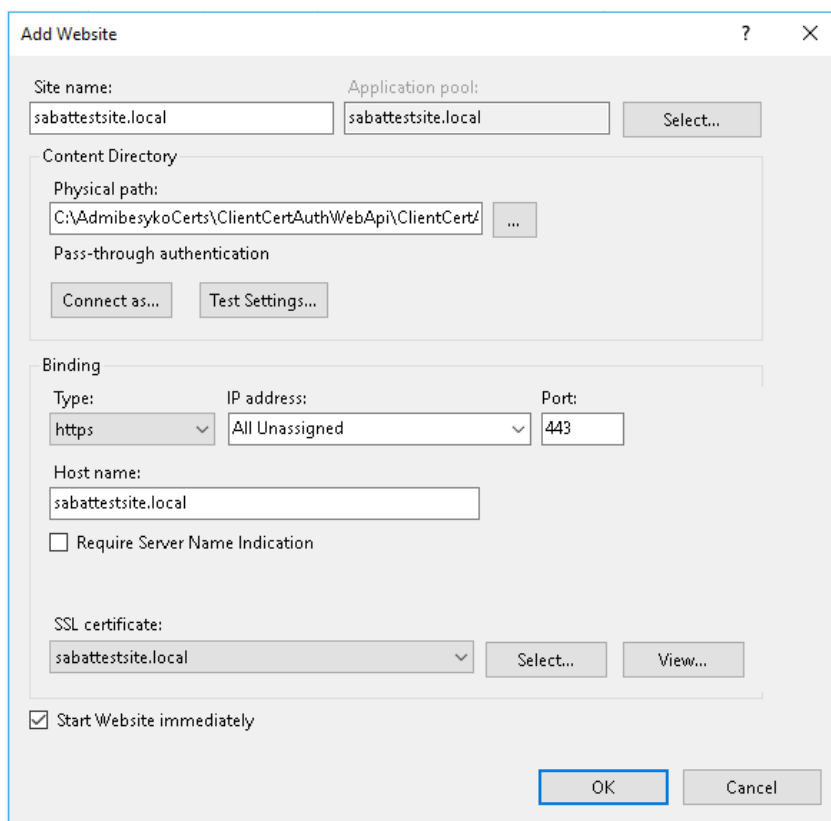


*Turn Windows features on or off*

Do uruchomienia szyfrowania w IIS konieczny jest również certyfikat serwera, na którym aplikacja jest uruchomiona. Mimo, że projekt dotyczy certyfikatów klienta należy więc wygenerować go także dla serwera, zapewniając tym samym również jego wiarygodność. Dokonujemy tego analogicznie jak w punkcie 2.1, wykorzystując OpenSSL. Certyfikat główny importujemy następnie jak w punkcie 2.2 do podfolderu „Trusted Root Certification Authorities”. Tym razem wybieramy jednak przestrzeń certyfikatów maszyny. Wygenerowaną parę certyfikatu i klucza prywatnego strony internetowej importujemy do podfolderu Personal w tej samej przestrzeni.

W Administrative Tools należy uruchomić następnie Internet Information Services i dodać nową stronę o nazwie jaka została użyta w jako „Common Name” utworzonego uprzednio certyfikatu. „Physical Path” powinna wskazywać na lokalizację solucji

projektu. W sekcji „Binding” ustawiamy typ na https z opcją adresu „All Unassigned” i portem 443(domyślnym dla połączeń szyfrowanych). Wybieramy też nazwę hosta(jak nazwa strony ) i certyfikat strony.



The screenshot shows the 'Add Website' dialog box with the following settings:

- Site name:** sabattestsite.local
- Application pool:** sabattestsite.local
- Content Directory:**
  - Physical path:** C:\Admibeszko\Certs\ClientCertAuthWebApi\ClientCert
  - Pass-through authentication:** Connect as... Test Settings...
- Binding:**
  - Type:** https
  - IP address:** All Unassigned
  - Port:** 443
  - Host name:** sabattestsite.local
  - Require Server Name Indication:** ☐
  - SSL certificate:** sabattestsite.local
- Start Website immediately:** ☒

*Utworzenie nowej strony*

Należy zmienić ustawienia SSL z opcji „Ignore” na „Accept”:



## SSL Settings

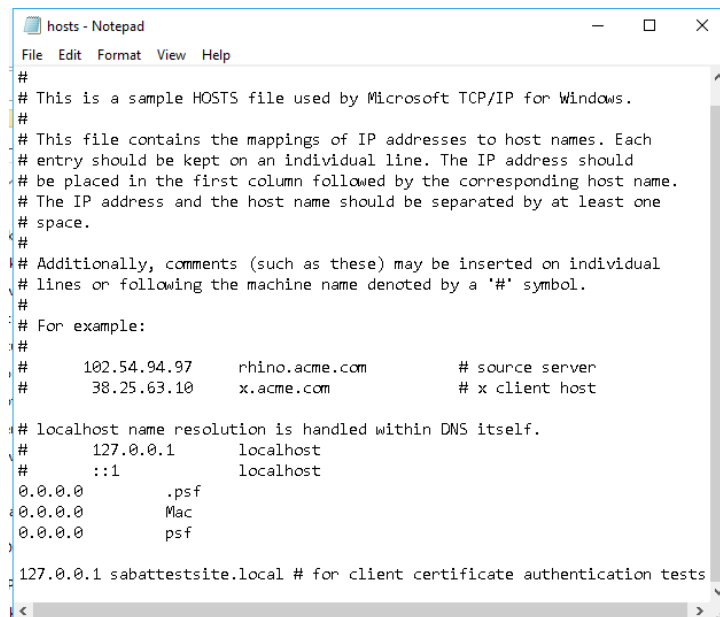
This page lets you modify the SSL settings for the content of a website or application.

☒ Require SSL

Client certificates:

- ☐ Ignore
- ☒ Accept
- ☐ Require

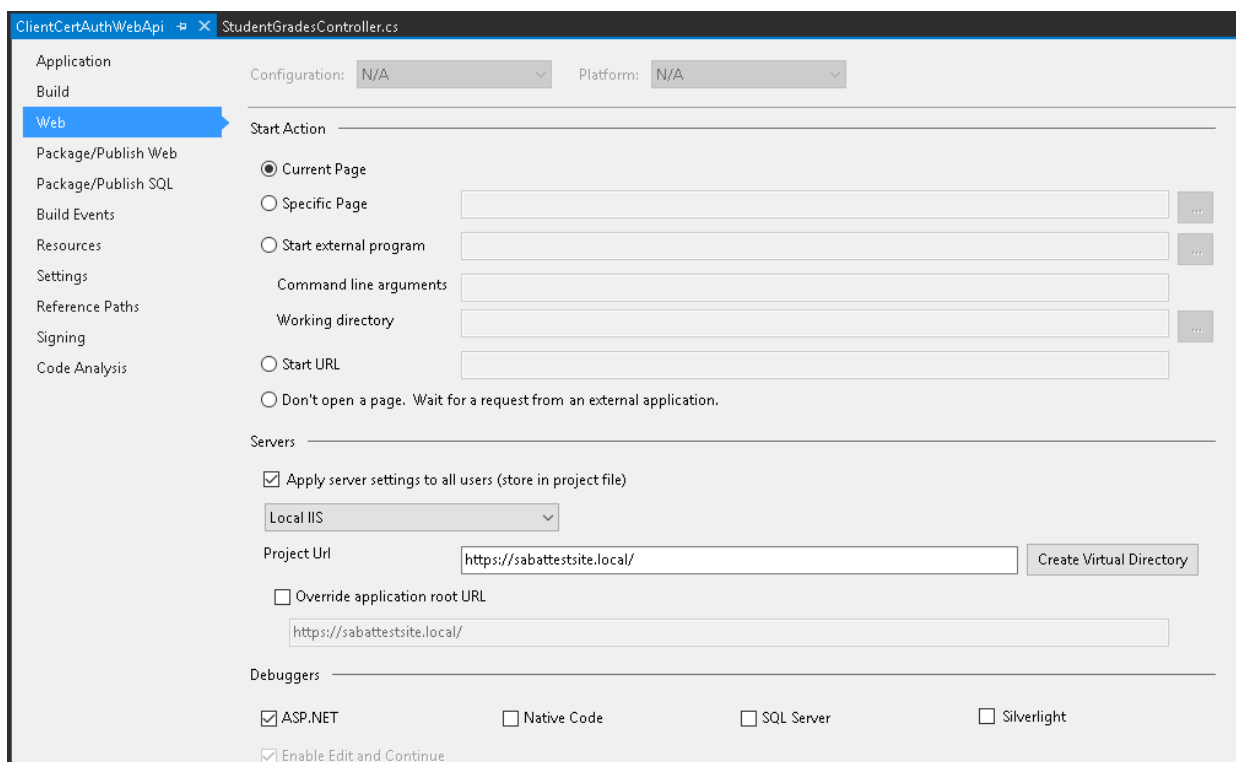
Dla utworzonej strony należy zmapować nazwę domeny na adres IP localhost (127.0.0.1). Dokonuje się tego przez dodanie linii z adresem IP i domeną do pliku konfiguracyjnego hosts.txt, znajdującego się w ścieżce C:\Windows\System32\drivers\etc.



```
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97   rhino.acme.com   # source server
#       38.25.63.10   x.acme.com      # x client host
#
# localhost name resolution is handled within DNS itself.
#       127.0.0.1     localhost
#       ::1           localhost
0.0.0.0               .psf
0.0.0.0               Mac
0.0.0.0               psf
127.0.0.1 sabattestsite.local # for client certificate authentication tests
```

*Plik hosts.txt*

Ostatnią czynnością jest zmiana ustawień serwera w projekcie aplikacji. Aby móc używać pełnoprawnego serwera IIS, środowisko Visual Studio musi za każdym razem być uruchamiane za pomocą konta administratora. We właściwościach projektu, w sekcji Web zmieniamy serwer z IIS Express na Local IIS i uzupełniamy adres strony:

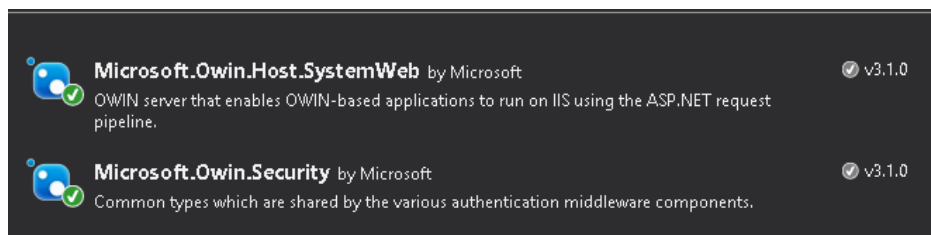


*Właściwości projektu aplikacji*



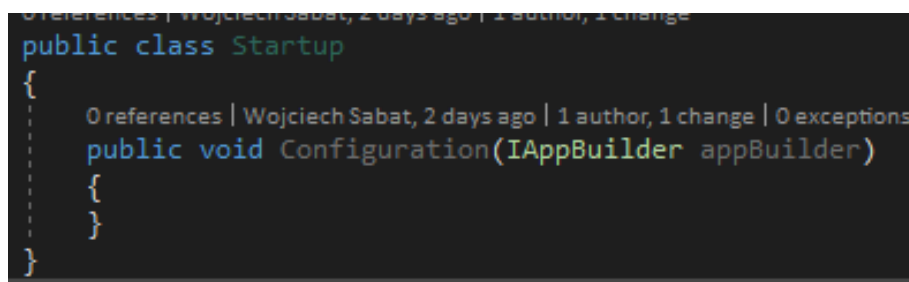
## 2.7 Owin i Katana

Katana to otwarty, tworzony przez Microsoft framework oparty o specyfikację Owin opisującą abstrakcje nad serwerem IIS pozwalającą na przenośność aplikacji i instalowanie ich na dowolnym środowisku. Katana udostępnia szereg funkcjonalności w tym opcje bezpieczeństwa, które wykorzystane zostały w projekcie. Należy więc z pakietów Nuget dodać referencję do pakietu „Microsoft.Owin.Host.System.Web” oraz „Microsoft.Owin.Security”.



*Nuget*

Do poprawnego działania katany należy do głównego katalogu projektu dodać klasę „Startup.cs” z metodą „Configuration”:

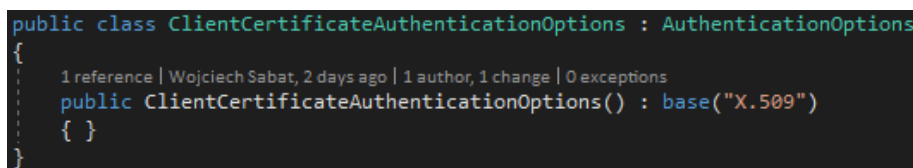


*Startup.cs*

Dodane biblioteki zawierają szereg klas, które wykorzystane zostaną w kolejnych punktach.

## 2.8 Implementacja klas bazowych

Klasa bazowa „AuthenticationOptions.cs” z bibliotek Owin to klasa definiująca opcje i sposób uwierzytelniania. Do celów projektu wystarczy dziedziczącą po niej klasę wyposażyć w konstruktor wywołujący konstruktor bazowy z argumentem „X.509”.



*ClientCertificateAuthenticationOptions.cs*

„AuthenticationHandler.cs” to klasa z tej samej biblioteki, która odpowiada za akcje wywoływaną podczas żądania przechwytywanego przez oprogramowanie pośredniczące obsługujące uwierzytelnianie. Posiada ona obiekt walidatora certyfikatu, którego implementacja wykonania zostanie w następnych krokach. Używa go do sprawdzenia certyfikatu w metodzie „Validate”.

```
public class ClientCertificateAuthenticationHandler : AuthenticationHandler<ClientCertificateAuthenticationOptions>
{
    private readonly IClientCertificateValidator _clientCertificateValidator;
    private readonly string _owinClientCertKey = "ssl.ClientCertificate";

    1 reference | Wojciech Sabat, 14 minutes ago | 1 author, 2 changes | 0 exceptions
    public ClientCertificateAuthenticationHandler(IClientCertificateValidator clientCertificateValidator)...

    0 references | Wojciech Sabat, 14 minutes ago | 1 author, 2 changes | 0 exceptions
    protected override async Task<AuthenticationTicket> AuthenticateCoreAsync()...

    1 reference | Wojciech Sabat, 2 days ago | 1 author, 1 change | 0 exceptions
    private ClientCertificateValidationResult ValidateCertificate(IDictionary<string, object> owinEnvironment)...
}
```

*ClientCertificateAuthenticationHandler.cs*

„ClientCertificateValidationResult.cs” to klasa opakowująca wynik walidacji dokonanej przez klasę walidatora. Zawiera pole typu boolean z informacją o poprawności certyfikatu oraz metody zajmujące się przechowywaniem ewentualnych wyjątków.

```
9 references | Wojciech Sabat, 2 days ago | 1 author, 1 change
public class ClientCertificateValidationResult
{
    private readonly bool isCertificateValid;
    private readonly IEnumerable<string> _validationExceptions;

    2 references | Wojciech Sabat, 2 days ago | 1 author, 1 change | 0 exceptions
    public ClientCertificateValidationResult(bool isCertificateValid)...

    1 reference | Wojciech Sabat, 2 days ago | 1 author, 1 change | 0 exceptions
    public void AddValidationExceptions(IEnumerable<string> validationExceptions)...

    1 reference | Wojciech Sabat, 2 days ago | 1 author, 1 change | 0 exceptions
    public void AddValidationException(string validationException)...

    1 reference | 0 changes | 0 authors, 0 changes | 0 exceptions
    public bool IsCertificateValid => _isCertificateValid;
}
```

*ClientCertificateValidationResult.cs*

Klasa „ClientCertificateValidator.cs” to klasa, w której następuje sprawdzenie ważności certyfikatu klienta. Wykorzystuje ona obiekt klasy X509Chain, który odpowiada za obsługę przestrzeni certyfikatów zainstalowanych w systemie. Jeśli certyfikat okaże się prawidłowa metoda „Validate” zwróci obiekt klasy „ClientCertificateValidationResult.cs” z właściwością „IsCertificateValid” ustawioną na wartość „true”. W przeciwnym wypadku do instancji tej klasy doda listę wyjątków pobranych ze statusu elementów obiektu X509Chain.

```

public class ClientCertificateValidator : IClientCertificateValidator
{
    2 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public ClientCertificateValidationResult Validate(X509Certificate2 certificate)
    {
        var isValid = false;
        var exceptions = new List<string>();
        try
        {
            var chain = new X509Chain();
            var chainPolicy = new X509ChainPolicy...;
            chain.ChainPolicy = chainPolicy;
            if (chain.Build(certificate))
                isValid = true;
            else
                foreach (X509ChainElement chainElement in chain.ChainElements)
                {
                    foreach (X509ChainStatus chainStatus in chainElement.ChainElementStatus)
                    {
                        exceptions.Add(chainStatus.StatusInformation);
                    }
                }
        }
        catch (Exception ex)
        {
            exceptions.Add(ex.Message);
        }
        var result = new ClientCertificateValidationResult(isValid);
        result.AddValidationExceptions(exceptions);
        return result;
    }
}

```

*ClientCertificateValidator.cs*

Kolejnym krokiem jest dopisanie używania systemu walidacji do klasy „Startup.cs”, metody konfiguracyjnej:

```

public class Startup
{
    0 references | Wojciech Sabat, 45 minutes ago | 1 author, 2 changes | 0 exceptions
    public void Configuration(IApplicationBuilder appBuilder)
    {
        appBuilder.UseClientCertificateAuthentication(new ClientCertificateValidator());
    }
}

```

*Startup.cs*

Ostatni krok polega na dekoracji akcji kontrolerów, lub też całych kontrolerów atrybutem [Authorize]. Każda akcja udekorowana tym atrybutem lub znajdująca się w kontrolerze nim udekorowanym, będzie wywoływana dopiero po sprawdzeniu wiarygodności użytkownika. W przeciwnym wypadku użytkownikowi odmówiony zostanie dostęp do zasobów zwracanych przez akcję.

```

[Authorize]
0 references | 0 changes | 0 authors, 0 changes
public class StudentGradesController : Controller
{
    private StudentRepository repository = new StudentRepository();

    0 references | 0 changes | 0 authors, 0 changes | 0 requests | 0 exceptions
    public ActionResult Index()
    {
        var students = _repository.GetAll();
        return View(students);
    }
}

```

*StudentGradesController.cs*