

Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH



Instytut Telekomunikacji

Praca dyplomowa inżynierska

na kierunku Telekomunikacja
w specjalności Teleinformatyka i Zarządzanie w Telekomunikacji

Projekt i implementacja systemu do głosowania elektronicznego.

Jakub Bańka

Numer albumu 261112

promotor
prof. dr hab. inż. Zbigniew Kotulski

Warszawa 2017

Projekt i implementacja systemu do głosowania elektronicznego.

Streszczenie

Rozwój technologii na przestrzeni ostatnich lat diametralnie zmienił styl życia i sposób myślenia ludzi. Możliwość zrobienia zakupów, legalnego korzystania z dóbr naszej kultury, wykonywanie opłat staje się coraz wygodniejsze i prostsze. Jednak mimo to nie wszystkie dziedziny naszego życia wydają się to zauważać. W wielu krajach wybory wciąż przeprowadzane są na podstawie papierowych kart głosowania liczonych później ręcznie. Podobnie w wielu firmach głosowania przeprowadza się poprzez podniesienie ręki. W takich sytuacjach warto zastanowić się czy nie skorzystać z elektronicznego systemu do głosowania.

Celem tej pracy jest zaprojektowanie i implementacja prostego systemu do głosowania elektronicznego. System ten ma być wygodny w użyciu, przy czym musi on zapewniać bezpieczeństwo prowadzonego głosowania. Aby spełnić to kryterium zostaną użyte algorytmy kryptograficzne.

Proces powstawania systemu jest opisywany od fazy koncepcyjnej, przez dobór narzędzi oraz implementację. W tej części pracy opisano jakie założenia przyjął autor pracy oraz sposób działania systemu.

W ostatnich rozdziałach znajdzie się również instrukcja instalacji systemu i obsługi aplikacji klienta. Zawarto tam również opis przeprowadzonych testów. Praca kończy się podsumowaniem, w którym autor analizuje stopień wykonania założeń przyjętych we wstępie pracy, wymienia trudności, które napotkał podczas tworzenia systemu oraz formułuje wnioski.



Politechnika Warszawska

załącznik nr 10 do zarządzenia
nr 46 /2016 Rektora PW

.....
miejscowość i data

.....
imię i nazwisko studenta

.....
numer albumu

.....
kierunek studiów

OŚWIADCZENIE

Świadomy odpowiedzialności karnej za składanie fałszywych zeznań, oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

Jednocześnie oświadczam, że:

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem w sposób niedozwolony,
- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płyce kompaktowej) oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

.....
czytelny podpis studenta

Spis treści

Projekt i implementacja systemu do głosowania elektronicznego	2
Streszczenie	2
Spis treści	4
1. Wstęp	7
1.1. Wprowadzenie	7
1.2. Motywacja i cel pracy	7
1.3. Zawartość i układ pracy	8
2. Przegląd istniejących rozwiązań	9
2.1. Prêt à Voter	9
2.2. System do głosowania elektronicznego stosowany w Estonii	9
3. Koncepcja systemu	11
3.1. Wybór typu głosowania	11
3.2. Koncepcja działania systemu i implementacji	11
3.3. Bezpieczny protokół komunikacyjny	11
3.4. Używane narzędzia i technologie	12
3.4.1. Język programowania C#	12
3.4.2. Entity Framework	12
3.4.3. Microsoft SQL Server	13
3.4.4. Microsoft Visual Studio 2015	13
3.4.5. Algorytm AES	13
3.4.6. Algorytm RSA	15
4. Architektura systemu	16
4.1. Założenia ogólne	16
4.2. Aplikacja serwera (urna)	17
4.3. Aplikacja klienta (głosujący)	17
4.4. Element publikujący (komisja)	19
4.5. Baza danych	19
4.6. Sieć miksująca	20
5. Implementacja	22
5.1. Elementy wspólne systemu	22
5.1.1. Biblioteka eVote.Database	22
5.1.2. Biblioteka eVote.Messages	23
5.2. Aplikacja serwera	24

5.2.1.	Obsługa połączeń TCP	24
5.2.2.	Uwierzytelnianie użytkownika	24
5.2.3.	Dodanie nowego głosu	25
5.2.4.	Dodanie nowego głosowania	25
5.2.5.	Dodanie nowego głosującego	25
5.2.6.	Zakończenie głosowania.....	25
5.3.	Aplikacja klienta	25
5.3.1.	Widok początkowy	26
5.3.2.	Uwierzytelnianie użytkownika	26
5.3.3.	Wybór głosowania.....	27
5.3.4.	Widok głosowania i dodawanie nowego głosu	28
5.3.5.	Widok zakończonego głosowania	29
5.3.6.	Dodanie nowego głosowania	30
5.3.7.	Dodawanie nowego głosującego.....	31
5.4.	Element publikujący	32
5.4.1.	Zakończenie głosowania i przeliczenie głosów.	32
6.	Instrukcja dla użytkownika	34
6.1.	Przygotowanie do uruchomienia aplikacji	34
6.2.	Korzystanie z aplikacji.....	35
6.2.1.	Tworzenie konta lub logowanie się	35
6.2.2.	Wybór głosowania.....	35
6.2.3.	Widok głosowania i oddawanie głosu	35
6.2.4.	Sprawdzanie wyników głosowania.....	36
6.2.5.	Dodawanie głosowania	36
6.2.6.	Wylogowanie się	36
7.	Testy	37
7.1.	Poprawność działania systemu	37
7.2.	Funkcjonalność	37
7.3.	Bezpieczeństwo	37
8.	Podsumowanie.....	39
8.1.	Ocena stopnia realizacji założeń	39
8.2.	Zaistniałe problemy i sposób ich rozwiązania	39
8.3.	Wnioski i perspektywy rozwoju	40
	Bibliografia	42

Spis rysunków	43
Spis tabel	44

1. Wstęp

1.1. Wprowadzenie

Obecnie stawiamy coraz większy nacisk na wygodę. Każda z dziedzin naszego życia stara się nam je ułatwić w maksymalny sposób. Większość informacji jest dostępna od ręki i dzieli nas od nich jedno kliknięcie, aby zamówić obiad lub taksówkę nie musimy już nigdzie dzwonić, ponieważ strona internetowa lub aplikacja na naszym smartphonie nie dość, że znajdzie naszą lokalizację, to również pobierze odpowiednią opłatę i rachunek wyśle na nasz adres e-mail. Jednak nie wszystkie aspekty naszego życia są równie wygodne. Wiele spraw urzędowych lub biznesowych wciąż wymaga naszej obecności, setek podpisów, itd. Sytuację tą poprawia rosnąca popularność podpisów elektronicznych i prób wprowadzenia systemów informatycznych, np. Elektroniczna Platforma Usług Administracji Publicznej (ePUAP). Jednakże system ten nie jest zbyt popularny wśród obywateli, co można łatwo zmienić poprzez dodanie do niego możliwości oddania głosu w wyborach.

Systemy służące do przeprowadzania głosowań drogą elektroniczną stają się coraz bardziej popularne. Pierwsze systemy były używane już w latach 60 dwudziestego wieku [1]. Możemy je podzielić na następujące grupy:

- Systemy oparte o karty papierowe – najstarszy rodzaj systemu. Do jego działania potrzebne są odpowiednio przygotowane karty. Oddanie głosu może odbyć się przez zrobienie otworu w odpowiednim miejscu lub zamalowanie odpowiedniego pola.
- Systemy DRE (Direct-recording electronic voting system) – systemy, w których głos oddawany jest za pomocą specjalnego urządzenia. Urządzenie to może następnie przechowywać głosy w swojej pamięci i po zakończeniu głosowania odpowiednio je przetwarzać.
- Systemy głosowania przez Internet – systemy, w których głosowanie odbywa się przy użyciu strony internetowej lub odpowiedniej aplikacji. Nie wymagają specjalnych urządzeń czy organizowania lokalów wyborczych.

Dziś przeważają rozwiązania DRE (używane np. w Brazylii i Indiach) lub internetowe (używane w USA i Estonii [2]). W tej pracy skupiono się na projektowaniu i implementacji prostego systemu głosowania przez Internet.

1.2. Motywacja i cel pracy

Główną motywacją do powstania tej pracy była chęć rozwoju umiejętności programistycznych, które autor zdobył w toku studiów oraz chęć praktycznego wykorzystania wiedzy wówczas zdobytej. Autorowi szczególnie zależało na rozwinięciu wiedzy z dziedziny kryptografii oraz cyberbezpieczeństwa, ponieważ są to dziedziny o coraz większym znaczeniu w dzisiejszym świecie, a ich znajomość jest atutem na rynku pracy.

Celem pośrednim było zaprojektowanie systemu do głosowania, który mógłby mieć zastosowanie komercyjne. Użytkownik tego systemu wchodziłby na odpowiednią stronę internetową, gdzie logowałby się do systemu lub tworzyłby konto. Następnie mógłby on wybrać głosowanie, w którym mógłby zagłosować lub stworzyć nowe głosowanie. Podczas oddawania głosu wybierałby opcje zgodnie ze swoimi preferencjami. W czasie trwania głosowania można oddać wiele głosów, jednak tylko ostatni oddany w czasie głosowania będzie liczony. Podczas tworzenia głosowania użytkownik mógłby wpisać opcje do głosowania oraz zaprosić innych użytkowników systemu do głosowania.

Efektem końcowym miał być system, który umożliwiłby wszystkie te operacje w bezpieczny sposób. System powinien składać się z kilku niezależnych modułów, które autor musiał zdefiniować. Po zdefiniowaniu odpowiednich modułów, autor musiał zdecydować się, jakie algorytmy kryptograficzne chce wykorzystać w poszczególnych modułach. Musiały one zapewnić dobry stosunek czasu łamania danego algorytmu do jakości czasu odszyfrowywania. Autor zdecydował się również na wykorzystanie protokołu TCP do połączenia modułów.

1.3. Zawartość i układ pracy

Niniejsza praca to opis tworzenia systemu do głosowania. Przedstawia ona wszystkie fazy tego procesu od koncepcji, przez krótki przegląd istniejących rozwiązań, określenie architektury systemu po implementację. Treść podzielono na siedem rozdziałów.

Rozdział pierwszy zawiera krótki wstęp do tematyki tej pracy oraz motywację i cele jej powstania.

Rozdział drugi zawiera krótki opis założeń jakie przyjął autor oraz przegląd istniejących już rozwiązań. Zawarto w nim również opisy wykorzystywanych narzędzi i technologii.

Rozdział trzeci wyjaśnia jak ma działać system opisywany w tej pracy. Skupia się na pokazaniu jaka jest architektura systemu i jak wygląda baza danych do niego dołączona.

Rozdział czwarty skupia się na implementacji modułów systemu.

Rozdział piąty to instrukcja, która ma na celu pokazać użytkownikowi jak używać klienta webowego to tworzenia nowych głosowań i oddawania w nim głosów.

Rozdział szósty opisuje testy jakie zostały wykonane w celu sprawdzenia działania aplikacji.

Rozdział siódmy składa się z podsumowania pracy i wniosków, do których doszedł autor pracy. Znajdują się tam m. in. ocena stopnia realizacji założeń pracy, możliwości rozwoju i wnioski końcowe.

2. Przegląd istniejących rozwiązań

2.1. Prêt à Voter

Informacje na temat tego systemu pochodzą z publikacji [3]. System Prêt à Voter to system, którego głównym założeniem jest zapewnienie wysokiego poziomu transparentności procesu przy zachowaniu jego tajności. Głosy oddane w tym systemie pozostają tajne, nawet jeżeli istnieje prawdopodobieństwo, że manipulacji całym głosowaniem. Sam system ma kilka wariacji służących do przeprowadzania różnych rodzajów głosowań.

Karta do głosowania (Rysunek 1) w tym systemie składa się z dwóch kolumn. Kartę można łatwo rozdzielić na połowę lewą i prawą. W lewej połowie mamy wiersze z opcjami głosowania (np. imionami kandydatów). Opcje te są w losowej kolejności, różnej dla każdej karty. Prawa strona zawiera pola, w których zaznaczamy jak głosujemy. W przypadku możliwości wyboru tylko jednej opcji wystarczy odpowiednio oznaczyć wiersz np. symbolem X, a w przypadku listy preferencji możemy wpisać numery odpowiadające naszemu wyborowi. Znajduje się tam również generowana kryptograficznie informacja jaka permutacja opcji jest użyta w danej karcie. Informacja ta zwana jest również *onion*.

Czesław		Czesław	
Alicja		Alicja	X
Donald		Donald	
Bartek		Bartek	
	S219DK89SA		S219DK89SA

Rysunek 1 Przykład karty do głosowania w systemie Prêt à Voter przed i po głosowaniu.

Lewa połowa karty zostaje zniszczona po oddaniu głosu, natomiast prawa zostaje zeskanowana lub skopiowana. Dzięki zniszczeniu lewej połowy karty do głosowania przed skanowaniem możemy być pewni, że jedyną osobą znającą kolejność kandydatów na swojej karcie jest sam głosujący. Prawa strona karty jest potwierdzeniem oddanego głosu i może służyć do późniejszej weryfikacji naszego głosu.

Po zeskanowaniu prawej połowy karty głos jest podpisywany elektronicznie i zapisywany w bazie danych. Jednocześnie jest on publikowany, tak aby dowolna osoba z potwierdzeniem mogła sprawdzić, czy nie został zmieniony.

Następnie wszystkie oddane głosy zostają poddane permutacji do oryginalnej wersji, na podstawie danych z *oniona*. Po permutacji danych głosy są zliczane i możemy opublikować wynik głosowania.

Niestety system ten wymaga, aby głos został oddany w zaufanej przestrzeni, przez co nie możemy go zaimplementować w tej pracy.

2.2. System do głosowania elektronicznego stosowany w Estonii

Informacje na temat systemu głosowania w Estonii pochodzą z oficjalnej strony internetowej poświęconej temu systemowi [2]. Estonia jako pierwszy kraj na świecie dał możliwość głosowania obywatelom przez Internet w roku 2005. Władze tego kraju postanowiły oprzeć swój system na używanym dotychczas systemie głosowania korespondencyjnego i wykorzystać fakt, że tamtejsze dowody osobiste mają wbudowany chip elektroniczny. System ten polegał na wysłaniu swojego głosu za pomocą tradycyjnej poczty. Jednak żeby zachować tajność głosu, każda osoba, która zarejestrowała chęć głosowania w ten sposób otrzymywała

2 koperty i kartę do głosowania. Wypełnioną kartę należało wrzucić do czystej koperty bez naszych danych osobowych. Dopiero tę kopertę należało włożyć do kolejnej (ta koperta miała już dane osobowe) i wysłać do swojego okręgu wyborczego. Po otrzymaniu takiej koperty przedstawiciel komisji niszczył kopertę z danymi osobowymi, a czystą kopertę wrzucał do urny. System ten nie pozwalał na powtórzenie głosu lub zagłosowanie w sposób tradycyjny.

System elektroniczny działa analogicznie do systemu korespondencyjnego. Użytkownik potrzebuje aplikacji, którą może pobrać z Internetu. Aplikacja weryfikuje tożsamość użytkownika, korzystając np. z certyfikatów wgranych na jego dowódzie osobistym. Użytkownik następnie oddaje głos na wybraną opcję. Następnie głos ten jest podpisywany elektronicznie i szyfrowany. Podpis elektroniczny możemy traktować jako analogię zewnętrznej koperty w systemie korespondencyjnym, kiedy szyfrowanie to ekwiwalent koperty wewnętrznej. Użytkownik może zagłosować dowolną ilość razy, ale pod uwagę bierze się tylko ostatni głos.

Zaszyfrowany głos jest następnie wysyłany na centralny serwer. Użytkownik w każdej chwili może sprawdzić w aplikacji mobilnej czy jego głos został dodany do bazy zgodnie z jego wyborem. [4]

Zaszyfrowane głosy mogą zostać odblokowane tylko przy pomocy tajnego klucza. Klucz ten został podzielony na części i rozdany członkom Państwowego Komitetu Wyborczego. W celu otworzenia głosów każdy członek Komitetu musi użyć swojej części klucza.

Następnie ze wszystkich głosów wybiera się tylko te, które dany wyborca dostarczył jako ostatnie. Sprawdza się również, czy dany wyborca nie zagłosował w sposób tradycyjny. Jeżeli tak się stało, głosem wiążącym jest głos oddany metodą tradycyjną. Po usunięciu wszystkich głosów niekwalifikujących się pozostałe są liczone.

System ten cieszy się dużą popularnością w Estonii. Dzięki łatwym w obsłudze aplikacjom, oraz faktowi, że jedyny dodatkowy koszt to czytnik kart z chipem elektronicznym w ostatnich wyborach około 30% głosów zostało oddanych przez ten system.

3. Koncepcja systemu

3.1. Wybór typu głosowania

Pierwszą rzeczą, o której należało zdecydować, był rodzaj głosowania obsługiwany przez projektowany system. Głosowania możemy podzielić ze względu na tajność na jawne lub poufne, ze względu na liczbę możliwych głosowań na te z jedną szansą głosowania lub głosowanie typu ostatni głos się liczy i ze względu na liczbę możliwych wyborów na głosowanie tylko na jedną lub na grupę opcji.



Rysunek 2 Rodzaje głosowań z zaznaczonym wyborem autora

Autor stwierdził, że w implementowanym systemie głosowania będą tajne, z możliwością oddania kilku głosów i możliwością wyrażenia swojej preferencji co oznacza wybór kilku opcji.

3.2. Koncepcja działania systemu i implementacji

Po zapoznaniu się z literaturą i przykładami istniejących już systemów, autor podjął decyzję, iż system projektowany w tej pracy będzie działać w oparciu o architekturę klient – serwer z dodatkiem bazy danych.

Zadaniem klienta będzie zbieranie informacji od użytkownika, np. tego na jaką opcję chciałby zagłosować lub jakie głosowanie chciałby dodać.

Zadaniem serwera będzie przetwarzanie zapytań klienta. Na podstawie tych zapytań serwer będzie odpytywać bazę danych o odpowiednie dane i przekazywał je klientowi lub będzie dodawać dane do bazy.

Autor stwierdził również, że system ten powstanie przy użyciu języka programowania C# oraz bazy danych MS SQL.

3.3. Bezpieczny protokół komunikacyjny

W celu zapewnienia bezpiecznej komunikacji pomiędzy elementami systemu, autor postanowił zaszyfrować przesyłane wiadomości. Postanowiono, że najlepiej będzie je zaszyfrować za pomocą algorytmu symetrycznego. Ponieważ algorytmy symetryczne

wymagają bezpiecznego kanału komunikacji do przesłania tajnego klucza, autor stwierdził również, że trzeba będzie użyć algorytmu asymetrycznego, aby go zapewnić. W tym projekcie użyte zostały symetryczny algorytm AES i asymetryczny algorytm RSA.

Podczas inicjalizacji każdego z elementów generowana jest para kluczy RSA. Serwer dodatkowo generuje klucz AES. Następnie każdy z elementów wysyła swój publiczny klucz RSA do serwera, który odpowiada swoim kluczem publicznym. Dzięki temu możemy zapewnić bezpieczny kanał komunikacji do przesłania tajnego klucza AES. Używając publicznych kluczy RSA serwer szyfruje klucz AES. Dla zapewnienia autentyczności klucza wiadomość ta jest dodatkowo podpisywana przez serwer. W następnym kroku serwer wysyła zaszyfowaną i podpisaną wiadomość do pozostałych elementów. Każdy z elementów odszyfrowuje wiadomość i weryfikuje jej poprawność. Jeżeli sprawdzenie podpisu wiadomości się powiedzie klucz AES jest zapisywany przez element.

3.4. Używane narzędzia i technologie

3.4.1. Język programowania C#

Aplikacje zaimplementowane w tej pracy zostały napisane przy użyciu obiektowego języka programowania C#. Język ten powstawał w latach 1998-2001 i został zaprojektowany dla firmy Microsoft. Informacje na temat tego języka pochodzą ze standardu ECMA 334. [10]

Program napisany w języku C# kompilowany jest do języka pośredniego, którym jest kod Common Intermediate Language (CIL). Dopiero kod CIL jest uruchamiany przez środowisko .Net, w związku z tym na systemie wymagana jest instalacja tego środowiska. Programy napisane w tej pracy są kompatybilne ze środowiskiem .Net w wersji 4.5 i nowszych.

Jednymi z najważniejszych zalet języka C# są:

- Duży wybór dodatkowych bibliotek;
- System automatycznego czyszczenia pamięci (tak zwany garbage collector). System ten sam zarządza pamięcią i cyklem życia obiektu, przez co deweloper nie musi martwić się o rezerwowanie i zwalanie pamięci;
- Możliwość używania słowa kluczowego *var* kiedy deweloper nie jest pewny jakiego typu powinna być dana zmienna. Pomaga to również przy późniejszej refaktoryzacji.
- Integracja z systemem zapytań LINQ, które znacznie ułatwiają pracę na kolekcjach danych;
- Wsparcie techniczne firmy Microsoft;
- Duża społeczność używająca tego języka.

Technologia użyta do implementacji aplikacji klienta to ASP.NET – framework służący do tworzenia aplikacji internetowych. Aplikacje tworzone z jego pomocą tego łączą ze sobą wiele obiektowych języków programistycznych, np. C#, VisualBasic oraz języków skryptowych, np. JavaScript. Graficzny interfejs użytkownika (GUI) może być definiowane przez strony internetowe tworzone w językach HTML i CSS lub przy użyciu technologii Windows Presentation Foundation (WPF) w języku XAML. Dzięki temu logika aplikacji jest odseparowana od interfejsu, co ułatwia pracę i zwiększa przejrzystość kodu.

Aplikacja serwera to zwykła aplikacja konsolowa systemu Windows.

3.4.2. Entity Framework

Entity Framework (EF) to dodatek do ASP.NET pozwalający na łatwą integrację aplikacji z relacyjną bazą danych. Został zaprojektowany przez firmę Microsoft, a pierwszą wersję wydano w roku 2008. Głównym zadaniem EF jest przeprowadzenie mapowania obiektowo – relacyjnego, dzięki czemu w naszym systemie możemy korzystać z obiektów opartych na

tabelach używanej przez nas bazy danych. Entity Framework pozwala również stworzyć bazę danych na podstawie klas naszych obiektów. [5] Dodatek ten jest wyposażony we własną wersję systemu zapytań LINQ To Entities. Entity Framework pozwala nie wykorzystywać języka SQL do komunikacji bazy danych w kodzie programu, co zmniejsza ryzyko ingerencji przez osoby trzecie. [11]

3.4.3. Microsoft SQL Server

Microsoft SQL Server (MSSQL) to system zarządzania relacyjnymi bazami danych zaprojektowany przez firmę Microsoft. Używa ona języka Transact-SQL, który jest jedną z wariacji podstawowego języka SQL. Oprócz standardowych usług takich jak przechowywanie tabel, widoków i procedur, posiada on również usługi raportujące i analityczne.

W tej pracy użyto wersji Microsoft SQL Server Enterprise 2014. [12]

3.4.4. Microsoft Visual Studio 2015

Visual Studio to środowisko programistyczne rozwijane przez firmę Microsoft. Umożliwia ono tworzenie wielu typów aplikacji, np. aplikacji webowych, WPF, WCF lub nawet mobilnych. Oprócz wielu typów aplikacji środowisko to wspiera wiele języków programowania w tym C#, Visual Basic i C++. Co więcej środowisko to jest w stanie otworzyć i okazać w sposób przejrzysty większość typów plików, na które programista może trafić podczas pracy, np. pliki *.xml, *.json, *.js czy *.html. Inną przydatną funkcją jest możliwość łatwego debugowania pisanej aplikacji. Do jej wad należy duże wykorzystywanie zasobów, jednak przy ogromie funkcji, które oferuje Visual Studio nie warto korzystać z innych opcji. [13]

3.4.5. Algorytm AES

Advanced Encryption Standard (w skrócie AES) jest symetrycznym szyfrem blokowym przyjęty jako standard FIPS-197 [14] w 2001 roku. Powstał on na potrzeby konkursu ogłoszonego przez amerykański Narodowy Instytut Standaryzacji i Technologii (ang. National Institute of Standards and Technology, w skrócie NIST) w 1997, kiedy to organizacja Electronic Frontier Foundation stwierdziła, że ówczesny standard (algorytm DES) nie jest już wystarczająco silny. Do konkursu przystąpił pod nazwą Rijndael, co jest zlepkiem nazwisk autorów – Joana Daemena i Vincenta Rijmena. Algorytm AES jest wersją algorytmu Rijndael.

AES to algorytm kryptografii symetrycznej – oznacza to, że używa tylko jednego klucza, który powinny znać tylko zainteresowane strony. Konsekwencją tego jest fakt, że klucz ten musi być dystrybuowany poprzez zaufane medium. AES jest również szyfrem blokowym operującym na blokach 128 bitowych z 128 bitowym, 192 bitowym lub 256 bitowym kluczem. Od długości klucza zależy również liczba wykonywanych rund – 10 dla klucza 128 bitowego, 12 dla 192 bitowego i 14 dla 256 bitowego. Bloki bitów są traktowane jak macierz.

Pierwsza runda składa się tylko z operacji dodania klucza. Każda następna runda składa się z operacji substytucji bajtów, zamiany wierszy, mieszania kolumn i dodawania podklucza rundy. W ostatniej rundzie nie wykonuje się operacji mieszania kolumn.

Operacja dodania klucza lub podklucza polega na dodaniu odpowiadających sobie bajtów bloku i podklucza danej rundy za pomocą operacji XOR.

Operacja substytucji bajtów, polega na zamianie wartości bajtu na wartość z tabeli S-BOX (przedstawiona w Tabeli 1). Nowa wartość jest ustalana na podstawie wartości 4 starszych bitów, które definiują wiersz w tabeli S-BOX, a 4 młodsze bity definiują jej kolumnę.

Tabela 1 Tabela S-BOX algorytmu AES

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Operacja zamiany wierszy polega na zmianie kolejności bajtów w wierszu. Bajty w pierwszym wierszu pozostają za swoich miejscach. Bajty w wierszu drugim przesuwamy o jedną pozycję w lewo, w trzecim o dwie, w czwartym o trzy. Skrajnie lewe bajty przechodzą na skrajnie prawą stronę tego samego wiersza.

Operacja mieszania kolumn polega na pomnożenie bloku przez stałą macierz (patrz Tabela 2).

Tabela 2 Macierz używana w kroku miksowania kolumn algorytmu AES

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

Aby stworzyć podklucz danej rundy potrzebny jest nam klucz poprzedniej rundy lub klucz tajny jeżeli jest to runda początkowa. Zaczynamy od przepisania 4 ostatnich bajtów starego podklucza. Następnie wykonujemy rotację bajtów o jeden w lewo. Przeprowadzamy operację substytucji bajtów. Następnie do najbardziej lewego bajtu za pomocą operacji XOR dodajemy dwójkę do potęgi „numer iteracji -1”. Operacja ta nazywa się RCON. Ostatnia operacja to operacja dodania odpowiadających sobie bajtów otrzymanej macierzy do odpowiadających im bajtów ze starego podklucza. Aby otrzymać pozostałe bajty naszego podklucza trzykrotnie powtarzamy te operacje z pominięciem przesuwania bajtów w wierszu przepisując kolejne bajty.

Deszyfracja polega na wykonaniu operacji odwrotnych do tych opisanych. Oznacza to, że odwracamy podstawienie bajtu, następnie przesuwamy je w prawo, a po sumowaniu z podkluczem, wykonujemy odwrotne mnożenie kolumn. [6]

3.4.6. Algorytm RSA

Algorytm RSA to algorytm szyfrowania asymetrycznego. Jego trudność opiera się na problemie faktoryzacji dużych liczb pierwszych. Algorytm ten został zaprojektowany przez Rona Rivesta, Adi Shamira i Leonarda Adelmanna w roku 1977. Jego nazwa pochodzi od pierwszych liter nazwisk autorów. Został on przyjęty jako standard i znajduje się między innymi w IEEE 1363 [9].

Jako algorytm asymetryczny RSA używa pary kluczy wyliczanych na podstawie dwóch dużych, tajnych liczb pierwszych. Klucze te mają postać pary liczb. Jeden z nich jest publikowany i dostępny do użytku dla każdego. Jest to klucz publiczny zwany również szyfrującym. Drugi klucz zwany prywatnym lub deszyfrującym, musi być znany tylko właścicielowi. W związku z tym nie ma możliwości obliczenia wartości jednego klucza na podstawie znajomości drugiego.

Algorytm RSA może zostać wykorzystany do szyfrowania danych oraz do ich podpisywania. Niestety słaba wydajność tego algorytmu sprawiła, że używa się go raczej do przesyłania zaszyfrowanych kluczy tajnych algorytmów symetrycznych zamiast szyfrować dane za jego pomocą. Obliczanie tych kluczy odbywa się za pomocą następującego algorytmu:

1. Wybieramy dwie duże liczby pierwsze p i q .
2. Obliczamy liczbę n będącą ich iloczynem: $n = p * q$.
3. Obliczamy wartość funkcji Eulera liczby n : $\phi(n) = (p-1)(q-1)$.
4. Wybieramy liczbę e taką, że $1 < e < \phi(n)$ oraz względnie pierwszą z $\phi(n)$.
5. Obliczamy liczbę d , która jest odwrotnością liczby $e \bmod(\phi(n))$ ($e * d \bmod(\phi(n)) = 1$).
6. Opublikujemy parę (n, e) jako klucz publiczny i używamy parę (n, d) jako klucz prywatny.

Aby zaszyfrować dane za pomocą algorytmu RSA nadawca potrzebuje klucza publicznego odbiorcy. Po jego zdobyciu nadawca używa go do wykonania operacji szyfrowania i przesyła taką wiadomość do odbiorcy. Odbiorca używa swój klucz prywatny do odszyfrowania szyfrowania. Operacje te odbywają się zgodnie z algorytmem:

1. Jako dane wejściowe używamy wiadomości m oraz kluczy publicznego (n, e) i prywatnego (n, d) .
2. Generujemy zaszyfrowaną wiadomość c za pomocą klucza publicznego. Służy do tego wzór $c = m^e \bmod(n)$.
3. Przekształcenie wiadomości c w wiadomość m odbywa się z użyciem klucza prywatnego. Wykorzystujemy wzór $m = c^d \bmod(n)$.

Aby użyć algorytmu RSA do podpisu elektronicznego odbiorca potrzebuje klucz publiczny nadawcy. Nadawca tworzy skrót danych za pomocą funkcji skrótu (np. SHA256). Następnie otrzymany klucz szyfruje swój kluczem prywatnym i wysyła dane wraz z otrzymanym podpisem. Odbiorca otrzymuje dane i robi ich skrót. Następnie odszyfrowuje podpis kluczem publicznym nadawcy i porównuje skrót z podpisu do skrótu, który sam wykonał. Podpisywanie wiadomości odbywa się za pomocą następującego algorytmu:

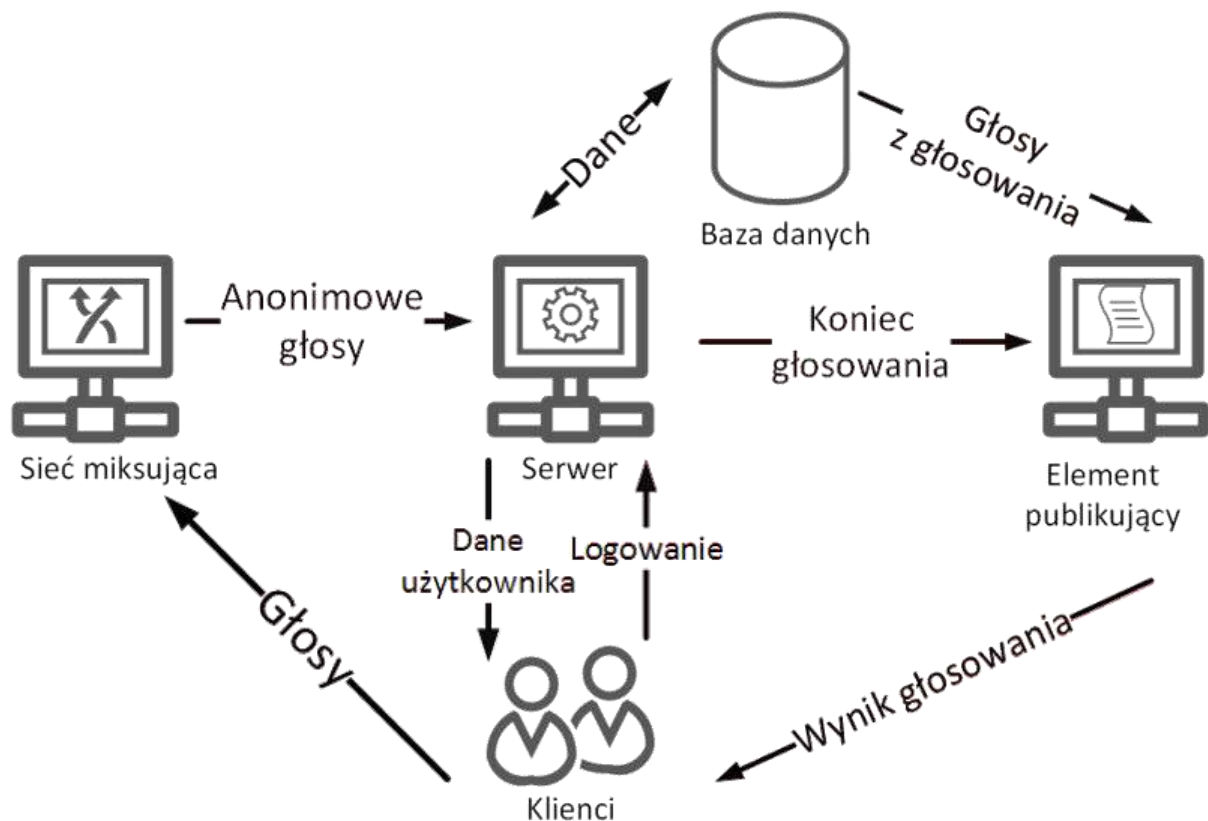
1. Jako dane wejściowe używamy wiadomości m oraz kluczy publicznego (n, e) i prywatnego (n, d) .
2. Obliczamy h wartość funkcji skrótu wiadomości m . Skrót h szyfrujemy za pomocą klucza prywatnego (odwrotnie do szyfrowania) otrzymując podpis s .
3. Weryfikacja podpisu s polega na odszyfrowaniu go za pomocą klucza publicznego do postaci h_1 . Następnie obliczymy skrót h wiadomości m i porównujemy $h_1 = h$. Jeżeli są one równe oznacza to, że podpis był prawidłowy.

4. Architektura systemu

4.1. Założenia ogólne

System tworzony podczas tej pracy inżynierskiej został podzielony na moduły. Zrobiono to w celu uproszczenia zarządzaniem systemem. Podział ten powstał w wyniku analizy zadań jakie trzeba wykonać. Na podstawie tej analizy sformułowano następujące cechy:

- System zostanie podzielony na moduły: aplikację klienta, aplikację serwera i elementu publikującego.
- Każdy moduł implementowany będzie jako grupa projektów, z czego niektóre projekty będą wspólne.
- Baza danych zostanie stworzona przez system ORM.
- Moduły korzystają z bazy danych przechowujące wszystkie dane.
- Większość operacji na bazie danych wykonuje serwer. Jednak dopuszcza się połączenie do bazy danych innych modułów.
- Moduły komunikują się między sobą za pomocą protokołu TCP.
- Wysyłane wiadomości są szyfrowane za pomocą algorytmu AES.
- Dane poufne są przechowywane w bazie danych w sposób niejawny – zaszyfrowane lub w postaci skrótu funkcji skrótu.



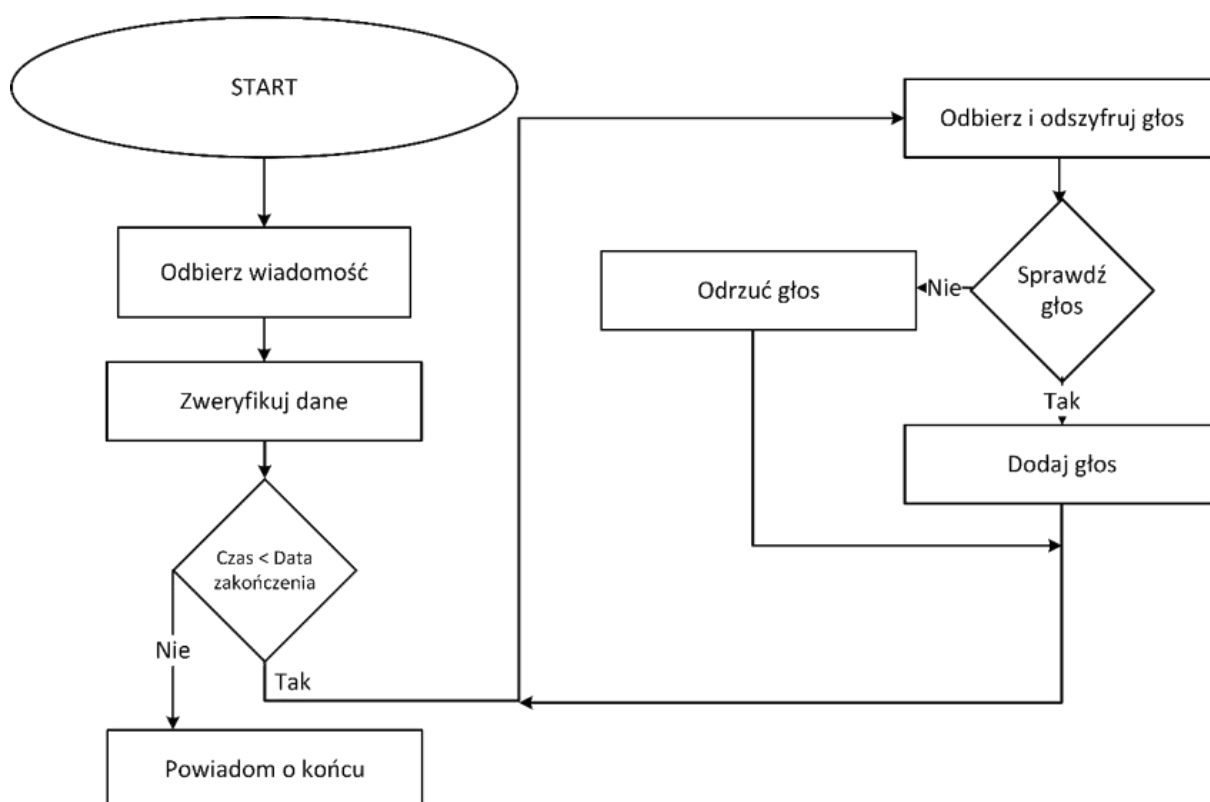
Rysunek 3 Schemat architektury systemu

Schemat systemu przedstawia rysunek 3. Aplikacja klienta loguje się do systemu poprzez odpytanie serwera czy dane logowania dostarczone przez klienta są prawidłowe. Serwer porównuje z danymi przechowywanymi w bazie danych i odsyła wynik porównania. Jeżeli dane się zgadzają klient może odpytywać serwer o pozostałe dane. Następnie klient może oddać głos, przejrzeć w jakich głosowaniach uczestniczy, sprawdzić swoje ostatnie głosy i dodać

nowe głosowanie. Jeżeli użytkownik odda głos przechodzi on przez sieć miksującą w celu zapewnienia anonimowości. W przypadku dodawania nowego głosowania serwer zaczyna odliczać czas do jego zakończenia. Po zakończeniu czasu głosowania serwer powiadamia element publikujący o konieczności liczenia głosów. Element publikujący pobiera głosy, przetwarza je i po zakończeniu jego pracy wyniki głosowania są dostępne w aplikacji klienta.

4.2. Aplikacja serwera (urna)

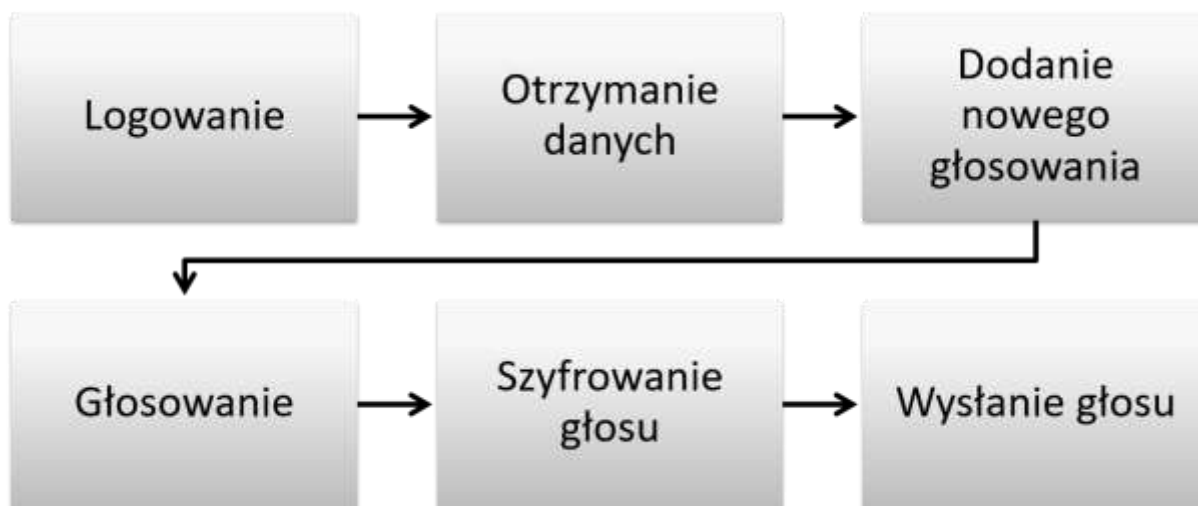
Aplikacja serwera to standardowa aplikacja konsolowa systemu Windows. Jej głównym zadaniem jest pobieranie danych z bazy danych i przesyłanie ich do innych modułów. Oprócz tego odlicza ona czas do zakończenia głosowań i informuje o tym element publikujący. Przybliżony sposób działania pokazuje rysunek 4.



Rysunek 4 Schemat działania serwera

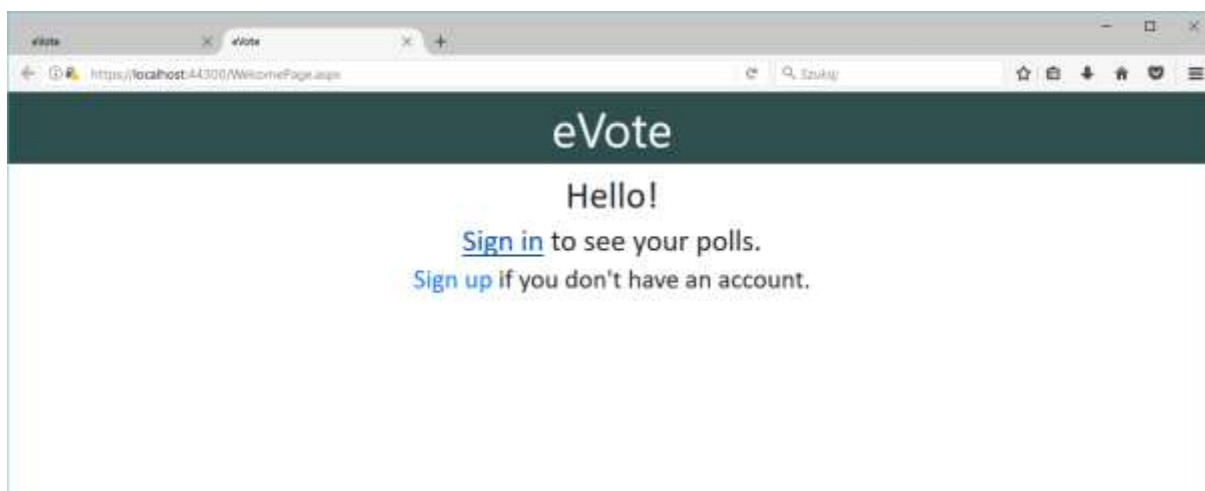
4.3. Aplikacja klienta (głosujący)

Aplikacja klienta to aplikacja webowa frameworku .NET. Służy ona do interakcji z użytkownikiem i pozwala mu na wykonanie podstawowych akcji. Składa się z kilku stron internetowych. Pobieranie stron internetowych odbywa się za pomocą zabezpieczonego protokołu HTTPS w celu zachowania prywatności. Rysunek 5 pokazuje schemat działania tej aplikacji.

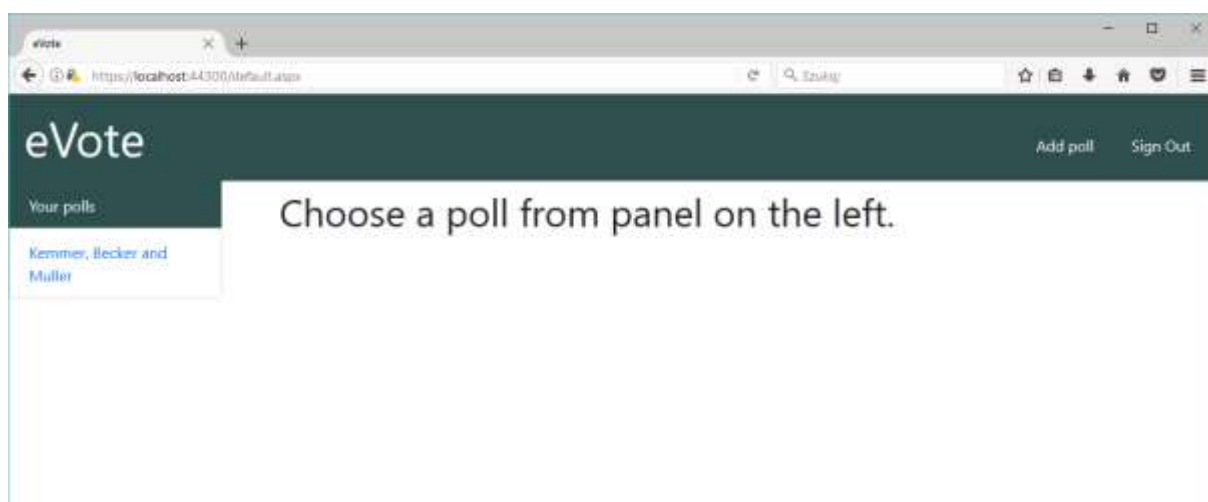


Rysunek 5 Schemat działania aplikacji klienta

Jeżeli użytkownik nie jest zalogowany to próba przejścia do widoku głosowania lub dodania głosowania automatycznie przekieruje do strony logowania. Sprawdzenie czy użytkownik jest zalogowany odbywa się za pomocą sprawdzenia pliku cookie. Widok stron dla użytkownika zalogowanego został wzbogacony o przycisk wylogowania, przycisk dodania nowego głosowania i listę dostępnych głosowań. Przykłady widoku użytkownika zalogowanego i niezalogowanego pokazują rysunki 6 i 7.



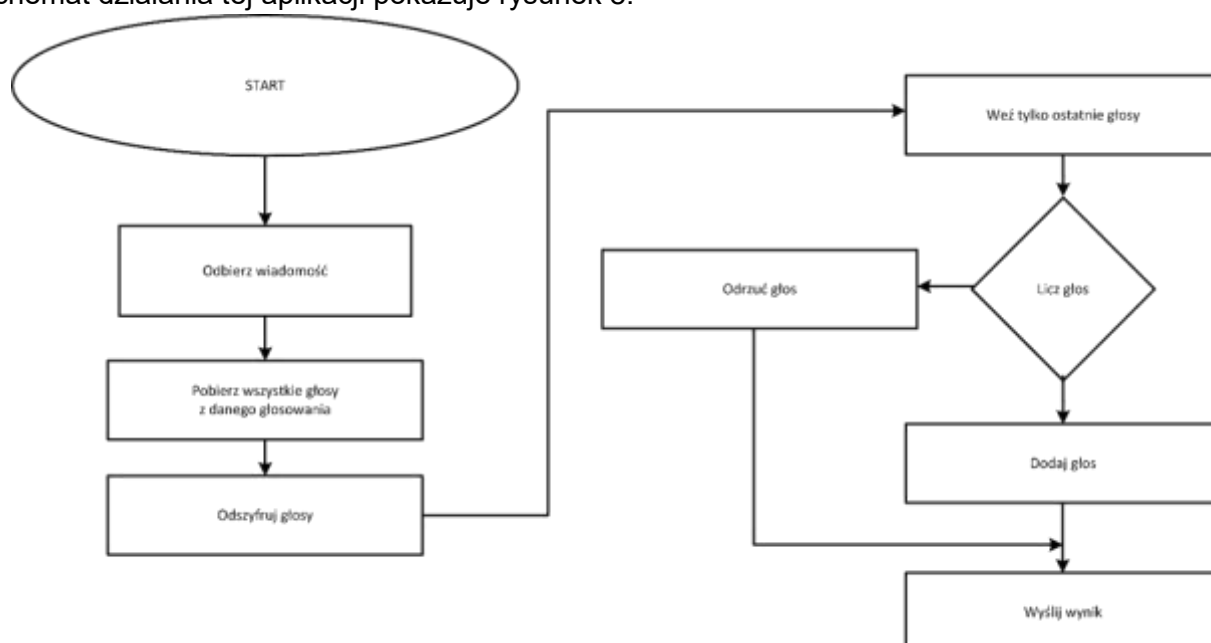
Rysunek 6 Widok użytkownika niezalogowanego



Rysunek 7 Widok użytkownika zalogowanego

4.4. Element publikujący (komisja)

Element publikujący to standardowa aplikacja konsolowa. Jego zadaniem jest przeliczenie wyników głosowań. Oczekuje on na wiadomość, która powiadamia go o zakończeniu głosowania. Następnie pobiera on głosy z zakończonego głosowania. Następnie głosy te są odszyfrowywane i liczone według zapisanych preferencji. Przeliczone głosy są publikowane. Schemat działania tej aplikacji pokazuje rysunek 8.



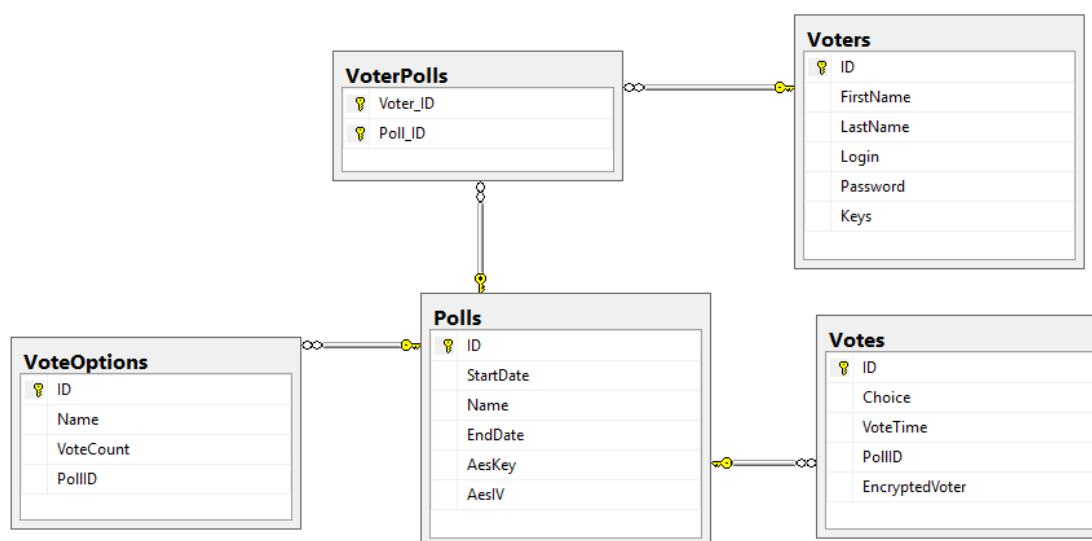
Rysunek 8 Schemat działania elementu publikującego.

4.5. Baza danych

Baza danych dla tego systemu została stworzona przez system ORM Entity Framework na podstawie zdefiniowanych klas. Składa się ona z 5 tabel i jest zarządzana przez Microsoft SQL Server 2014. Tabela 3 pokazuje jakie tabele są w bazie danych, a rysunek 9 ich relacje.

Tabela 3 Tabele w bazie danych

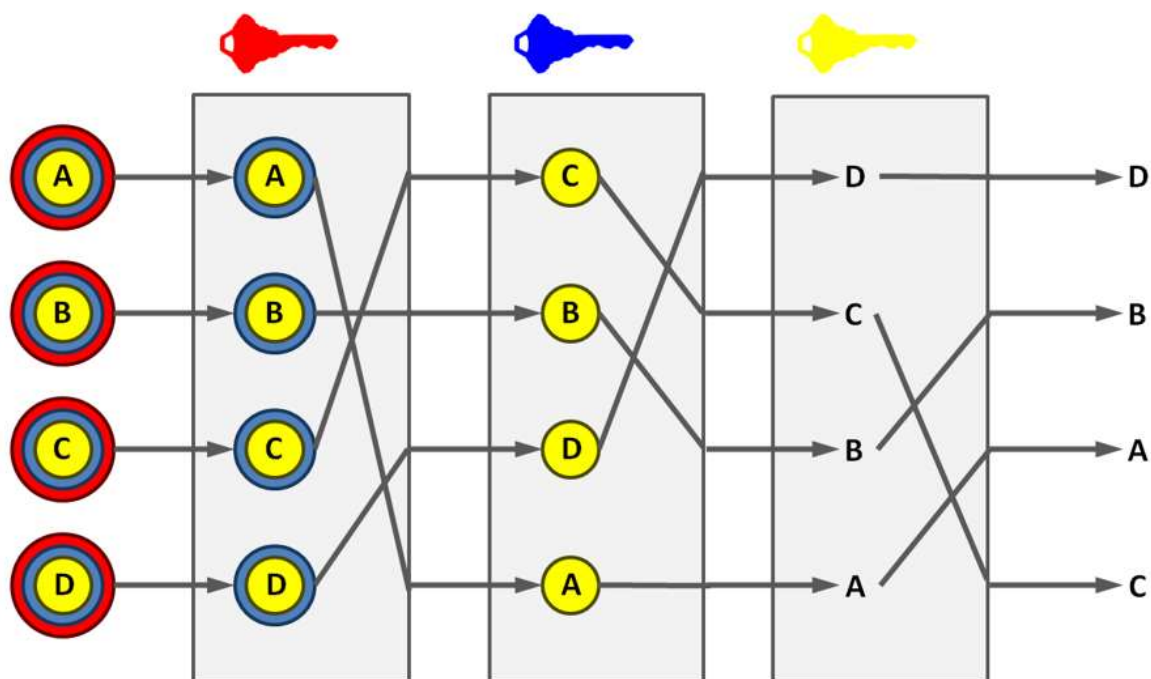
Nazwa tabeli	Zawartość
Votes	Dane dotyczące głosów, m. in. wybór użytkownika, data głosowania.
Voters	Dane użytkowników: nazwisko, imię, login, hasło, zaszyfrowane klucze RSA
VoteOptions	Opcje wyboru w głosowaniach i liczba zdobytych punktów
Polls	Data rozpoczęcia i zakończenia głosowania, nazwa głosowania, zaszyfrowany tajny klucz głosowania.
VoterPolls	Tabela łącząca tabele Voters i Polls. Niezbędna do wykonania relacji wiele do wielu.



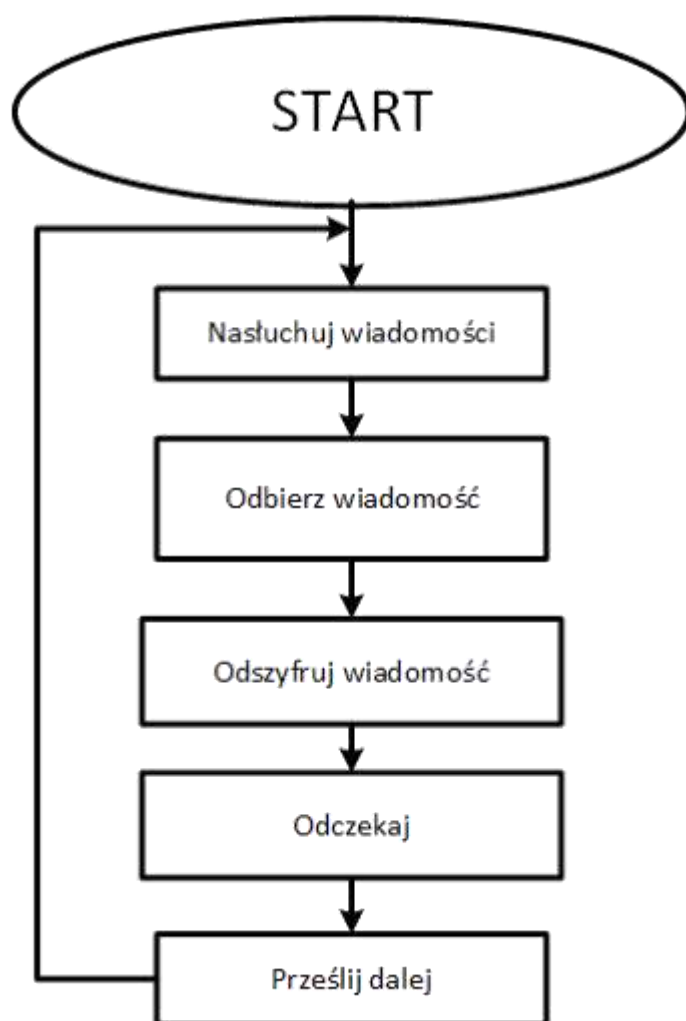
Rysunek 9 Schemat relacji w bazie danych

4.6. Sieć miksująca

Sieć miksująca, to sieć składająca się z tak zwanych węzłów miksujących. Jest to jeden z najpopularniejszych sposobów zapewnienia anonimowości w Internecie. Pakiety w takiej sieci są kilkakrotnie szyfrowane kluczami węzłów sieci. Następnie są one rozsyłane przez te węzły. Każdy węzeł miksujący ma za zadanie odebrać pakiet, odszyfrować go swoim węzłem i przesłać go do innego węzła lub odbiorcy. Schemat działania sieci miksującej pokazuje rysunek , a rysunek pokazuje jak działa węzeł miksujący.



Rysunek 10 Schemat działania sieci miksującej.



Rysunek 11 Schemat działania węzła sieci miksującej.

5. Implementacja

5.1. Elementy wspólne systemu

Wszystkie aplikacje w implementowanym systemie korzystają z tych samych klas obiektów. Dla ułatwienia pracy nad implementacją klasy te zostały wydzielone do dwóch bibliotek dll. Biblioteki te są później dołączane do gotowych aplikacji. Pierwsza biblioteka nazywa się eVote.Database i zawiera definicję klas używanych do łączenia się z bazą danych. Druga biblioteka nazywa się eVote.Messages i zawiera definicję klasy wiadomości przesyłanej pomiędzy obiektami. Klasa ta zawiera też funkcje szyfrujące.

5.1.1. Biblioteka eVote.Database

Biblioteka eVote.Database to biblioteka służąca do komunikacji aplikacji z bazą danych. Na podstawie klas stworzonych w tej bibliotece Entity Framework stworzył tabele w bazie danych, wraz z relacjami pomiędzy nimi. Poniżej omówiono każdą z tych klas:

- **Klasa Poll**

Klasa Poll to klasa odwzorowująca rekord z tabeli Polls w bazie danych. Służy ona jako model głosowania przeprowadzanego w implementowanym systemie.

Pola prywatne i właściwości tej klasy pokazuje tabela 4.

Tabela 4 Pola klasy Poll

Nazwa pola	Zawartość
ID	Pole kucza głównego w tabeli.
Name	Nazwa głosowania.
StartDate	Data rozpoczęcia głosowania.
EndDate	Data zakończenia głosowania.
AesKey	Zaszyfrowany klucz tajny algorytmu AES.
AesIV	Zaszyfrowany wektor inicjalizujący algorytmu AES..

Oprócz tego zawiera ona kolekcje obiektów VoteOption, i Voter. Zawarcie tych kolekcji w klasie przekazuje Entity Framework informację, że spodziewamy się relacji z ich tabelami w bazie danych.

- **Klasa Voter**

Klasa Voter odwzorowuje rekord z tabeli Voters. Ta klasa służy jako model głosującego w systemie.

Tabela 5 pokazuje jakie pola i właściwości ma ta klasa.

Tabela 5 Pola klasy Voter

Nazwa pola	Zawartość
ID	Pole kucza głównego w tabeli.
FirstName	Imię głosującego
LastName	Nazwisko głosującego.
Login	Login użytkownika (adres e-mail).
Password	Skrót hasła użytkownika.
Keys	Zaszyfrowane klucze RSA.

Klasa ta zawiera jeszcze kolekcję obiektów Poll. Dzięki niej zostanie utworzona relacja wiele do wielu tabel Polls i Voters. Relacja wiele do wielu nie jest obsługiwana przez MSSQL serwer, więc entity Framework stworzy nową tabelę łączącą VoterPolls.

- **Klasa Vote**

Klasa Vote odwzorowuje rekord z tabel Voters. Jest ona jednocześnie modelem głosu oddanego w głosowaniu.

Tabela 6 pokazuje jakie pola i właściwości ma ta klasa:

Tabela 6 Pola klasy Vote

Nazwa pola	Zawartość
ID	Pole kucza głównego w tabeli.
Choice	Zaszyfrowany wybór głosującego
VoteTime	Czas oddania głosu.
EncryptedVoter	Login głosującego zaszyfrowany kluczem głosowania.
PollID	Klucz obcy potrzebny do relacji z tabelą Polls.

- **Klasa VoteOption**

Klasa VoteOption odwzorowuje rekord z tabeli VoteOptions. Jest modelem opcji wyboru w głosowaniu.

Tabela 7 pokazuje jej pola i właściwości.

Tabela 7 Pola klasy VoteOption

Nazwa pola	Zawartość
ID	Pole kucza głównego w tabeli.
Name	Treść opcji.
VoteCount	Ilość zdobytych punktów w głosowaniu.
PollID	Klucz obcy potrzebny do relacji z tabelą Polls.

- **Klasa eVoteModel**

Klasa eVoteModel została automatycznie stworzona przez Entity Framework. To w niej definiujemy kolekcje, które później zostaną zmapowane na tabele w bazie danych. Klasa ta dziedziczy po klasie DbContext, dzięki czemu może pobierać dane z bazy.

5.1.2. Biblioteka eVote.Messages

Biblioteka eVote.Messages składa się tylko z jednej klasy o nazwie Message. Klasa ta odwzorowuje wiadomość przesyłaną pomiędzy modułami aplikacji. Zawiera ona pola Sender, Receiver, Subject i Data. Przechowują one odpowiednio nadawcy, odbiorcę, temat i dane wiadomości.

Oprócz pól Klasa Message zawiera grupę metod służącą do szyfrowania danych. Metody te są statyczne, ponieważ nie muszą się one odwoływać do żadnego z pól klasy. Umożliwia to również wykorzystywanie ich w innych programach bez potrzeby tworzenia obiektu Message. Spis tych metod znajduje się w tabeli 8.

Tabela 8 Metody statyczne klasy Message

Nazwa	Działanie
EncryptStringToBytes_Aes	Funkcja szyfrująca łańcuch znaków do tablicy bajtów za pomocą algorytmu AES
DecryptStringFromBytes_Aes	Funkcja deszyfrująca tablicę bajtów do łańcucha znaków za pomocą algorytmu AES
EncryptStringToBytes_RSA	Funkcja szyfrująca łańcuch znaków do tablicy bajtów za pomocą algorytmu RSA
DecryptStringFromBytes_RSA	Funkcja deszyfrująca tablicę bajtów do łańcucha znaków za pomocą algorytmu RSA
SHA512	Funkcja generująca skrót łańcucha znaków do innego łańcucha znaków.

5.2. Aplikacja serwera

Głównym zadaniem serwera jest przesyłanie danych pomiędzy modułami i bazą danych. Zadanie to można jednak podzielić na kilka mniejszych:

- obsługa połączeń TCP,
- uwierzytelnianie użytkownika,
- dodanie nowego głosu,
- dodanie nowego głosowania,
- dodanie nowego użytkownika,
- zakończenie głosowania.

Poniżej znajduje się opis poszczególnych zadań.

5.2.1. Obsługa połączeń TCP

Klasa serwera znajduje implementowanego systemu się w bibliotece eVote.Server i nazywa się DatabaseServer. Obiekt tej klasy zawiera w sobie obiekt typu TcpListener, służący do nasłuchiwanie nadchodzących połączeń na wybranym porcie. W celu obsługi nawiązanych połączeń zaimplementowano dwie metody – Initialize i HandleConnection.

Metoda Initialize rozpoczyna nasłuchiwanie na wybranym porcie. Kiedy nadejdzie nowe połączenie tworzy ona nowy wątek, który ma za zadanie wykonać metodę HandleConnection. Następnie utworzony wątek jest ustawiony na działanie w tle i rozpoczynany.

Metoda HandleConnection to prawidłowa metoda obsługi połączenia TCP. Odczytuje ona nadchodzącą wiadomość, deszyfruje ją i sprawdza czy jej podpis elektroniczny się zgadza. Następnie sprawdza temat wiadomości i nakazuje wykonanie innej funkcji w zależności od tematu.

Dla zachowania bezpieczeństwa wszystkie wiadomości wysyłane są w postaci zaszyfrowanej z wyjątkiem kluczy publicznych modułów.

5.2.2. Uwierzytelnianie użytkownika

Uwierzytelnianie użytkownika odbywa się za pomocą metody VoterLogin. Metoda ta otrzymuje jako parametr dane użytkownika i klienta TCP.

Wykonanie metody zaczyna się od pobrania z bazy danych obiektu głosującego o loginie podanym w parametrze. Następnie sprawdza się poprawność hasła użytkownika. Jeżeli użytkownik nie istnieje lub hasło jest nieprawidłowe uwierzytelnianie kończy się porażką. W

przeciwnym wypadku uwierzytelnienie kończy się sukcesem. Metoda kończy się po wysłaniu wiadomości z wynikiem uwierzytelnienia za pomocą klienta TCP otrzymanego w parametrze.

5.2.3. Dodanie nowego głosu

Dodanie nowego głosu odbywa się za pomocą metody AddNewVote. Metoda ta przyjmuje jako parametr głos do dodania.

Wykonywanie metody zaczyna się od sprawdzenia czy głos nie został oddany po zakończeniu głosowania. Jeżeli tak się stanie metoda kończy się wykonywać. Następnie głos jest dodawany do bazy danych. Ta metoda nie wysyła niczego do klienta.

5.2.4. Dodanie nowego głosowania

Za dodanie nowego głosowania odpowiada metoda AddNewPoll przyjmująca jako parametr obiekt typu Poll.

Na początku wykonywania metody sprawdzany jest czy data zakończenia głosowania nie jest ustawiona w przeszłości. Jeżeli tak się stało metoda kończy działanie. Następnie tworzony jest nowy Timer, który odlicza czas do zakończenia głosowania. W następnym kroku tworzy się tajny klucz algorytmu AES, który będzie służył do szyfrowania głosów tego głosowania. Metoda kończy się wykonywać po dodaniu głosowania do bazy danych.

5.2.5. Dodanie nowego głosującego

Dodanie nowego głosującego do bazy danych zajmuje się metoda AddNewVoter. Metoda ta przyjmuje jako parametr obiekt Voter i klienta TCP.

Rozpoczynając wykonywanie metody sprawdzamy, czy użytkownik nie jest już dodany w naszej bazie. Jeżeli znajdziemy login użytkownika w bazie odsyłamy wiadomość zwrotną z taką informacją. Po sprawdzeniu czy login taki znajduje się w naszej bazie i otrzymaniu odpowiedzi negatywnej tworzymy nowe klucze algorytmu RSA dla użytkownika i dodajemy go do bazy danych. Wykonywanie metody kończy się po wysłaniu wiadomości zwrotnej przy użyciu klienta TCP.

5.2.6. Zakończenie głosowania

Zakończeniem głosowania zajmuje się metoda PollFinished. Wykonuje się ona po ukończeniu działania minutnika, który został włączony przy tworzeniu głosowania. Przyjmuje ona jako parametr obiekt Poll.

Pierwszą rzeczą, jaką robi ta metoda jest odszyfrowanie klucza tajnego używanego w głosowaniu. Następnie obiekt głosowania jest serializowany do łańcucha znaków za pomocą biblioteki JSON. Ten łańcuch znaków jest wysyłany w wiadomości do elementu publikującego za pomocą połączenia TCP. Aby połączenie zostało nawiązane tworzy się nowego klienta TCP.

5.3. Aplikacja klienta

Aplikacja klienta to aplikacja internetowa. Składa się ona z kilku projektów z czego najważniejszymi są eVote.Client i eVote.WebClient. Projekt eVote.Client to biblioteka dll, w której znajduje się logika klienta, natomiast projekt eVote.WebClient to aplikacja internetowa odpowiedzialna za interfejs graficzny i interakcję z użytkownikiem.

W projekcie eVote.WebClient znajdziemy pliki stron internetowych, które będą wyświetlane użytkownikowi. Strony te to:

- strona domowa,
- strona logowania użytkownika,

- strona rejestracji,
- strona głosowania,
- strona wyników głosowania,
- strona dodawania głosowania.

Strony te są napisane w języku HTML z wyglądem zdefiniowanym w języku CSS.

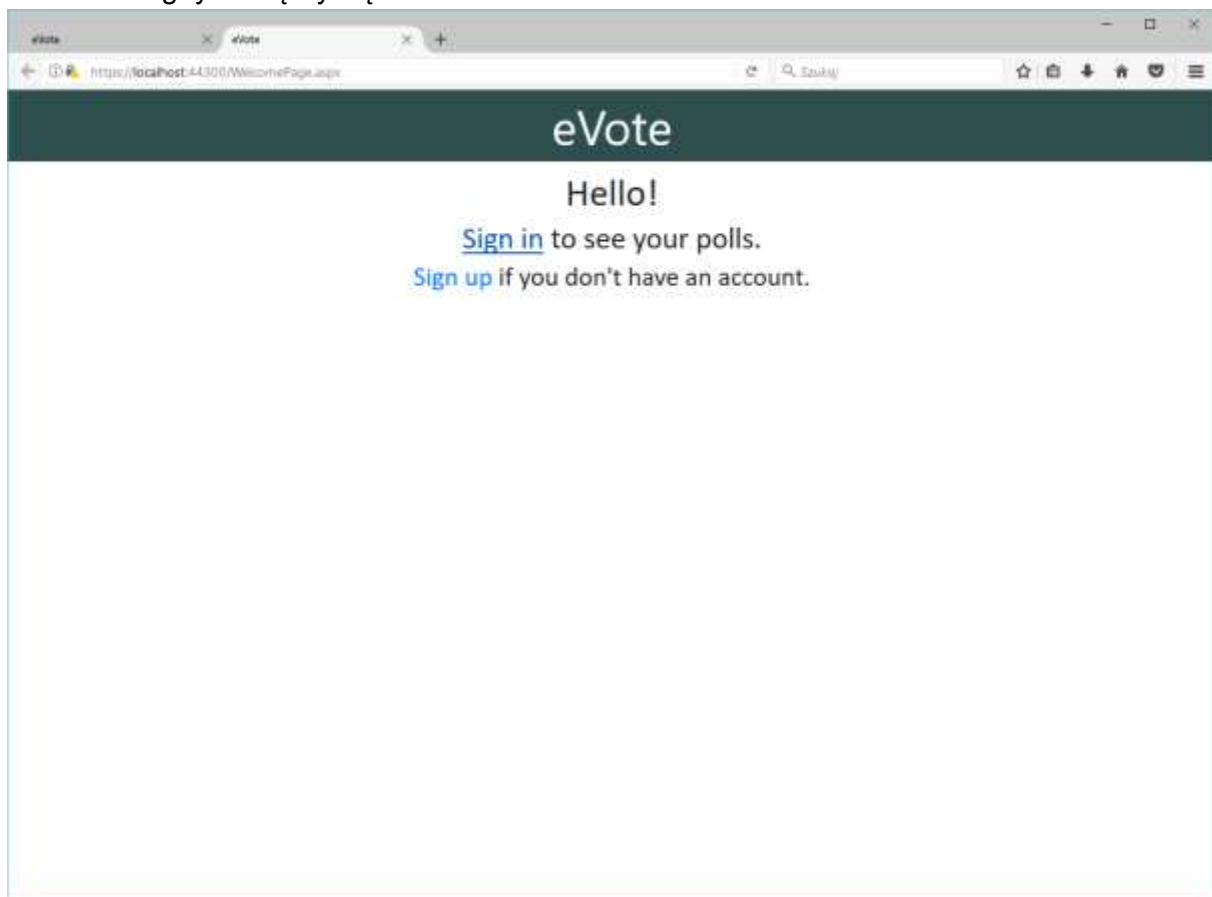
W projekcie eVote.Client znajduje się klasa Client, która odpowiada za logikę aplikacji.

Wszystkie przejścia pomiędzy stronami są wykonywane za pomocą metody `HttpResponse.Redirect` biblioteki `System.Web` języka C#.

5.3.1. Widok początkowy

Na początku użytkownik pobiera stronę `WelcomePage.aspx`. Jej widok można zobaczyć na rysunku 12. Z tej strony może przejść do strony logowania poprzez naciśnięcie przycisku `Sign in`. Jeżeli użytkownik nie ma konta może je stworzyć za pomocą strony rejestracji, do której przejdzie naciskając przycisk `Sign up`.

Ta strona nigdy nie łączy się z serwerem.

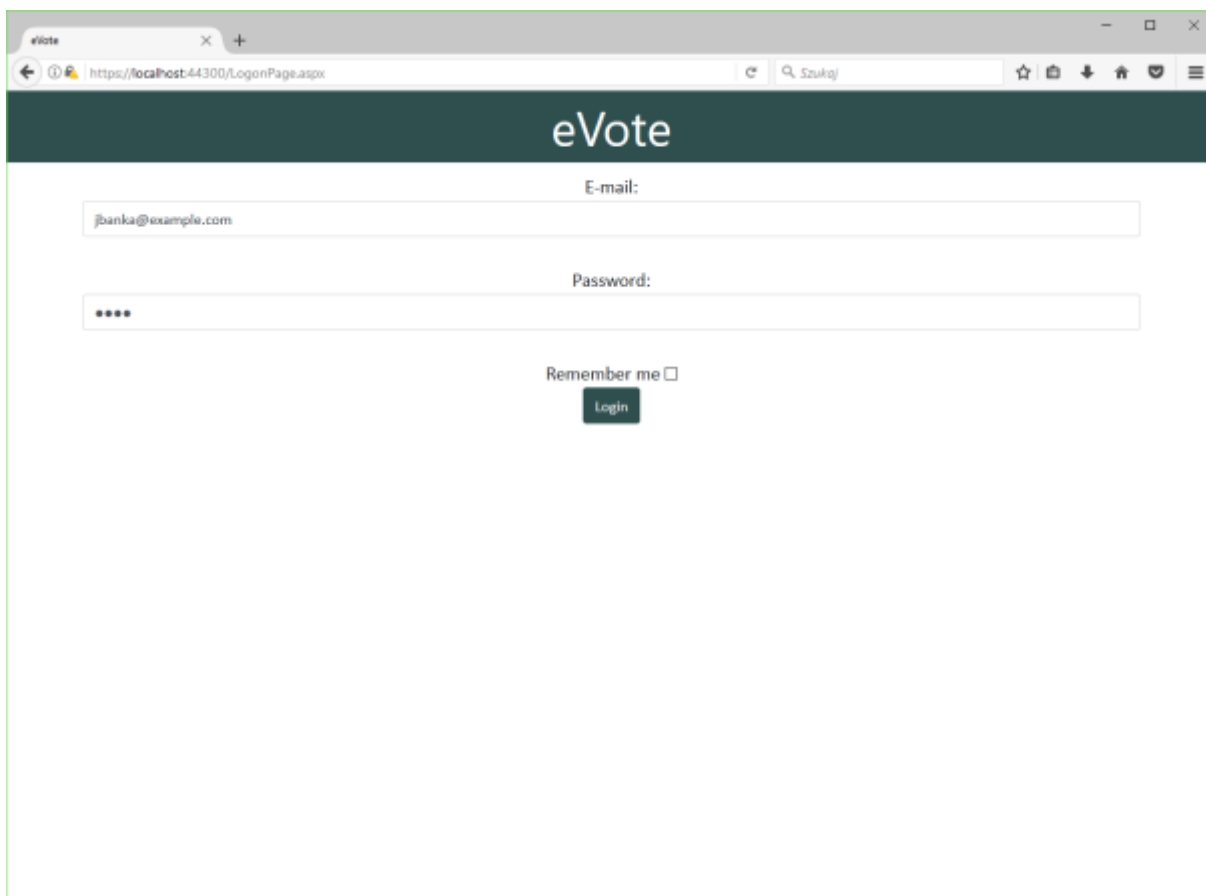


Rysunek 12 Widok strony domowej.

5.3.2. Uwierzytelnianie użytkownika

Uwierzytelnianie użytkownika odbywa się na podstawie danych strony nazwanej `LogonPage.aspx`. Strona ta służy do logowania się do systemu i pokaże się zawsze, gdy niezalogowany użytkownik spróbuje wejść na stronę dla użytkownika zalogowanego.

Strona składa się z dwóch pól tekstowych, jednego pola wyboru i jednego przycisku. Są to miejsca, w których użytkownik może wprowadzić swoje dane. Wygląd strony widać na rysunku 13.

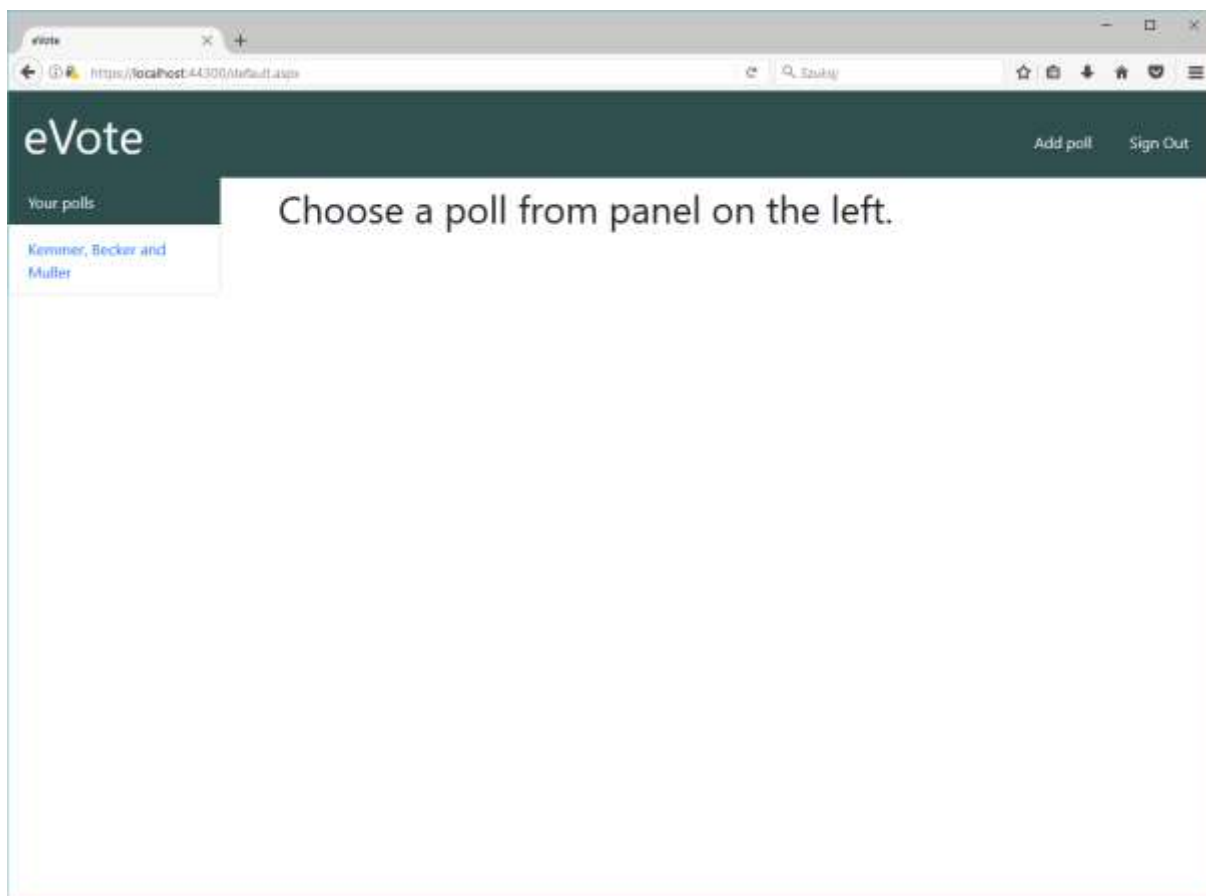


Rysunek 13 Strona logowania.

Zanim strona logowania połączy się z serwerem, sprawdza czy pola tekstowe nie są puste. Jeżeli tak aplikacja nie łączy się z serwerem. Jeżeli dane użytkownika są wprowadzone aplikacja pobiera je ze strony i przekazuje do metody Login klasy Client. Metoda ta zwraca wartość logiczną odpowiadającą wynikowi sprawdzenia danych podanych przez użytkownika. Aby sprawdzić czy dane są prawidłowe, metoda Login pobiera login i hasło użytkownika. Hasło jest przetwarzane funkcją skrótu w celu jego zabezpieczenia. Następnie tworzy obiekt klasy Voter i wypełnia pola loginu i hasła. Obiekt ten jest następnie wysyłany do serwera za pomocą klienta TCP. Po otrzymaniu odpowiedzi od serwera użytkownik jest przekierowany na inną stronę lub wyświetla się informacja o nieprawidłowych danych.

5.3.3. Wybór głosowania

Wybór głosowania odbywa się po zalogowaniu. Użytkownik może przejść do innego głosowania z każdej strony, do której ma dostęp po zalogowaniu. Odbywa się to poprzez naciśnięcie nazwy głosowania z listy, która znajduje się po lewej stronie ekranu. Na rysunku 14 widzimy stronę Default.aspx, która pokazuje się po zalogowaniu.

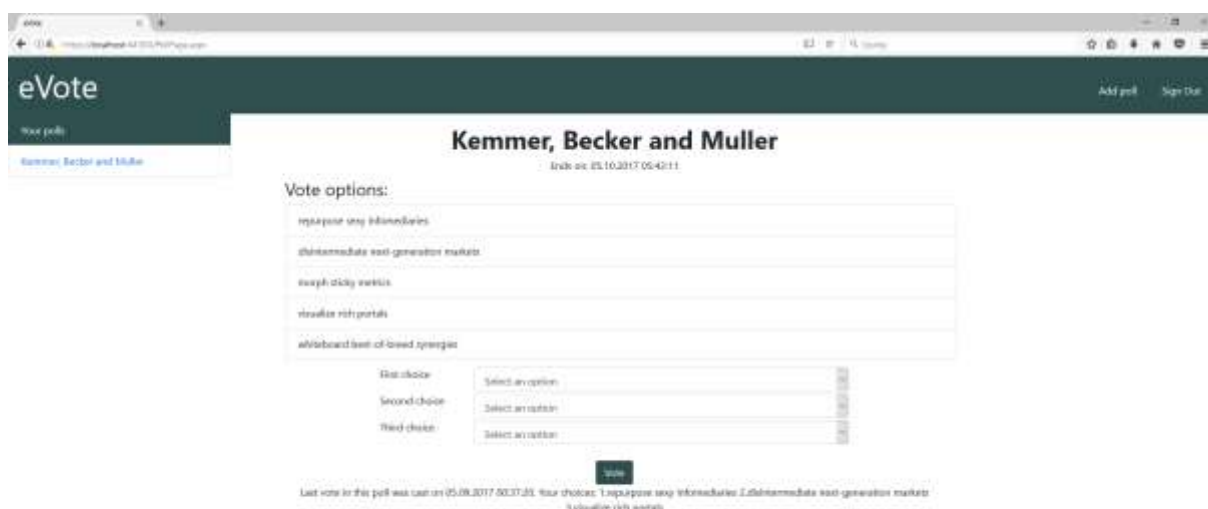


Rysunek 14 Strona wyboru głosowania.

Ponieważ lista głosowań nie potrzebuje danych, które są zaszyfrowane pobierana są one bezpośrednio z bazy danych. Pobierane głosowania zależą od głosującego. Po dokonaniu wyboru użytkownik jest przekierowany na stronę głosowania lub na stronę z wynikami głosowania, jeżeli głosowanie jest zakończone.

5.3.4. Widok głosowania i dodawanie nowego głosu

Strona pokazująca szczegóły głosowania nazywa się PollPage. Służy ona jednocześnie do oddania głosu. Składa się z dużego nagłówka z nazwą głosowania. Poniżej nazwy jest informacja o dacie zakończenia głosowania. Widoczna również jest lista możliwych opcji do wyboru. Pod listą znajdują się trzy pola wyboru, dzięki którym użytkownik może wybrać swoje preferencje w tym głosowaniu. Poniżej znajduje się przycisk do oddania głosu. Pod przyciskiem wyświetla się informacja o ostatnim oddanym przez użytkownika głosie.



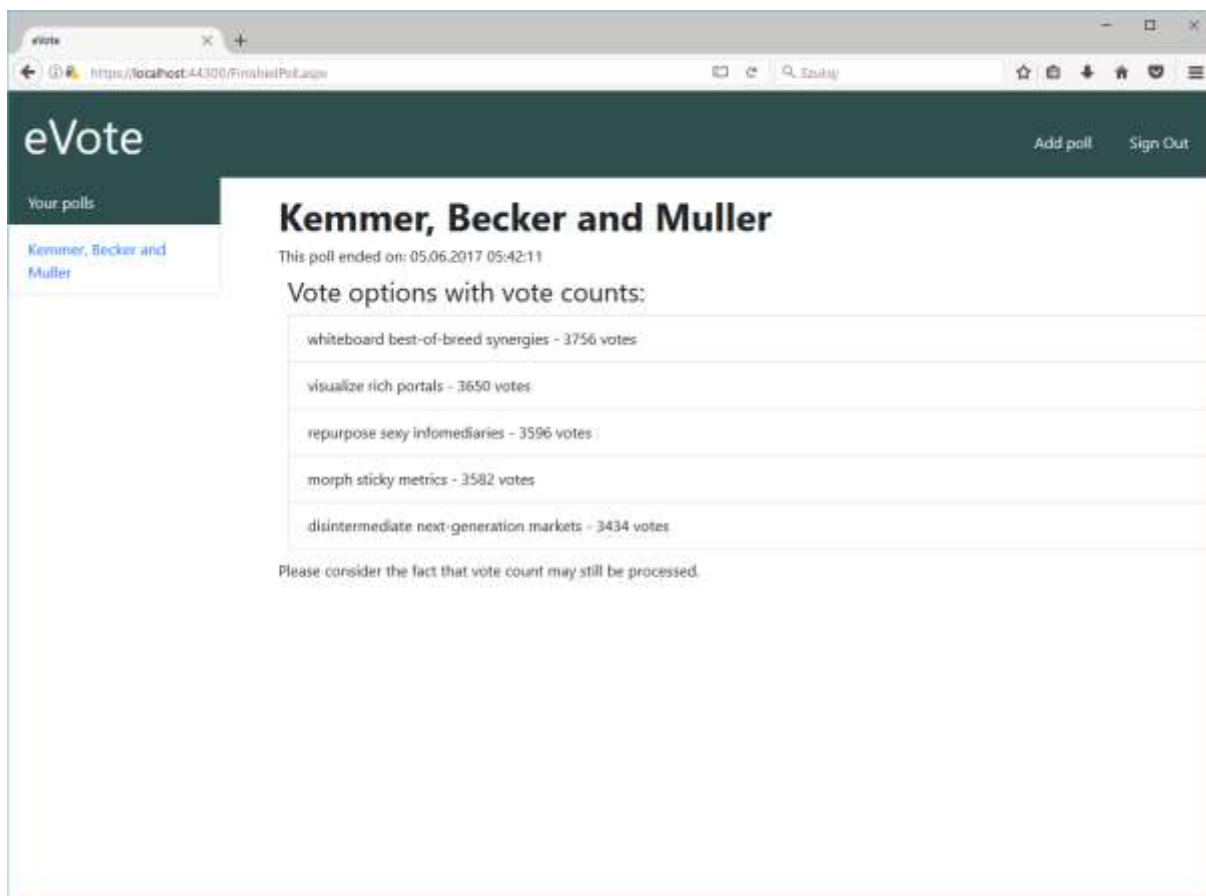
Rysunek 15 Strona głosowania.

Aby prawidłowo wyświetlić tę stronę aplikacja klienta musi połączyć się z bazą danych w celu pobrania danych o głosowaniu – jego nazwy, daty zakończenia i opcji głosowania. Aplikacja łączy się również z serwerem w celu pobrania ostatniego głosu głosującego i dowiedzenia się jak głosował. W celu zdobycia tej informacji aplikacja wysyła do serwera ID głosowania i login użytkownika. Serwer, po przetworzeniu danych, odpowiada informacją, która później jest przypisywana do kontrolki. Informacje o ostatnim głosie wydobywa metoda `GetLastVote` klasy `Client`.

W celu dodania nowego głosu do bazy danych używana jest metoda `CastVote` klasy `Client`. Aplikacja pobiera ze strony dane na temat preferencji użytkownika. Po sprawdzeniu czy są one prawidłowo wybrane, aplikacja wysyła do serwera zapytanie o klucz tajny wybranego głosowania. Po otrzymaniu klucza od serwera aplikacja generuje zaszyfrowany głos i wysyła go do serwera, który przetwarza go zgodnie z logiką opisaną w punkcie 4.2.3.

5.3.5. Widok zakończonego głosowania

Widok głosowania, którego czas trwania już minął pokazuje stronę `FinishedPoll.aspx`. Jest ona bardzo podobna do strony niezakończzonego głosowania – również zawiera nagłówek z nazwą głosowania i informację o dacie zakończenia głosowania. Różnice zaczynają się w liście możliwych opcji głosowania – zostały one uporządkowane według liczby zdobytych punktów, ale dodano do nich właśnie tę wartość. Zamiast reszty kontrolerek dodano informację, że nasze głosy mogą jeszcze być liczone.



Rysunek 16 Strona zakończonego głosowania.

Strona ta ponownie nie komunikuje się z serwerem. Wyniki głosowania nie są zaszyfrowane, więc dane te są pobierane bezpośrednio z bazy danych i przetwarzane w aplikacji klienta.

5.3.6. Dodanie nowego głosowania

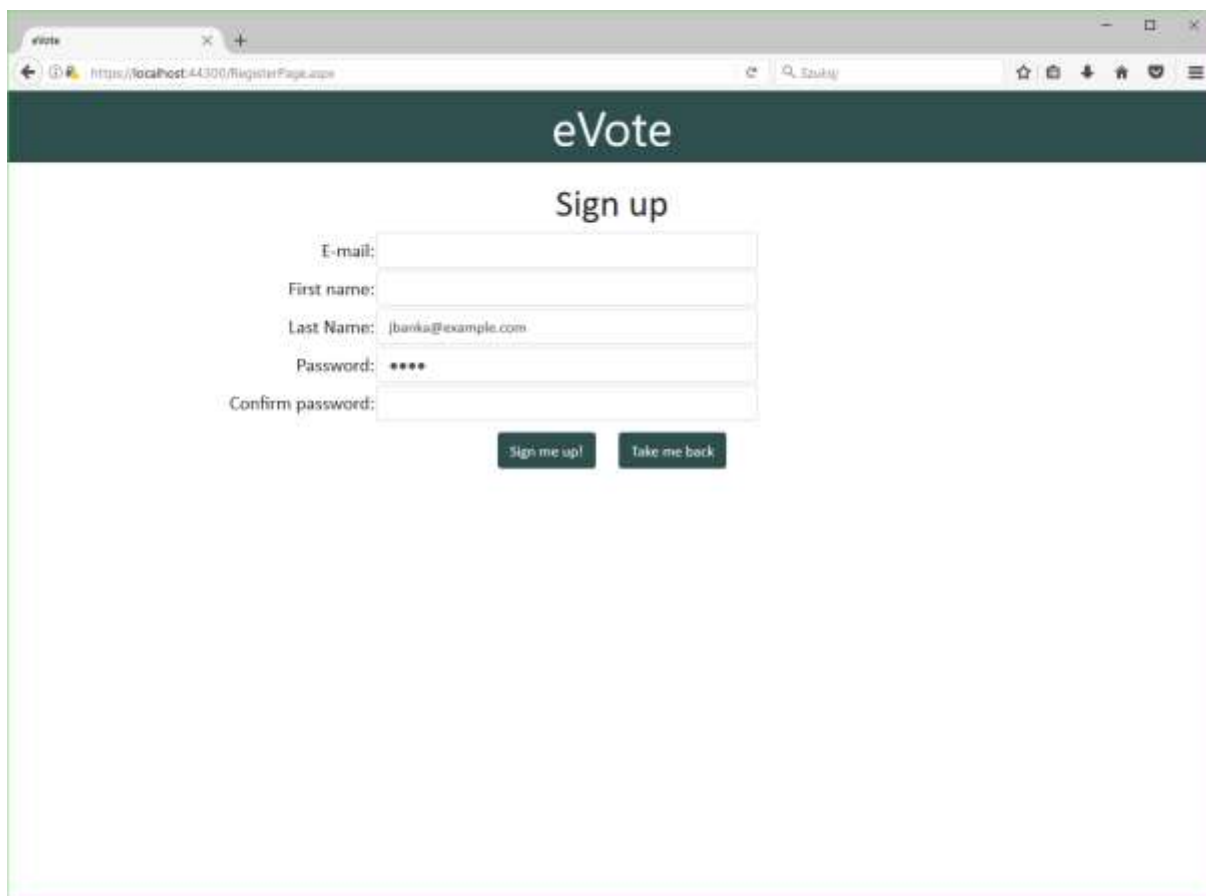
Strona służąca do dodania nowego głosowania stworzono stronę `AddNewPoll.aspx`. Składa się ona z grupy pól tekstowych, które użytkownik powinien wypełnić i przycisku dodania nowego głosowania.

Rysunek 17 Strona dodawania nowego głosowania.

Po naciśnięciu przycisku dodania nowego głosowania aplikacja pobiera dane z pól tekstowych i sprawdza ich poprawność. Na podstawie danych pobranych z bazy danych, usuwa wszystkie adresy e-mail (loginy), które nie występują w bazie danych. Usuwa również powtarzające się opcje wyboru. Następnie zostaje wywołana metoda `AddNewPoll` klasy `Client`. Metoda ta serializuje przygotowany wcześniej obiekt typu `Poll` i wysyła go do serwera.

5.3.7. Dodawanie nowego głosującego

W celu dodania nowego głosującego stworzono stronę `RegisterPage.aspx`. Ta strona nie wymaga logowania się. Składa się z pięciu pól tekstowych i dwóch przycisków. Pola tekstowe służą do wprowadzania danych użytkownika, a przyciski służą do zatwierdzenia danych do przesłania lub cofnięcia się na stronę domową.



Rysunek 18 Strona rejestracji użytkownika

Strona ta sprawdza czy któreś z pól nie jest puste oraz czy wpisywane hasła są identyczne zanim naciśnie się przycisk zatwierdzający. Robi to za pomocą kontrolek walidacyjnych frameworku .NET. Po naciśnięciu przycisku anulującego cofamy się do strony głównej. Po naciśnięciu przycisku akceptującego aplikacja wywołuje metodę `CastVote` klasy `Client`. Metoda ta pobiera dane ze strony i prosi serwer o klucz do zaszyfrowania ich. Po otrzymaniu klucza od serwera, klient tworzy właściwy głos, serializuje go i wysyła do serwera w celu dodania go do bazy danych. Aplikacja klienta nie oczekuje na odpowiedź serwera odnośnie oddanego głosu.

5.4. Element publikujący

Element publikujący to standardowa aplikacja konsolowa. Cała logika tego modułu znajduje się w klasie `VoteCounter`. Jedynym zadaniem tego modułu jest przeliczenie głosów z zakończonego głosowania. Podobnie jak klasa serwera ma ona obiekt `TcpListener` aby można było połączyć się z modułem. Obsługa połączeń TCP odbywa się tak samo jak w serwerze. Jedyna różnica to wiadomości jakich spodziewa się element publikujący. Jedyna wiadomość, której spodziewa się ten moduł to wiadomość o zakończeniu głosowania.

5.4.1. Zakończenie głosowania i przeliczenie głosów.

Po otrzymaniu wiadomości, że głosowanie zostało zakończone element publikujący rozpoczyna przeliczanie głosów. Zaczyna od pobrania wszystkich głosów z zakończonego głosowania. Następnym krokiem jest pobranie wszystkich głosujących w tym głosowaniu. Na podstawie kluczy otrzymanych w wiadomości definiuje, które głosy zostały oddane jako

ostatnie przez głosującego. Kolejny krok polega na odszyfrowaniu wyborów głosującego i przydzielenie punktów według wyboru zapisanego w głosie. Następnie zmiany są przekazywane do bazy danych.

6. Instrukcja dla użytkownika

6.1. Przygotowanie do uruchomienia aplikacji

Ta część instrukcji przeznaczona jest dla osoby, która przygotowuje system do użytku.

Aby aplikacja działała poprawnie należy ją odpowiednio przygotować. Dla ułatwienia tego zadania przygotowano katalog Deploy, który zawiera pliki potrzebne do przygotowania aplikacji. Katalog zawiera 3 podkatalogi: client, server i publish element. Znajdują się w nich pliki potrzebne do uruchomienia naszych aplikacji. Działania tu opisane wymagają, aby na komputerze były zainstalowane następujące usługi:

- środowisko .NET w wersji 4.5.2 lub nowszej,
- Microsoft SQL Server w wersji 2014
- Internetowe Usługi Informacyjne (IIS) w wersji 7.0.

Do prawidłowego działania systemu, potrzebna jest baza danych. W celu stworzenia odpowiedniej bazy danych wystarczy wykonać plik *CreateDatabase.sql*. Składa się on z poleceń języka SQL tworzących bazę danych o nazwie eVote-Deploy, dodaje do niej tabele wraz z związkami między nimi. Jeżeli chcemy zmienić nazwę naszej bazy, to plik *CreateDatabase.sql* otwieramy w edytorze tekstowym i za pomocą opcji *Wyszukaj i zamień* zmieniamy wszystkie wystąpienia.

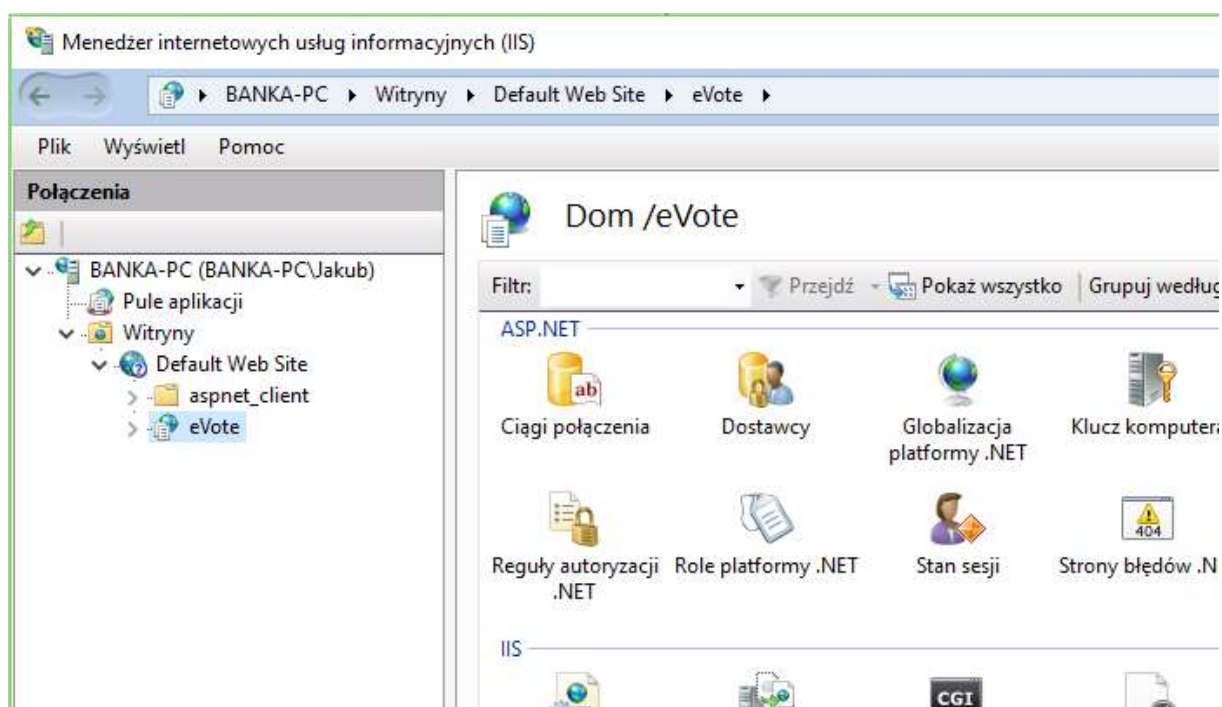
Po dodaniu bazy danych możemy uruchomić pozostałe nasze aplikacje. Jednak zanim to zrobimy, musimy upewnić się czy podłączamy się do odpowiedniej bazy danych. Dla aplikacji serwera i elementu publikującego ustawienie te znajduje się w pliku *.config w katalogu odpowiedniej aplikacji. W tym pliku należy zmienić właściwość connectionString. Następnie możemy odpalić plik *.exe – eVote.PublishElement.exe dla elementu publikującego i ConsoleApplication1.exe dla serwera.

Do działania aplikacji klienta musimy zmienić właściwość connectionString w pliku *eVote.WebClient.SetParameters.xml*. Miejsce edycji zaznaczono na rysunku 19.

```
<?xml version="1.0" encoding="UTF-8"?>
- <parameters>
  <setParameter value="Default Web Site/eVote" name="IIS Web
    Application Name"/>
  <setParameter value="data source=(local);initial
    catalog=eVoteDatabase;integrated
    security=True;MultipleActiveResultSets=True;App=EntityFramework"
    name="eVoteDatabase-Web.config Connection String"/>
</parameters>
```

Rysunek 19 Connection string w pliku eVote.WebClient.SetParameters.xml

Po zmianie tej wartości należy uruchomić wiersz poleceń jako administrator, przejść do folderu client i wykonać komendę *eVote.WebClient.deploy.cmd /Y*. Dla pewności sprawdzamy czy strona eVote została dodana w menedżerze usługi IIS. W menedżerze konfigurujemy również stronę do korzystania z protokołu HTTPS według instrukcji z [tej strony](#).



Rysunek 20 Witryna eVote w menedżerze usług IIS.

6.2. Korzystanie z aplikacji

Do korzystania z aplikacji użytkownik potrzebuje przeglądarki internetowej. Wchodzi on na stronę aplikacji i wyświetla mu się strona domowa. Następnie użytkownik może się zalogować lub stworzyć nowe konto.

6.2.1. Tworzenie konta lub logowanie się

Aby stworzyć konto użytkownik podaje swoje imię, nazwisko, adres e-mail i wymyślone przez siebie hasło. Hasło nie może być puste. Logując się użytkownik używa adresu e-mail i hasła.

6.2.2. Wybór głosowania

Po zalogowaniu się lub rejestracji użytkownik zostaje przekierowany na stronę wyboru głosowania. Głosowanie wybiera się z listy po lewej stronie ekranu. Klikając na nazwę głosowania zostaniemy przekierowany na widok głosowania lub wyników głosowania.

6.2.3. Widok głosowania i oddawanie głosu

Jeżeli wybrane głosowanie wciąż trwa głosującemu pokazuje się stronę głosowania. Strona ta pozwala mu na oddanie głosu. W tym celu użytkownik wybiera swoje preferencje za pomocą pól wyboru.

Użytkownik może wybrać wszystkie 3 opcje, jednak nie jest to wymagane. Wymagane jest tylko wybranie jednej opcji w pierwszym polu. Pierwsze pole wyboru wybiera opcję wartą pięć punktów, drugie opcję wartą trzy, a trzecie opcję wartą jeden punkt.

Po dokonaniu wyboru użytkownik naciska przycisk głosowania. Pod tym przyciskiem jest informacja o ostatnim głosie, który użytkownik oddał w tym głosowaniu.

6.2.4. Sprawdzanie wyników głosowania

Jeżeli wybrane głosowanie się zakończyło, to użytkownik jest przekierowany na stronę wyników głosowania. Na tej stronie nie może nic zrobić – informuje go ona tylko do kiedy można było głosować i jakie są wyniki głosowania.

6.2.5. Dodawanie głosowania

Po zalogowaniu, użytkownik może dodać nowe głosowanie. W celu dodania głosowania należy nacisnąć przycisk *Add poll*. Po naciśnięciu przycisku, głosujący dostaje widok dodawania nowego głosowania. Składa się on z czterech pól tekstowych. Każde pole tekstowe musi być wypełnione, żeby dodać głosowanie.

Pierwsze pole to nazwa głosowania – może zawierać dowolny, niepusty łańcuch znaków. Drugie pole służy do określenia daty i godziny zakończenia głosowania. Preferowana forma daty jest zapisana przy etykiecie.

Kolejne pole to wybór głosujących w głosowaniu. W tym polu należy wpisać adresy e-mail osób z systemu, które mają mieć możliwość oddania głosu. Adresy należy wpisywać oddzielone średnikami. Jeżeli podamy adres, którego nie ma w bazie danych aplikacja nie bierze go pod uwagę. Aplikacja usuwa również białe znaki i powtarzające się adresy e-mail. W celu zwiększenia wygody użytkownika pole to przyjmuje wiele linijek tekstu.

Ostatnie pole służy do dodania opcji głosowania. Opcje dodajemy podobnie jak głosujących. Wszystkie opcje należy wpisać po średniku. Jeżeli opcje się powtórzą zostaną usunięte przy dodawaniu.

6.2.6. Wylogowanie się

Aby się wylogować należy nacisnąć przycisk *Sign out*. Zostaniemy wtedy wylogowani z aplikacji i przekierowani na stronę domową.

7. Testy

7.1. Poprawność działania systemu

Ważną częścią tworzenia implementowanego systemu było testowanie. Funkcjonuje wiele koncepcji weryfikacji poprawności pisanego kodu od sprawdzania każdego fragmentu w momencie jego implementacji do testów końcowych. Implementując ten system, każdy moduł był testowany z pomocą aplikacji konsolowej mającej udawać inne moduły w celu testowania pojedynczych metod. Kiedy poziom aplikacji wrastał aplikacja testowa została zastąpiona innymi modułami. Dzięki temu można było wynajdywać i poprawić błędy we wszystkich modułach na bieżąco.

Po ustaleniu architektury zaczęto tworzyć oprogramowanie, jednocześnie rozpoczynając proces testowania. Testowanie aplikacji tworzonej w Visual Studio było stosunkowo prostym zadaniem dzięki możliwościom tego środowiska. Najprzydatniejszą funkcją Visual Studio była możliwość zatrzymania wykonania metody w wybranym przez nas momencie. Kiedy wykonywanie było zatrzymane bez problemu mogliśmy sprawdzać wartości zmiennych i weryfikować, czy zgadzają się z naszymi oczekiwaniami. W celu sprawdzania czy dane były prawidłowo przetwarzane w bazie danych używano SQL Server 2014 Management Studio. Za pomocą zapytań języka SQL sprawdzano, czy dane zostały przetworzone w sposób prawidłowy.

Testy te były szczególnie przydatne, ponieważ zmniejszyły one ilość poprawek wprowadzonych po innych testach. Dzięki tym testom znaleziono kilka błędów w tym błędy związane z nieprawidłowym zapisem do bazy danych

7.2. Funkcjonalność

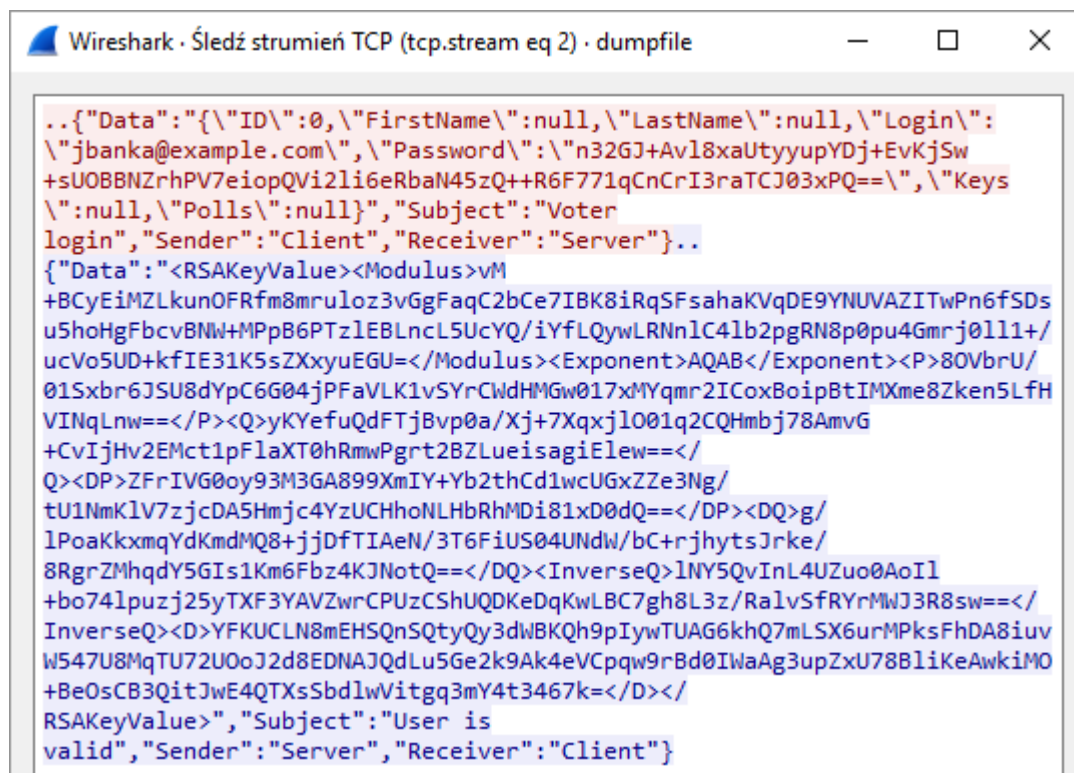
Ostatnim krokiem było przetestowanie gotowych aplikacji. Po uruchomieniu wszystkich modułów, zaczęto analizować krok po kroku, czy system działa prawidłowo zaczynając na logowaniu użytkownika, kończąc na wylogowaniu.

Aplikację oddano do użytku grupie testerów. Zaowocowało to sugestiami usprawnień i zmian, które zostały zaimplementowane lub opisane w dalszej części pracy. Testerzy zgodzili się, że aplikacja jest prosta w obsłudze, mimo kilku niewygodnych rozwiązań.

Dzięki tym testom udało się ustalić, że początkowo serwer nie był wystarczająco wydajny oraz odkryto możliwość dodania głosu po zakończeniu głosowania. Doprowadziło to również do kilku zmian w interfejsie graficznym.

7.3. Bezpieczeństwo

Po sprawdzeniu działania poszczególnych modułów za pomocą analizatora pakietów sprawdzono czy połączenia TCP są zaszyfrowane. Okazało się, że część ruchu nie była szyfrowana co widać na rysunku 21. Okazało się również, że w kilku przypadkach używano złych kluczy szyfrujących.



Wireshark · Śledź strumień TCP (tcp.stream eq 2) · dumpfile

```
..{"Data":{"ID":0,"FirstName":null,"LastName":null,"Login":
"\jbanka@example.com","Password":"n32GJ+Avl8xaUtyyupYDj+EvKjSw
+sUOBBNZrhPV7eiopQVi2li6eRbaN45zQ++R6F771qCnCrI3raTCJ03xPQ==","Keys
":null,"Polls":null},"Subject":"Voter
login","Sender":"Client","Receiver":"Server"}..
{"Data":{"RSAKeyValue":{"Modulus":vM
+BCyEiMZLkunOFRfm8mruloz3vGgFaqC2bCe7IBK8iRqSFsahaKVqDE9YNUVAZITwPn6fSDs
u5hoHgFbcvBNw+MPpB6PTzLEBLncL5UcYQ/iYfLQywLRNn1C4lb2pgRN8p0pu4Gmrj0111+/
ucVo5UD+kfIE31K5sZXxyuEGU=</Modulus><Exponent>AQAB</Exponent><P>80VbrU/
01Sxbr6JSU8dYpC6G04jPFaVLK1vSYrCwdHMGw017xMYqmr2ICoxBoipBtIMXme8Zken5LfH
VINqLnw==</P><Q>yKYefuQdFTjBvp0a/Xj+7Xqxjl001q2CQHmbj78AmvG
+CvIjHv2EMct1pFlaXT0hRmwPgrrt2BZLueisagiElew==</
Q><DP>ZFrIVG0oy93M3GA899XmIY+Yb2thCd1wcUGxZZe3Ng/
tU1NmKlV7zjcDA5Hmjc4YzUCHhoNLHbRhMDi81xD0dQ==</DP><DQ>g/
lPoakKxmQYdKmdMQ8+jjDfTIAeN/3T6FiUS04UNdW/bC+rjhytsJrke/
8RgrZMhqdy5GIs1Km6Fbz4KJNotQ==</DQ><InverseQ>lNY5QvInL4UZuo0AoI1
+bo74lpuzj25yTXF3YAVZwrCPUzCSHUQDKedQKwLBC7gh8L3z/RalvSfRYrMWJ3R8sw==</
InverseQ><D>YFKUCLN8mEHSQnSQtyQy3dWBKQh9pIywTUAG6khQ7mLSX6urMPksFhDA8iuv
W547U8MqTU72U0oJ2d8EDNAJQdLu5Ge2k9Ak4eVCpqw9rBd0IwaAg3upZxU78BliKeAwkiMO
+BeOsCB3QitJwE4QTXsSbdlwVitgq3mY4t3467k=</D></
RSAKeyValue"},"Subject":"User is
valid","Sender":"Server","Receiver":"Client"}
```

Rysunek 21 Niezaszyfrowane połączenie TCP.

8. Podsumowanie

8.1. Ocena stopnia realizacji założeń

Po skończeniu implementacji i testowania autor stwierdził, że większość założeń dokonanych na etapie powstawania koncepcji systemu została zrealizowana. Użytkownik może zagłosować w wybranym przez siebie głosowaniu wyrażając swoje preferencje. W celu uproszczenia aplikacji ustalono, że głosujący może wybrać 3 preferencje i aby to osiągnąć założono, że w jednym głosie użytkownik ma do dyspozycji 9 punktów. Najsilniejsza preferencja warta jest 5 punktów, średnia 3, a najslabsza 1. W danym głosowaniu możemy zagłosować więcej niż jeden raz, ale do wyników wliczany jest tylko ostatni głos.

Architektura systemu również została stworzona w większości zgodnie z założeniami. System składa się z 3 modułów oraz bazy danych. Większość danych pobieranych jest z bazy za pomocą serwera i przesyłana do pozostałych modułów, a ruch pomiędzy modułami jest zabezpieczony. Każdy moduł działa zgodnie z przewidywaniami, a komunikacja między nimi przebiega sprawnie. Postarano się, aby aplikacja klienta była prosta w obsłudze i łatwo dostępna. Zrezygnowano z implementacji sieci miksującej, ponieważ według autora użycie zewnętrznej sieci miksującej, np. Tor nie wpływa na anonimowość głosowania. W momencie oddania głosu komisja nie wie, z którego urządzenia głos został oddany. Urządzenia nie są również na stałe przypisane do głosujących – użytkownik może oddać głos z dowolnego urządzenia podłączonego do sieci. Co więcej użycie zewnętrznej sieci może zwiększyć bezpieczeństwo w przypadku kiedy komisja chce wykorzystać swoją przewagę do manipulacji głosowaniem.

8.2. Zaistniałe problemy i sposób ich rozwiązania

Podczas procesu projektowania i implementacji systemu do głosowania autor odnalazł pewne trudności.

Pierwszą trudnością była możliwość umożliwienia zmiany głosu przy zachowaniu jego anonimowości. Ponieważ głosy miały być anonimowe nie powinno być możliwości łatwego przypisania głosu do głosującego przez co nie można było być pewnym czy tylko ostatni głos danej osoby zostanie policzony. Rozwiązano to za pomocą dodania kolumny EncryptedVoter w tabeli Votes. Kolumna ta przechowuje login głosującego zaszyfrowany kluczem tajnym głosowania. Dzięki temu nie można w łatwy sposób połączyć użytkownika z głosem, który oddał. Jednocześnie element publikujący jest w stanie rozszyfrować tę wartość i wziąć tylko ostatnie oddane głosy tych użytkowników.

Druga trudność wiązała się z przechowywaniem i przesyłaniem kluczy użytkowników i głosowań. Klucze te nie mogą być przechowywane w postaci jawnej. W związku z tym postanowiono zaszyfrować kolumny przechowujące te wartości w bazie danych, a klucz przechować bezpośrednio w kodzie programu. Ponieważ komunikacja pomiędzy modułami jest szyfrowana, nie ma problemu z przesyłaniem tajnych kluczy.

Kolejny problem wyniknął podczas testowania systemu. Polegał on na małej wydajności serwera związanej z jego dużym obciążeniem. W celu rozwiązania tego problemu przepisano sposób działania serwera, dodając mu obsługę wielu wątków. Stwierdzono również, że niektóre dane (np. lista opcji w danym głosowaniu) mogą być pobierane bez użycia serwera. Dodanie obsługi wielu wątków oraz zmniejszenie ilości danych pobieranych przez serwer usprawniła jego działanie.

Innym błędem, który został znaleziony podczas testowania był brak szyfrowania części wiadomości. Błąd ten wyniknął z faktu, że metoda wysyłająca te wiadomości nie wywoływała innej metody odpowiedzialnej za szyfrowanie tekstu. Pomyłka ta została szybko naprawiona.

Ostatnim błędem, o którym warto wspomnieć była możliwość dostępu do strony głosowania bez zalogowania się. Użytkownik był w stanie bez większych trudności przejść do stron zarezerwowanych tylko dla użytkowników zalogowanych. Aby rozwiązać ten problem autor dodał wymagania autoryzacji użytkowników w projekcie aplikacji klienta. Po zastosowaniu tej zmiany, przy próbie przejścia na stronę wymagającą zalogowania się sprawdzane jest ciasteczko uwierzytelniające. Jeżeli okaże się, że użytkownik nie jest zalogowany to zostanie przekierowany do strony logowania.

8.3. Wnioski i perspektywy rozwoju

Celem pracy inżynierskiej było zaprojektowanie i implementacja prostego systemu do głosowania elektronicznego. System ten pozwala na logowanie się użytkowników, ich rejestrację, dodawanie nowych głosowań oraz oddanie głosu w głosowaniu. Co więcej projektowany system nie jest skomplikowany, a użytkownicy nie powinni mieć problemu z dostępem do systemu lub jego obsługą. W celu zapewnienia bezpieczeństwa użytkowników użyte zostały standardy kryptograficzne w celu utajnienia komunikacji wewnątrz systemu.

Niniejsza praca jest opisem kolejnych etapów powstawania tego systemu, zaczynając od koncepcji architektury, przez implementowanie kończąc na testowaniu. Niektóre z założeń z fazy koncepcyjnej zostały odrzucone, ponieważ niepotrzebnie skomplikowałyby one działanie systemu. Można jednak stwierdzić, że główne założenia koncepcyjne zostały zrealizowane.

Powstały w tej pracy system może zostać ulepszony w celu jeszcze lepszego zapewnienia bezpieczeństwa oraz wygody użytkownika. Dzięki podziałowi systemu na moduły jego rozbudowa i modyfikacja nie powinna stwarzać wielu problemów. Jednym z kierunku rozwoju może być próba użycia kryptografii homomorficznej do głosowania, co pozwalałoby na dodawanie do siebie głosów przed odszyfrowaniem. Inną modyfikacją może być modyfikacja wszystkich modułów do zgodności z pracą w chmurze.

Podczas testowania systemu znaleziono jeszcze kilka innych aspektów, które można rozwinąć. Poniżej znajdują się najważniejsze z nich:

- możliwość dodania głosujących z listy wyświetlanej w sposób podobny do listy głosowań,
- możliwość edycji głosowania w czasie jego trwania,
- możliwość dodania głosującego do głosowania w czasie jego trwania,
- możliwość wyboru typu głosowania przy jego tworzeniu,
- możliwość wyboru liczby preferencji w głosowaniu,
- stworzenie aplikacji komórkowej sprzężonej z systemem,
- możliwość zarządzania kontem użytkownika,
- możliwość sprawdzenia całej historii głosów,
- integracja z systemami podpisu elektronicznego,
- możliwość sprawdzenia swojego ostatniego głosu po zakończonym głosowaniu.

Podczas prac nad tym systemem autor pogłębił swoją wiedzę dotyczącą systemów i algorytmów kryptograficznych. Autor poznał również nowe platformy i frameworki używane do tworzenia oprogramowania, napotkał problemy, których rozwiązanie przyczyniło się do

rozwoju umiejętności programistycznych autora oraz dostał lepszy wgląd do świata projektowania systemów informatycznych, co również należało do celów tej pracy.

Ostatecznie zaimplementowany system stanowi dobrą podstawę do modyfikacji i rozszerzania jego funkcjonalności. Pozwala on na przeprowadzenie głosowań w sposób bezpieczny i wygodny. Oczywiście jest kilka aspektów wymagających większego rozbudowania związanych głównie z wyglądem aplikacji klienta oraz jej możliwościami. Mimo to rezultaty, które uzyskano spełniają oczekiwania i można je uznać za zadowalające.

Bibliografia

1. Saltman R., *Effective use of computing technology in vote-tallying*, http://csrc.nist.gov/publications/nistpubs/NBS_SP_500-30.pdf, 1975;
2. *Internet voting in Estonia*, <https://www.valimised.ee/en/internet-voting/internet-voting-estonia>
3. Ryan P. Y. A., Bismark D., Heather J., Schneider S., Xia Z., *Prêt à Voter: a Voter-Verifiable Voting System*, 2009
4. *What is Verification of I-votes?*, http://vvk.ee/public/Verification_of_I-Votes.pdf
5. *What is Entity Framework?*, <http://www.entityframeworktutorial.net/what-is-entityframework.aspx>
6. *AES (ang. Advanced Encryption Standard)*, <http://www.cryptoit.net/pl/symetryczne/aes.html?tab=0>
7. Z. Kotulski: *Plansze do wykładów z przedmiotu PKRY*,
8. *Tor: Overview*, <https://www.torproject.org/about/overview.html.en>
9. *IEEE P1363: Standard Specifications For Public Key Cryptography*, <http://grouper.ieee.org/groups/1363/IBC/material/P1363.3-D1-200805.pdf> ,
10. *Standard ECMA-334 C# Language Specification*, <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf> ,
11. *Entity Framework Documentation*, [https://msdn.microsoft.com/en-us/library/ee712907\(v=vs.113\).aspx](https://msdn.microsoft.com/en-us/library/ee712907(v=vs.113).aspx) ,
12. *SQL Server Documentation*, <https://docs.microsoft.com/pl-pl/sql/sql-server/sql-server-technical-documentation> ,
13. *Funkcje programu Visual Studio*, <https://www.visualstudio.com/pl/vs/features/> ,
14. *Standard FIPS-197*, <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>

Spis rysunków

Rysunek 1 Przykład karty do głosowania w systemie Prêt à Voter przed i po głosowaniu.....	9
Rysunek 2 Rodzaje głosowań z zaznaczonym wyborem autora	11
Rysunek 3 Schemat architektury systemu	16
Rysunek 4 Schemat działania serwera	17
Rysunek 5 Schemat działania aplikacji klienta	18
Rysunek 6 Widok użytkownika niezalogowanego.....	18
Rysunek 7 Widok użytkownika zalogowanego.....	19
Rysunek 8 Schemat działania elementu publikującego.	19
Rysunek 9 Schemat relacji w bazie danych	20
Rysunek 10 Schemat działania sieci miksującej.	21
Rysunek 11 Schemat działania węzła sieci miksującej.....	21
Rysunek 12 Widok strony domowej.....	26
Rysunek 13 Strona logowania.	27
Rysunek 14 Strona wyboru głosowania.	28
Rysunek 15 Strona głosowania.	29
Rysunek 16 Strona zakończonego głosowania.....	30
Rysunek 17 Strona dodawania nowego głosowania.....	31
Rysunek 18 Strona rejestracji użytkownika.....	32
Rysunek 19 Connection string w pliku eVote.WebClient.SetParameters.xml	34
Rysunek 20 Witryna eVote w menedżerze usługi IIS.	35
Rysunek 21 Niezaszyfrowane połączenie TCP.....	38

Spis tabel

Tabela 1 Tabela S-BOX algorytmu AES	13
Tabela 2 Macierz używana w kroku miksowania kolumn algorytmu AES.....	14
Tabela 3 Tabele w bazie danych	20
Tabela 4 Pola klasy Poll.....	22
Tabela 5 Pola klasy Voter.....	22
Tabela 6 Pola klasy Vote	23
Tabela 7 Pola klasy VoteOption.....	23
Tabela 8 Metody statyczne klasy Message	24