Politechnika Poznańska
Wydział Elektryczny
Instytut Automatyki i Inżynierii Informatycznej

Dokumentacja projektu „CompSpy” system monitorowania pracowni laboratoryjnej

Podstawy Teleinformatyki

Norbert Langner (121995)
norbert.langner@student.put.poznan.pl

Marcin Stupin (122086)
marcin.stupin@student.put.poznan.pl

Katarzyna Sroka (122100)
katarzyna.sroka@student.put.poznan.pl

Grupa: BSI-2

2 lipca 2017

Spis treści

1 Opis projektu

Celem projektu było utworzenie systemu pozwalającego prowadzącemu zajęcia monitorować komputery w pracowni laboratoryjnej. Aplikacja do pracy wymaga skonfigurowania serwera aplikacji (IIS z obsługą ASP.NET), połączonego siecią lokalną z komputerami w laboratorium.

Serwer udostępnia serwis internetowy, do którego prowadzący zajęcia loguje się za pomocą przeglądarki internetowej. Udostępnia on podgląd pulpitów i informacji o procesach uruchomionych na aktywnych stanowiskach komputerowych. W domyślnym widoku dostępny jest podgląd ekranów z wybranej pracowni, każdy w oddzielnej komórce siatki – podobnie jak w systemach monitoringu wizyjnego CCTV. Po wybraniu odpowiedniego stanowiska, program wyświetla podgląd z pulpitu danego komputera w wyższej jakości wraz z dodatkowymi danymi. Są to uruchomione aplikacje oraz otwarte karty w przeglądarkach komputerowych. W czasie, gdy prowadzący będzie zalogowany na swój panel, aplikacje klienckie zbierają dane i przesyłają je na serwer. Monitoring nie odbywa się w momencie, gdy żaden prowadzący nie prowadzi monitoringu – oszczędność zasobów sprzętowych oraz przepustowości sieci lokalnej.

System umożliwia definiowane tzw. „czarnej listy" procesów, a w momencie wykrycia, że użytkownik uruchomił jakikolwiek proces z tej listy, zapisuje we właściwej tabeli informacje o naruszeniach. Dodatkowo, do każdego naruszenia zapisywany jest zrzut ekranu.

1.1 Wybór tematu

Temat został wybrany posługując się dwoma kryteriami: potencjalnej przydatności oraz poziomu umiejętności grupy projektowej. Wybrany projekt może być podstawą dla aplikacji do kompleksowego zbierania danych o pracy pracowników w miejscu pracy lub monitoringu publicznie dostępnych komputerów.

Przykładem takiego powszechnie wykorzystywanego systemu jest system podglądu kas samoobsługowych w supermarketach. Udostępniane klientom jest zwykle kilka kas, będących pod ciągłym nadzorem pracownika sklepu, który oprócz rozwiązywania problemów i pomocy klientom, jest odpowiedzialny także za sprawdzanie, czy produkty są odpowiednio skanowane.

Należy wspomnieć o aspekcie prawnym stosowania takich aplikacji. Zgodnie z polskim prawem monitoring musi szanować godność oraz dobra osobiste pracownika (art. 11 Kodeksu Pracy).

1.2 Architektura

W tym rozdziale krótko przedstawiono architekturę projektowanego systemu. W dalszej części niniejszej dokumentacji będzie to punkt odniesienia do sposobu działania aplikacji.

Rysunek 1: Projekt architektury systemu

Rysunek 1. przedstawia architekturę projektowanego systemu. Stanowiska klienckie to komputery użytkowników np. w sali komputerowej, na których uruchamiane są aplikacje klienckie. Stanowisko administratora, jest to komputer, który potrzebuje dostępu do przeglądarki internetowej, aby skorzystać z aplikacji. Serwer natomiast udostępnia serwis internetowy, do którego łączą się administratorzy/nauczyciele i stanowiska kliencie. Tutaj znajduje się także baza danych systemu.

2 Wymagania funkcjonalne

W poniższej sekcji przedstawiono wymagania funkcjonalne projektowanego systemu – zbiór funkcjonalności, które zestaw aplikacji powinien spełnić. Wymagania dotyczą danych obszarów działania oraz związane są z aktorami (zewnętrzny obiekt wchodzący w interakcję z systemem).

Na początku przedstawiono aktorów. Ich lista wraz z opisem znajdują się w Tabeli 1.

Tabela 1: Aktorzy systemu

Aktor	Opis aktora
Aplikacja agenta	Program zainstalowany na stanowisku komputerowym. Zbiera dane z maszyny klienta.
Użytkownik zalogowany	Jest to użytkownik, który ma możliwość zalogowania się na stronie, która znajduje się na serwerze i umożliwia podgląd stanowisk połączonych z serwerem.
Administrator	Jest to użytkownik zalogowany, ale oprócz jego funkcjonalności może także zarządzać klientami, grupami stanowisk oraz użytkownikami.

Tabela 2. przedstawia zebrane wymagania funkcjonalne projektowanego systemu. W kolejnych kolumnach opisano żądaną funkcjonalność, aktora (lub aktorów) (według Tabeli 1) uczestniczącego w danej funkcjonalności oraz obszar działania tożsamy z modułami aplikacji.

Tabela 2: Lista wymagań funkcjonalnych

l.p.	Funkcjonalność	Aktor	Obszar
1.	Wykonywanie zrzutów ekranu w określonej częstotliwości i przesyłanie ich do serwera	Aplikacja agenta	Agent, Serwer
2.	Definiowanie stanowisk komputerowych w systemie.	Administrator	Serwis internetowy
3.	Tworzenie i zarządzanie grupami stanowisk.	Administrator	Serwis internetowy
4.	Podgląd wielu stanowisk jednocześnie w ramach wybranej grupy.	Administrator, Użytkownik zalogowany	Serwis internetowy
5.	Podgląd obrazu w lepszej jakości ze stanowisk wraz z listą uruchomionych aplikacji oraz kart z przeglądarek internetowych	Administrator, Użytkownik zalogowany	Serwis internetowy
6.	Zarządzanie czarną listy procesów, które umożliwią alarmowanie monitorującego o naruszeniu zasad.	Administrator	Serwis internetowy
7.	Wysyłanie wiadomości tekstowej do wybranego komputera lub grupy stanowisk.	Administrator, Użytkownik zalogowany	Serwis internetowy
8.	Zarządzanie użytkownikami i administratorami.	Administrator	Serwis internetowy
9.	Zarządzanie uprawnieniami użytkowników do sal.	Administrator	Serwis internetowy

3 Wymagania pozafunkcjonalne

W tej sekcji zostały przedstawione kryteria i ograniczenia, jakie są nałożone na system. Ze względu na architekturę zaprezentowaną w sekcji 1.2, wymagania zostały rozdzielone na trzy kategorie: aplikację klienta, serwera systemu oraz aplikacji webowej. Poniższe wymagania są niezbędne do poprawnego działania wszystkich aplikacji.

Dla aplikacji klienta:

- System operacyjny Windows 7 lub nowszy
- minimalna wersja .NET Framework 4.5.2
- aplikacja okienkowa uruchamiana wraz ze startem systemu
- polska wersja językowa

Dla serwera systemu:

- System operacyjny Windows Server 2012 (ew. Windows 8) lub nowszy
- minimalna wersja .NET Framework 4.5.2
- minimalna wersja Entity Framework 6.1
- Microsoft SQL Server 2016
- serwer IIS z zainstalowaną obsługą ASP.NET

Dla aplikacji webowej:

- przeglądarka internetowa (minimalne wersje: Google Chrome 56, Mozilla Firefox 52 lub Opera 43)
- aplikacja dostępna wyłącznie w języku polskim

4 Środowisko i użyte technologie

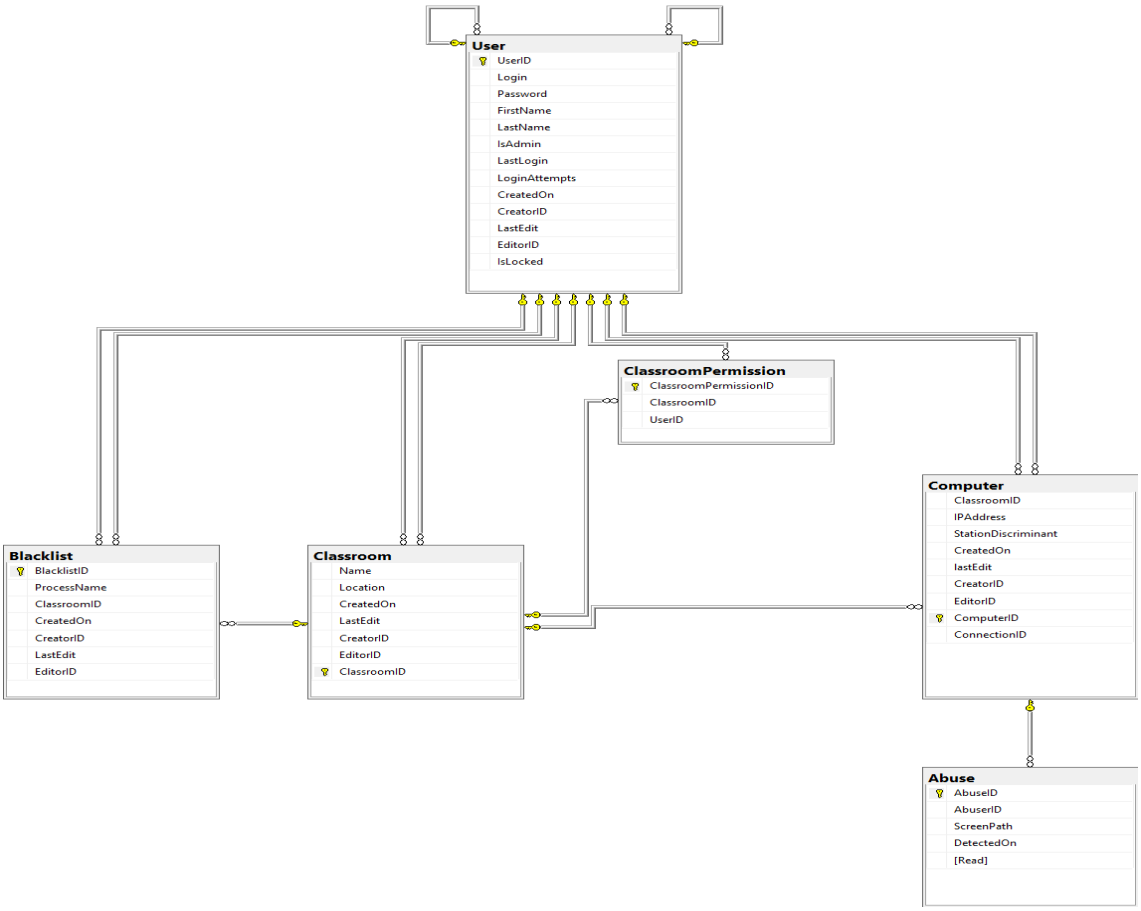
Wykonanie projektu zakłada wykorzystanie kilku narzędzi, bibliotek, edytorów oraz środowiska programistycznego. Odpowiednią listę przedstawiono poniżej.

- Microsoft Visual Studio 2015 (IDE),
- język programowania: C#,
- Microsoft SQL Server 2016 (baza danych),
- Microsoft SQL Server Management Studio (zarządzanie bazą danych),
- .NET Framework 4.5.2,
- Entity Framework 6.1 (ORM),
- ASP.NET MVC 4.6,
- serwis GitHub (repozytorium kodu źródłowego),
- narzędzie sharelatex.com (sporządzenie niniejszej dokumentacji),
- biblioteka SignalR.

5 Projekt bazy danych

Poniższy rozdział zawiera projekt bazy danych systemu. Będzie ona przechowywała informacje o użytkownikach, grupach laboratoryjnych, stanowiskach komputerowych oraz uprawnieniach użytkowników do sal.

Rysunek 2 przedstawia diagram relacyjny projektowanej bazy danych. Opis poszczególnych pól oraz ich znaczeń opisano w Tabelach 3 (User), 4 (Computer), 5 (Classroom), 6 (ClassroomPermission), 7 (Blacklist) oraz 8 (Abuse).



Rysunek 2: Diagram relacyjny bazy danych

Tabela 3: Opis kolumn w tabeli User

Pole	Opis
UserID	unikatowy identyfikator użytkownika
Login	krótki ciąg znaków używany przez użytkowników do logowania
Password	hasło do konta
FirstName	imię użytkownika
LastName	nazwisko użytkownika
IsAdmin	pole logiczne, którego wartość „Prawda" określa, że użytkownik jest administratorem
LastLogin	data ostatniego poprawnego logowania do serwisu
LoginAttempts	licznik błędnych prób logowania
CreatedOn	data utworzenia użytkownika
CreatorID	identyfikator użytkownika, który utworzył to konto
LastEdit	data ostatniej edycji danych tego użytkownika
EditorID	identyfikator użytkownika, który ostatnio edytował dane tego konta
IsLocked	pole logiczne określające, czy użytkownik jest zablokowany. Gdy wartością jest prawda, użytkownik nie może się zalogować do systemu.

Tabela 4: Opis kolumn w tabeli Computer

Pole	Opis
ID	identyfikator komputera
IPAdress	adres IP komputera
StationDiscriminant	krótki tekstowy wyróżnik
CreatedOn	data utworzenia tego stanowiska
LastEdit	data ostatniej modyfikacji danych stanowiska
CreatorID	identyfikator administratora, który utworzył to stanowisko
EditorID	identyfikator administratora, który ostatnio edytował dane tego stanowiska
IsConnected	oznacza, czy dany komputer jest aktualnie podłączony do serwera

Tabela 5: Opis kolumn w tabeli Classroom

Pole	Opis
ClassroomID	identyfikator sali komputerowej
Name	nazwa sali komputerowej (np. jej numer)
Location	opis miejsca, gdzie znajduje się sala (np. budynek, piętro)
CreatedOn	data utworzenia sali
LastEdit	data ostatniej modyfikacji danych sali
CreatorID	identyfikator administratora, który utworzył salę
EditorID	identyfikator administratora, który ostatnio edytował dane sali

Tabela 6: Opis kolumn w tabeli ClassroomPermission

Pole	Opis
ClassroomPermID	identyfikator uprawnienia do edycji danych sali
ClassroomID	identyfikator sali, której dotyczy to uprawnienie
UserID	identyfikator użytkownika, którego dotyczy to uprawnienie

Tabela 7: Opis kolumn w tabeli Blacklist

Pole	Opis
BlacklistID	identyfikator pozycji czarnej listy
ProcessName	nazwa procesu niedozwolonego
ClassroomID	identyfikator sali, której dotyczy ta pozycja
CreatedOn	data utworzenia
LastEdit	data ostatniej modyfikacji danych czarnej listy
CreatorID	identyfikator administratora, który utworzył pozycję czarnej listy
EditorID	identyfikator administratora, który ostatnio edytował dane pozycji

Tabela 8: Opis kolumn w tabeli Abuse

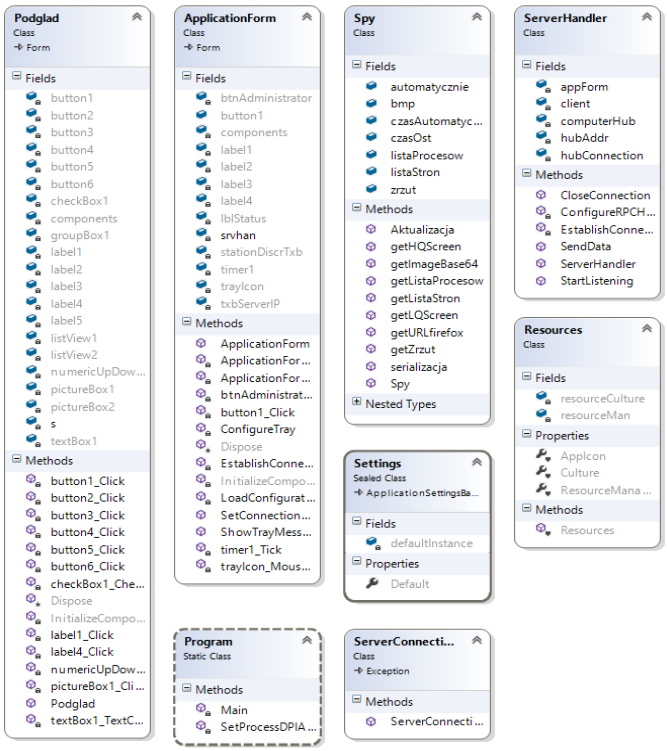
Pole	Opis
AbuseID	identyfikator naruszenia
AbuserID	identyfikator komputera, którego dotyczy to naruszenie
ScreenPath	nazwa pliku ze zrzutem ekranu wykrytego naruszenia
DetectedOn	data i godzina wykrycia naruszenia

6 Diagramy klas

W poniższym rozdziale zaprezentowane zostały diagramy klas. Ukazano uproszczoną strukturę systemu ze względu na podział na klasy. Podstawowym kryterium podziału jest program, w którym znajdują się dane klasy. W pierwszej części przedstawiono klasy użyte w aplikacji klienta, w drugiej natomiast wykorzystane w serwisie internetowym.

6.1 Aplikacja klienta

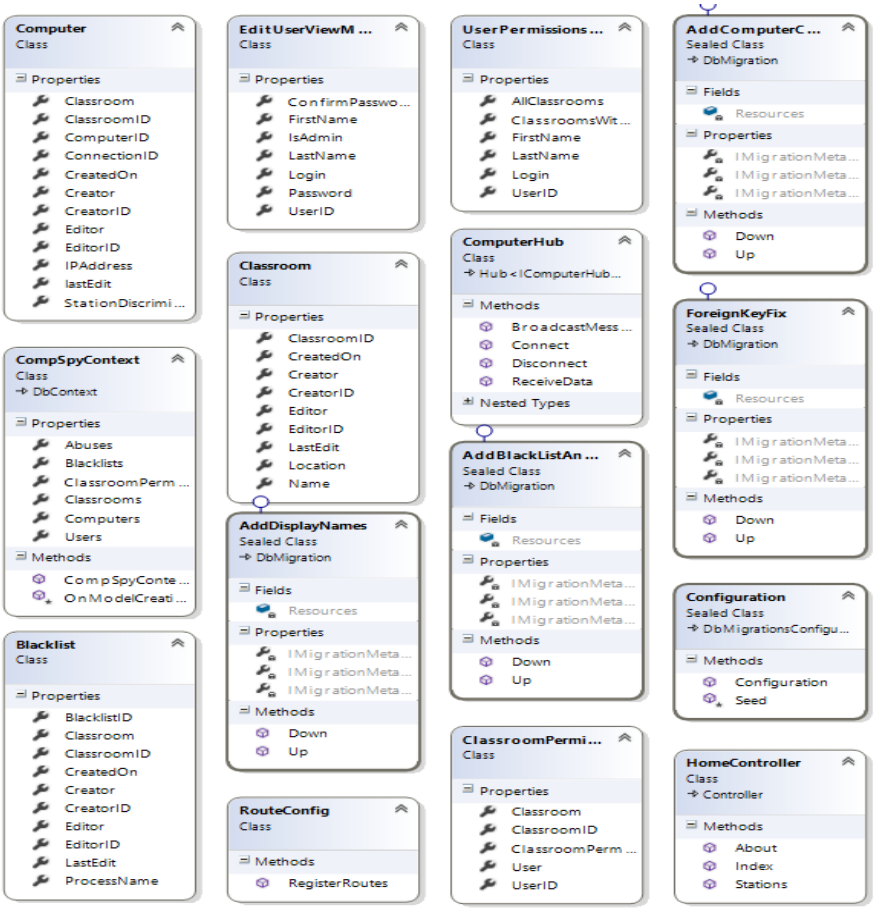
Poniżej przedstawiono diagram klas znajdujących się w aplikacji klienta. Klasy ApplicationForm i Podglad są oknami Windows Form. Klasy ServerHandler oraz ServerConnection są wykorzystywane do połączenia z serwerem za pomocą biblioteki SignalR. Klasa Spy udostepnia za to metody odpowiedzialne za tworzenie zrzutu ekranu, pobieranie listy otwartych stron oraz aktywnych procesów.



Rysunek 3: Diagram klas - aplikacja klienta

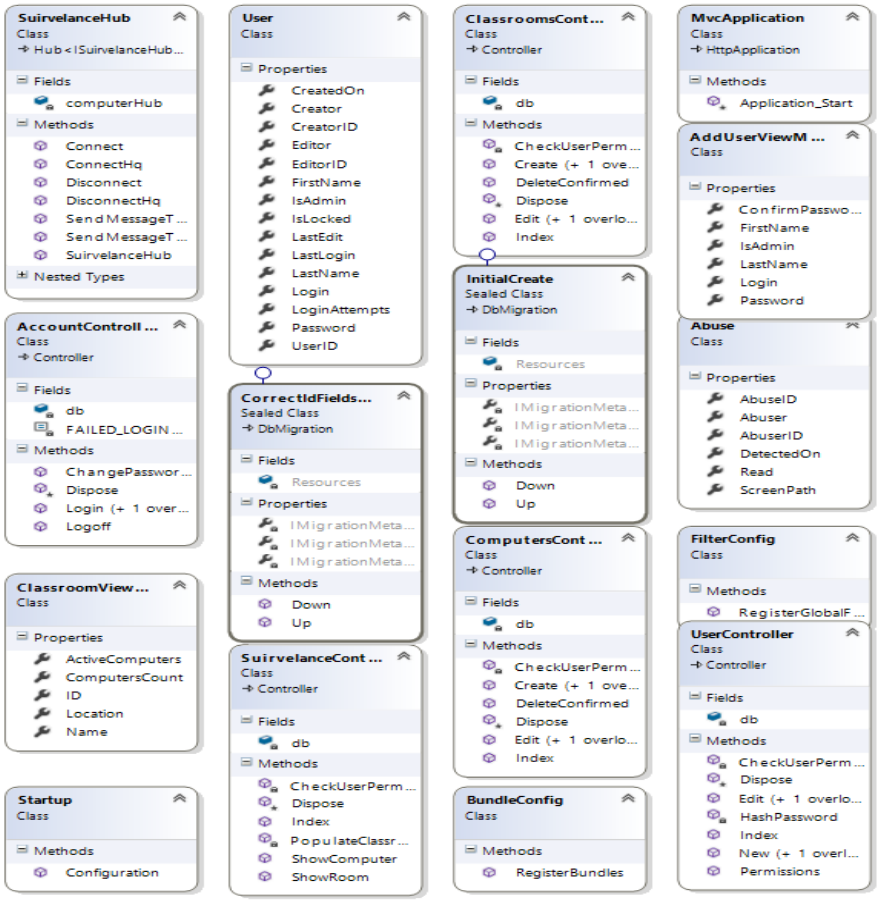
6.2 Serwer

Poniżej natomiast znajdują się diagramy klas w aplikacji serwera. Zostały one podzielone na dwie części, ze względu na ilość klas, jaka została użyta. Klasy Computer, Classroom, ClassroomPermision oraz User zostały napisane do obsługi bazy danych zgodnie z metodą CodeFirst przy wykorzystaniu Entity Framework. Oprócz tego znajdują się kontrolery odpowiedzialne za każdy widok oraz same widoki. Dodatkowo są zaimplementowane huby potrzebne do połączenia z klientami.



Rysunek 4: Diagram klas - aplikacja serwera cz. 1

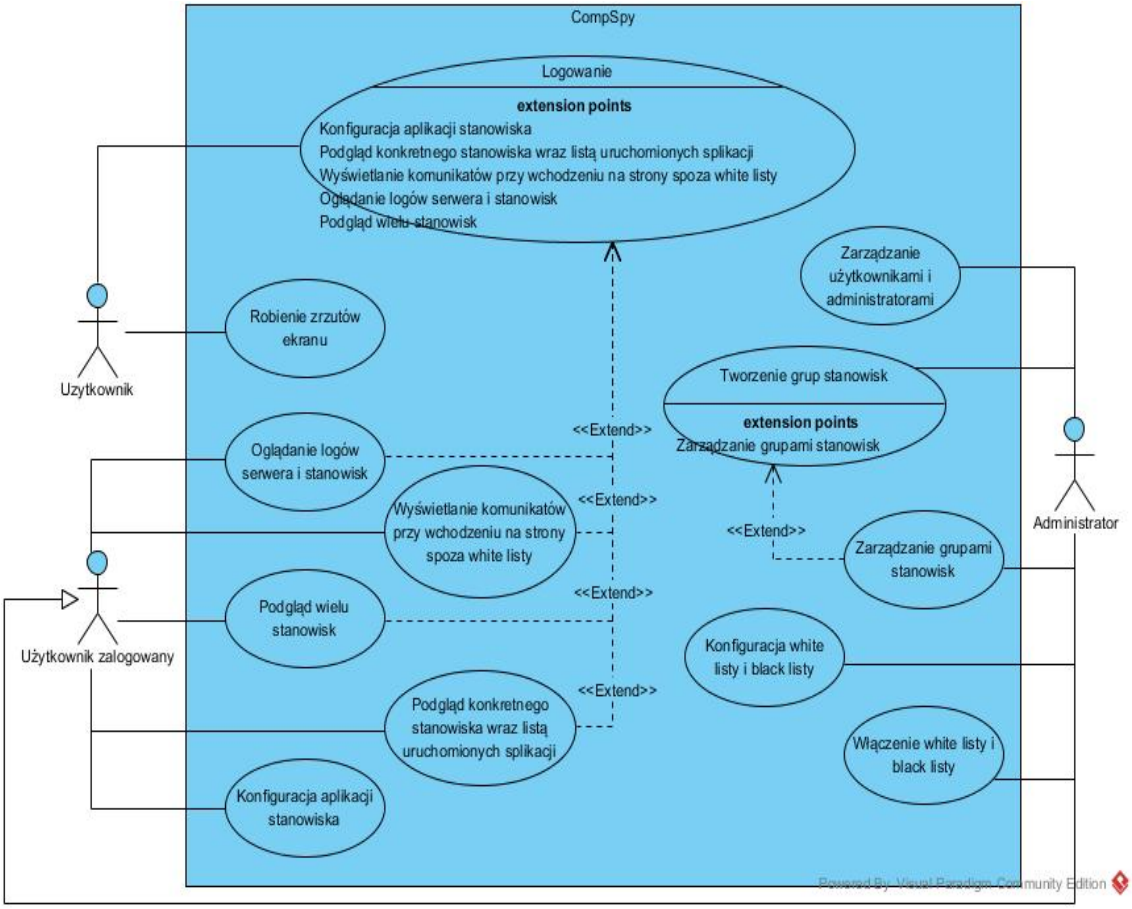
W modelach oprócz tych klas związanych z bazą danych znajdują się jeszcze klasy Abuse i Blacklist. Natomiast jeśli chodzi o widoki, to zostały podzielone na kategorie: Account, Classrooms, Computers, Home, Shared, Suirvalance oraz User.



Rysunek 5: Diagram klas - aplikacja serwera cz. 2

7 Diagram przypadków użycia

Rozdział przedstawia diagram przypadków użycia projektowanego systemu sporządzony w języku UML. Wykorzystuje się go w celu zaprezentowania, jaką rolę mogą wykonywać poszczególni aktorzy w kontakcie z opisywanym systemem oraz przejrzyście prezentuje, w jaki sposób zachowuje się dany system i jak wygląda jego interakcja z użytkownikami w konkretnych sytuacjach.



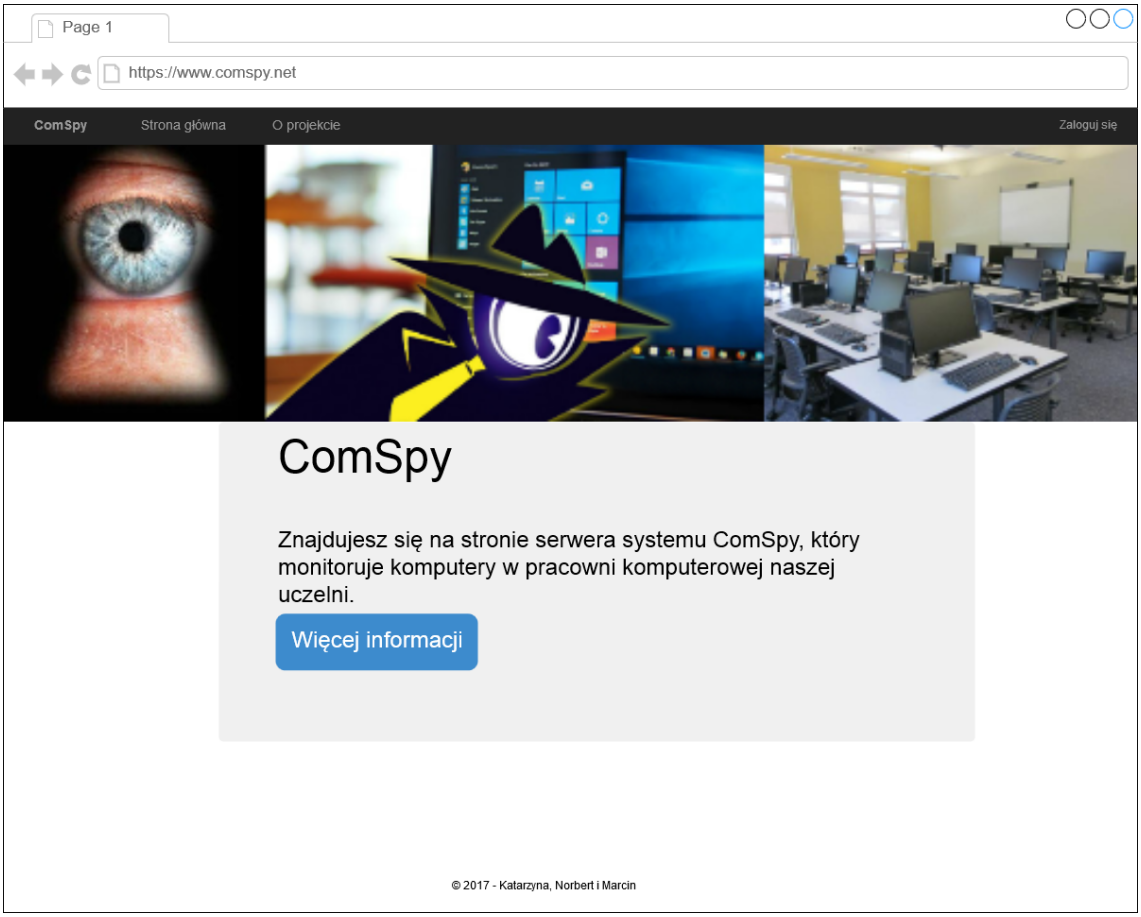
Rysunek 6: Diagram przypadków użycia

8 Szkic interfejsu użytkownika

Poniżej przedstawiony jest projekt interfejsu użytkownika. Pierwsza część przedstawia widoki w serwisie internetowym. Druga natomiast część przedstawia okno programu klienta na stanowisku klienckim.

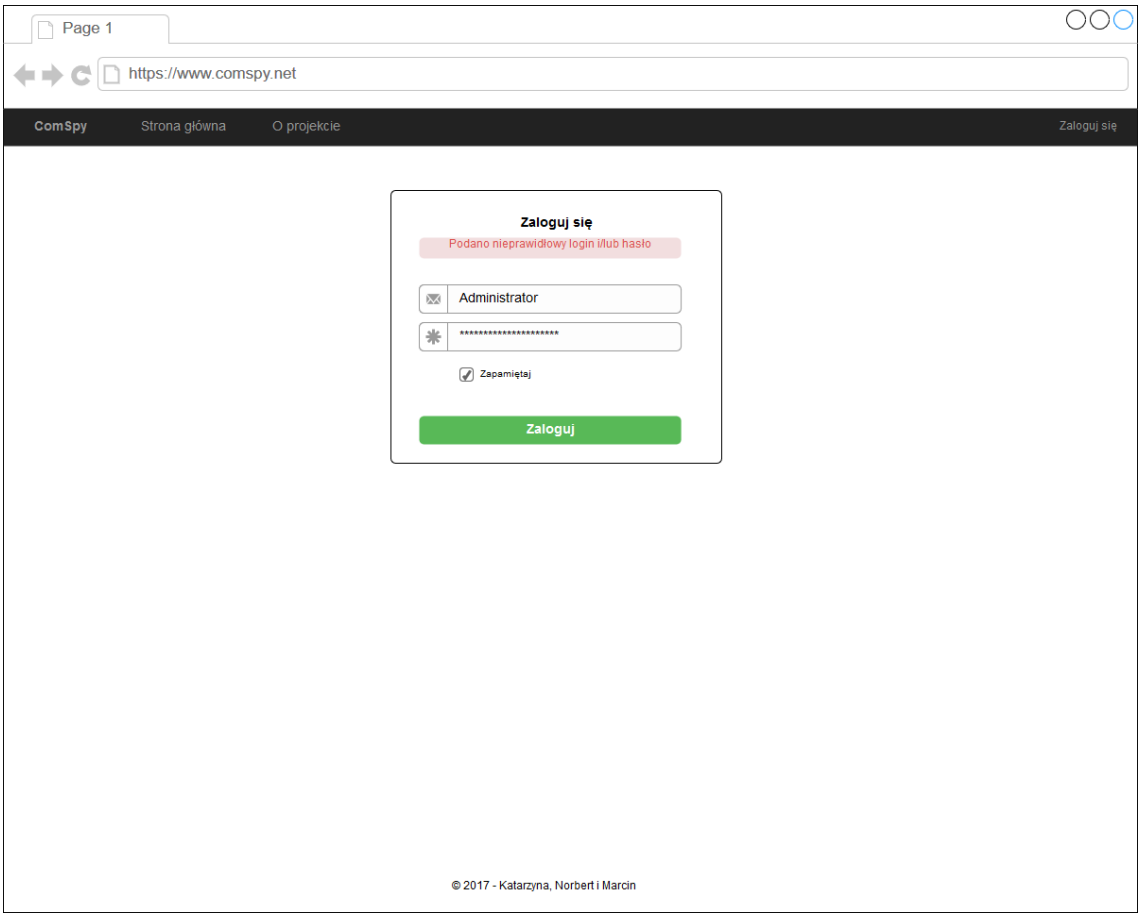
8.1 Serwis internetowy

Na poniższych rysunkach przedstawiono projekty poszczególnych widoków strony internetowej serwisu. Rysunek nr 8 przestawia projekt strony głównej, którą każdy zobaczy po wpisaniu adresu serwisu internetowego.



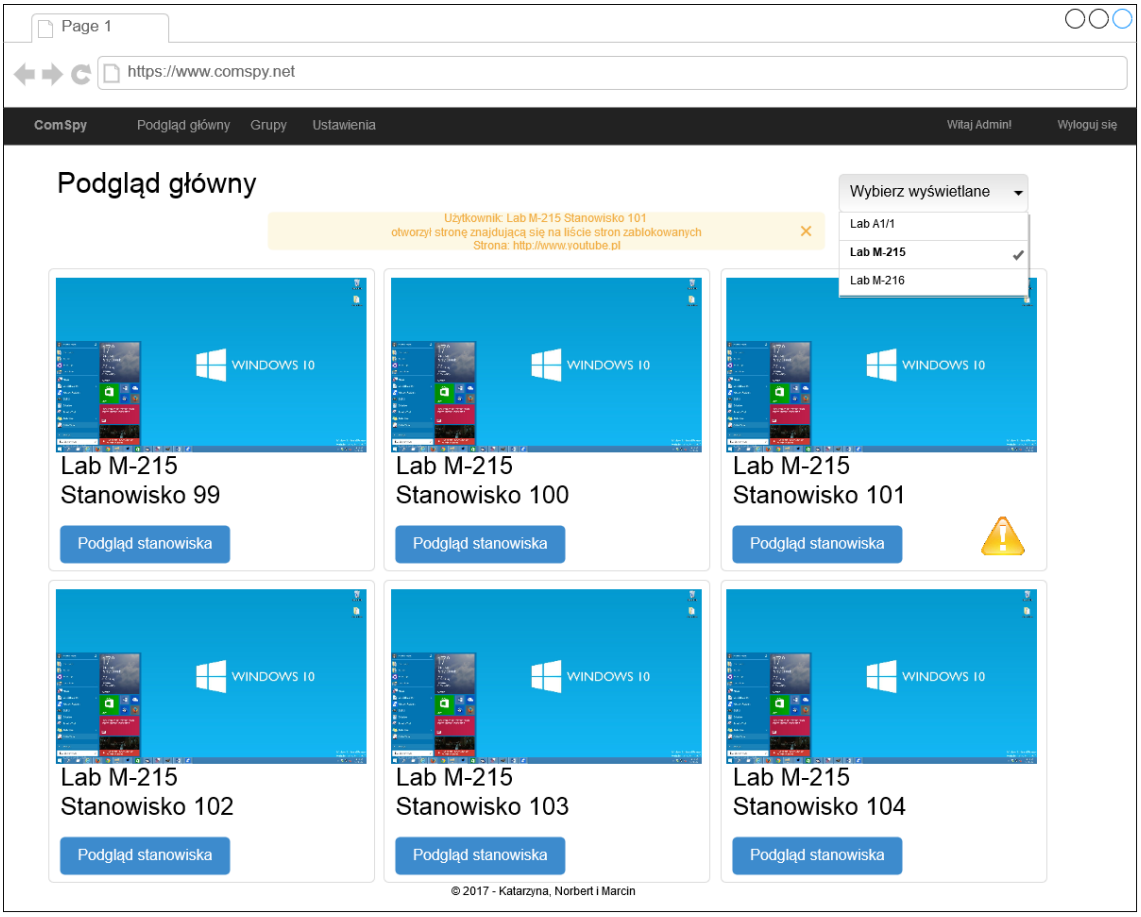
Rysunek 7: Projekt strony głównej

Rysunek nr 9 przedstawia panel logowania. W celu zalogowania się użytkownik jest proszony o podanie loginu i hasła. W przypadku wprowadzenia złych danych jest wyświetlany komunikat o błędzie i niepowodzeniu logowania.



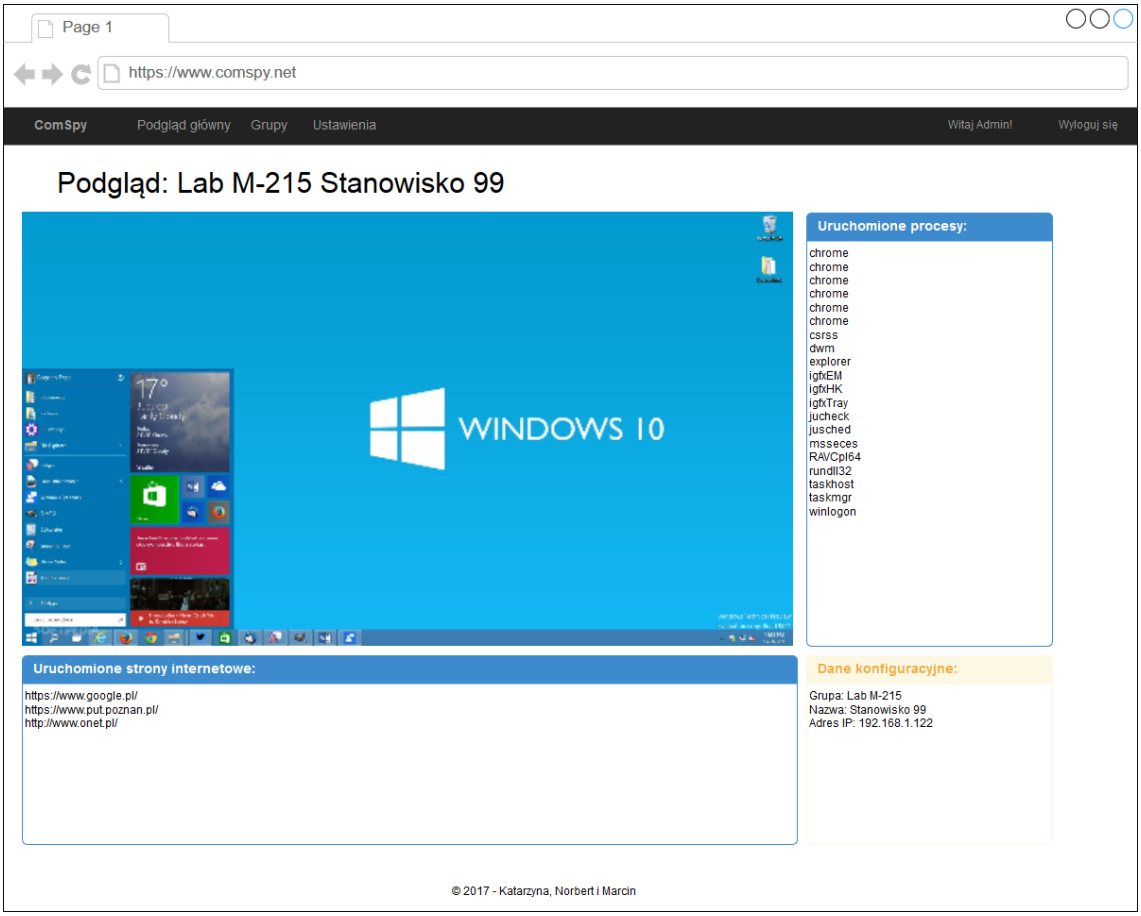
Rysunek 8: Projekt strony logowania

Rysunek nr 10 przedstawia projekt strony poglądu głównego, który jest widoczny dla użytkowników zalogowanych (oraz Administratorów). Widać tutaj podglądy wybranych przez użytkownika grup stanowisk oraz okienka z ostrzeżeniami związanymi z naruszeniem listy dozwolonych stron do przeglądania (blacklist)



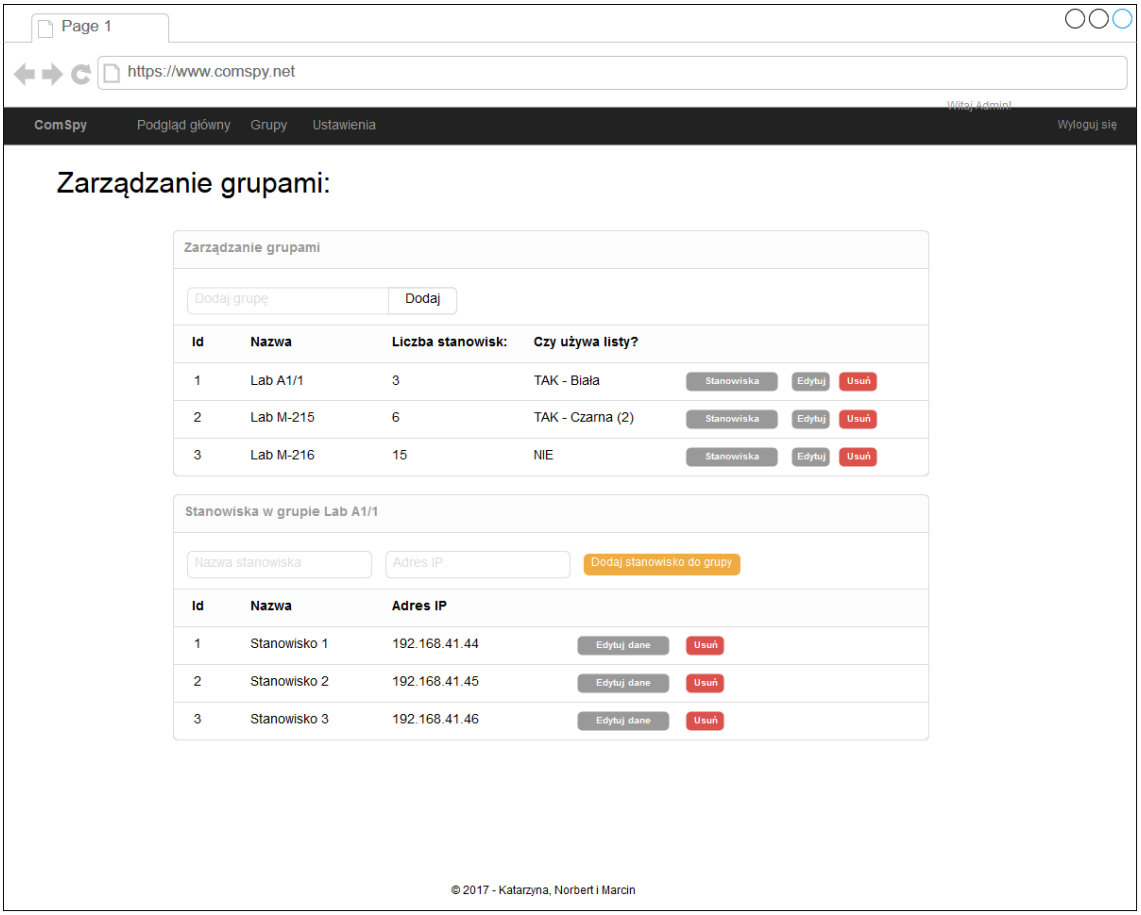
Rysunek 9: Projekt strony - podgląd główny (dostępny po zalogowaniu)

Rysunek nr 11 przedstawia projekt podglądu szczegółowego jednego stanowiska. Tutaj oprócz samego zrzutu ekranu wyświetlane są informacje o uruchomionych procesach, otwartych stronach internetowych oraz podstawowe dane konfiguracyjne.



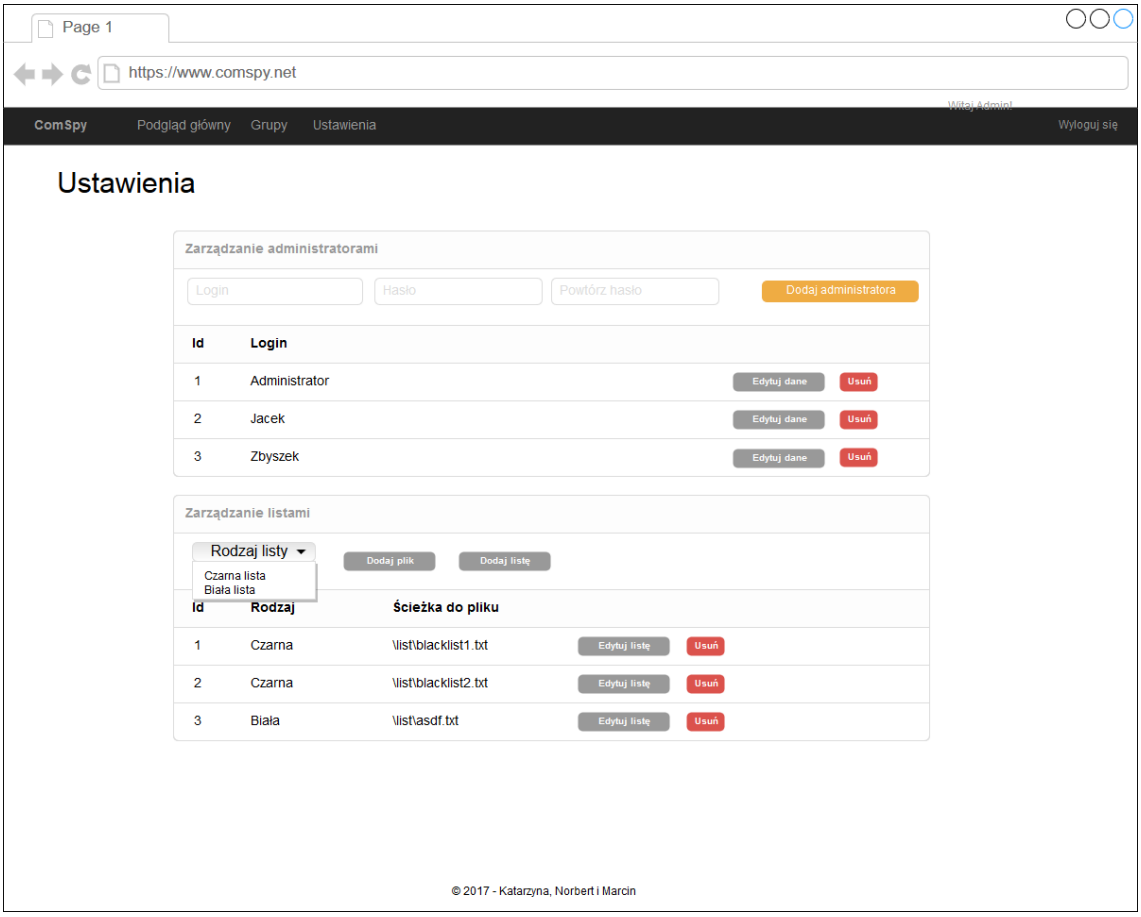
Rysunek 10: Projekt strony - podgląd stanowiska (dostępny po zalogowaniu)

Rysunek nr 12 przedstawia zarządzanie grupami oraz stanowiskami w ramach jednej z nich. Ten widok dostępny jest tylko dla administratorów.



Rysunek 11: Projekt strony - zarządzanie grupami i stanowiskami (dostępny po zalogowaniu)

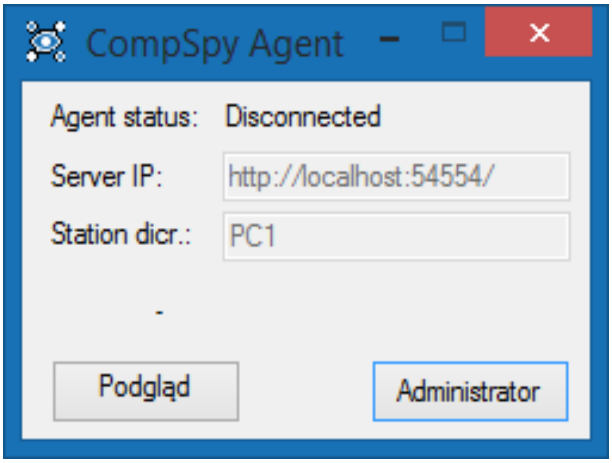
Rysunek nr 13 przestawia panele dotyczące zarządzania użytkownikami oraz blacklistą.



Rysunek 12: Projekt strony - ustawienia (dostępny po zalogowaniu)

8.2 Aplikacja kliencka

Ostatni szkic przedstawia program klienta, na którym widać niepodłączonego jeszcze klienta do serwera. Oprócz samego statusu połączenia widać adres IP serwera oraz identyfikator stanowiska. Widać także przyciski Podgląd, która przeniesie do okna, w którym można podejrzeć chwilowy stan tego, co jest następnie przesyłane do serwera. Po kliknięciu w przycisk Administrator natomiast można natomiast w przyszłości zaimplementować podstawowy panel administracyjny lub inne funkcje, które dostępne będą tylko dla administratora systemu.



Rysunek 13: Projekt strony - ustawienia (dostępny po zalogowaniu)

9 Bezpieczeństwo

Podczas tworzenia projektu postarano się, aby zapewnić w bezpieczeństwo pod wieloma aspektami. Poniżej przedstawiono najważniejsze z nich oraz sposoby ich zapewnienia.

Hasła przechowywane w bazie danych nie są przechowywane w formie jawnej, lecz w postaci heksadecymalnego skrótu powstałego przy użyciu funkcji skrótu SHA-256. Podczas wpisywania samego hasła w polu tekstowym, hasło jest zakropkowane.

W serwisie internetowym zapewniono, aby dostęp do poszczególnych funkcjonalności był możliwy tylko dla osób z odpowiednimi uprawnieniami (np. podgląd możliwy tylko dla użytkowników zalogowanych). Same widoki nie powinny być również dostępne po wpisaniu adresu w pasku adresu przeglądarki.

Innym rodzajem bezpieczeństwa jest zabezpieczenie przed zbędnym przesyłaniem pakietów i zmniejszeniu przepustowości sieci. Polega to na tym, że gdy żaden użytkownik w serwisie internetowym nie podgląda stanowisk, to stanowiska nie przesyłają w tym czasie dużych pakietów ze zrzutami ekranów, listą włączonych procesów oraz listą otwartych kart w przeglądarkach.

Kolejnym zabezpieczeniem jest rozdzielenie użytkowników na użytkownika zalogowanego oraz administratora. Dzięki temu użytkownik ma dostęp do poglądu, ale nie może usunąć ani zarządzać grupami oraz stanowiskami. Uprawnienia do ich dodawania, edycji i usuwania ma tylko konto z uprawnieniami administratora.

10 Napotkane problemy

W poniższym rozdziale opisane zostały najważniejsze problemy z jakimi spotkała się grupa projektowa w trakcie projektowania oraz już samej implementacji.

10.1 Tworzenie zrzutów ekranu oraz pobieranie listy procesów aktywnych

Pierwszym problemem jaki wystąpił, było tworzenie zrzutów ekranu oraz listy procesów na komputerze klienta. To był najłatwiejszy problem z jakim spotkała się grupa. Aby rozwiązać problem wystarczyło wykorzystać dostępne w środowisku .NET funkcje. Samo zrobienie zrzutu to tak na prawdę wykorzystanie dokładnie jednej funkcji.

```
bmp = new Bitmap(Screen.PrimaryScreen.Bounds.Width,
    Screen.PrimaryScreen.Bounds.Height,
    System.Drawing.Imaging.PixelFormat.Format32bppRgb);
    zrzut = Graphics.FromImage(bmp);
    zrzut.CopyFromScreen(0, 0, 0, 0,
        Screen.PrimaryScreen.Bounds.Size,
        CopyPixelOperation.SourceCopy);
```

Listing 1: Fragment kodu odpowiedzialny za tworzenie zrzutów ekranu

```
public Process[] listaProcesow;
listaProcesow = Process.GetProcesses();
```

Listing 2: Fragment kodu odpowiedzialny za pobranie listy aktywnych procesów

10.2 Uzyskanie adresów kart w przeglądarkach internetowych

Drugim problemem było pytanie, jak uzyskać adresy stron internetowych. Samo wyświetlenie tytułów było łatwe, ponieważ wykorzystano podobnie jak w poprzednim problemie dostępne funkcje w środowisku. NET. Samo zdobycie tytułów otwartych okien zostało pokazane na przykładzie przeglądarki Chrome na listingu nr 3.

```
foreach (var p in listaProcesow)
{
    if (Convert.ToString(p.ProcessName) == "chrome")
    {
        if (p.MainWindowTitle != "" ||
            p.MainWindowTitle == "␣")
        {
            listaStron.Add("[chrome]");
            listaStron.Add ("Title:␣" +
                p.MainWindowTitle);
        }
    }
}
```

Listing 3: Fragment kodu odpowiedzialny za pobranie tytułu otwartej strony w przeglądarce Chrome.

Powyższy sposób został użyty do pobrania tytułów z przeglądarek: Chrome, Opera, Internet Explorer oraz Microsoft Edge (działanie tej ostatniej nie zostało potwierdzone, ze względu na brak testów w systemie Windows 10). Dzięki wykorzystaniu biblioteki NDde udało się pobrać nie tylko tytuł, ale także adres strony internetowej otwartej w przeglądarce Mozilla Firefox. Ze względu na ograniczony czas pracy nad systemem i wystąpieniem tak wielu problemów, obecnie nie działa to dla innych przeglądarek.

```
using NDde.Client;

public void getURLfirefox()
{
    try
    {
        DdeClient dde = new DdeClient("firefox",
            "WWW_GetWindowInfo");
        dde.Connect();
        string url = dde.Request("URL", int.MaxValue);
        string[] urls = url.Split(new string[]
            { ",", "\"" },
            StringSplitOptions.RemoveEmptyEntries);
        dde.Disconnect();

        listaStron.Add("[firefox]");
        listaStron.Add("URL:␣" + urls[0]);
```

```
            listaStron.Add("Title: " + urls[1]);
        }
        catch {}
}
```

Listing 4: Fragment kodu odpowiedzialny za pobranie adresu URL oraz tytułu otwartej strony w przeglądarce Mozilla Firefox

10.3 Konwersja zrzutu ekranu

Następnym problemem, jaki napotkała grupa projektowa, było to to w jaki sposób przesłać zrzut ekranu. Okazało się że najlepszym sposobem będzie konwersja zdjęcia (już po zmianie formatu oraz rozmiaru w zależności od tego w jakiej jakości żąda serwer) do formatu tekstowego za pomocą konwersji do typu Base64.

```
public String getImageBase64(Bitmap img)
{
    Graphics g = Graphics.FromImage(img);

    System.IO.MemoryStream stream =
        new System.IO.MemoryStream();
    img.Save(stream,
        System.Drawing.Imaging.ImageFormat.Bmp);
    byte[] imageBytes = stream.ToArray();

    return Convert.ToBase64String(imageBytes);
}
```

Listing 5: Fragment kodu odpowiedzialny za konwersję obrazu.

10.4 Wykorzystanie SignalR

Po konwersji następuje już budowa samego komunikatu. Następnie następuje serializacja komunikatu i wysłanie go do serwera. Ale nie jest to takie łatwe, ponieważ problemem okazała się kwestia, aby nie przesyłać niepotrzebnie pakietów, gdy żaden użytkownik nie korzysta z serwisu internetowego i nie podgląda stanowisk. Rozwiązaniem okazało się wykorzystanie biblioteki SignalR. Pierwszym listingiem, który został ukazany jest sama konfiguracja uchwytu serwera, który zawiera w sobie hub do obsługi połączenia.

```
public ServerHandler
    (string hubAddr, ApplicationForm appForm)
 {
    this.hubAddr = hubAddr;
    this.appForm = appForm;

    hubConnection = new HubConnection(hubAddr);

    computerHub =
        hubConnection.CreateHubProxy("ComputerHub");

    ConfigureRPCHandlers();
}
```

Listing 6: Fragment kodu - konstruktor klasy ServerHandler

Ważnym elementem wykorzystania tej klasy jest metoda ustanawiająca połączenie. Podczas nawiazywania tego połączenia jest przesyłana informacja o identyfikatorze stanowiska oraz sekret.

```
private void EstablishConnection()
{
    var parameters = new Dictionary<string, string>
    {
        { "stationId", ConfigurationManager.AppSettings
                ["stationDiscr"]},

        { "secret", ConfigurationManager.AppSettings
                ["secret"] }
    };

    computerHub.Invoke("Connect",
        ConfigurationManager.AppSettings
        ["stationDiscr"]);
}
```

Listing 7: Fragment kodu metody nawiązującej połączenie

Poniżej przedstawiono sposób wysyłania komunikatu dla żądania rozpoczęcia transmisji niskiej jakości. Fragment pochodzi z metody odpowiedzialnej za konfigurację uchwytu RPC.

```
computerHub.On("StartLowQualityTransmission", () =>
{
    Spy spy = new Spy();
    spy.Aktualizacja();
    var data = spy.serializacja(false);
    computerHub.Invoke("ReceiveData", data);
});
```

Listing 8: Fragment kodu przedstawiający wysyłanie komunikatu dla rozpoczęcia transmisji niskiej jakości.

11 Komunikacja

W poniższym rozdziale zostanie w krótki sposób opisany sposób komunikacji pomiędzy poszczególnymi częściami systemu. Wyróżniamy dwa rodzaje komunikacji: pierwszy to komunikacja użytkownika serwisu internetowego z serwerem tego systemu, drugi to natomiast komunikacja klientów z serwerem.

Komunikacja pomiędzy serwisem internetowym i jej klientem odbywa się w standardowy sposób dla technologii ASP.NET. Postawiony system na serwerze IIS(Internet Information Services) zawiera pliki napisane z wykorzystaniem HTML, JavaScript oraz C. Komunikacja odbywa się za pośrednictwem protokołu HTTP.

Komunikacja pomiędzy klientami a serwisem internetowym wykorzystuje bibliotekę SignalR. Wykorzystanie jej jest opisane w rozdziale 12 zatytułowanym "Napotkane problemy". Poniżej natomiast przedstawiona jest budowa komunikatu, który jest wysyłany od klienta do serwera.

```
[DataContract]
public class Komunikat
{
    [DataMember]
    public String image { get; set; }
    [DataMember]
    public List<String> listaProcesow { get; set; }
    [DataMember]
    public List<String> listaStron { get; set; }

    public Komunikat(String img, Process[] procesy,
        List<String> strony) {
        image = img;
        listaProcesow = new List<String>();
        listaStron = new List<String>();
        foreach (var p in procesy)
            listaProcesow.Add(p.ProcessName);
        foreach (var s in strony)
            listaStron.Add(s);
    }
}
```

Listing 9: Fragment kodu przedstawiający budowę klasy Komunikat

Podczas tworzenia powyższego komunikatu wprowadzane są dane aktualne z listy procesów, listy stron oraz tworzony jest zrzut ekranu. Po tym procesie komunikat jest serializowany. Format samego komunikatu, który jest już przesyłany do serwera jest w formacie XML. W kolejnym listingu przedstawiony jest przykład takiego pliku.

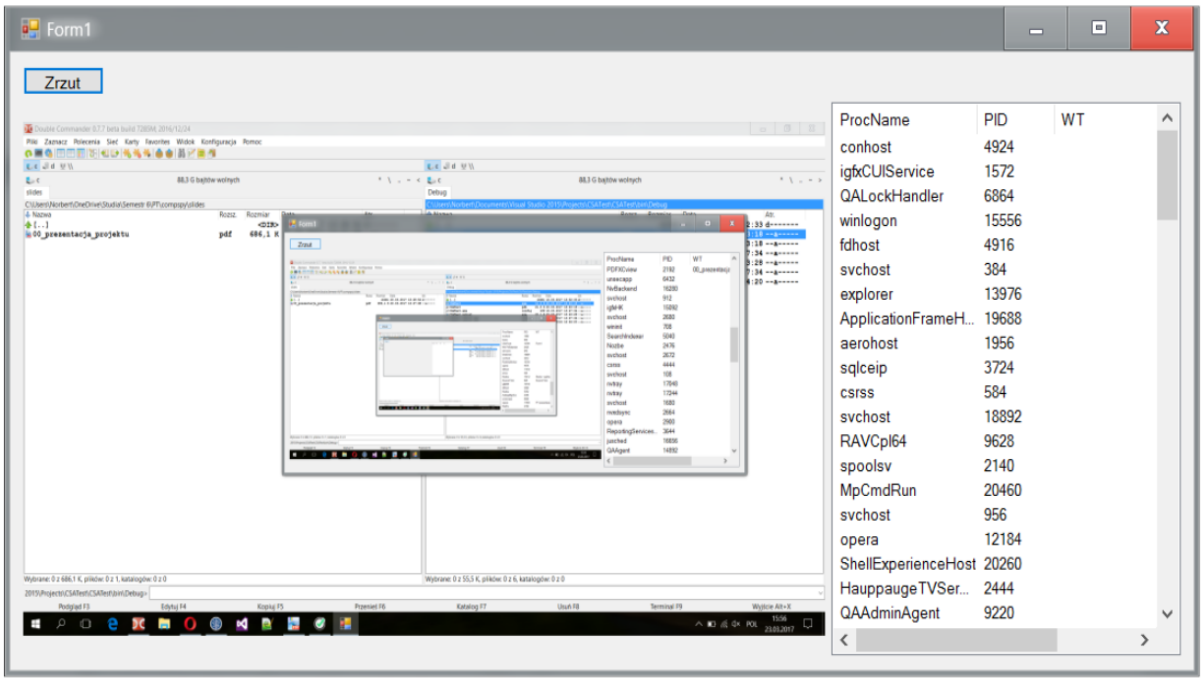
```
<Spy.Komunikat
    xmlns="http://schemas.datacontract.org/2004/07/
        CompSpyAgent"
   xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  <image>
    ...
    zrzut ekranu jako base64
    ...
  </image>
  <listaProcesow xmlns:a="http://schemas.microsoft.com/2003/
      10/Serialization/Arrays">
    <a:string>svchost</a:string>
    <a:string>notepad++</a:string>
    <a:string>isesrv</a:string>
    <a:string>services</a:string>
    <a:string>winlogon</a:string>
    <a:string>chrome</a:string>
    <a:string>conhost</a:string>
    <a:string>explorer</a:string>
  </listaProcesow>
  <listaStron xmlns:a="http://schemas.microsoft.com/
      2003/10/Serialization/Arrays">
    <a:string>[firefox]</a:string>
    <a:string>URL: https://www.google.pl</a:string>
    <a:string>Title: Google</a:string>
    <a:string>[chrome]</a:string>
    <a:string>Title: Onet.pl - Google Chrome</a:string>
  </listaStron>
</Spy.Komunikat>
```

Listing 10: Przykładowy komunikat przesyłany do serwera w formacie XML

12 Testy

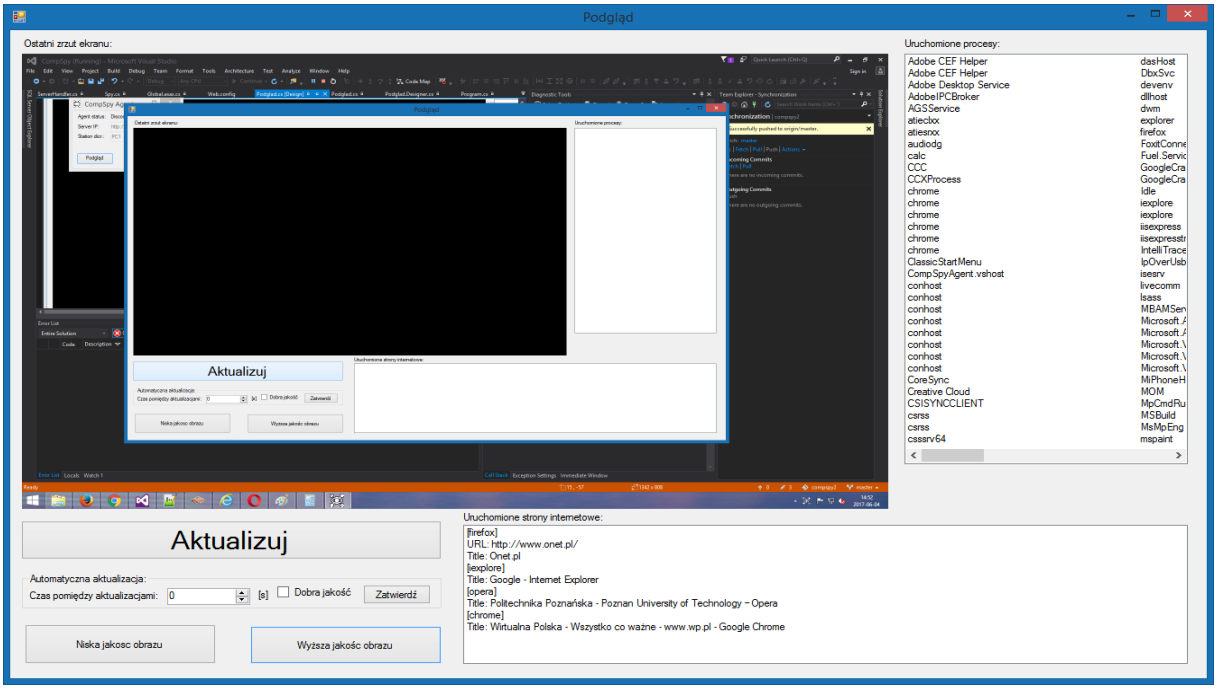
W poniższym rozdziale przedstawione zostaną testy systemu. Zostały one wykonane na różnych etapach prac. Część widoków została utworzona specjalnie do tego, aby przetestować różne funkcjonalności systemu, a nie znajdą się w finalnej wersji produktu.

Do pierwszych testów tworzenia zrzutów ekranu oraz pobierania listy aktywnych procesów napisano aplikację Windows Form, która po kliknięciu na przycisk "Zrzut" wyświetlany jest zrzut ekranu oraz lista procesów.



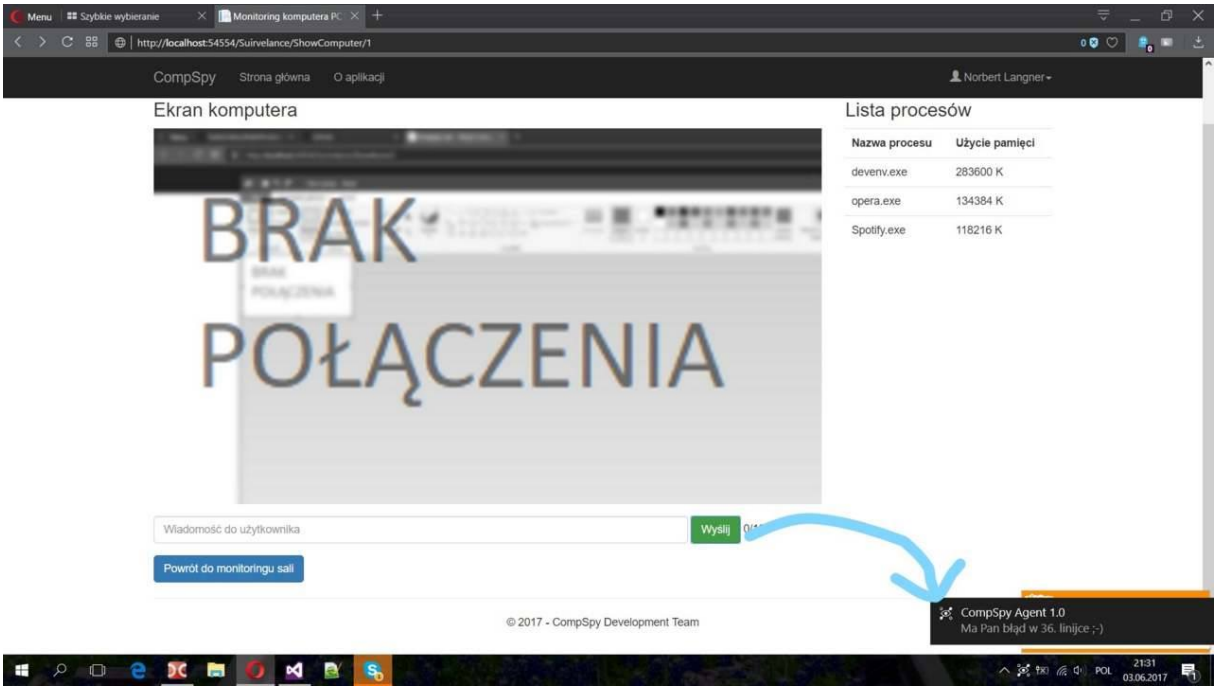
Rysunek 14: Testowanie funkcjonalności tworzenia zrzutów ekranu oraz pobierania listy aktywnych procesów.

Gdy już udało się wybrać odpowiednie rozwiązanie postanowiono zbudować okno do podglądu w aplikacji klienta, aby można było sprawdzić działanie jeszcze przed połączeniem się do serwera i wysłaniem komunikatu. W tej wersji (w porównaniu do poprzedniego programu) działa również podgląd adresów i tytułów otwartych kart w przeglądarkach, a także zmiana jakości obrazu (niska i wysoka jakość). We wcześniejszej wersji korzystano także z funkcjonalności zegara do aktualizacji co określony czas.



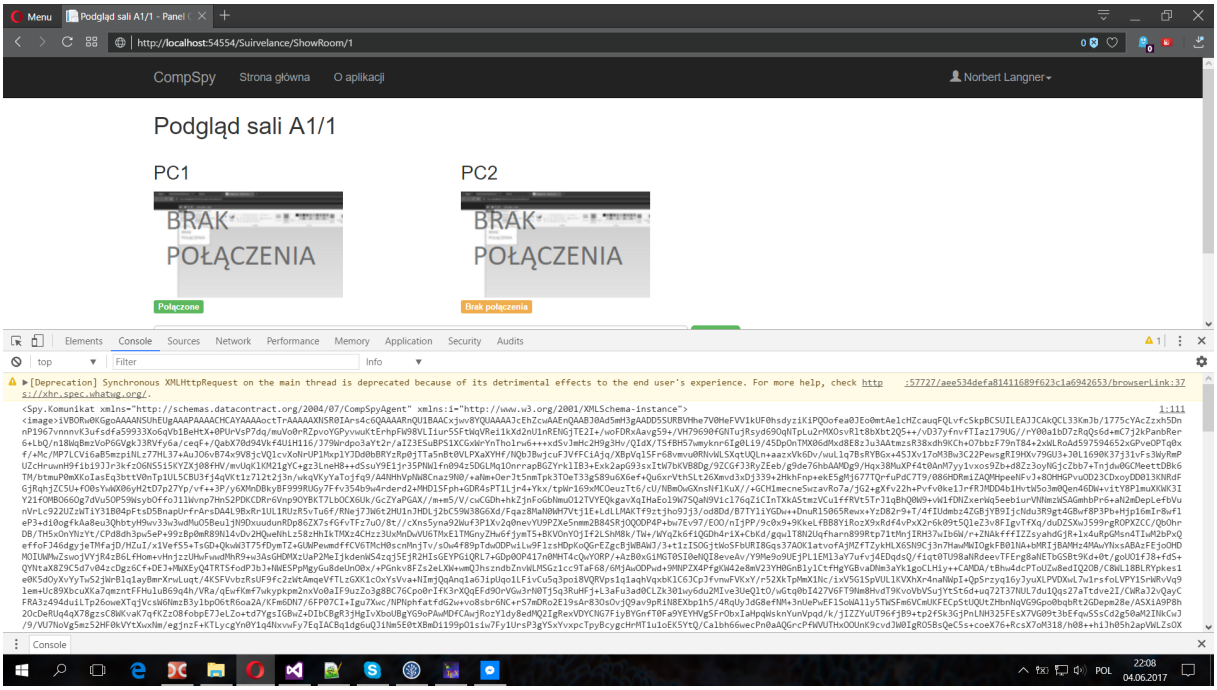
Rysunek 15: Widok Podgląd w aplikacji klienta

Gdy udało się połączyć aplikację klienta i serwer z pomocą biblioteki SignalR zaczęto także testować uruchamianie z poziomu strony metod wywoływanych u klienta. Rysunek nr 17 przedstawia możliwe przykładowe wykorzystanie takiego połączenia.



Rysunek 16: Testowanie funkcjonalności SignalR za pomocą wysyłania wiadomości do aplikacji klienta.

Ostatni przedstawiony test miał już miejsce po wysłaniu poprawnego komunikatu przez klienta do serwera. Za pomocą narzędzi deweloperskich dostępnych w przeglądarce Opera sprawdzono otrzymany komunikat.



Rysunek 17: Podgląd komunikatu otrzymanego od klienta

13 Podział prac

W poniższej tabeli nr 5 przedstawiono podział prac pomiędzy osoby należące do grupy projektowej.

Tabela 9: Podział prac

Część projektu	Norbert	Kasia	Marcin
Tworzenie dokumentacji	X	X	X
Utworzenie projektu bazy danych	X	X	X
Utworzenie diagramu przypadków użycia		X	
Utworzenie diagramów klas			X
Rozwiązanie problemu tworzenia zrzutów ekranu	X		
Rozwiązanie problemu tworzenia listy aktywnych procesów		X	
Rozwiązanie problemu pobierania adresów otwartych stron internetowych			X
Implementacja programu klienta	X		X
Implementacja serwisu internetowego i serwera	X	X	

14 Możliwości rozwoju

Poniższy rozdział przedstawia możliwe kierunki rozwoju dla systemu. Podzielono te kierunki na dwie kategorie: rozszerzające lub dodające nowe funkcjonalności oraz zwiększające przyjazność obsługi dla użytkownika. Dodatkowo można wspomnieć o możliwości optymalizacji wykorzystania zasobów komputera oraz wykorzystania łącza sieciowego.

14.1 Funkcjonalności

W tym podrozdziale skupiono się na możliwościach rozwoju działających już funkcjonalności oraz takim, które można dodać do systemu.

Pierwszą rzeczą, jaką można utworzyć jest to utworzenie aplikacji klienckiej dla systemu operacyjnego Linux. Dzięki takiemu rozwiązaniu, system będzie wspierał sale, w których korzysta się z innego systemu operacyjnego niż Windows.

Drugim rozwiązaniem może być połączenie systemu z innymi, w celu np. identyfikacji studenta/ucznia i powiązaniem go ze zdarzeniami związanymi w nim. W takim przypadku można np. wykorzystać do sprawdzenia w późniejszym terminie historii ostrzeżeń studenta bez pamiętania jego stanowiska, przy którym siedział.

Kolejnym rozszerzeniem mogłoby być dodanie większej ilości informacji o procesach i wykorzystaniu komputera przez użytkownika. Może to być informacja u procencie wykorzystania procesora, ilości wykorzystywanej pamięci oraz łącza internetowego. To ostatnie może być także użyte do prowadzenia statystyk związanych z wykorzystywaniem systemu CompSpy w salach laboratoryjnych.

Dosyć ważnym rozszerzeniem byłoby utworzenie rozszerzeń do przeglądarek internetowych wykorzystywanych w salach laboratoryjnych, aby była możliwość pobrania wszystkich otwartych kart w przeglądarkach, a nie tylko jednej aktualnie otwartej.

14.2 Przyjazność dla użytkownika

W tej sekcji postanowiono opisać możliwości rozwoju związane z samym interfejsem użytkownika i funkcjonalnościami, które maj poprawić wygodę korzystania z systemu.

Jedną z możliwości polepszenia interfejsu użytkownika jest stworzenie aplikacji okienkowej, która będzie służyć tylko do podglądu stanowisk. Można by taką aplikację udostępniać w danych salach laboratoryjnych na np. jednym stanowisku komputerowym, gdzie dostęp do podglądu nie wymagałby konta w systemie ale byłby powiązany tylko z daną salą. Dzięki takiemu rozwiązaniu nauczyciele nie musieliby się logować do systemu, lecz taki program mógłby działać od początku uruchomienia systemu operacyjnego i w każdym momencie mogliby do niego przejść.

Drugim rozszerzeniem, które zostało uznane przez grupę za ciekawe jest utworzenie wielu wersji językowych systemu, szczególnie wersji angielskiej. Dzięki takiemu rozwiązaniu nauczyciele, którzy są np. z zagranicznych uczelni mogliby też korzystać.