

**WYŻSZA SZKOŁA KOMUNIKACJI I ZARZĄDZANIA
W POZNANIU**

Dawid Kaleta

Internetowy system rezerwacji biletów

Praca dyplomowa

Promotor: dr. inż. Jarosław Warczyński

Instytut: Wydział technologii informacyjnych

Kierunek: Informatyka

Specjalność: Technologie i zastosowanie Internetu

Poznań 2010

Spis treści

| | |
|--|----|
| 1. Wstęp..... | 2 |
| 1.1. Wprowadzenie..... | 2 |
| 1.2. Cel i zakres pracy..... | 3 |
| 1.2.1. Przegląd istniejących rozwiązań..... | 3 |
| 1.2.2. Język i narzędzia programistyczne | 4 |
| 1.2.3. Zakres prac obejmujący zagadnienia szczegółowe | 4 |
| 1.3. Charakterystyka źródeł..... | 4 |
| 2. Przegląd istniejących rozwiązań | 6 |
| 2.1. Multikino..... | 6 |
| 2.2. Cinema City..... | 7 |
| 3. Opis języków i narzędzi programistycznych przydatnych do konstrukcji aplikacji..... | 9 |
| 3.1. Technologia ASP..... | 9 |
| 3.2. Język programowania C#..... | 11 |
| 3.3. Visual Studio..... | 13 |
| 3.4. Microsoft SQL Server 2005..... | 19 |
| 4. Projekt aplikacji..... | 23 |
| 4.1. Założenia projektowe | 23 |
| 4.2.1. Funkcjonalność systemu | 24 |
| 4.2.3. Działanie systemu | 27 |
| 4.2.4. Projekt bazy danych..... | 31 |
| 5. Konstrukcja aplikacji..... | 36 |
| 5.1 Dostęp do danych..... | 36 |
| 5.2. Logika aplikacji | 37 |
| 5.3. Warstwa prezentacji..... | 51 |
| 5.3.1. Formularze użytkownika..... | 51 |
| 5.3.2. Formularze administratora..... | 57 |
| Testy aplikacji..... | 60 |
| 7. Instalacja..... | 63 |
| 8. Podsumowanie..... | 65 |
| Literatura..... | 66 |
| Załączniki..... | 67 |

1.Wstęp

1.1. Wprowadzenie

Internet jest ogromnym medium informacyjnym, dzięki któremu, zanikają bariery narodowościowe i kulturowe, informacje mogą wymieniać ludzie z odległych krańców świata. Trudno jest określić ilość usług dostępnych w Internecie a ich ilość stale wzrasta. Do jednych z najważniejszych z pewnością należy usługa WWW, dzięki której można w atrakcyjny sposób promować różnego rodzaju instytucje czy przedsiębiorstwa, docierając do szerokiego grona odbiorców. Rozwój Internetu stwarza wiele możliwości w konkretnych branżach handlowo-usługowych, niesie ze sobą zapotrzebowanie na informatyczne narzędzia ułatwiające codzienną pracę. Dlatego powstał pomysł stworzenia aplikacji, ułatwiającej użytkownikowi korzystanie z oferty instytucji rozrywkowej, jaką jest kino. Posiadanie przez kino witryny internetowej, dzięki której, może promować swoje zdarzenia kulturalne, jest praktyką coraz częściej stosowaną i odbierana przez społeczeństwo pozytywnie.

Internetowe systemy rezerwacji to znaczna oszczędność czasu i pieniędzy, raz zainwestowane środki szybko się zwracają. System może służyć przez lata ulegając modernizacji. Dodatkową zaletą jest niewielka liczba osób obsługująca witrynę. W praktyce, jeden administrator może zarządzać portalem, co powoduje ograniczenie kosztów i ogranicza błędy w administracji związane z rozbudowaną infrastrukturą.

Internetowe systemy rezerwacji ułatwiają pracę umożliwiając rezerwację miejsc z dowolnego punktu, w którym mamy dostęp do Internetu. Systemy rezerwacji internetowej działają w czasie rzeczywistym, dzięki temu, użytkownik unika problemów z podwójnymi rezerwacjami biletów na seanse kinowe.

Posiadanie przez kino strony internetowej pozwala nawiązać kontakt z szeroką gamą klientów, którym łatwiej dokonać wyboru po wcześniejszym zapoznaniu się z ofertami widocznymi w portalu.

Jeżeli system rezerwacji działa w modelu ASP, to po stronie klienta nie jest wymagana jest żadna instalacja, wszystko dostępne jest z poziomu przeglądarki internetowej, dzięki temu nie obciążamy użytkownika dodatkowymi problemami związanymi z czasem instalacji dodatkowego oprogramowania i potrzebną do tego wiedzą. Ze strony klienta cały proces rezerwacji składa się wykonania kilku prostych kroków, w których podaje interesujące go daty, seanse i miejsca na sali. Dlatego strony powinny być funkcjonalne, oryginalne. Za pomocą kolorowych szablonów, dowolnych podgrup łatwiej wypromować najciekawsze oferty.

Rezerwacja biletów kinowych za pośrednictwem Internetu pozwala na zajęcie najlepszych miejsc bez konieczności wizyty w kinie. Umożliwia wygodne zapoznanie się z repertuarem i zaplanowanie udanej wizyty. W dzisiejszym świecie gdzie coraz częściej liczy się oszczędność czasu takie rozwiązania stają się coraz bardziej potrzebne i zyskują rzesze zwolenników.

1.2. Cel i zakres pracy

Głównym celem pracy jest stworzenie systemu rezerwacji biletów dla kina, który umożliwi użytkownikowi zapoznanie się z repertuarem, a po zalogowaniu, wybór filmu i rezerwację biletów poprzez formularz na stronie internetowej kina. Jest to rozwiązanie które, cieszy się obecnie dużą popularnością, charakteryzuje je łatwy dostęp, krótki czas wyboru i uniknięcie problemu braku biletów z wybranego repertuaru.

W ramach pracy zostanie stworzona strona internetowa, na której będą się znajdowały, dostępne w danym czasie filmy i ich opis, ceny biletów, można będzie wybrać film i godzinę seansu po wcześniejszym zalogowaniu się w oknie logowania. Jeśli dana osoba nie posiada swojego konta zostanie poproszona o wypełnienie formularza rejestracji.

Zakres pracy obejmuje następujące zadania szczegółowe:

1. Przegląd istniejących rozwiązań. Celem tego zadania jest zapoznanie się jak działają podobne rozwiązania dostępne na rynku. Zostanie przeanalizowany sposób działania popularnych systemów rezerwacji biletów Multikina i Cinema City.
2. Opis narzędzi programistycznych przydatnych do konstrukcji aplikacji. Scharakteryzowany zostanie język programowania C#, oraz rozbudowane narzędzie programistyczne Microsoft Visual Web Developer Express 2008, użyte do tworzenia aplikacji a także język baz danych SQL jak i środowisko SQL Server 2005.
3. Projekt aplikacji. Zadanie to obejmuje swoim zakresem specyfikacje funkcjonalności poszczególnych modułów. Wykonanie schematu bazy danych w postaci graficznej zawierającej definicje danych, oraz opis tabel i relacji. Sporządzenie modelu projektowanego systemu. Przedstawiona zostanie architektura systemu.
4. Konstrukcja aplikacji. Celem zadania jest implementacja projektu aplikacji w technologii ASP. Prace obejmować będą wykonanie bazy danych stanowiącej model

dziedzinowy aplikacji. Opracowanie warstwy dostępu do danych wykorzystując między innymi Microsoft .NET Framework w wersji 3.5. Implementacja warstwy logiki aplikacji. Ostatnim zadaniem będzie napisanie interfejsu użytkownika końcowego, bazującego na technologii ASP.

5. Opis testów i sprawdzenie zachowania aplikacji podczas działania, przy wprowadzaniu i odczytywaniu danych.

1.3. Charakterystyka źródeł

Informację na temat działalności i historii kin Multikino i Cinema City, zostały zaczerpnięte ze stron: Multikino <http://www.multikino.pl/poznan/tekst,pokaz,18.html> [6] oraz <http://www.cinemacity.pl/index.php?module=cinema&action=info&cid=1078&id=4>, dla Cinema City[1].

Wiedza na temat platformy NET. w której, uruchamia się aplikację oraz przegląd komponentów .NET Framework które, wspomagają budowę i uruchamianie aplikacji opartych na technologii .NET, zaczerpnięta została z książki Andrzeja Stefańczyka *Sekrety języka C#* [7]. W tym tytule, przedstawione zostały również zagadnienia związane z tworzeniem projektu, kompilacją programu, pracą z rozbudowanymi oknami narzędziowymi, które zilustrowano pomocnymi slajdami.

Niezbędne informacje na temat historii, składni, podstaw języka i komponentów wchodzących w skład języka programowania C#, zostały zaczerpnięte z książki Marcina Lisa *C# ćwiczenia*[5] oraz Wiktora Zychła *Programowanie pod Windows*[9] .

Do budowy samej aplikacji jak i właściwego doboru komponentów, bardzo przydatna okazała się wiedza nabyta na uczelni, jak również informacje zdobyte po przeczytaniu, obfitującej w przykłady, bardzo przydatnej książki Pawła Chłosty *Ćwiczenia z ASP.NET i kolekcje C#*[2].

Podczas projektowania, tworzenia i pracy z bazą danych SQL niezwykle pomocna okazała się wiedza nabyta podczas ćwiczeń i wykładów odbywających się na uczelni, poszerzona po przeczytaniu książki panów Richarda Waymire i Ricka Sawtella *SQL Server 2000 dla każdego*[8] i artykułów pana Grzegorza Chuchra *Kurs SQL* [3]. Natomiast techniczny pomysł na połączenie z bazą danych, formułowanie zapytań oraz wyświetlanie

danych i pracę z komponentami do obsługi języka baz danych SQL zaczerpnięto z kursów Piotra Gaszewskiego *Kurs ASP.NET 2* [4].

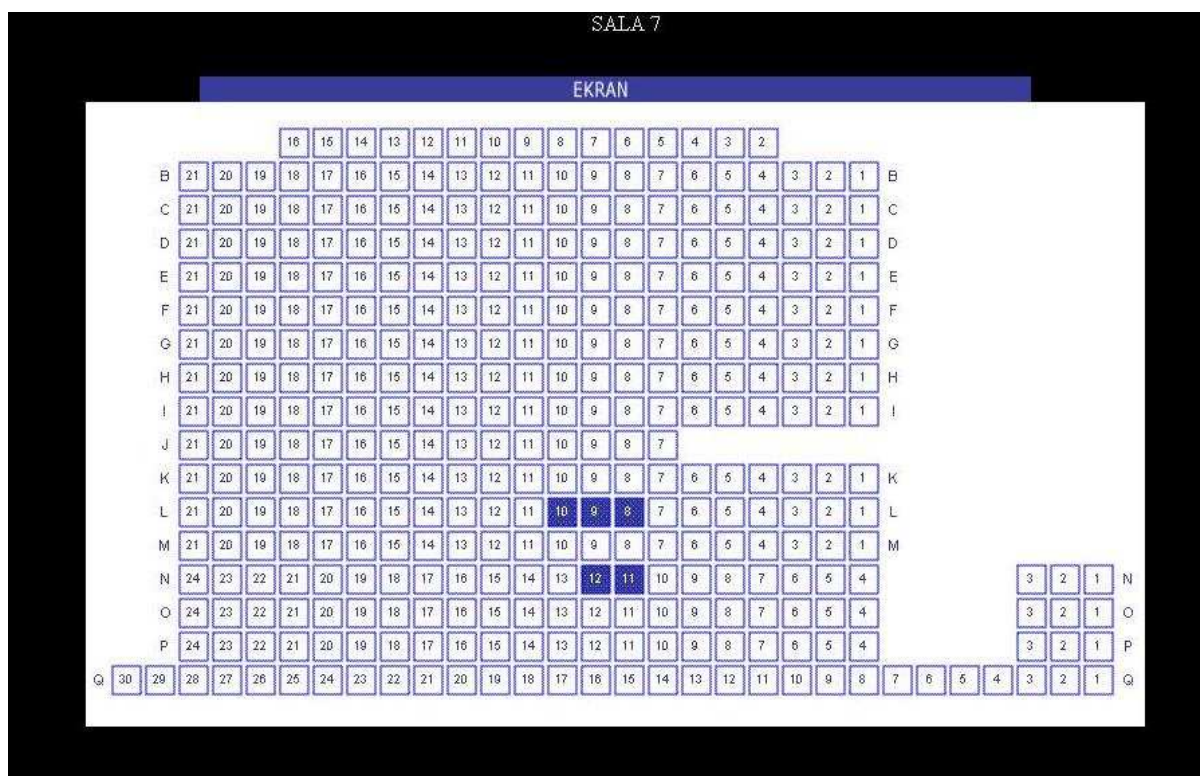
2. Przegląd istniejących rozwiązań

Obecnie, sieci kin lub większe kina, starają się ułatwić klientowi dostęp do ich repertuaru, promocji, zareklamować swoje usługi, dlatego większość z nich posiada swoje strony internetowe, czasami nawet bardzo rozbudowane z wieloma funkcjami dotyczącymi wyboru repertuaru, ale również możliwością komentowania filmów. Na stronach tych można również znaleźć oferty pracy.

2.1 Multikino

Spółka Multikino jest jednym z największych operatorów kin w Polsce. W sieci działa obecnie 21 nowoczesnych multipleksów w największych miastach w kraju. Spółka powstała w 1996 rok, jako wspólne przedsięwzięcie pomiędzy UCI – jednym z największych na świecie operatorów multipleksów i ITI – liderem na polskim rynku mediów. W 1998 roku otworzono w Poznaniu pierwszy w Polsce multipleks. W roku 2004 grupa ITI przejęła udziały w spółce Multikino od UCI i stała się jedynym udziałowcem firmy. Rok 2007 Multikino wprowadziło projekcje cyfrowe w technologii Dolby 3D Digital Cinema. W roku 2008 spółka stała się liderem na rynku, Multikino połączyła się z Apollo Silver Screen World Cinemas. Multikino to nie tylko filmy ale to różnego rodzaju imprez i rozrywek : koncerty, transmisje wydarzeń sportowych, festiwale oraz imprezy okolicznościowe.

Witryna sieci Multikino oferuje możliwość rezerwacji miejsc na wybrany film. Na stronie Multikina w zakładce repertuar można zobaczyć oferowane obecnie przez sieć tytuły filmów, przy których widoczne są godziny seansów a w górnej części strony istnieje możliwość wybrania dnia tygodnia, na który planowana jest rezerwacja. Po kliknięciu na godzinę seansu przy wybranym filmie, zostaje generowany podgląd sali z numeracją miejsc i rzędów. Po wybraniu miejsca lub miejsc, zmienia się kolor kwadracika symbolizującego siedzenie, następnie klient, za pomocą komunikatu jest proszony o wypełnienie formularza danych kontaktowych i ostateczne potwierdzenie rezerwacji.[8]



Rysunek 2.1. Przedstawia podgląd sali Multikina.

2.2. Cinema City

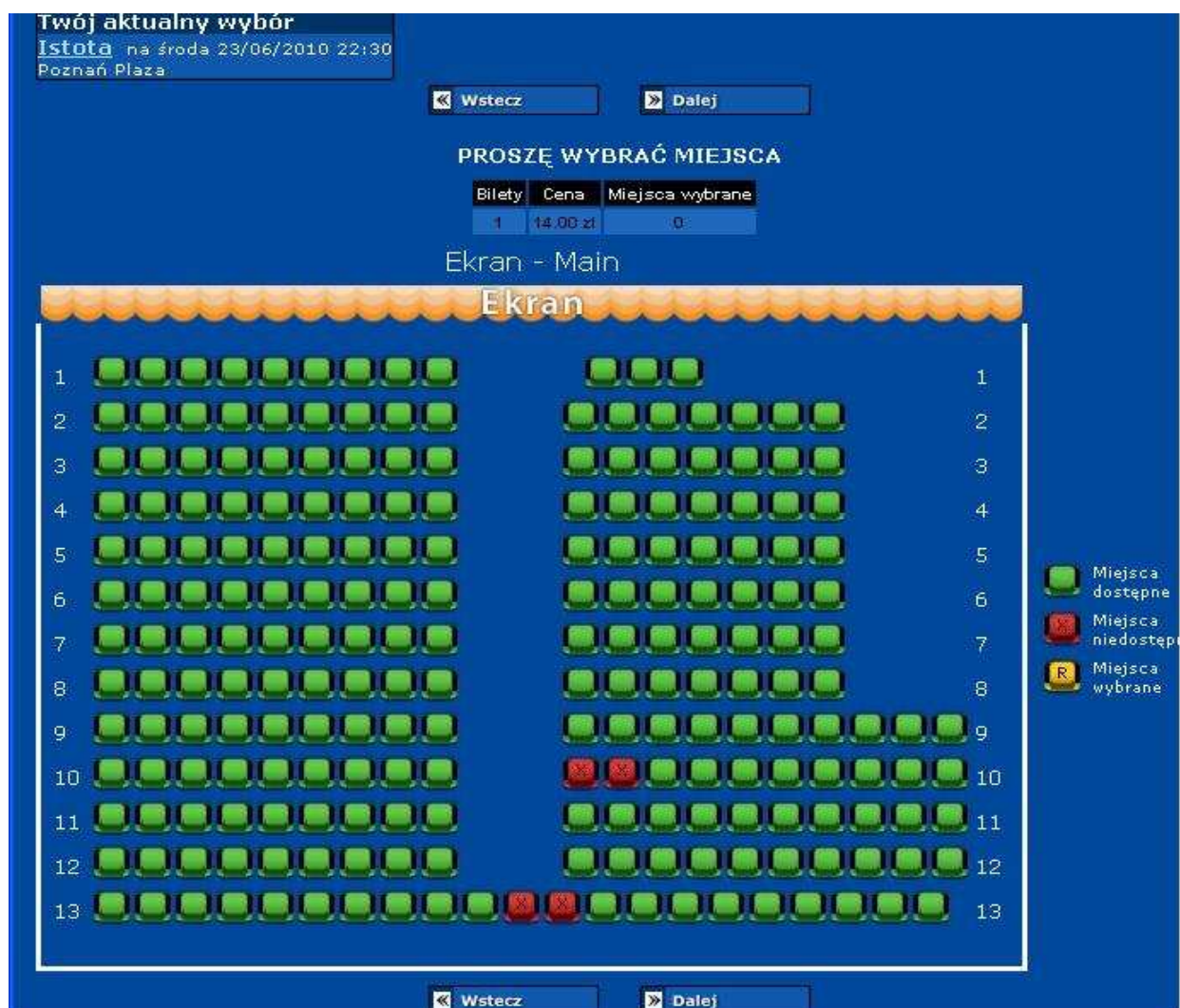
Cinema City International jest największym w Europie Środkowo-Wschodniej i Izraelu operatorem kin wielosalowych, posiada swoje kina w Polsce, Czechach, Bułgarii i na Węgrzech. Poznańskie Cinema City Kineapolis jest w Polsce dziewiętnastą inwestycją międzynarodowej firmy Cinema City, tworzącej sieć kin - multipleksów. Jest to największe kino w Polsce posiadające 18 sal kinowych, które mogą jednocześnie pomieścić ponad 6500 widzów, mają pojemność od 208 do 750 miejsc. Wszystkim salom przyznano certyfikat Georga Lucasa THX, który zapewnia:

- doskonały obraz
- najlepszej klasy dźwięk Dolby Digital Surround EX,
- najwygodniejsze fotele kinowe z podwójnymi podłokietnikami
- układ widowni o optymalnym przewyższeniu gwarantujący idealną widoczność

Obiekt jest w pełni klimatyzowany, oraz przystosowany do obsługi osób niepełnosprawnych. Dla kierowców dostępny jest strzeżony parking, dysponujący miejscami

na 1500 samochodów oraz parking przeznaczony dla autobusów przywożących grupy zorganizowane.

Na stronie Cinema City wybierając zakładkę repertuar kina jest dostępna lista filmów aktualnie wyświetlanych, przy których znajdują się godziny wyświetlania danego filmu. Po wyborze godziny, która wydaje się najbardziej odpowiednia, klient zostaje poproszony o zgodę na przetwarzanie danych osobowych na potrzeby kina. Następnie wybiera typ biletu: normalny, studencki lub ulgowy i ilość miejsc które chce zarezerwować. Kolejnym krokiem jest wybranie miejsc na wygenerowanym podglądzie sali, program prosi o wybranie tyłu miejsc ile zadeklarowaliśmy wcześniej. Istnieje też możliwość automatycznego wyboru, gdzie program zrobi to za nas. Na koniec wypełniamy formularz z naszymi danymi kontaktowymi.[1]



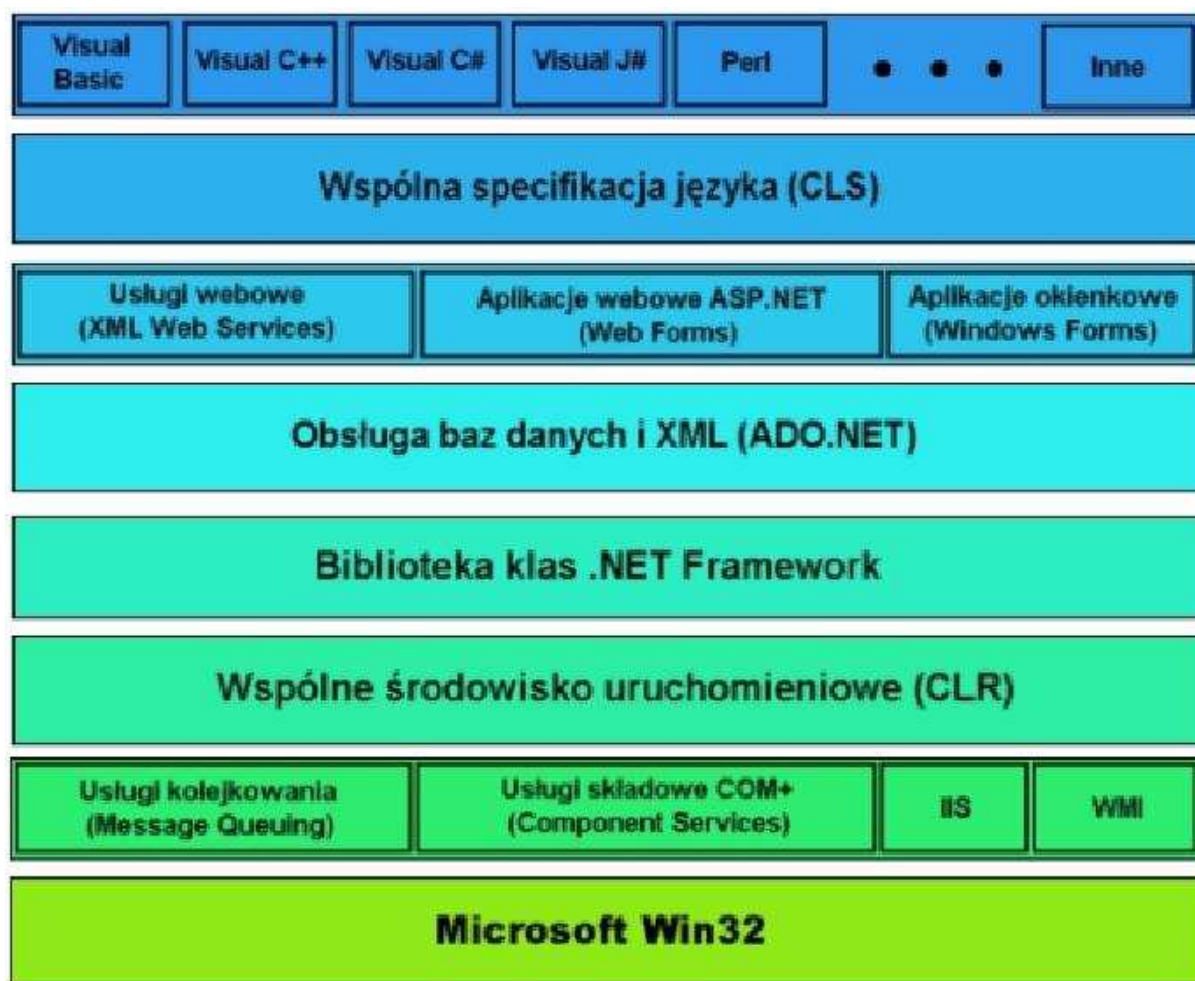
Rysunek 2.2. Przedstawia podgląd sali Cinema City.

3. Opis narzędzi programistycznych przydatnych do konstrukcji aplikacji

3.1. Technologia ASP

Obecnie dzięki takim technologiom jak ASP, która jest częścią platformy Microsoft .NET możemy stosunkowo łatwo stworzyć aplikację internetową, komunikującą się z użytkownikiem za pośrednictwem formularzy a dzięki zastosowaniu programowania zdarzeniowego, możliwe jest oddzielenie kodu aplikacji od warstwy prezentacji, co zwiększa czytelność oraz daje możliwość wielokrotnego wykorzystania kodu. Zaletą szybkiego i znacznie uproszczonego sposobu tworzenia zaawansowanych aplikacji internetowych, gdzie część kodu dotyczącego wizualizacji i rozmieszczenia kontrolek generowana jest automatycznie, spowodowała docenienie i znaczny wzrost popularności technologii ASP.NET, będąca częścią platformy Microsoft .NET[4].

.NET Framework jest platformą programistyczną, stworzoną przez firmę Microsoft w której skład wchodzi środowisko uruchomieniowe oraz biblioteki klas. Zadaniem .NET Framework jest zarządzanie przeróżnymi elementami systemu takimi jak kod aplikacji, pamięć, zabezpieczenia. Do tej pory obsługa tych zagadnień zajmowała programistom najwięcej czasu. W środowisku tym można tworzyć oprogramowanie działające po stronie serwera oraz pracujące na systemach, na które istnieje działająca implementacja tej platformy. Aplikacje stworzone na bazie ASP.NET są niezależne od systemu operacyjnego po stronie klienta, jak również od przeglądarki internetowej której on używa. Aplikacja webowa składa się z elementów takich jak : stron webowych z rozszerzeniem .aspx, zawierających interfejs użytkownika. Plików z kodem zawierających funkcjonalność po stronie serwera z rozszerzeniem .aspx.cs. Plików konfiguracyjnych Web.config. Pliku global.aspx w którym, znajduje się kod od obsługi zdarzeń aplikacji zgłoszonych przez ASP.NET, oraz połączeń z bazą danych[7].



Rysunek 3.1. przedstawia architekturę platformy .NET Framework.

CLR (Common Language Runtime) czyli wspólne środowisko uruchomieniowe zostało stworzone w celu zintegrowania wielu języków programowania w znacznym stopniu ułatwia tworzenie aplikacji gdyż nadzoruje proces wykonywania kodu. Kod zostaje skompilowany do kodu pośredniego, a nie kodu procesora[7].

Technologia ASP.NET służy do tworzenia w pełni dynamicznych stron internetowych, wykorzystującą środowisko uruchomieniowe CLR oraz wielką funkcjonalność platformy .NET Framework. Głównym celem zespołu, który był odpowiedzialny za stworzenie ASP.NET, było umożliwienie programistom szybkiego i prostego tworzenia rozbudowanych serwisów internetowych. Zastosowanie programowania zdarzeniowego umożliwiło oddzielenie kodu aplikacji, tworzonego w językach programowania, takich jak Visual Basic czy C#, od warstwy prezentacji. Dało to programistom możliwość, zależnie od potrzeb, wielokrotnego wykorzystania tego kodu a zautomatyzowany sposób jego formatowania znacznie zwiększył jego czytelność.

3.2. Język programowania C#

Język C# został stworzony przez programistów firmy Microsoft, głównym jego twórcą jest Anders Hejlsberg, osoba odpowiedzialna za powstanie Delphi firmy Borland, wywodzi się z rodziny języków C, choć zawiera element Javy.

Kod programu w C# jest zbiorem klas, podobnie tak jak w Javie, hierarchia dziedziczenia opiera się na istnieniu jednej klasy - object (System.Object), która jest elementem nadrzędnym tej hierarchii. Całe bogactwo i możliwości tego języka opierają się na występowaniu dużej, gotowej liczby klas z których można dziedziczyć, oraz możliwości definiowania swoich własnych, co pozwala na pisanie najbardziej zaawansowanych programów i aplikacji internetowych.

C# jest językiem w pełni obiektowym posiada mechanizmy odzyskiwania pamięci oraz obsługę wyjątków, współdziała ze środowiskiem uruchomieniowym .NET. W skład języka wchodzi: klasy, interfejsy, delegacje, przestrzenie nazw, właściwości, indeksatory, zdarzenia, atrybuty, operatory przeciążania, wsparcie dla wywołań kodu niebezpiecznego oraz mechanizmy generacji dokumentów XML[5].

Wykaz słów kluczowych języka C#, czyli takich które mają zastrzeżoną nazwę oraz specjalne znaczenie w danym języku programowania[7].

| | | | | |
|-----------------|-----------------|-----------------|----------------|------------------|
| <i>abstract</i> | <i>as</i> | <i>base</i> | <i>bool</i> | <i>break</i> |
| <i>byte</i> | <i>case</i> | <i>catch</i> | <i>char</i> | <i>checked</i> |
| <i>class</i> | <i>const</i> | <i>continue</i> | <i>decimal</i> | <i>default</i> |
| <i>delegate</i> | <i>do</i> | <i>double</i> | <i>else</i> | <i>enum</i> |
| <i>event</i> | <i>explicit</i> | <i>extern</i> | <i>false</i> | <i>finally</i> |
| <i>fixed</i> | <i>float</i> | <i>for</i> | <i>foreach</i> | <i>goto</i> |
| <i>if</i> | <i>implicit</i> | <i>in</i> | <i>int</i> | <i>interface</i> |

| | | | | |
|-----------------|---------------|-------------------|------------------|------------------|
| <i>internal</i> | <i>is</i> | <i>lock</i> | <i>long</i> | <i>namespace</i> |
| <i>new</i> | <i>null</i> | <i>object</i> | <i>operator</i> | <i>out</i> |
| <i>override</i> | <i>params</i> | <i>private</i> | <i>protected</i> | <i>public</i> |
| <i>readonly</i> | <i>ref</i> | <i>return</i> | <i>sbyte</i> | <i>sealed</i> |
| <i>short</i> | <i>sizeof</i> | <i>stackalloc</i> | <i>static</i> | <i>string</i> |
| <i>struct</i> | <i>switch</i> | <i>this</i> | <i>throw</i> | <i>true</i> |
| <i>try</i> | <i>typeof</i> | <i>uint</i> | <i>ulong</i> | <i>unchecked</i> |
| <i>unsafe</i> | <i>ushort</i> | <i>using</i> | <i>virtual</i> | <i>volatile</i> |
| <i>void</i> | <i>while</i> | | | |

Tabela 3.2. Słowa kluczowe występujące w języku c#

| Słowo kluczowe | Alias do struktury | Znaczenie |
|----------------|----------------------|---|
| <i>sbyte</i> | <i>System.SByte</i> | liczba całkowita ze znakiem z przedziału od -128 do 127 |
| <i>byte</i> | <i>System.Byte</i> | liczba całkowita bez znaku z przedziału od 0 do 255 |
| <i>short</i> | <i>System.Int16</i> | liczba całkowita ze znakiem z przedziału od -32 768 do 32 767 |
| <i>ushort</i> | <i>System.UInt16</i> | liczba całkowita bez znaku z przedziału od 0 do 65 535 |
| <i>int</i> | <i>System.Int32</i> | liczba całkowita ze znakiem z przedziału |

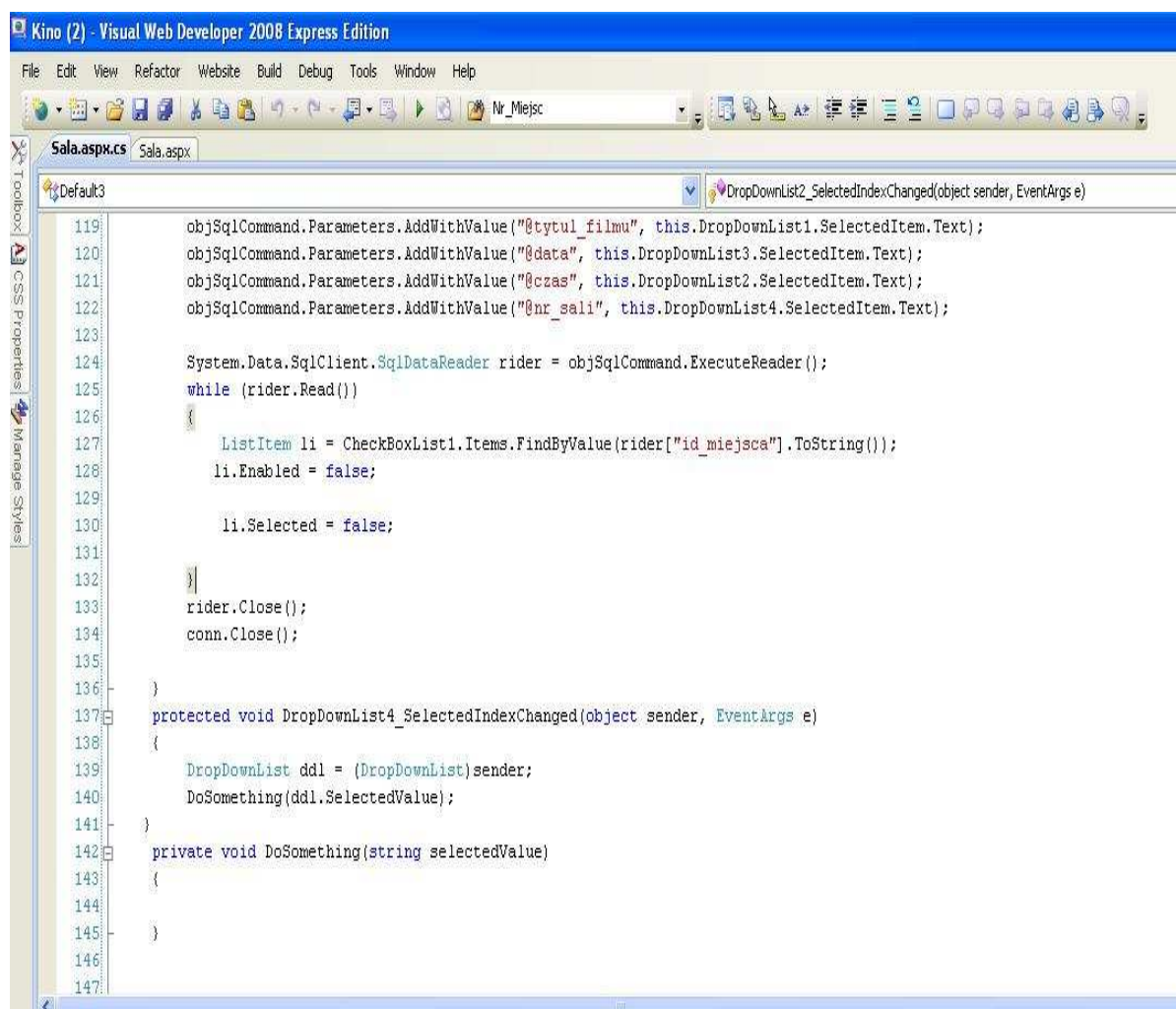
Tabela 3.3. Opisuje słowa kluczowe i zakres dozwolonych wartości

| | | |
|----------------|-----------------------|--|
| <i>uint</i> | <i>System.UInt32</i> | liczba całkowita bez znaku z przedziału od 0 do 4 294 967 295 |
| <i>long</i> | <i>System.Int64</i> | liczba całkowita ze znakiem z przedziału od -9 223 372 036 854 775 808 do 9 223 372 036 854 775 807 |
| <i>ulong</i> | <i>System.UInt64</i> | liczba całkowita bez znaku z przedziału od 0 do 18 446 744 073 709 551 615 |
| <i>char</i> | <i>System.Char</i> | znak, wartości można wprowadzać w postaci znaku (np. 'A'), sekwencji szesnastkowej (np. '\x0065') lub sekwencji Unicode (np. '\u0065') |
| <i>float</i> | <i>System.Single</i> | liczba zmiennoprzecinkowa z przedziału od $1.5 \cdot 10^{-45}$ do $3.4 \cdot 10^{38}$ |
| <i>double</i> | <i>System.Double</i> | liczba zmiennoprzecinkowa z przedziału od $5.0 \cdot 10^{-324}$ do $1.7 \cdot 10^{308}$ |
| <i>bool</i> | <i>System.Boolean</i> | wartość logiczna przyjmuje wartości literalne <i>true</i> (prawda) lub <i>false</i> (fałsz) |
| <i>decimal</i> | <i>System.Decimal</i> | liczba o dużej dokładności z przedziału od $1.0 \cdot 10^{-28}$ do $7.9 \cdot 10^{28}$ |

Tabela 3.3. Opisuje słowa kluczowe i zakres dozwolonych wartości

3.3 Visual Studio

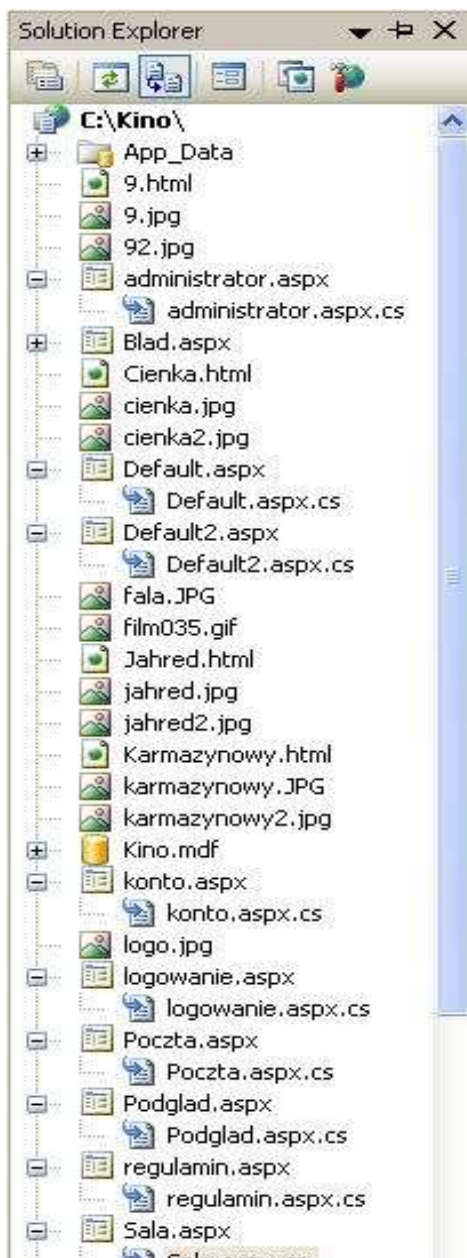
Microsoft Visual Studio to potężne narzędzie programistyczne, wspierające tworzenie programowania w językach takich jak Visual Basic, Visual J# , C#. W skład środowiska wchodzi bardzo rozbudowany edytor z szeregiem pomocnych funkcji jak podświetlanie składni, system podpowiedzi, wyszukiwania wyrazów. Środowisko można konfigurować i dopasowywać do potrzeb użytkownika. Wszystko odbywa się w głównym oknie gdzie u góry widać podstawowy pasek narzędziowy, po otwartych dokumentach aplikacji można się poruszać jak po zakładkach. Istnieje możliwość zakotwiczenia okien narzędziowych w dowolnym miejscu okna głównego, jeśli taki widok jest mało przejrzysty należy użyć automatycznego ukrywania.



Rysunek 3.4. Okno główne środowiska Microsoft Visual Studio

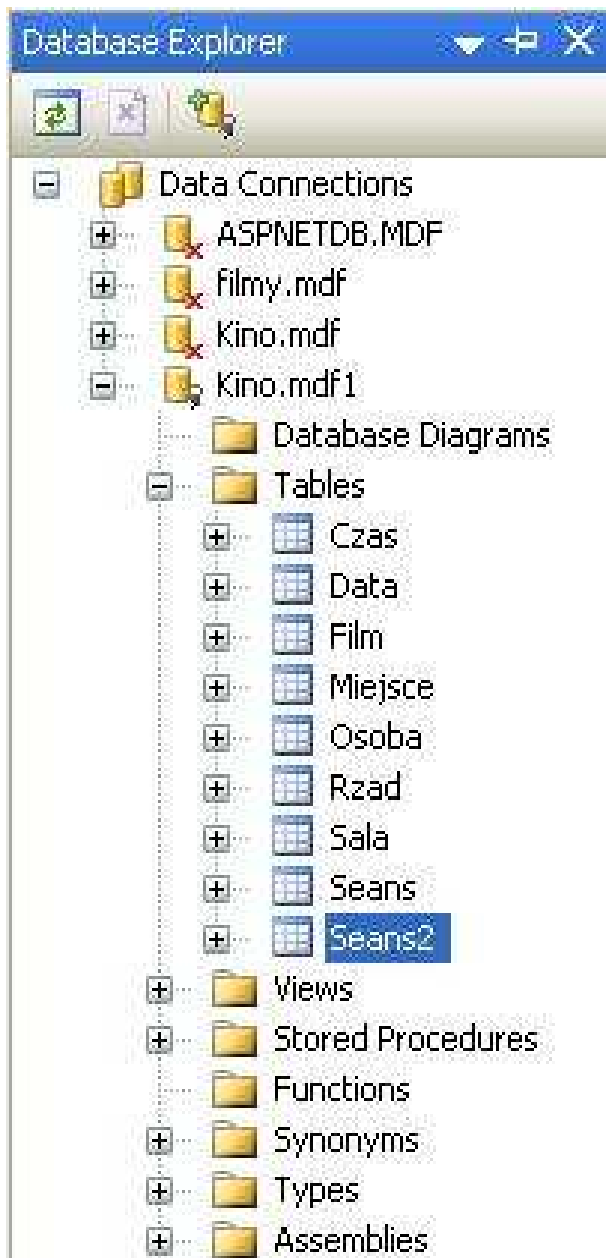
Właściwe korzystanie z okien narzędziowych, ich dopasowanie i rozmieszczenia, może bardzo ułatwić pracę. Oto kilka z podstawowych, najważniejszych okien narzędziowych:

Solution Explorer – w oknie tym widoczne są projekty i pliki należące do danego projektu, widok w postaci drzewa. Dwukrotne kliknięcie myszka na wybrany plik powoduje otwarcie go w trybie edycji, zaś użycie prawego przycisku uruchamia menu podręczne, pozwalające na wykonanie dodatkowych czynności jak zmiana nazwy, skopiowanie, czy usunięcie pliku.



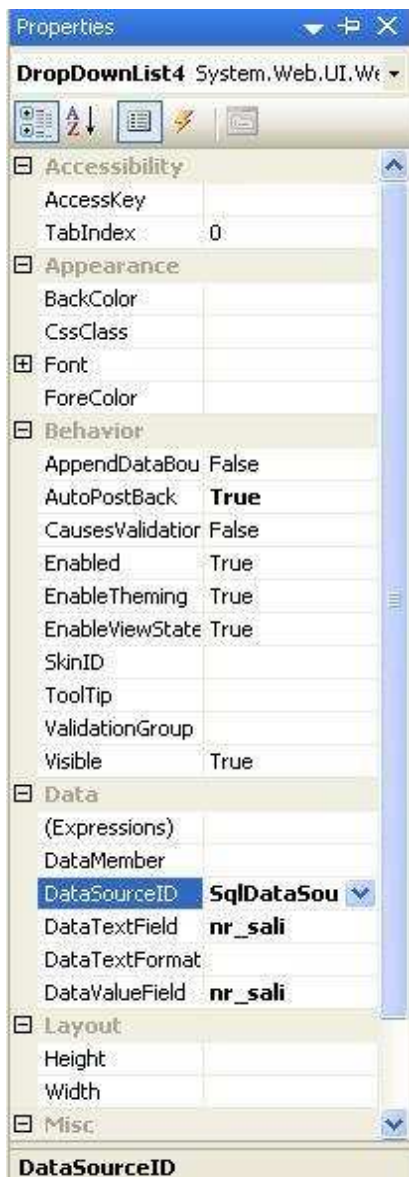
Rysunek 3.5. Okno narzędziowe Solution Explorer.

Database Explorer – okno wyświetlające listę baz danych z którymi po kliknięciu na wybraną można się połączyć. Po połączeniu, klikając na znak plus widoczna staje się lista tabel występujących w danej bazie. Najeżdżając na ikonkę tabeli, używając prawego klawisza mamy do wyboru opcje: dodawania nowej tabeli, kasowania, budowania nowego zapytania, otwierania tabeli w celu wprowadzania danych lub zmieniania nazw czy typów danych.



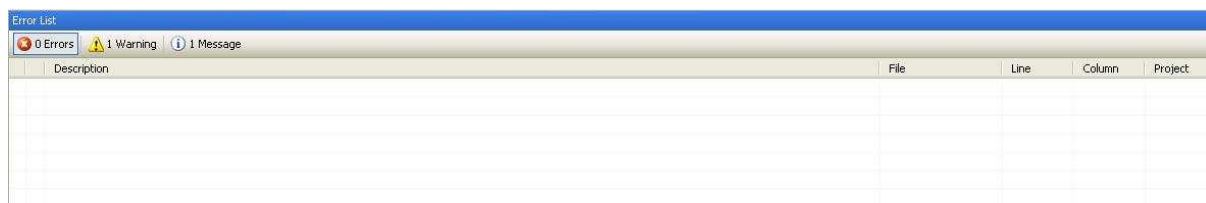
Rysunek 3.6. Okno narzędziowe Database Explorer.

Properties – wyświetla i służy do zmiany właściwości elementów. Właściwości jakie można konfigurować w tym oknie zależą od elementu który został wybrany do edycji. W przypadku gdy nie mam zaznaczonego elementu na formatce, okno nie wyświetla niczego. Okna używa się najczęściej do edytowania kontrolek np. przycisków, etykiet itp. Ale również do konfiguracji i ustawiania skąplikowanych parametrów kontrolek, służących do współpracy z bazami danych takich jak: DetailsView, czy GridView.



Rysunek 3.4. Okno narzędziowe Properties.

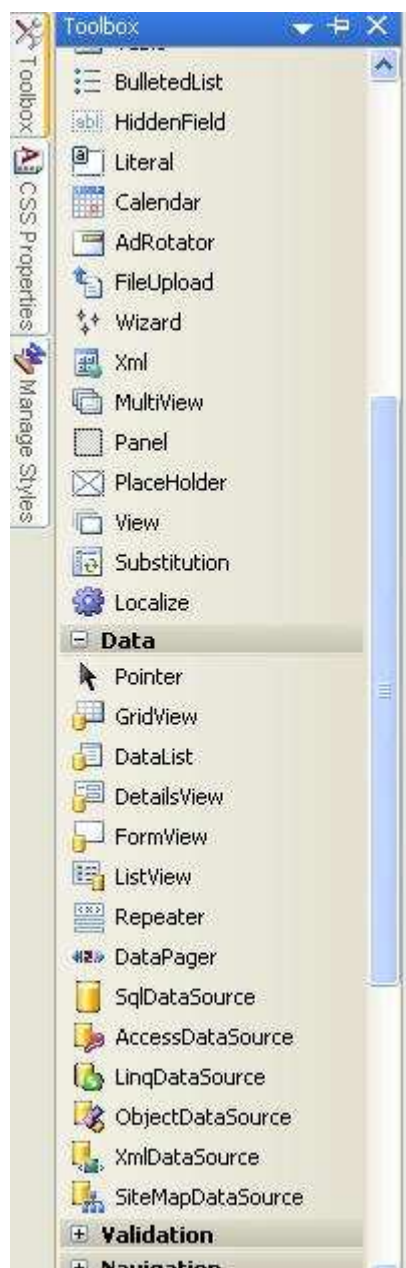
Error List – wyświetla listę błędów, ostrzeżeń oraz komunikaty o jakich informuje kompilator. Podwójne kliknięcie na element z listy powoduje przejście do linii kodu w której, prawdopodobnie pojawił się błąd.



Rysunek 3.7. Okno informacyjne Error List.

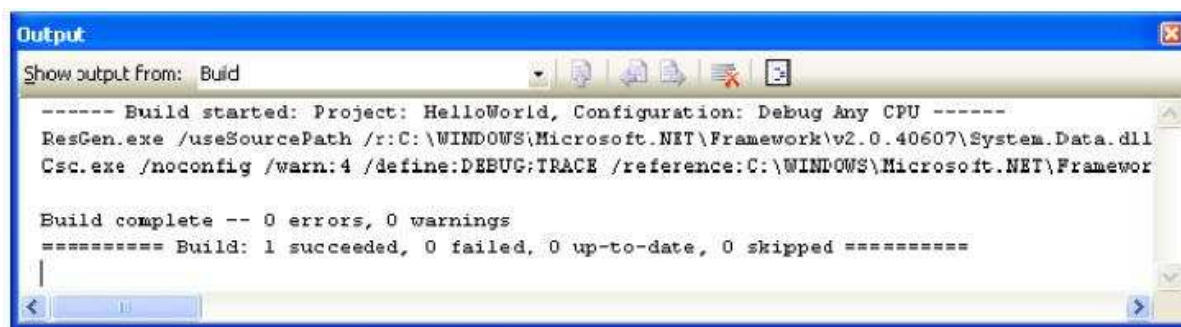
Toolbox – Jedna z najważniejszych okienek, zawiera listę gotowych komponentów

interfejsowych, które za pomocą metody pociągnij i upuść, mogą być łatwo dodawane do projektu. Zapobiega to żmudnemu pisaniu wielu linijek kodu. Komponenty podzielone są na grupy tematyczne co znacznie ułatwia wyszukiwanie.



Rysunek 3.8. Okno narzędziowe Toolbox.

Output – Okno wyświetla pomocne informacje o statusie kompilowanego projektu jak i informacje w czasie debugowania[7].



Rysunek 3.9. Okno narzędziowe Output.

Aby utworzyć nowy projekt uruchamiamy Visual Studio z zakładki File wybieramy *New File* – np. gdy ma zostać utworzona strona w języku HTML lub XML. W przypadku gdy chcemy stworzyć aplikację internetową wybieramy *New Web Site*, gdzie możemy, wybrać miejsce położenia plików, nazwę aplikacji i język w którym będziemy ją pisać.

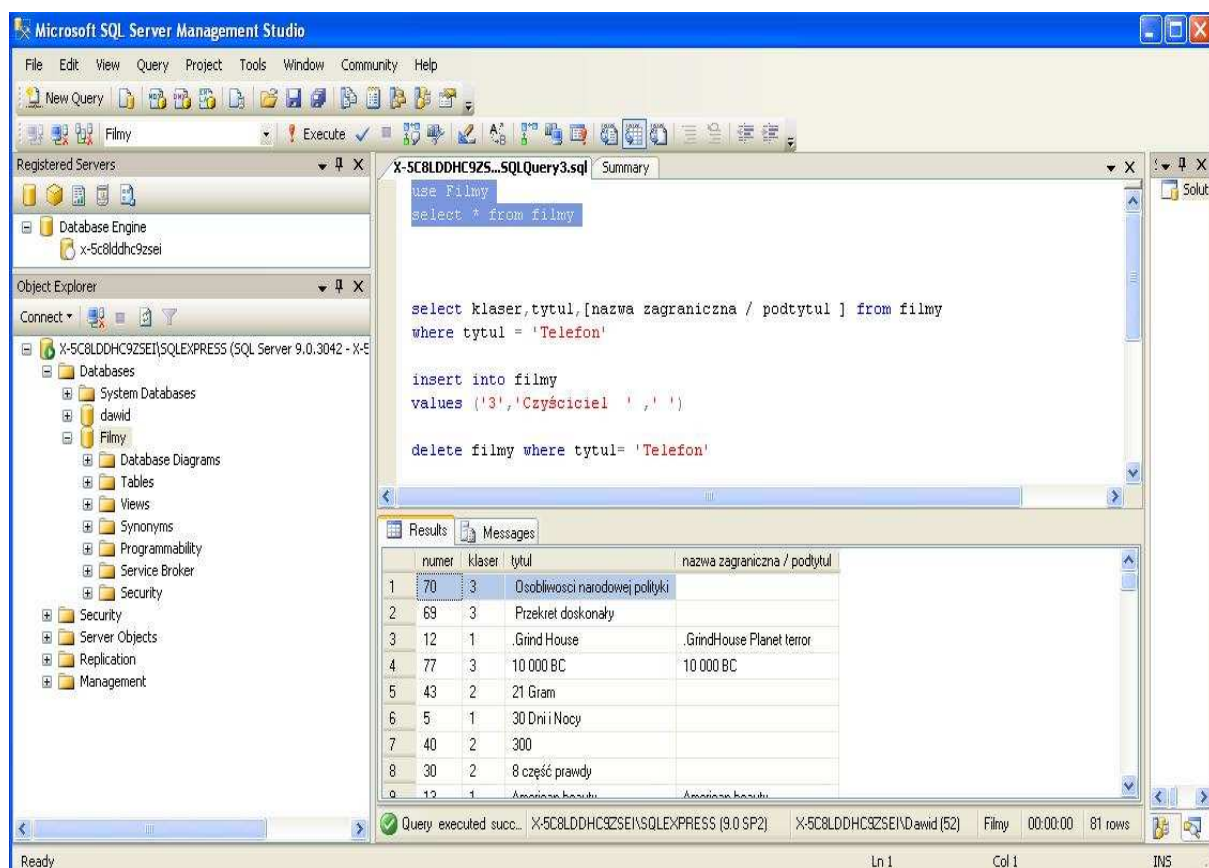
Kompilacja programu w środowisku Visual Studio odbywa się poprzez wybranie opcji *Bulid Page* lub *Bulid Web Site* z menu *Bulid*. Jeżeli program został napisany poprawnie i w *Output* nie pojawią się informacje o błędach, będzie możliwe uruchomienie programu. Uruchamianie jest dostępne w dwóch wersja ze śledzeniem i bez śledzenia błędów, są to odpowiednio opcje: *Start Debugging* - klawisz *F5* i *Start Without Debugging* – klawisz *Ctrl + F5*. Opcja *Start Debugging* pozwala na kontrolę programu po jego uruchomieniu, śledzenie programu w krokach i obserwowanie jego stanu.

3.4 Microsoft SQL Server 2005

W ostatnich latach, kiedy coraz większa rzesza użytkowników zajmuje się zagadnieniami związanymi z bazami danych, zna języki takie jak SQL i pochodne, oraz używa oprogramowania typu SQL Server, coraz mniej wyraźna staje się granica między projektantami baz danych a administratorami zarządzającymi serwerem na co dzień.

Microsoft SQL Server 2005 to zaawansowane narzędzie do tworzenia i obsługi baz danych, która oprócz podstawowej funkcji przechowywania, przetwarzania i efektywnego wyszukiwania danych posiada szereg dodatkowych narzędzi ułatwiających jej eksploatację, zarządzanie, analizę i raportowanie.

SQL Server używa bazy danych typu relacyjnego, dane są zgrupowane w tabelach. Tabele są tworzone poprzez grupowanie danych z tego samego tematu i zawierają kolumny oraz wiersze. Gdy uruchamiane jest zapytanie, tabele są ze sobą wiązane za pomocą mechanizmów bazy danych. Tabele są ściśle związane z pojęciami relacji lub encji. W serwerze SQL Server, baza danych nie koniecznie jest związana z pojedynczym plikiem, jest to pojęcie logiczne, oparte na zbiorze powiązanych obiektów. Baza danych na serwerze SQL Server zawiera nie tylko pierwotne dane, ale także strukturę bazy danych, wszelkie indeksy, zabezpieczenia bazy w niektórych przypadkach, obiekty takie jak widoki lub procedury składowane związane z określoną bazą[7].



Rysunek 3.10. przedstawia okno Query Analizera.

Obiekty relacyjnej bazy danych

Najważniejsze obiekty wchodzące w skład relacyjnej bazy danych :

- Kolumny są integralną częścią tabel przechowującą dane, muszą posiadać określony typ danych i unikalną nazwę, oraz informację, czy jest w niej dopuszczalna wartość pusta.

- Typy danych określają rodzaj przechowywanych danych. Można korzystać z wielu typów danych, takich jak typ znakowy, numeryczny czy data. Każdej kolumnie w tabeli przypisany jest pojedynczy typ danych.
- Widoki są to głównie zapytania przechowywane w bazie danych, odnoszące się do jednej lub wielu tabel. Widoki zwykle wykluczają kolumny z tabeli lub łączą dwie lub więcej tabel. Stosowane są również jako mechanizmów zabezpieczeń, przez ograniczenie dostępu do określonej kolumny lub wiersza tabeli.
- Indeksy usprawniają zorganizowanie danych, powodują efektywniejsze wykonywanie zapytań.
- Klucze podstawowe mają zasadnicze znaczenie dla relacyjnych baz danych. Powodują że wiersze stają się unikalne, oraz umożliwiają identyfikację każdego przechowywanego elementu.
- Klucze obce to jedna lub więcej kolumn, do których odnoszą się klucze podstawowe. , Podczas wykonywania zapytania SQL Server wykorzystuje klucze podstawowe i obce do określenia relacji między danymi z odrębnych tabel.

Typy danych dostępne w SQL Server 2005

| | |
|---------------|---|
| Dane tekstowe | char, varchar, nchar, ntext, nvarchar |
| Liczbowe | int, smallint, bigint, tinyint, float, real, decimal, numeric |
| Data i czas | datetime, smalldatetime |
| Binarne | binary, varbinary |
| Waluta | money, smallmoney |
| Specjalne | text, image, xml, bit |

Rysunek 3.11. przedstawia typy danych języka SQL.

Typy tekstowe składają się z dwóch podtypów: łańcuchowego i unicode. Typ łańcuchowy *char*, *varchar*, może zawierać litery, liczby, symbole. Typ *char* ma możliwość przechowywania określonej, wcześniej zadeklarowanej ilości znaków. Zaleca się stosowanie typów o zmiennej ilości znaków takich jak *varchar*, gdyż ze względu na sposób ich zapisu przyczyniają się do efektywniejszego wykorzystania pamięci serwera.

Typ *nchar* zapisywany zostaje w standardzie unicode, czyli obsługującym znaki diakrytyczne, charakterystyczne dla języków danego kraju. Wadą jest tu wzrost miejsca w pamięci potrzebnego do zapisu takiego znaku.

Typy liczbowe dzielimy na całkowite, przybliżone i dokładne. Jeśli istnieje taka możliwość powinno stosować się typ całkowity, dzięki temu zwiększa się wydajność i efektywność pracy serwera.

Jeśli w bazie przeprowadzane są operacje na liczbach w których wystąpi przecinek zaleca się stosowanie liczb typu *decimal* lub *numeric*. Typ *numeric* powinien mieć określone dwa parametry precyzję i skalę. Pierwszy parametr określa, ile cyfr znajduje się przed przecinkiem, drugi parametr wskazuje ile cyfr znajduje się po nim. Przybliżone typy danych to *float* i *real*. Liczby przechowywane za pomocą tych typów używane są do składowania danych statystycznych, gdzie dokładność nie jest najważniejszą sprawą, a najważniejsza jest sama rozpiętość liczb, jakie można składować.

Data i czas. Do przechowywania dat, służą dwa typy danych. Pierwszym z nich to *datetime*. Przy pomocy tego typu, można zapisywać informacje z dokładnością do milisekundy. Jeżeli istnieje potrzeba zapisu danych, dotyczących daty a nie jest potrzebna aż tak duża dokładność, w zupełności wystarczy pole typu *smalldatetime*, określa ono czas z dokładnością do jednej minuty. Jeżeli istnieje taka możliwość a trzeba zaoszczędzić ilość miejsca na zapisywanie danych i uzyskać większą efektywność pracy należy zastanowić się nad użyciem typu *int*.

Typy binarne służą do przechowywania danych binarnych, czyli danych reprezentowanych w postaci szesnastkowej tworzonej ze znaków od 0-9 i A-F. Typ binarny pozwala na przechowywanie wszelkiego rodzaju danych binarnych, dokumentów, zdjęć, całych plików do 2 GB. [3].

Podstawowe polecenia języka SQL

- CREATE – utworzenie struktury bazy, tabeli, indeksu.
- DROP – całkowite usunięcie struktury.
- ALTER – zmiana struktury (dodanie kolumny do tabeli, zmiana typu danych w kolumnie tabeli).
- SELECT – pobranie z bazy danych.
- INSERT – umieszczenie danych w bazie.
- UPDATE – zmiana danych.
- DELETE – usunięcie danych z bazy[3].

4. Projekt aplikacji

4.1 Założenia projektowe

Odbiorcą oprogramowania może być dowolne kino, liczbę sal kinowych czy ilość filmów reguluje się wpisując odpowiednie wartości do bazy danych. Zleceniodawca zażyczył sobie aby w salach kina była równa ilość miejsc siedzących. Dla celów pracy zostało wymyślone kino o nazwie *Venus*. Z punktu widzenia projektu nie jest istotne ile kino posiada pracowników, istotna jest osoba klienta i administratora oprogramowania, oraz bazy danych.

Ponieważ kino jest nowo powstałe, nie wykorzystuje jeszcze żadnego oprogramowania. Dyrekcja kina postanowiła zakupić odpowiednie oprogramowanie, aby mieścić się w obecnych standardach posiadając swoją stronę internetową.

Głównym zadaniem systemu jest udostępnienie przez internet użytkownikom kinowego repertuaru i rezerwacji miejsc, kontrola, sprawowanie nadzoru nad kontami użytkowników, oraz informowanie ich, w razie zajścia nieprzewidzianych wypadków typu awarie sprzętu i inne okoliczności, uniemożliwiające odbyciu się seansów w planowanym terminie, przy pomocy informacji telefonicznej lub mailowej.

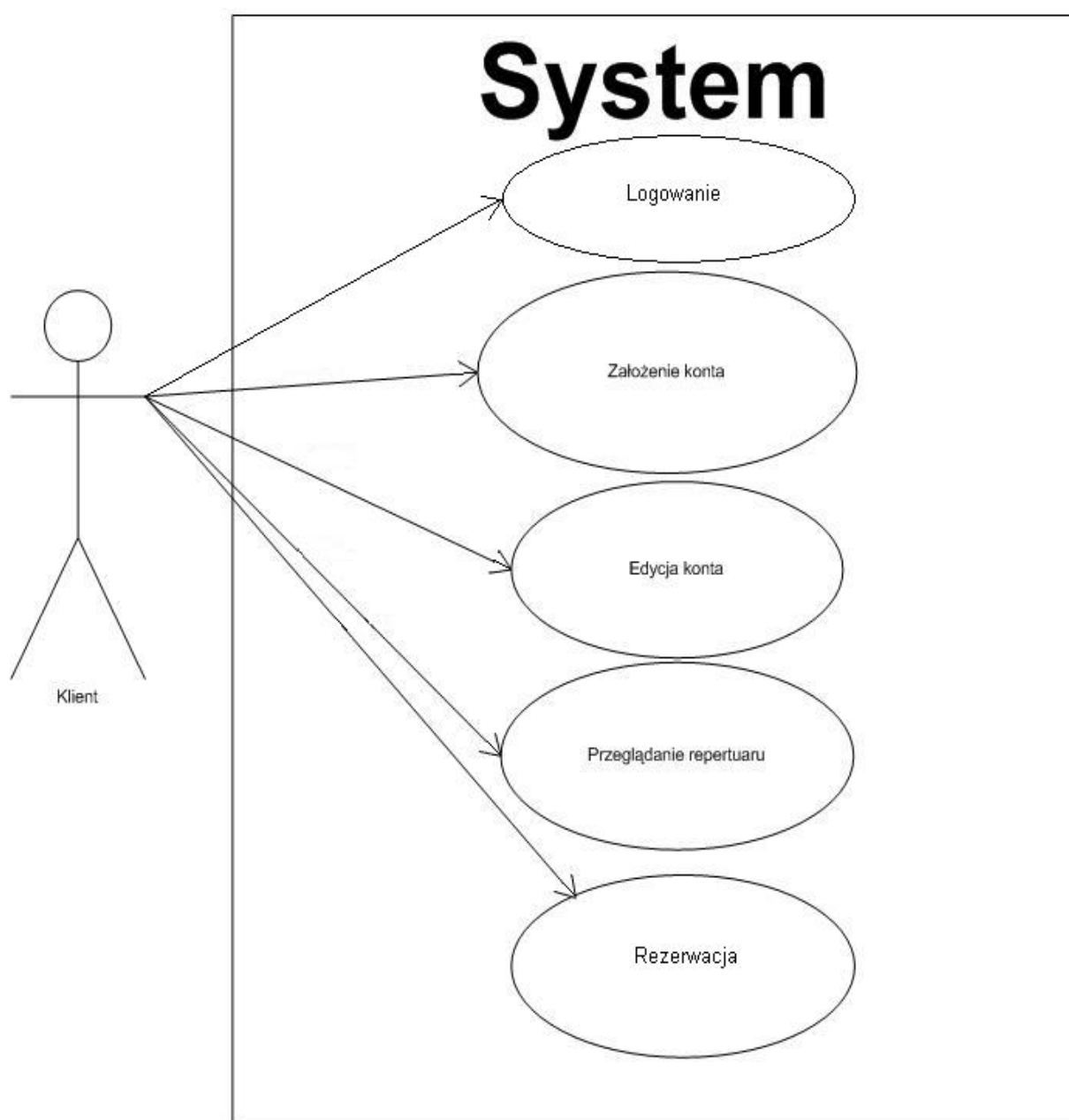
System może być gotowy do użytku po zainstalowaniu na serwerze kina, jeśli zdecyduje się, na zainstalowanie swojego serwera w budynku kina lub może działać na płatnych serwerach stron ASP. Powinien być dostępny, dla osób pragnących dokonać rezerwacji, siedem dni w tygodniu przez dwadzieścia cztery godziny na dobę. Powinno się uwzględnić czas na przerwę techniczną, najlepiej w późnych godzinach nocnych ze względu na małą używalność w tym czasie, na archiwizację lub wprowadzanie zmian w repertuarze.

Główną zaletą systemu rezerwacji będzie jego dostępność, prostota obsługi, niewymagająca specjalnych umiejętności informatycznych, niezawodność, niska cena i koszty eksploatacji. System może być administrowany, przez jedną lub dwie osoby np. w systemie dwuzmianowym.

System będzie pobierał dane od użytkowników w postaci formularzy dostępnych na stronach internetowych. Do jego obsługi wystarcza podstawowe umiejętności obsługi i przeglądania stron internetowych.

4.2.1 Funkcjonalność systemu

Klient- osoba, która na stronie kina ma możliwość zapoznania się dostępnym repertuarem, cenami biletów. Poprzez formularz internetowy ma możliwość założenia konta i po właściwym wypełnieniu pól, jeśli konto zostanie zaakceptowane może się zalogować, dokonać wyboru filmu i miejsca na sali.

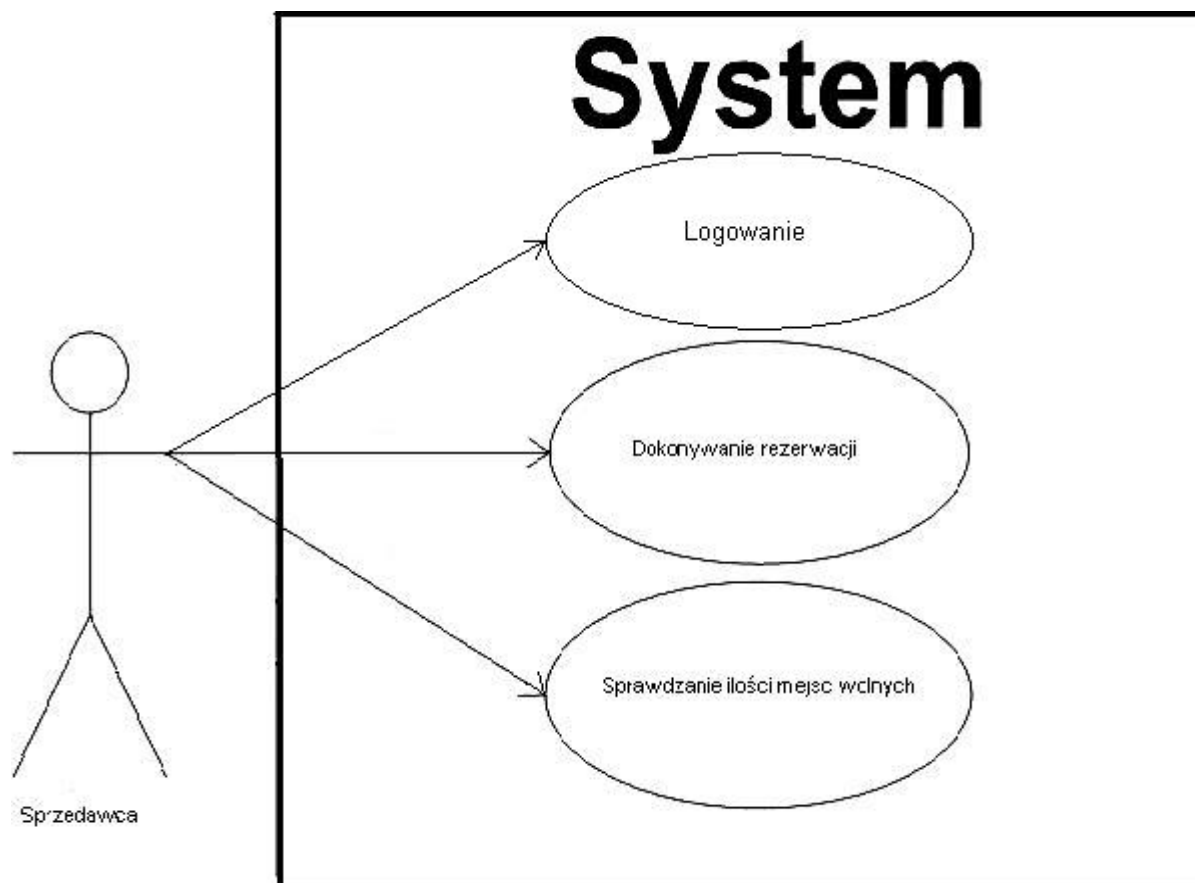


Rysunek 4.1. przedstawia diagram możliwych działań klienta.

Do istotnych operacji dotyczących klienta należy zaliczyć:

- przeglądanie repertuaru na głównej stronie kina, czynność ta jest dostępna dla wszystkich użytkowników Internetu i nie wymaga operacji logowania ani posiadania konta.
- zakładanie konta, niezbędne aby mieć dostęp do systemu rezerwacji. Klient zostaje poproszony o wypełnienie formularza z danymi kontaktowymi i podwójne wpisanie hasła w celu uniknięcia pomyłki. Zaleca się hasło, składające się z liczb i liter.
- edycja konta, przeprowadzana przez użytkownika w razie zmiany adresu zameldowania, numeru telefonu, czy adresu mailowego, danych niezbędnych w celach kontaktowych.
- wybór miejsca, po uprzednim wybraniu interesującego filmu, daty i godziny projekcji, osoba zalogowana może dokonać wyboru miejsca lub kilku, jeśli na seans wybiera się więcej niż jedna. O swoim wyborze zostanie poinformowana, wyświetlonym komunikatem z danymi dotyczącymi numerów miejsc i daty projekcji.

Sprzedawca – zadaniem tej osoby jest pobranie opłaty od klienta lub grupy za seans według obowiązującej taryfy. Sprzedawca przy pomocy programu, korzystając z podglądu sali, może poinformować osobę dokonującą zakupu o ilości miejsc wolnych na dany film.



Rysunek 4.2. przedstawia diagram możliwych działań sprzedawcy.

Do zadań sprzedawcy należą:

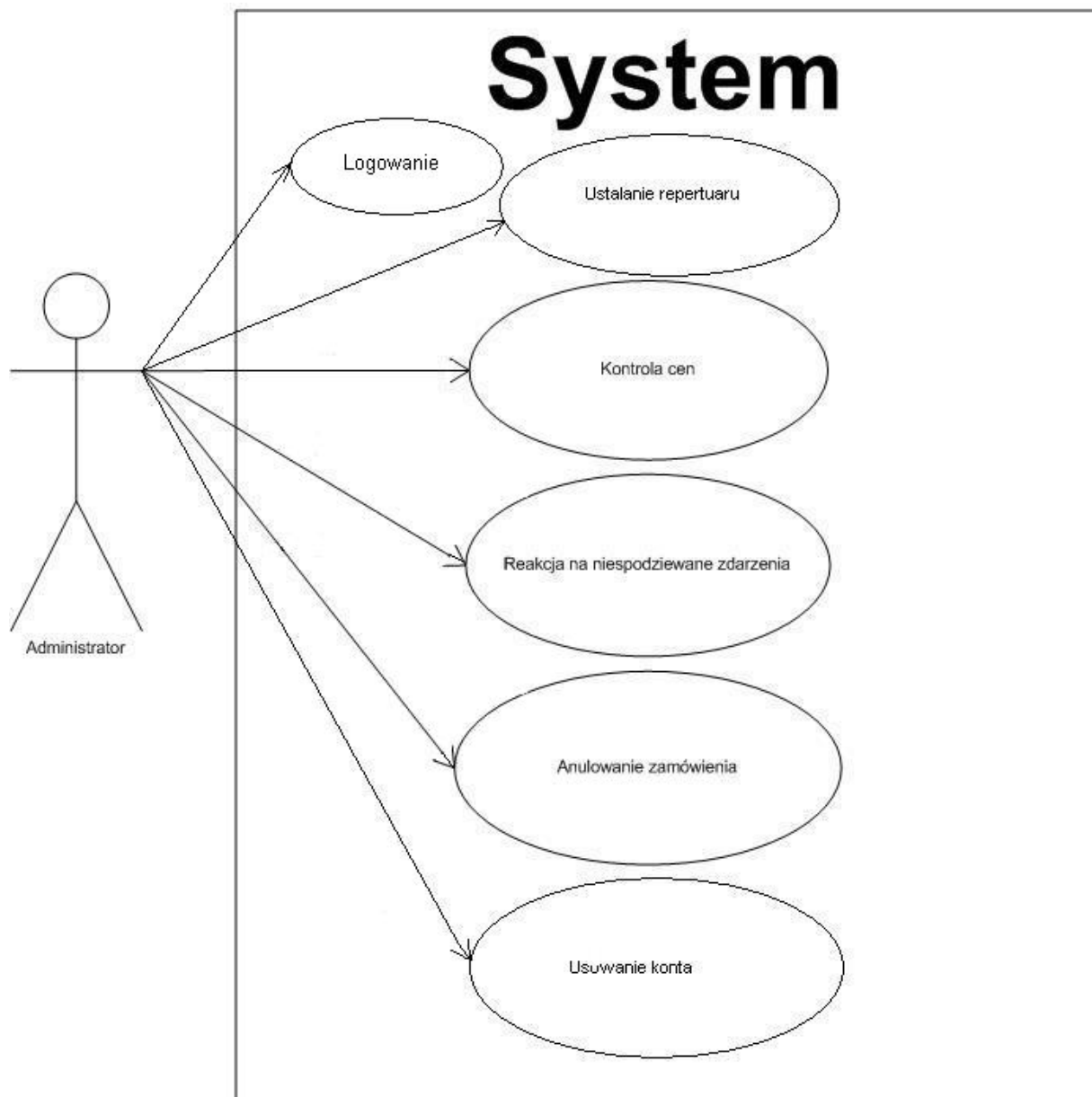
- dokonywanie rezerwacji za pomocą systemu, jeśli zostanie o to poproszony przez osoby znajdujące się przy kasie, zainteresowane wyborem miejsc na dany film, które nie dokonały wcześniejszej rezerwacji przez internet w domu a pragną zakupić bilet na chwilę przed projekcją.
- poproszony, sprzedawca ma możliwość sprawdzenia stanu sali na dany film, którym interesuje się klient, który nie dokonał wcześniej rezerwacji i chce to zrobić bezpośrednio w kinie.

Administrator – administrator systemu pełni tu rolę osoby pełniącej kontrolę nad kontami użytkowników, może je usunąć, korygować błędy powstałe, przy wprowadzaniu danych kontaktowych, popełnione przez użytkowników. Do jego zadań należy informowanie o promocjach, czy ewentualnych niespodziewanych zmianach w terminach projekcji. Zajmuje się też wprowadzaniem zmian do repertuaru kina, sprawuje kontrole nad cenami widocznymi na stronie kina w razie zmian wprowadza korekty .

Zapoznać się z ofertą kina może każdy, natomiast dostęp do systemu mogą mieć tylko użytkownicy posiadający swoje konta. Autoryzacja odbywać się będzie przy użyciu loginu i hasła. Zwykły użytkownik będzie miał dostęp tylko do tych funkcji, które są mu niezbędne. Administrator po uwierzytelnieniu, przekierowany zostanie do panelu zarządzania, gdzie dysponuje odpowiednimi narzędziami do sprawowania swoich zadań.

Do operacji dotyczących administratora należy zaliczyć :

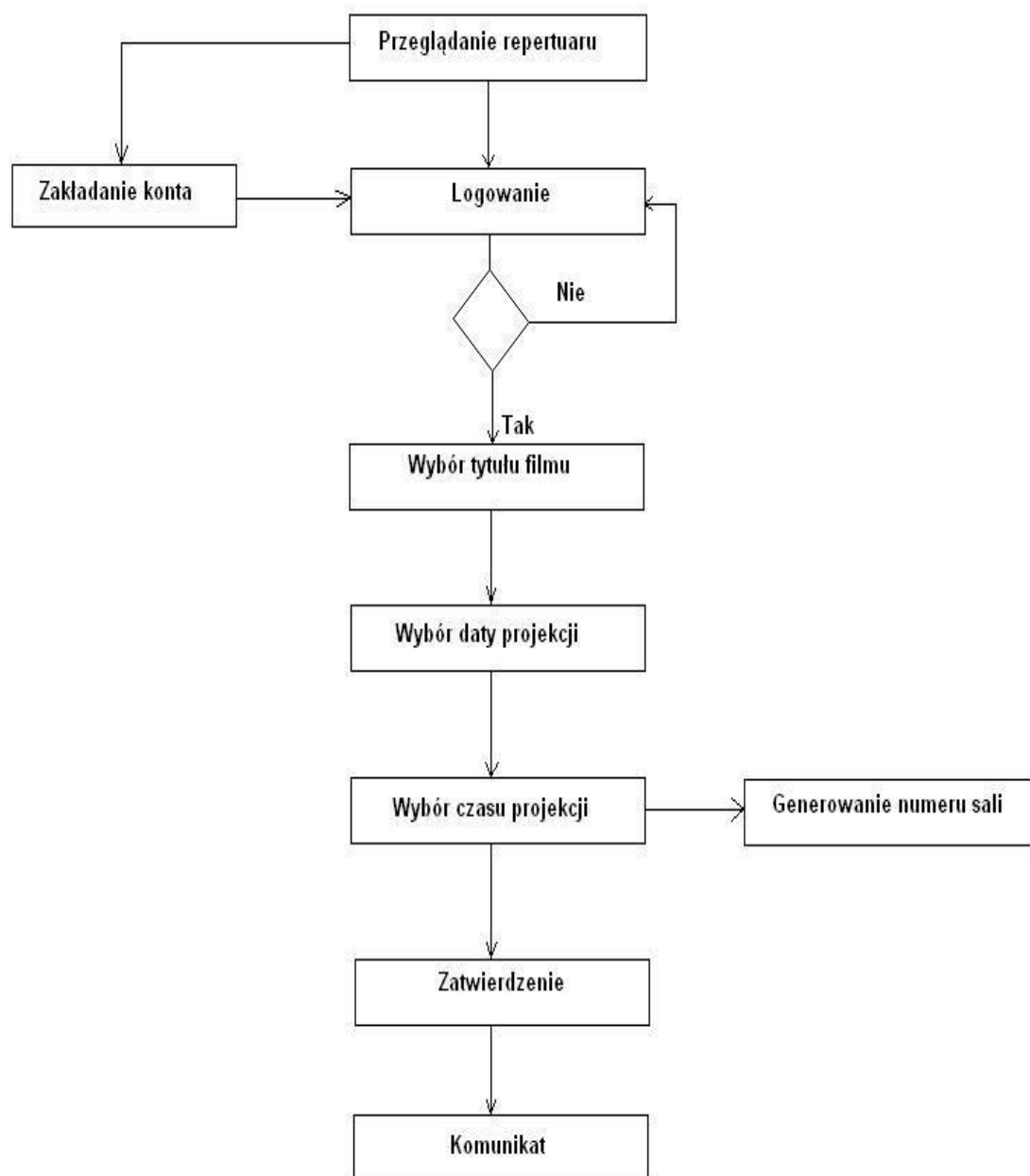
- Wprowadzanie nowych tytułów filmów na głównej stronie kina *Venus*, jaki i godzin ich projekcji.
- Wprowadzanie korekt cen seansów na stronie jeśli te ulegną zmianie.
- Reagowanie na zdarzenia losowe, czyli informowanie klienta, drogą poczty elektronicznej lub telefonicznie w razie konieczności odwołania lub przesunięcia terminu projekcji z przyczyn niezależnych.
- Usuwanie konta w przypadku łamania regulaminu przez danego użytkownika.



Rysunek 4.3. przedstawia diagram możliwych działań administratora.

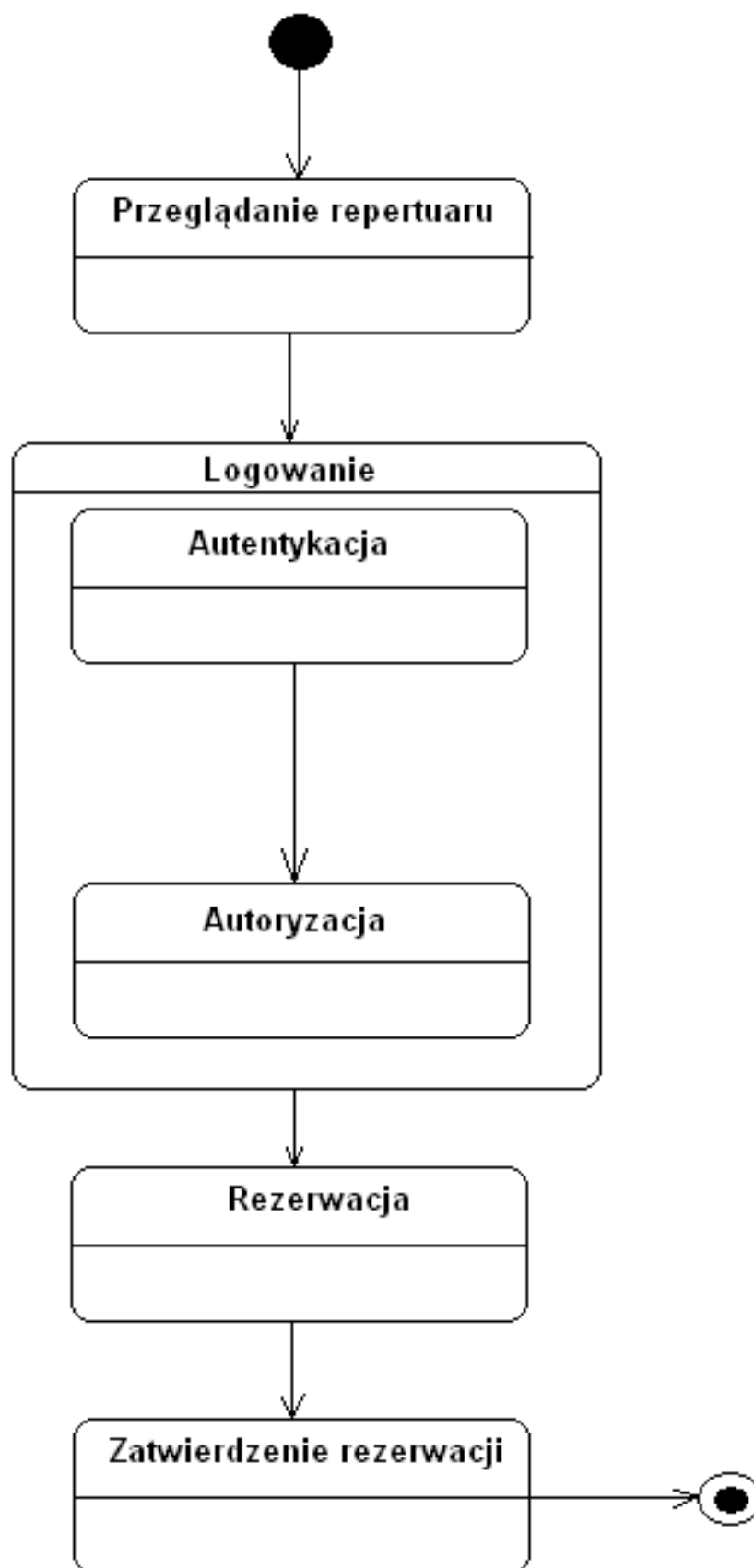
4.2.2. Działanie systemu

Głównym celem systemu jest danie użytkownikowi możliwości przeglądania filmowej oferty kina, oraz rezerwacji biletu lub ich większej ilości, na film w dowolnym wybranym przez siebie czasie. Operacja ta powinna się odbywać za pośrednictwem internetu. Najważniejszą osobą jest klient to on dokonuje wyborów, które na koniec zatwierdza. Następuje finalizacja i końcowy zapis wyników wyborów w bazie.

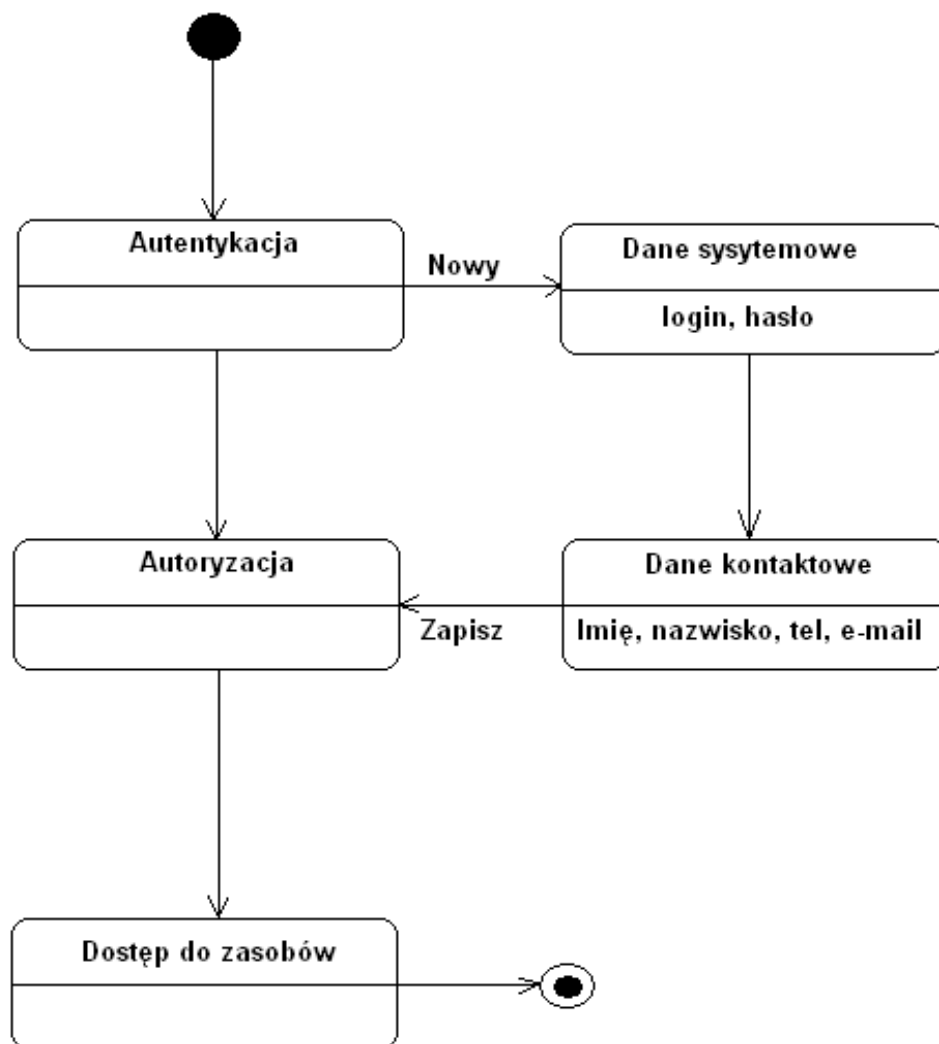


Rysunek 4.4. przedstawia algorytm działania aplikacji .

System umożliwia, wszystkim osobom odwiedzającym stronę kina zapoznanie się z dostępnym ogólnie repertuarem, cenami i promocjami, natomiast wybór miejsc i rezerwacja, przysługuje już tylko użytkownika zalogowanym czyli takim, którzy wcześniej założyli konto i posiadają dane niezbędne do autoryzacji.



Rysunek 4.5. przedstawia diagram stanów inicjowany przez klienta .



Rysunek 4.6. przedstawia diagram stanów podczas logowania .

Jeśli dana osoba zdecyduje się na rezerwowanie biletu, musi przejść proces logowania, będzie to tylko możliwe w przypadku gdy ma aktywne konto. Powinna podać prawidłowo login i hasło, dane te zostaną sprawdzone czy zgadzają się z tymi w bazie, wtedy osoba otrzymuje dostęp do zasobów.

W przypadku gdy nie posiada konta, może je założyć, wpisując w formularzu dane kontaktowe i zatwierdzając je, wtedy zostanie poproszona o zalogowanie.

4.2.3. Projekt bazy danych

System będzie wykorzystywał relacyjną bazę danych SQL, grupującą dane w tabele. Baza nosi nazwę *Kino*. Każda tabela wchodząca w skład bazy posiada klucz główny, który określony jest na jednym atrybucie, klucze są podstawową koncepcją w teorii relacyjnych baz danych. Gdyż zapewniają możliwość powiązania ze sobą dwóch lub więcej tabel. Poruszanie się i działania w relacyjnej bazie danych, zależy od możliwości identyfikacji określonego wiersza w tabeli za pomocą klucza głównego. Baza została zaprojektowana tak by ilość zbędnych danych nie została powielana, co zapobiega nadmiernemu się jej rozrastaniu i przyspiesza szybkość zapytań.

W skład bazy wchodziły tabele :

Osoba: tabela zawiera dane kontaktowe użytkownika, oraz dane potrzebne do logowania w systemie, bez których nie można dokonać rezerwacji.

-login : klucz główny, identyfikuje w sposób unikalny użytkownika w tej tabeli.

-hasło: chroni konto użytkownika przed niepowołanym dostępem.

-hasło2: powtórzenie pierwszego hasła w celu uniknięcia pomyłki i sprawdzeniu poprawności.

-imię: imię użytkownika podawane w celach kontaktowych.

-nazwisko : nazwisko osoby zakładającej konto.

-miasto: miasto zamieszkania.

-ulica: nazwa ulicy na której dana osoba mieszka.

-nr_domu: numer domu w miejscu zameldowania.

-wiek: wiek przyszłego użytkownika, może zastać użyty do badań statystycznych.

-telefon: numer telefonu, przydatny w celach kontaktowych.

-email: adres mailowy do prowadzenia korespondencji z klientem kina, przesyłaniu mu informacji promocyjnych.

| | |
|----------|---|
| login | ciąg tekstowy, 50 znakowy o zmiennej długości, klucz główny |
| haslo | ciąg tekstowy, 50 znakowy o zmiennej długości |
| haslo2 | ciąg tekstowy, 50 znakowy o zmiennej długości |
| imie | ciąg tekstowy, 20 znakowy o zmiennej długości |
| nazwisko | ciąg tekstowy, 30 znakowy o zmiennej długości |
| miasto | ciąg tekstowy, 30 znakowy o zmiennej długości |
| ulica | ciąg tekstowy, 30 znakowy o zmiennej długości |
| nr_domu | typ całkowity |
| wiek | typ całkowity |
| telefon | typ całkowity |
| email | ciąg tekstowy, 40 znakowy o zmiennej długości |

Rysunek 4.7. przedstawia widok tabeli Osoba.

Film: Kolejna z istotnych tabel zawierających nazwy filmów wyświetlanych w kinie Venus. Osoba wprowadzająca dane do tej tabeli ma wpływ na repertuar filmów dostępnych w kinie.

-id_film: klucz główny, identyfikujący w sposób unikalny film w tabeli.

-tytuł_filmu: kolumna zawierająca tytuły wyświetlanych filmów.

| | |
|-------------|---|
| id_filmu | typ całkowity, klucz główny |
| tytuł_filmu | ciąg tekstowy, 50 znakowy o zmiennej długości |

Rysunek 4.8. przedstawia widok tabeli Film.

Seans: Jedna z najważniejszych tabel, zawiera informacje, jaka osoba, zarezerwowała ile miejsc na dany seans w określonym, wybranym przez siebie czasie.

-id_seansu; klucz główny tabeli Seans, identyfikujący w sposób unikalny dany seans.

-login: identyfikuje użytkownika dokonującego rezerwacji.

-tytuł filmu: kolumna zawierająca tytuły wyświetlanych filmów.

-data: data projekcji danego filmu.

-czas: czas projekcji.

-nr_sali: numer sali w której będzie odbywał się seans.

-nr_miejsca: numer siedzenia na które zostanie wykupiona rezerwacja.

-id_miejsca: identyfikator miejsca.

| | |
|-------------|--|
| id_seansu | typ całkowity, klucz główny |
| login | ciąg tekstowy, 40 znakowy o zmiennej długości, obowiązkowy |
| tytul_filmu | ciąg tekstowy, 50 znakowy o zmiennej długości, obowiązkowy |
| data | ciąg tekstowy, 20 znakowy o zmiennej długości, obowiązkowy |
| czas | ciąg tekstowy, 20 znakowy o zmiennej długości, obowiązkowy |
| nr_sali | typ całkowity, obowiązkowy |
| nr_miejsca | typ całkowity, obowiązkowy |
| id_miejsca | typ całkowity, obowiązkowy |

Rysunek 4.9. przedstawia widok tabeli Seans.

Dane potrzebne do wypełnienia tabeli *Seans* pobierane są z tabel wcześniej już opisanych *Osoba* i *Film*, oraz tabel : *Data*, *Czas*, *Sala*, *Miejsce* i *Rzqd*.

Data: encja z której pobieramy datę kiedy grane są poszczególne filmy.

-id_daty: identyfikator daty, klucz główny tabeli w unikalny sposób definiuje przyporządkowanie daty do filmu .

-data: data projekcji danego filmu.

-id_filmu :unikalny identyfikator filmu.

| | |
|----------|--|
| id_daty | typ całkowity, klucz główny |
| data | ciąg tekstowy, 20 znakowy o zmiennej długości, obowiązkowy |
| id_filmu | typ całkowity, obowiązkowy |

Rysunek 4.10. przedstawia widok tabeli Data.

Czas: encja z której pobieramy czas występowania seansów.

-id_czasu: identyfikator czas, klucz główny tabeli w unikalny sposób definiuje przyporządkowanie czasu do filmu .

-data: data projekcji danego filmu.

-id_filmu : unikalny identyfikator filmu.

| | |
|----------|---|
| id_czasu | typ całkowity, klucz główny |
| czas | ciąg tekstowy, 20 znakowy o zmiennej długości |
| id_filmu | typ całkowity, obowiązkowy |

Rysunek 4.11. przedstawia widok tabeli Czas.

Sala: w tabeli tej są zapisane numery sal które, znajdują się w kinie.

-id_sali : unikalny identyfikator numeru sali kinowej.

-nr_sali : numer sali.

-id_filmu : identyfikator filmu.

| | |
|----------|-----------------------------|
| id_sali | typ całkowity, klucz główny |
| nr_sali | typ całkowity, obowiązkowy |
| id_filmu | typ całkowity, obowiązkowy |

Rysunek 4.12. przedstawia widok tabeli Sala.

Radz: w tabeli tej są zapisane numery sal które, znajdują się w kinie.

-id_rzedu : unikalny identyfikator numeru rzędu.

-rzed : literowe oznaczenie rzędu, które jest tu kluczem głównym, gdyż litery rzędu się nie powtarzają.

| | |
|----------|---|
| Rzad | typ znakowy, o stałej długości równej jeden, klucz główny |
| id_rzedu | typ całkowity, obowiązkowy |

Rysunek 4.13. przedstawia widok tabeli Rzad.

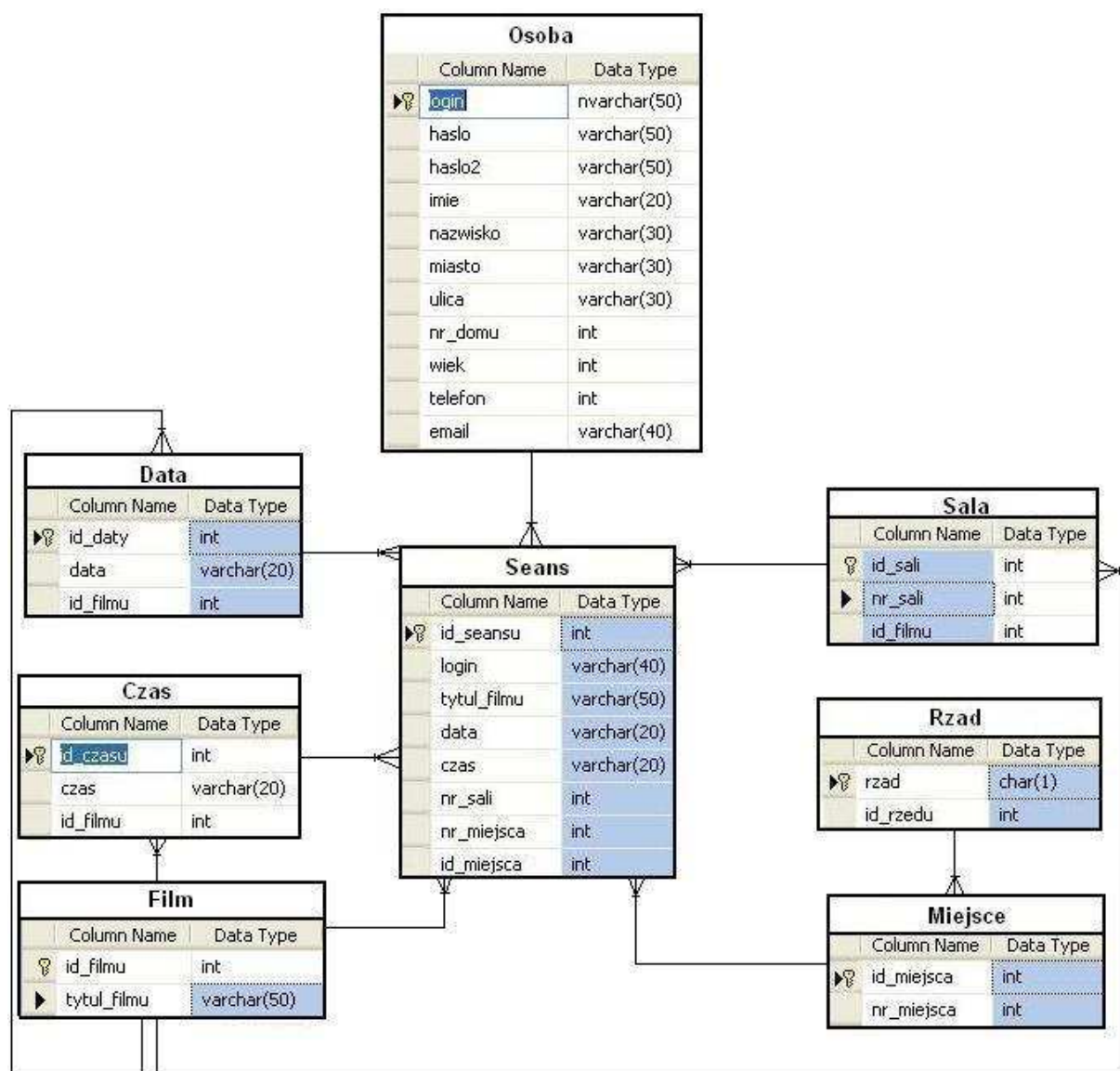
Miejsce: w tabeli tej znajdują się numery miejsc sali kinowej.

-id_miejsca : unikalny identyfikator numeru rzędu.

-nr_miejsca : numer miejsca znajdującego się na sali.

| | |
|------------|-----------------------------|
| id_miejsca | typ całkowity, klucz główny |
| nr_miejsca | typ całkowity, obowiązkowy |

Rysunek 4.14. przedstawia widok tabeli Miejsce.



Rysunek 4.15. przedstawia schemat związków encji.

Tabele *Sala*, *Rząd* i *Miejsce* w zasadzie nie zmieniają swoich wartości, gdyż ilość miejsc na sali, czy samych sal nie powinna się zmieniać, chyba, że dyrekcja kina podejmie się jego rozbudowy. Na tabelę *Data*, *Czas* i *Film* wpływ ma osoba zajmująca się zmianą repertuaru. To dzięki tym tabelą klienci kina wiedzą, jaki film i o jakim czasie będzie grany. Tabele *Osoba* wypełnia sam użytkownik, sprawdzana jest poprawność wprowadzania danych i ich kompletność. Wszystkie dane wprowadzone i wybrane przez użytkownika systemu trafiają na koniec do tabeli wynikowej *Seans*, gdzie przechowywane są dane, kto, ile miejsc, w jakim czasie zarezerwował na dany film.

5 . Konstrukcja aplikacji

5.1. Dostęp do danych

To tworzenia bazy danych i przeprowadzenia na niej operacji posłużono się SQL Server 2005 oraz Serwerem SQL wbudowanym w Visual Studio Web Developer 2008 i językiem zapytań SQL.

Budowa aplikacji i jej prawidłowe działanie wymaga istnienia bazy danych, jest to niezbędny atrybut służący do zapisu i pobierania danych, bez którego program nie mógłby funkcjonować.

Baza danych nosi nazwę *Kino.mdf* i znajduje się w katalogu *Kino*, łącznie z wszystkimi plikami projektu, co ułatwia instalację na dysku C lub serwerze stron ASP. który jest rozwiązaniem płatnym.

Definicje parametrów połączenia z bazą danych znajdują się w pliku konfiguracyjnym projektu Web.config.

```
<connectionStrings>
" providerName="System.Data.SqlClient" />

<add name="polaczenie" connectionString=
"Data Source=\SQLEXPRESS;AttachDbFilename=C:\Kino\Kino.mdf;
Integrated Security=True;Connect Timeout=20;
User Instance=True"

providerName="System.Data.SqlClient" />

</connectionStrings>
<system.web>
```

Kod w pliku Web.config generuje się automatycznie po użyciu komponentów służących do połączenia z bazą danych przy pomocy obiektów typu *SqlDataSource*. W niektórych przypadkach gdy nie stosuje się kontrolek dostępnych w środowisku programistycznym Visual Studio, należy samodzielnie zdefiniować połączenie z baza danych.

```
System.Data.SqlClient.SqlConnection conn = new
System.Data.SqlClient.SqlConnection();
conn.ConnectionString = conn.ConnectionString =
ConfigurationManager.ConnectionStrings["@polaczenie"].ConnectionString;
conn.Open();
```

Obiektem wykorzystanym do komunikacji między aplikacją, a systemem bazodanowym jest *SqlConnection*, który reprezentuje połączenie z bazą danych. Jednym z jego najważniejszych parametrów jest *ConnectionString*. Jest to ciąg znaków, zawierający przeważnie adres serwera bazodanowego, nazwę użytkownika oraz sposób autoryzacji.

5.2. Logika aplikacji

W tej części pracy zostaną zaprezentowane i omówione kluczowe fragmenty kodu źródłowego, sterującego aplikacją. Kod źródłowy został napisany w języku C#, jest on standardowo przechowywany w plikach z rozszerzeniem .aspx.cs środowiska programistycznego Visual Studio. Warto dodać, że kod, który generowany jest automatycznie, po dodaniu do formularza gotowego komponentu, dostępnego w Visual Studio, zapisywany jest w plikach z rozszerzeniem .aspx.

Poniżej zostanie omówiona droga od logowania do rezerwacji ze perspektywy analizy kodu źródłowego aplikacji w języku C#. Poniżej przedstawiono kod strony logowania.

```
protected void Button1_Click(object sender, EventArgs e)
{
    System.Data.SqlClient.SqlConnection conn = new
    System.Data.SqlClient.SqlConnection();

    conn.ConnectionString = ConfigurationManager.ConnectionStrings["@polaczenie"].ConnectionStr
    ing;

    conn.Open();
    System.Data.SqlClient.SqlCommand objSqlCommand = new
    System.Data.SqlClient.SqlCommand();
    objSqlCommand.Connection = conn;

    System.Data.SqlClient.SqlCommand objSqlCommand1 = new
    System.Data.SqlClient.SqlCommand();
    objSqlCommand1.Connection = conn;
    objSqlCommand.CommandType = System.Data.CommandType.Text;
    objSqlCommand1.CommandType = System.Data.CommandType.Text;
    objSqlCommand.CommandText = "SELECT login FROM Osoba where login =
    @nazwa ";
    objSqlCommand1.CommandText = "SELECT haslo FROM Osoba where
    haslo = @nazwa1 ";
    objSqlCommand.Parameters.AddWithValue("@nazwa ", TextBox1.Text);
    objSqlCommand1.Parameters.AddWithValue("@nazwa1 ", TextBox2.Text);
    object result = objSqlCommand.ExecuteScalar();
    object result1 = objSqlCommand1.ExecuteScalar();

    if (TextBox1.Text == "admin" && TextBox2.Text == "admin")
    {
        Session.Add("zmienna1", TextBox1.Text);
        Server.Transfer("administrator.aspx", true);
    }
}
```

```

    }

    else
    {

        if (result != null && result1 != null && TextBox2.Text == "admin")
        {
            Session.Add("zmienna1", TextBox1.Text);
            Server.Transfer("Sala.aspx", true);
        }

        else TextBox1.Text = " Niepoprawne dane , spróbuj ponownie ";

    }

    conn.Close();
}

```

W metodzie tej zdefiniowane jest i następuje otwarcie połączenia z bazą danych Kino.Mdf, tworzone są dwa obiekty, niezbędne do wykonania SQL-owego zapytania. Zadaniem metody ExecuteScalar jest zwrócenie wyników zapytania, które ma sprawdzić, czy dany login i hasło znajdują się w bazie danych i czy dane hasło przyporządkowane jest danemu loginowi. Jeśli login i hasło są prawdziwe, pokrywają się z tymi znajdującymi się w bazie, użytkownik zostanie przekierowany, odpowiedni na stronę z wyborem miejsc na sali kina, lub jako administrator do panelu zarządzania. W zmiennej sesyjnej zostaje zapamiętany login użytkownika, przekazany następnie na stronę Sala.

Do zakładania konta wykorzystano gotową kontrolkę *GridView*. Jest to kontrolka dostępna w programie Visual Studio, obsługująca bazy danych, współpracuje ona w tym przypadku z kontrolką *SqlDataSource* w której konfiguruje się połączenie z bazą danych.

Komponent *GridView* został jednak zmodyfikowany na potrzeby pracy. Co widać w pliku konto.aspx, gdzie przechowywane są informacje o gotowych, dodanych do formatki komponentach.

```

<Fields>
<asp:TemplateField HeaderText="login" SortExpression="login">
<EditItemTemplate>
<asp:Label ID="Label1" runat="server" Text='<%# Eval("login")
%>'></asp:Label>
</EditItemTemplate>
<InsertItemTemplate>
<asp:TextBox ID="TextBox9" runat="server" Text='<%# Bind("login")
%>'></asp:TextBox>
</InsertItemTemplate>
<ItemTemplate>
<asp:Label ID="Label9" runat="server" Text='<%# Bind("login")
%>'></asp:Label>
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="hasło" SortExpression="haslo">
<EditItemTemplate>

```

```

<asp:Label ID="Label2" runat="server" Text='<%# Eval("haslo")
%>'></asp:Label>
</EditItemTemplate> <InsertItemTemplate>
<asp:TextBox ID="TextBox10" runat="server" TextMode="Password" Text='<%#
Bind("haslo") %>'></asp:TextBox>
</InsertItemTemplate>
<ItemTemplate>
<asp:Label ID="Label10" runat="server" Text='<%# Bind("haslo")
%>'></asp:Label>
</ItemTemplate></asp:TemplateField>
<InsertItemTemplate>
<asp:TextBox ID="TextBox3" runat="server" Text='<%# Bind("email")
%>'></asp:TextBox>
<asp:RegularExpressionValidator ID="RegularExpressionValidator8"
runat="server"
ControlToValidate="TextBox3" ErrorMessage="Proszę wprowadzić adres poczty"
ValidationExpression="\w+([-+.' ]\w+)*@\w+([-.\w+)*\.\w+([-
.]\w+)*"></asp:RegularExpressionValidator>
</InsertItemTemplate>

```

Dla każdego pola, służącego do wprowadzania danych w kontrolce *DetailsView*, dodano sprawdzanie poprawności wprowadzanych danych. Zabespieczenie to zostało wprowadzone w celu uniknięcia, próby wprowadzenia do bazy niepoprwnego formatu danych przez użytkownika.

Po prawidłowym wypełnieniu pól kontrolki, dane zapisujemy za pomocą przycisku *Dodaj*, który również należy do komponentu. Opis samego przycisku został zmieniony na język polski, dodano komunikat-zapytanie o poprawne uzupełnienie danych kontaktowych.

```

<InsertItemTemplate>
<asp:Button ID="LinkButton1" runat="server" CausesValidation="True"
CommandName="Insert"
onclick="LinkButton1_Click1"
OnClientClick="return confirm('Pola nie mogą być puste, czy wszystkie
zostały wypełnione?');"></asp:Button>
<asp:Button ID="LinkButton2" runat="server" CausesValidation="False"
CommandName="Cancel" Text="Rezygnuję"></asp:Button>
</InsertItemTemplate>

```

Poniżej widoczny jest fragment kodu komponentu *SqlDataSource*, odpowiedzialny za aktualizację danych w tabeli *Osoba*.

```

<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%%$ ConnectionStrings:KinoConnectionString8 %>"
DeleteCommand="DELETE FROM [Osoba] WHERE [login] = @original_login AND
[haslo] = @original_haslo AND [haslo2] = @original_haslo2 AND [imie] =
@original_imie AND [nazwisko] = @original_nazwisko AND [miasto] =
@original_miasto AND [ulica] = @original_ulica AND [nr_domu] =
@original_nr_domu AND [wiek] = @original_wiek AND [telefon] =
@original_telefon AND [email] = @original_email"
InsertCommand="INSERT INTO [Osoba] ([login], [haslo], [haslo2], [imie],
[nazwisko], [miasto], [ulica], [nr_domu], [wiek], [telefon], [email])
VALUES (@login, @haslo, @haslo2, @imie, @nazwisko, @miasto, @ulica,
@nr_domu, @wiek, @telefon, @email)"
SelectCommand="SELECT * FROM [Osoba]"
UpdateCommand="UPDATE [Osoba] SET [haslo] = @haslo, [haslo2] = @haslo2,
[imie] = @imie, [nazwisko] = @nazwisko, [miasto] = @miasto, [ulica] =

```



```

@ulica, [nr_domu] = @nr_domu, [wiek] = @wiek, [telefon] = @telefon, [email]
= @email WHERE [login] = @original_login AND [haslo] = @original_haslo AND
[haslo2] = @original_haslo2 AND [imie] = @original_imie AND [nazwisko] =
@original_nazwisko AND [miasto] = @original_miasto AND [ulica] =
@original_ulica AND [nr_domu] = @original_nr_domu AND [wiek] =
@original_wiek AND [telefon] = @original_telefon AND [email] =
@original_email"
ConflictDetection="CompareAllValues"
OldValuesParameterFormatString="original_{0}">
<DeleteParameters>
<asp:Parameter Name="original_login" Type="String" />
<asp:Parameter Name="original_haslo" Type="String" />
<asp:Parameter Name="original_haslo2" Type="String" />
<asp:Parameter Name="original_imie" Type="String" />
<asp:Parameter Name="original_nazwisko" Type="String" />
<asp:Parameter Name="original_miasto" Type="String" />
<asp:Parameter Name="original_ulica" Type="String" />
<asp:Parameter Name="original_nr_domu" Type="Int32" />
<asp:Parameter Name="original_wiek" Type="Int32" />
<asp:Parameter Name="original_telefon" Type="Int32" />
<asp:Parameter Name="original_email" Type="String" />

```

Osoba, która jest zalogowana, zostaje przekierowana na formatkę Sala.aspx, gdzie w polu *TextBox* wpisany jest login użytkownika, przeniesiony ze strony logowania w zmiennej sesyjnej.

```

Session.Add("zmienna1", TextBox1.Text);
Server.Transfer("administrator.aspx", true);

```

Ma na tej stronie możliwość przeprowadzenia operacji wyboru, dostępne są listy wyboru *DropDownList1* o numerach od 1 do 4, reprezentujące odpowiednio: tytuł filmu, date, godzinę i numer sali w której będzie odbywał się kinowy seans .

| Login | Wybierz film | Wybierz dzień | Wybierz godzinę | Sala |
|-------|---------------|---------------|-----------------|------|
| jan | Wojna Harta ▼ | ▼ | ▼ | ▼ |

Rysunek 5.1. przedstawia listy wyboru strony Sala.

Użytkownik ma możliwość oddziaływania na listy wyboru, wybierając na początku tytuł filmu. Powoduje to uruchomienie zapytania języka SQL : `SELECT [tytuł_filmu] FROM [Film]`, gdzie *Film* jest nazwą tabeli z bazy danych *Kino*, ponieważ lista wyboru *DropDownList1* połączona jest ze źródłem danych *SqlDataSource1*.

Gdy dokonany już zostanie wybór filmu, stają się dostępne kolejne pola z list wyboru, możliwość wyboru daty i miejsca seansu, gdyż numer sali generowany jest automatycznie w zależności od dokonanych, wcześniejszych wyborów.

| | | | | |
|-------|---------------|---------------|-----------------|------|
| Login | Wybierz film | Wybierz dzień | Wybierz godzinę | Sala |
| jan | Wojna Harta ▼ | 04-09-2010 ▼ | 13:00 ▼ | 3 ▼ |

Rysunek 5.2. przedstawia aktywne listy wyboru strony Sala.

Jeśli zostanie dokonana zmiana w polu *DropDownLis3*, odpowiadającym za wybór daty, zostanie uruchomiona metoda *SelectedIndexChanged*, przyporządkowana do tej kontrolki.

```
System.Data.SqlClient.SqlConnection conn = new
System.Data.SqlClient.SqlConnection();
conn.ConnectionString =
ConfigurationManager.ConnectionStrings["@polaczenie"].ConnectionString;
conn.Open();

System.Data.SqlClient.SqlCommand objSqlCommand = new
System.Data.SqlClient.SqlCommand();
objSqlCommand.Connection = conn;
objSqlCommand.CommandType = System.Data.CommandType.Text;

objSqlCommand.CommandText = "SELECT id_miejsca from Seans where
(tytul_filmu=@tytul_filmu and data=@data and czas=@czas and
nr_sali=@nr_sali )";

objSqlCommand.Parameters.AddWithValue("@tytul_filmu",
this.DropDownList1.SelectedItem.Text);
objSqlCommand.Parameters.AddWithValue("@data",
this.DropDownList3.SelectedItem.Text);
objSqlCommand.Parameters.AddWithValue("@czas",
this.DropDownList2.SelectedItem.Text);
objSqlCommand.Parameters.AddWithValue("@nr_sali",
this.DropDownList4.SelectedItem.Text);

System.Data.SqlClient.SqlDataReader rider = objSqlCommand.ExecuteReader();
while (rider.Read())
{
ListItem li =
CheckBoxList1.Items.FindByValue(rider["id_miejsca"].ToString());

li.Enabled = false;

li.Selected = false;
}

rider.Close();
conn.Close();
```

Na początku metody deklarowane jest i otwierane połączenie z baza danych. Otwarcie połączenia następuje poprzez wywołanie metody *Open*, obiektu *conn*. Ważną sprawą jest by pamiętać o jego zamknięciu po wykonanych operacjach, odpowiada za to metoda *Close*. Do

wykonywania instrukcji na bazie danych służy obiekt klasy *SqlCommand*, który reprezentuje instrukcje w języku SQL. Zapytanie *Select*, zwraca liczbę miejsc zapisanych w tabeli *Seans*.

```
"SELECT id_miejsca from Seans2 where (tytul_filmu=@tytul_filmu and  
data=@data and czas=@czas and nr_sali=@nr_sali )"
```

Dzięki klauzuli *where* dokonuje się selekcja wyników zapytania. Wynik zapytania zależy od tytułu filmu, daty, czasu i numeru sali. Parametry te są wybierane przez osobę dokonującą rezerwacji, pobierane są z list wyboru *DropDownList*.

Przy pomocy obiektu *DataReader* pobierany jest z bazy danych strumień danych tylko do odczytu. Wyniki działania zapytania są zapisywane w lokalnym buforze pamięci klienta. Przechowywane do momentu odczytania ich przez klienta przy użyciu metody *Read* obiektu *DataReader*. Wywołanie metody *ExecuteReader* na istniejącym obiekcie *Command* powoduje utworzenie obiektu *DataReader* i zwrócenie rekordów. Do pobrania rekordu z wyników zapytania służy metoda *Read* obiektu *DataReader*.

Wyniki zapytania zostają przedstawione wizualnie, przy pomocy kontrolki *CheckBoxList*, odbywa się to w sposób dynamiczny w reakcji na zmiany, przeprowadzone przez użytkownika w listach wyboru *DropDownList1*, *DropDownList2*, *DropDownList3*, *DropDownList4*. Zajęte miejsce, które jakiś użytkownik wcześniej już zarezerwował i zostało to zapisane w bazie, posiada status: `Enabled = false`; `Selected = false`; znaczy to, że nie można na nim wykonać operacji zaznaczenia. Po zakończeniu pracy z obiektem *DataReader* należy wywołać metodę *Close*, oraz zamknąć połączenie z bazą danych.

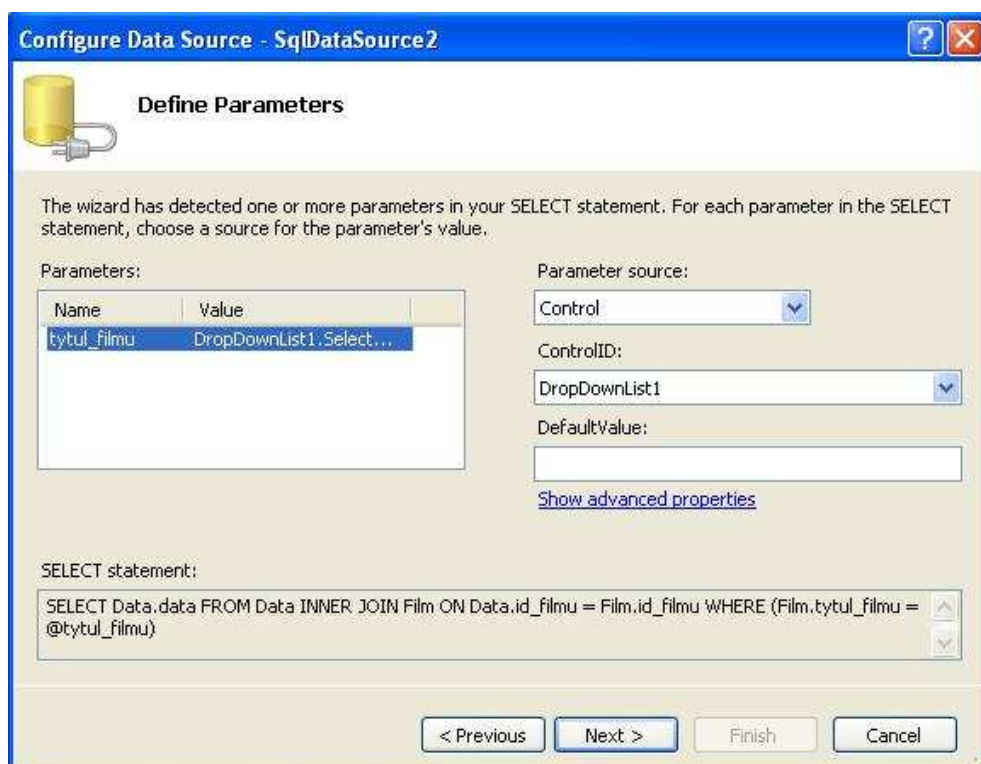
Lista wyboru *DropDownList2*, odpowiada za wybór godziny seansu. Kod tej kontrolki i metoda działania wygląda analogicznie. Różnicą jest tu wpływ użytkownika na wybór godziny a nie jak w poprzednim przypadku daty. Zmiana daty ma wpływ na wynik zapytania, co jest równoważne z inną interpretacją graficzną kontrolki *CheckBoxList*. Wszystkie listy wyboru mają ustawioną właściwość *AutoPostBack* na *True*, oznacza to że zawartość formularza wysyłana jest na serwer do powtórnego przetworzenia po przeprowadzeniu przez użytkownika zmian w którejkolwiek z list wyboru.

Grupa kontrolki *DropDownList* i przeprowadzane na nich operacje wyboru, ma bezpośredni wpływ na to co jest wyświetlane w *CheckBoxList* – zajęte miejsca na poszczególnych seansach. Omówiony zostanie kod, który ma wpływ na to co wyświetlają

kontrolki *DropDownLis*, jest to kod znajdujący się w *SqlDataSource*-kontrolce reprezentującej źródło danych w relacyjnej bazie danych SQL, ponieważ każda z list wyboru jest z taką połączona.

Lista wyboru *DropDownList2*, odpowiedzialna za możliwość wyboru daty seansu, pobiera dane z bazy *Kino* za pośrednictwem źródła *SqlDataSource3*, dzieje się tak za pośrednictwem właściwie skonstruowanego zapytania:

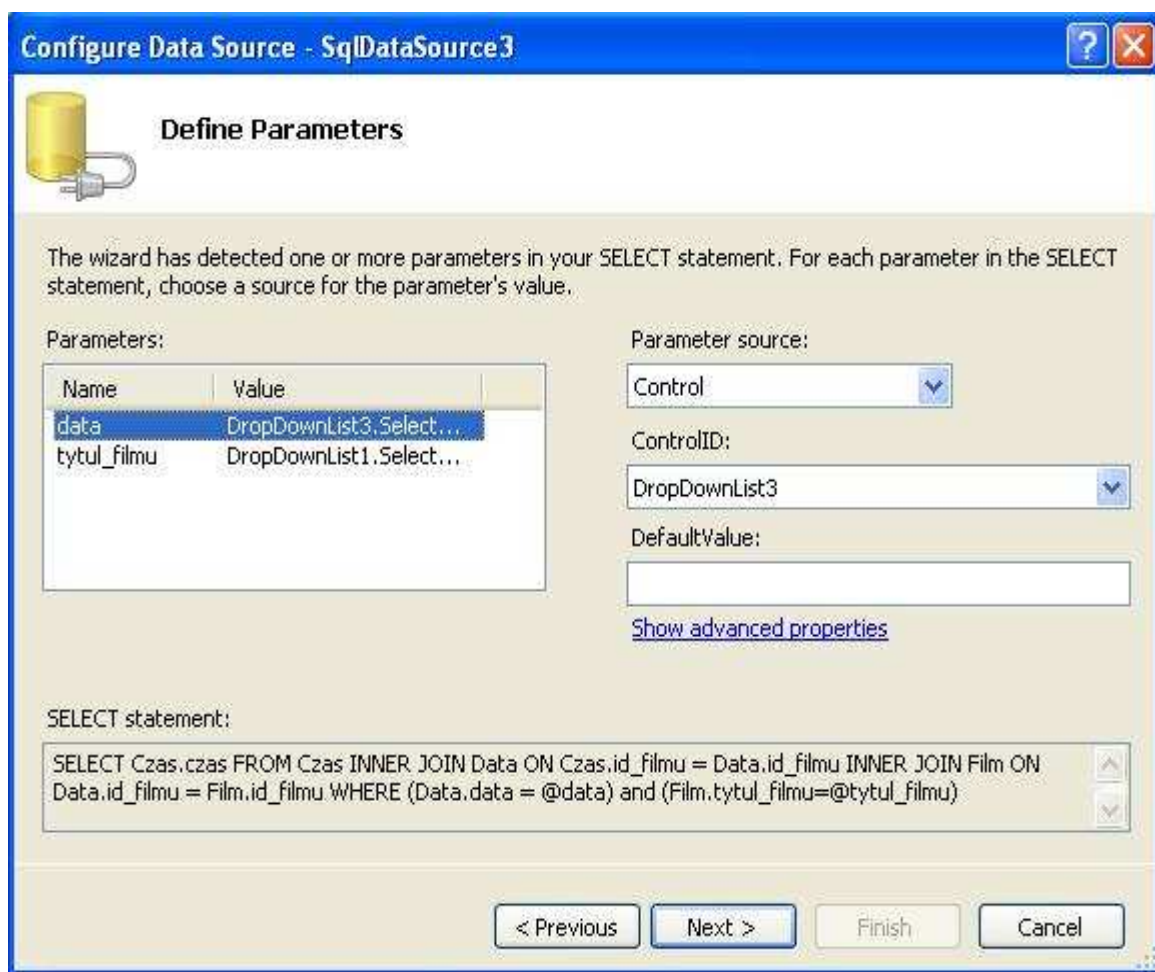
SELECT Data.data FROM Data INNER JOIN Film ON Data.id_filmu = Film.id_filmu WHERE (Film.tytul_filmu = @tytul_filmu). Kontrolka wyświetla możliwe daty seansu na film, którego tytuł wybrano z listy wyboru *DropDownList3*. Zostało to skonfigurowane we właściwościach *SqlDataSource2*.



Rysunek 5.3. przedstawia konfigurowanie źródła danych.

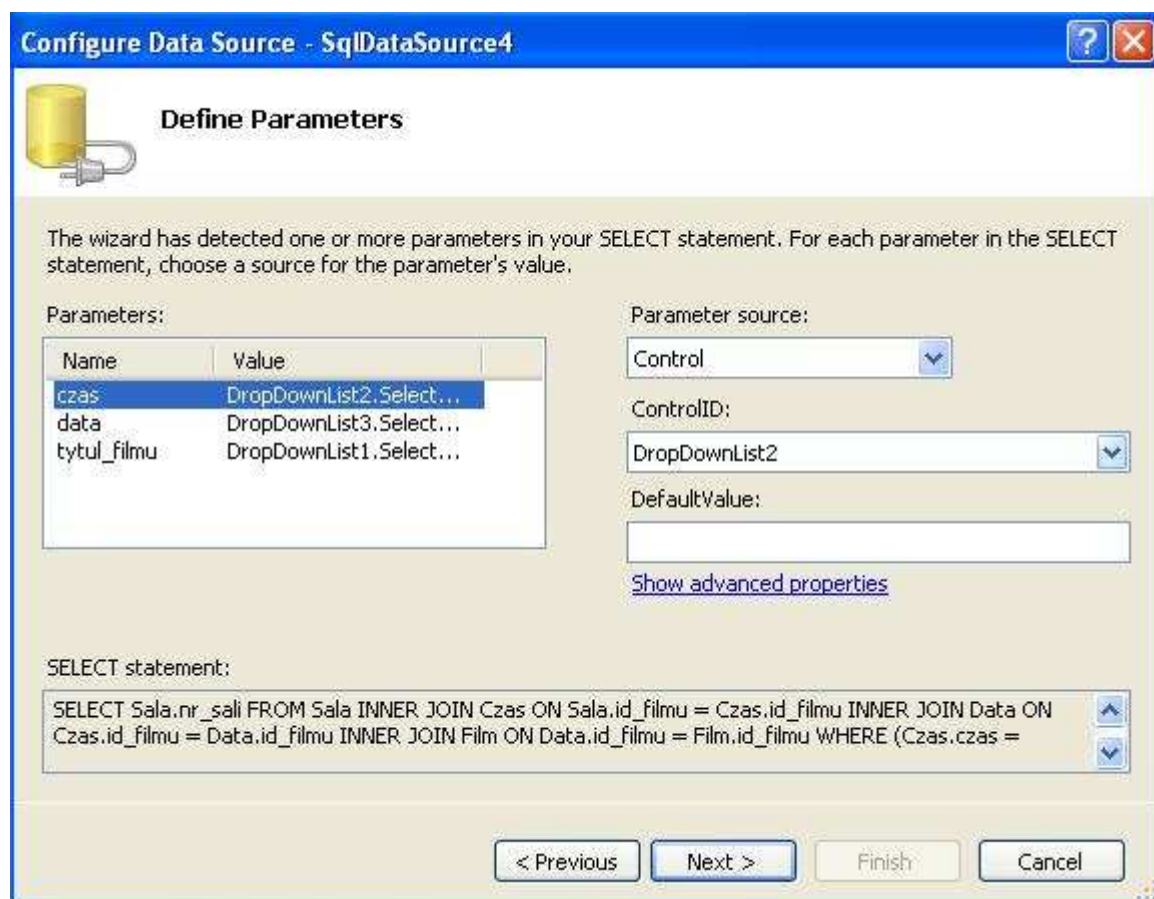
Za możliwość wyboru godziny wyświetlania filmu, którym zainteresowanie przejawia klient kina odpowiada lista *DropDownLis2* połączona ze źródłem danych *SqlDataSource3*. Odpowiada za to zapytanie w języku SQL o składni:

SELECT Czas.czas FROM Czas INNER JOIN Data ON Czas.id_filmu = Data.id_filmu INNER JOIN Film ON Data.id_filmu = Film.id_filmu WHERE (Data.data = @data) and (Film.tytul_filmu=@tytul_filmu) niezbędne było połączenie, typu *INNER JOIN*, tabel *Czas*, *Data* i *Film*. Kryteria wybierania do klauzuli *Where* pobierane są z *DropDownLis1* i *DropDownLis3* co skonfigurowano.



Rysunek 5.4. przedstawia konfigurowanie źródła danych *SqlDataSource3*.

Numer sali, w tabeli *DropDownLis4* generowany jest automatycznie, na podstawie zapytania w *SqlDataSour4*: `SELECT Sala.nr_sali FROM Sala INNER JOIN Czas ON Sala.id_filmu = Czas.id_filmu INNER JOIN Data ON Czas.id_filmu = Data.id_filmu INNER JOIN Film ON Data.id_filmu = Film.id_filmu WHERE (Czas.czas = @czas) AND (Data.data = @data) AND (Film.tytul_filmu = @tytul_filmu)`. Użytkownik nie dokonuje tego wyboru, ale ma na to pośredni wpływ dzięki zmianą wprowadzanym w listach odpowiedzialnych za wyświetlanie tytułu filmu, daty i czasu jego projekcji. Źródła kryteriów dla klauzuli *where* zostały również skonfigurowane.



Rysunek 5.5. przedstawia konfigurowanie źródła danych *SqlDataSource4*.

Wcześniej zostało zobrazowane w jaki sposób grupa kontrolki *DropDownList* wyświetla swoją zawartość, oraz jak w sposób dynamiczny oddziałuje to na komponent *CheckBoxList*. W następnym kroku zostanie scharakteryzowana zasada działania przycisku *Rezerwuj*, który służy do zapisu wybranej przez użytkownika konfiguracji do bazy danych.

```
System.Data.SqlClient.SqlConnection conn = new
System.Data.SqlClient.SqlConnection(); conn.ConnectionString =
ConfigurationManager.ConnectionStrings["@polaczenie"].ConnectionString;

conn.Open();

int ileZaz = 0;
try
{
    for (int i = 0; i < CheckBoxList1.Items.Count; i++)
    {
        if (CheckBoxList1.Items[i].Selected )
        {

            ileZaz += 1;
        }
    }
}
```



```

        System.Data.SqlClient.SqlCommand objSqlCommand = new
System.Data.SqlClient.SqlCommand();
        objSqlCommand.Connection = conn;
        objSqlCommand.CommandType =
System.Data.CommandType.Text;
        SqlTransaction tran = conn.BeginTransaction();
        objSqlCommand.Transaction = tran;

        objSqlCommand.CommandText = "INSERT INTO Seans
(tytul_filmu, data,czas,login,nr_sali,id_miejsca) VALUES
(@tytul_filmu,@data,@czas,@login,@nr_sali,@id_miejsca)";
        objSqlCommand.Parameters.AddWithValue("@tytul_filmu",
this.DropDownList1.SelectedItem.Text);
        objSqlCommand.Parameters.AddWithValue("@data",
this.DropDownList3.SelectedItem.Text);
        objSqlCommand.Parameters.AddWithValue("@czas",
this.DropDownList2.SelectedItem.Text);
        objSqlCommand.Parameters.AddWithValue("@login",
this.TextBox14.Text);
        objSqlCommand.Parameters.AddWithValue("@nr_sali",
this.DropDownList4.SelectedItem.Text);
        objSqlCommand.Parameters.AddWithValue("@id_miejsca",
this.CheckBoxList1.Items[i].Value);
        objSqlCommand.ExecuteNonQuery();
        tran.Commit();
    }
}

}
catch (Exception ex)
{
    TextBox10.Visible = true;
    TextBox10.Text = " Wystąpił błąd ";
}
finally { conn.Close(); }

TextBox10.Visible = true;
TextBox10.Text = "Rezerwacja na seans : " + DropDownList1.SelectedItem.Text
+ ", dnia " + DropDownList3.SelectedItem.Text + " na godzinę " +
DropDownList2.SelectedItem.Text + ", sala nr: " +
DropDownList4.SelectedItem.Text + " ,liczba wybranych miejsc " + ileZaz ;

foreach (ListItem li in CheckBoxList1.Items)
{
    li.Selected = false;
}

```

Na początku kodu zauważyć można deklarację połączenia z bazą danych i jego otwarcie. Definiowana jest zmienna *ileZaz* i zerowana jej wartość. Następnie zostaje otwarty blok *try {} catch*, którego zadaniem jest wyłapywanie i obsługa nieprzewidzianych sytuacji. W pętli zostają zliczone, zaznaczone pozycje z kontrolki *CheckBoxList*.

Przy pomocy SQL-owej komendy *INSERT* do bazy danych *Kino* w tabeli *Seans2* zostają zapisane dane znajdujące się aktualnie w listach wyboru, polu tekstowym oraz kontrolce *CheckBoxList*. Czyli w kolejności: login użytkownika, tytuł filmu, data i czas seansu, numer sali oraz zaznaczone miejsca. Cała operacja zapisu danych do bazy zachodzi w transakcji.

Transakcja gwarantuje że serwer przyjmie wszystkie polecenia będące jej częścią, jako niepodzielną całość, albo wszystkie je odrzuci. Transakcja pilnuje więc niepodzielność wykonania operacji na serwerze SQL.

Następuje zamknięcie połączenia a w polu tekstowym zostają wyświetlone informacje dotyczące rezerwacji. Zaznaczone pola w kontrolce *TextBoxList*, zostają oznaczone jako wybrane. Nie będzie już możliwe ponowne ich wybranie przy identycznej jak dokonana przez użytkownika konfiguracji.

Poniżej zostanie omówiona zasada działania, panelu sterowania administratora. Administrator może dodawać użytkowników za pośrednictwem komponentu *DetailsView* o identycznej zasadzie działania jak ten, który został użyty do wprowadzania danych kontaktowych przez użytkownika, zatem powtórne jego omawianie było by bezcelowe. Ta sama zasada dotyczy podglądu sali, gdyż administrator ma dostęp do komponentów i używa wcześniej już omówionych metod, które zostały przedstawione przy okazji działania podglądu sali i rezerwacji miejsc przez użytkownika.

Do edycji kont użytkowników został wykorzystana i zmodyfikowana kontrolka *GridView*. Opis przycisków, został zmieniony w pliku konfiguracyjnym, odpowiadającym za parametry *GridView*.

```
<asp:CommandField CancelText="Rezygnuje" DeleteText="" EditText="Edytuj"
ShowEditButton="True" UpdateText="Zatwierdzam" ButtonType="Button" />
<asp:TemplateField ShowHeader="False">
<ItemTemplate>
<asp:Button ID="Button1" runat="server" CausesValidation="False"
CommandName="Delete" Text="Usuń" OnClientClick="return confirm('Czy
naprawdę chcesz usunąć ?');" />
</ItemTemplate>
</asp:TemplateField>
```

Kontrolka pobiera dane za pomocą źródła danych *SqlDataSource* w ustawieniach tej kontrolki, zaznaczono opcje: *Enabled Sorting* i *Enabled Editing*, dzięki czemu uzyskano możliwość sortowania danych w kolumnach oraz pojawiły się w polu komponentu, przyciski edycji, których opis został zmieniony w pliku konfiguracyjnym na język polski, zostało też dodane potwierdzenie kasowania.

```
<asp:CommandField CancelText="Rezygnuje" DeleteText="" EditText="Edytuj"
ShowEditButton="True" UpdateText="Zatwierdzam" ButtonType="Button" />
<asp:TemplateField ShowHeader="False">
<ItemTemplate>
```



```

<asp:Button ID="Button1" runat="server" CausesValidation="False"
  CommandName="Delete" Text="Usuń" OnClientClick="return confirm('Czy
  naprawdę chcesz usunąć ?');"/>
</ItemTemplate>

<asp:SqlDataSource ID="SqlDataSource1" runat="server"

```

Oto kod odpowiedzialny za edycje i zapisywanie danych, przez kontrolke *GridView*.
DeleteParameters>

```

    <asp:Parameter Name="login" Type="String" />
  </DeleteParameters>
  <UpdateParameters>
    <asp:Parameter Name="haslo" Type="String" />
    <asp:Parameter Name="haslo2" Type="String" />
    <asp:Parameter Name="imie" Type="String" />
    <asp:Parameter Name="nazwisko" Type="String" />
    <asp:Parameter Name="miasto" Type="String" />
    <asp:Parameter Name="ulica" Type="String" />
    <asp:Parameter Name="nr_domu" Type="Int32" />
    <asp:Parameter Name="wiek" Type="Int32" />
    <asp:Parameter Name="telefon" Type="Int32" />
    <asp:Parameter Name="email" Type="String" />
    <asp:Parameter Name="login" Type="String" />
  </UpdateParameters>
  <InsertParameters>
    <asp:Parameter Name="login" Type="String" />
    <asp:Parameter Name="haslo" Type="String" />
    <asp:Parameter Name="haslo2" Type="String" />
    <asp:Parameter Name="imie" Type="String" />
    <asp:Parameter Name="nazwisko" Type="String" />
    <asp:Parameter Name="miasto" Type="String" />
    <asp:Parameter Name="ulica" Type="String" />
    <asp:Parameter Name="nr_domu" Type="Int32" />
    <asp:Parameter Name="wiek" Type="Int32" />
    <asp:Parameter Name="telefon" Type="Int32" />
    <asp:Parameter Name="email" Type="String" />
  </InsertParameters>
</asp:SqlDataSource>

```

```

ConnectionString="<%"$ ConnectionStrings:KinoConnectionString9 %">
DeleteCommand="DELETE FROM [Osoba] WHERE [login] = @login"
InsertCommand="INSERT INTO [Osoba] ([login], [haslo], [haslo2], [imie],
[nazwisko], [miasto], [ulica], [nr_domu], [wiek], [telefon], [email])
VALUES (@login, @haslo, @haslo2, @imie, @nazwisko, @miasto, @ulica,
@nr_domu, @wiek, @telefon, @email)"
SelectCommand="SELECT * FROM [Osoba]"
UpdateCommand="UPDATE [Osoba] SET [haslo] = @haslo, [haslo2] = @haslo2,
[imie] = @imie, [nazwisko] = @nazwisko, [miasto] = @miasto, [ulica] =
@ulica, [nr_domu] = @nr_domu, [wiek] = @wiek, [telefon] = @telefon, [email]
= @email WHERE [login] = @login">
<

```

Administrator w razie takiej potrzeby ma możliwość przejrzania ko dokonywał jakich rezerwacji. Do tego celu wykożystano kontrolkę *GridView* która wykorzystuje do połączenia z bazą danych *SqlDataSource* z włączona możliwością sortowania danych według loginu, identyfikatora seansu, daty, czasu czy numeru miejsca.

Do wyświetlania wyników wyszukiwania wykorzystano drugą kontrolkę *GridView*. Do

pola tekstowego, zostaje wprowadzony login osoby o której mają zostać wyświetlone dane na temat dokonanych przez nią rezerwacji. Login z pola tekstowego staje się parametrem wyszukiwania klauzuli *Where*, SQL-owego zapytania. Poniżej kod przedstawiający metodę przycisku.

```
System.Data.SqlClient.SqlConnection conn = new
System.Data.SqlClient.SqlConnection();
conn.ConnectionString =
ConfigurationManager.ConnectionStrings["@polaczenie"].ConnectionString;
conn.Open();
System.Data.SqlClient.SqlCommand objSqlCommand = new
System.Data.SqlClient.SqlCommand();
objSqlCommand.Connection = conn;
objSqlCommand.CommandType = System.Data.CommandType.Text;
objSqlCommand.CommandText = "select * from Seans2 where (login=@login)";
objSqlCommand.Parameters.AddWithValue("@login", this.TextBox4.Text);
SqlDataAdapter da = new SqlDataAdapter();
DataSet ds = new DataSet();
da.SelectCommand = objSqlCommand;
da.Fill(ds);
this.GridView2.DataSource = ds;
this.GridView2.DataBind();
conn.Close();
```

Zdefiniowanie i otwarcie połączenia z bazą *Kino*. Wykonanie zapytania w języku SQL z pobraniem parametru z pola tekstowego. Utworzenie obiektu *DataSet*, służącego do przechowywania danych, który staje się źródłem danych dla tabeli *GridView* z którego, przy pomocy polecenia *fill*, zostaje ona wypełniona danymi. Następnie zamknięcie połączenia z bazą danych.

W razie konieczności osoba zarządzająca portalem, ma możliwość urzycia narzędzia do wysyłania poczty. Na stronie znajduje się pole tekstowe z ustawioną opcją przewijania tekstu, służy ono do wprowadzania treści wiadomości. Drugie pole tekstowe do wprowadzania tytułu wiadomości. Lista wyboru *DropDownList*, Która wyświetla meile użytkowników z tabeli *Osoba*, za pomocą źródła danych *SqlDataSource*, przy pomocy zapytania: SELECT [email] FROM [Osoba].

Po wpisaniu w pole tekstowe treści i tytułu wiadomości, oraz wybraniu z listy adresu e-mail, wiadomość zostaje wysłana po naciśnięciu przycisku *Wyślij*.

```
System.Net.Mail.MailMessage wiadomosc = new
System.Net.Mail.MailMessage();
wiadomosc.IsBodyHtml = true;
wiadomosc.From = new MailAddress("dawidkal3@interia.eu");
wiadomosc.To.Add(DropDownList1.Text); //do
wiadomosc.Subject = (TextBox2.Text); //temat
```

```

wiadomosc.SubjectEncoding = System.Text.Encoding.UTF8;
wiadomosc.Body = (TextBox1.Text); //tresc
wiadomosc.BodyEncoding = System.Text.Encoding.UTF8;

SmtpClient smtp = new SmtpClient("poczta.interia.pl", 587)
{
    Credentials = new NetworkCredential("dawidka13@interia.eu",
    "2536mw2536mw");
};
smtp.Send(wiadomosc);

TextBox1.Text = " Wiadomość wysłano ";
TextBox2.Text = " ";

```

W metodzie użyto przestrzeni nazw System.Net.Mail, która zastąpiła swoją poprzedniczkę System.Web.Mail w stosunku do której wprowadzono zmiany, dodano nowe klasy, właściwości i metody, które pozwalają na bardziej elegancką oraz czystą formę operacji związanych z wysyłaniem meili, oraz możliwość asynchronicznego wysyłania meili. Przestrzeń nazw System.Net.Mail zawiera wszystkie klasy potrzebne do wysłania maila z aplikacji .NET. Należą do nich:

1. Attachment – przechowuje załączniki i jest używana z klasą MailMessage.
2. MailAddress – reprezentuje adres nadawcy lub odbiorcy (To, CC, BCC)
3. MailMessage – reprezentuje wiadomość e-mail, która ma być wysłana przy użyciu SmtpClient i zawiera własności takie jak: From, To, CC, BCC, Attachments, Subject i Body
4. SmtpClient – pozwala na wysyłanie wiadomości przy pomocy SMTP
5. SmtpException – reprezentuje wyjątki, kiedy nie można wysłać wiadomości.

Klasa MailMessage reprezentuje wiadomość e-mail, która ma być wysłana przy użyciu SmtpClient i zawiera w tym przypadku własności takie jak: From, To, Subject i Body. Wiadomość będzie w formacie HTML *wiadomosc.IsBodyHtml* = true; Utworzono instancję klasy MailMessage. Konstruktor klasy przyjmuje jako parametry adres nadawcy, adres odbiorcy – odczytywany jest z listy wyboru DropDownList, temat wiadomości pobierany z pola tekstowego TextBox2, oraz treść wiadomości, wpisywaną w pole tekstowe TextBox.

Do wysyłania poczty użyto konta na serwerze Interii, obsługiwanego przez protokół SMTP i ustawiono numer portu. Po wysłaniu wiadomości w polu tekstowym zostaje wyświetlony komunikat o tym zdarzeniu.

5.3. Warstwa prezentacji

5.3.1. Formularze użytkownika

System rezerwacji jest aplikacją WWW, komunikującą się z użytkownikiem za pomocą formularzy. W tym rozdziale pracy zostanie pokazane i omówione środowisko graficzne i działanie interfejsu widziane oczami użytkownika aplikacji.



Rysunek 5.6. przedstawia stronę startową aplikacji.

Po uruchomieniu widoczna jest strona startowa aplikacji, która dostępna jest dla wszystkich użytkowników zarówno tych posiadających konta jak i nie.

Na stronie kina można zapoznać się z tytułami wyświetlanych filmów, dokładnym ich opisem, cennikiem oraz regulaminem kina i zasadami rezerwacji. Dzieje się to po najechaniu kursorem myszki i kliknięciu na wybranym linku.





| Regulamin |
|---|
| 1. W celu wyboru repertuaru prosimy o zalogowanie , jest to wymagane do uruchomienia procedury |
| 2. Prosimy o poprawne dane kontaktowe. |
| 3. Wszystkie dane są chronione i nie będą rozpowszechniane. |
| 4. Cennik jest dostępny na stronie kina , kierownictwo może wprowadzać jego okresowe zmiany. |
| 5. Prosimy o odebranie biletów 30 min. przed seansem. |
| 6. Przy zakupie biletów prosimy okazać dokument uprawniający do zniżki. |
| 7. W razie nieprzewidzianych wypadków, zostaną państwo poinformowani telefonicznie lub drogą elektroniczną. |
| Zasady rezerwacji |
| 1. Rezerwacji można dokonać telefonicznie, dzwoniąc na numer podany na stronie kina lub przez internet. |
| 2. Rezerwacji może dokonać osoba posiadająca aktywne konto. |
| 3. Jeśli nie posiadają państwo konta, prosimy wypełnić formularz na stronie kina. |
| 4. Następnie należy się zalogować. |
| 5. Kolejnym krokiem jest wybranie interesującego nas filmu i czasu projekcji. |

Rysunek 5.7. przedstawia stronę z regulaminem i zasadami rezerwacji.

Na stronie głównej znajdują się odnośniki do poszczególnych stron ze szczegółowym opisem każdego filmu, czasem jego projekcji, nazwiskiem autora scenariusza i reżysera.

WOJNA HARTA





Strona poświęcona filmom wojennym


reżyseria: Gregory Hoblit , **scenariusz:** Billy Ray , Terry George , **zdjęcia:** Alar Kivilo , **muzyka:** Oliver Wallace , Rachel Portman , **na podstawie powieści:** Johna Katzenbacha "Wojna Harta" , **czas trwania:** 125 min

Występują :

Bruce Willis : Pułkownik William McNamara , Colin Farrell : Porucznik Tommy Hart , Terrence Howard : Porucznik Lincoln Scott , Cole Hauser : Bedford , Vicellous Reon Shannon : Lamar Archer




Opis filmu Wojna Harta
 II wojna światowa. Plk. William McNamara (Bruce Willis), amerykański żołnierz pochodzący z rodziny, której członkowie od czterech pokoleń służą w wojsku, trafia do niemieckiego obozu jenieckiego. Jako najwyższy rangą oficer obejmuje dowództwo nad swymi rodakami pod czujnym okiem komendanta obozu, bezwzględnego pułkownika Wenera Vissera (Marcel Iures). McNamara nie ma zamiaru poddać się bez walki - w tajemnicy przez wszystkich snuje plan prywatnej wojny i czeka na stosowny moment, by uderzyć na wroga... Okazja, by wprowadzić plan w życie nadarza się, gdy w obozie popełnione zostaje morderstwo. Aby odwrócić uwagę Vissera i niemieckich strażników, McNamara pozyskuje do współpracy młodego porucznika, Tommy'ego Harta (Colin Farrell) i z jego pomocą organizuje pokazowy proces wojskowy. W rzeczywistości ma zamiar podjąć próbę ucieczki i wysadzić w powietrze pobliską fabrykę amunicji.



Strona główna

Rysunek 5.8. przedstawia stronę z opisem wybranego filmu.

Każda osoba goszcząca na stronie może przeglądać cennik i ofertę kina, ale żeby dokonać rezerwacji trzeba być posiadaczem konta. Osoby posiadające konto, po kliknięciu na odnośnik *rezerwuj bilet* poproszone zostaną o wypełnienie formularza logowania, w celu identyfikacji.

The image shows a login panel for 'kina Venus' set against a blue background. At the top, the title 'Panel logowania kina Venus' is centered. Below it, there are two input fields: the first is labeled 'Podaj login:' and the second is labeled 'Podaj hasło:'. Under the password field, there is a 'Zatwierdź' button. Below that is a link 'Jeśli nie posiadasz' followed by a 'Założ konto' button. At the bottom of the panel is a link 'Strona główna'.

Rysunek 5.9. przedstawia panel logowania.

Jeśli dany użytkownik nie posiada konta, po kliknięciu przycisku *załóż konto*, będzie miał dostęp do formularza, służącego do wprowadzania danych. Należy uzupełnić pola : *login, hasło, imię, nazwisko, miasto, numer domu, nazwę ulicy, wiek, telefon i email*. Dane kontaktowe niezbędne są pracownikom kina, aby mogli oni informować klienta o różnych zdarzeniach: promocjach, konkursach, nowościach ale i awariach czy zmianie godzin seansów.

[Na stronę kina](#)

Zakładanie konta

| | |
|---------------|--------------------------|
| login | <input type="text"/> |
| hasło | <input type="password"/> |
| powtórz hasło | <input type="password"/> |
| imie | <input type="text"/> |
| nazwisko | <input type="text"/> |
| miasto | <input type="text"/> |
| ulica | <input type="text"/> |
| nr_domu | <input type="text"/> |
| wiek | <input type="text"/> |
| telefon | <input type="text"/> |
| email | <input type="text"/> |

Rysunek 5.10. przedstawia formularz zakładania konta.

Wprowadzana dane pomogą pracownikom kina na kontakt z oponentem droga mailową, telefoniczną czy za sprawą tradycyjnej poczty. Każdy użytkownik może wcześniej sprawdzić czy wybrany przez niego login już istnieje w bazie danych. W celu uniknięcia pomyłki, wypełnia się pole hasło i jego bliźniaczy odpowiednik.

Każda osoba posiadająca konto po zalogowaniu zostaje przekierowana na odpowiednią stronę w zależności czy jest to zwykły użytkownik, sprzedawca czy administrator.

Proszę wybrać z listy film, datę i godzinę

Login:
 Wybierz film:
 Wybierz dzień:
 Wybierz godzinę:
 Sala:

Karmazynowy przypływ
 Cierń czerwona linia
Wojna Harta
 Jahred - żołnierz piechoty morskiej
 Włóg u bram
 9 kompania

Rezerwuj

[Strona główna](#) [Załącznik](#) [Przebieg](#)

Rysunek 5.11. przedstawia formularz rezerwacji miejsc na seans.

Na formularzu służącym do rezerwacji w polu *login* można odczytać jaka aktualnie osoba jest zalogowana. Użytkownik z rozwijalnej listy wybiera film, będący w ofercie kina w tym momencie pojawi się zdjęcie z filmu aby lepiej zobrazować dokonany wybór. W listach, pojawiają się terminy i daty w których wyświetlany jest wybrany wcześniej film, numer sali zostaje wygenerowany automatycznie, osoba dokonująca rezerwacji nie ma na to wpływu. Po dokonaniu wyboru filmu, daty i czasu projekcji, w prawej części ekranu, można będzie zaobserwować które miejsca są wolne a które już zarezerwowane, przez osoby dokonujące rezerwacji wcześniej. Każdy wybór z listy data czy czas powoduje zmiany w widoku sali z zaznaczonymi miejscami, oczywiście jeśli miejsca na te właśnie seanse były zarezerwowane. Miejsca wcześniej zarezerwowane nie mają niebieskiej obwódki wokół kwadracika symbolizującego siedzenie, na polu kwadratu nie można dokonać już zaznaczenia, jest nieaktywne.

Po zalogowaniu się jako sprzedawca widzimy również formularz rezerwacji miejsc, gdzie możemy dokonać podobnych funkcji jak zwykły użytkownik. Konto sprzedawca zostało utworzone głównie z myślą o osobach które z jakiś przyczyn nie dokonały rezerwacji przez internet bądź nie mają założonego konta. Pracownik kina znajdujący się przy kasie dokonując wyboru filmu na określoną godzinę, ma dostęp do podglądu sali, widzi które miejsca są zajęte i ma możliwość zasygnalizowania tego klientowi. Sprzedawca poproszony przez osobę zainteresowaną kupnem biletu, może pokazać jej które miejsca są jeszcze wolne i zaznaczając odpowiednie numery siedzeń dokonać rezerwacji. Zostanie zapisane to w bazie, wtedy inna osoba nie będzie już mogła ich zarezerwować.

Po zakończeniu rezerwacji i związanych z tym operacji, można dokonać płatności za przez internet lub smesem, na specjalnej stronie www.ecard.pl. Odbywa się to po kliknięciu na link *Zapłata przelewem* w dolnej części panelu gdzie rezerwuje się miejsca na sali.



Rysunek 5.12 przedstawia stronę do płatności.

Firma eCard zajmują się nowoczesnymi rozwiązaniami w zakresie bezgotówkowego rozliczania transakcji finansowych, płatności on-line w internecie. eCard świadczy usługi zgodnie z normami międzynarodowych organizacji płatniczych takich jak Visa, MasterCard czy American Express. Firma stara się być na tyle elastyczna, aby być w stanie przyjąć prawie każdy schemat płatniczy zaproponowany przez klienta, dzięki temu obsługuje znane firmy takie jak: LOT, Merlin.pl, PKP, Polkomtel, Netia, Agora i PCK.

eCard oferuje :

- Płatności on-line
- Płatności w Call Centre i IVR

- Stałe zlecenie obciążenia karty
- 3D Secure dla wydawców kart płatniczych
- 3D Secure dla wydawców kart płatniczych
- SMS Premium
- Sieć bankomatów

Bezpieczeństwo transakcji dokonywanych za pomocą ePrzelewów jest zapewnione przez zastosowanie 128 bitowego protokołu SSL, jak również przez procedury identyfikacji klienta w banku (*login, hasło, hasło jednorazowe*). Dokonując przelewu on-line, klient loguje się bezpiecznie bezpośrednio na stronie swojego banku. Informacje o jego profilu posiada tylko i wyłącznie jego bank.

5.3.2. Formularze użytkownika

Administrator serwisu kina ma możliwość, podobnie jak sprzedawcy i zwykli użytkownicy, rezerwowania miejsc i wizualnego podglądu zajętości miejsc na sali. Opcje te zostały dokładniej omówione w opisie możliwości działań sprzedawcy i użytkownika. Oprócz tego ma możliwość dodawania i edycji poszczególnych pól, osób posiadających konto. Czynności te odbywają się w panelu administracyjnym.

[Na stronie kina](#)
 [Wyslij poczte](#)
 [Szukanie](#)
 [Podglad sali](#)

Panel administratora

| | |
|---|--------------------------|
| login | <input type="text"/> |
| haslo | <input type="password"/> |
| powtorz haslo | <input type="password"/> |
| imie | <input type="text"/> |
| nazwisko | <input type="text"/> |
| miao | <input type="text"/> |
| ulica | <input type="text"/> |
| nr domu | <input type="text"/> |
| wiek | <input type="text"/> |
| telefon | <input type="text"/> |
| email | <input type="text"/> |
| <input type="button" value="Dodaj"/> <input type="button" value="Rezygnuje"/> | |

| | <u>login</u> | <u>haslo</u> | <u>haslo2</u> | <u>imie</u> | <u>nazwisko</u> | <u>miao</u> | <u>ulica</u> | <u>nr domu</u> | <u>wiek</u> | <u>telefon</u> | |
|---------------------------------------|-------------------------------------|--------------|---------------|-------------|-----------------|-------------|--------------|----------------|-------------|----------------|------------|
| <input type="button" value="Edytuj"/> | <input type="button" value="Usuń"/> | adam | adam | adam | ss | sss | sss | 2 | 11 | 2222 | 2 |
| <input type="button" value="Edytuj"/> | <input type="button" value="Usuń"/> | admin | admin | admin | jan | www | www | 33 | 333 | 333 | d |
| <input type="button" value="Edytuj"/> | <input type="button" value="Usuń"/> | dany | dany | dany | aa | aa | aaa | aa | 22 | 2222 | a |
| <input type="button" value="Edytuj"/> | <input type="button" value="Usuń"/> | jan | jan | jan | ssss | sss | sss | 22 | 22 | 222 | q |
| <input type="button" value="Edytuj"/> | <input type="button" value="Usuń"/> | jas | jas | jas | dddd | poz | Blotna | 33 | 23 | 1212122 | s |
| <input type="button" value="Edytuj"/> | <input type="button" value="Usuń"/> | karol | karol | karol | kddd | pp | www | 22 | 222 | 222 | a |
| <input type="button" value="Edytuj"/> | <input type="button" value="Usuń"/> | sprzedawca | sprzedawca | sprzedawca | jan | kol | poz | Blotna | 2 | 25 | 29229922 s |

Rysunek 5.13. przedstawia panel administratora.

W przypadku decyzji o usunięciu konta, osoba zarządzająca zostaje zapytana o potwierdzenie decyzji. Po kliknięciu na przycisk edycji istnieje możliwość edycji wszystkich pól, wariant ten został wzięty pod uwagę ze względu na zaistnienie potrzeby zmiany adresu e-mail, czy innych danych niezbędnych w celach , kontaktu. Osoba wyrażająca chęć zmiany może poinformować o tym administratora telefonicznie lub za pośrednictwem meila, który podany jest na głównej stronie kina. Po przeprowadzeniu zmiany danych, petent zostanie poinformowany o tym fakcie drogą elektroniczną.

Wysyłanie meili odbywa się na specjalnie do tego celu stworzonej stronie, dostępnej wyłącznie osobie administrującej. Są tam stosowne pola w których wpisuje się treść i temat wiadomości a adresy e-mail wybiera się z rozwijalnej listy, która pokazuje wszystkie adresy poczty elektronicznej użytkowników znajdujących się w bazie danych.

Rysunek 5.14. przedstawia panel obsługi poczty.

Na stronie administratora, dostępna jest również zakładka, służąca do zapoznawania się z danymi dotyczącymi rezerwacji. Zobaczyć tam można kto dokonał rezerwacji na jaki film, ile miejsc i w jakim czasie. Istnieje możliwość odszukania interesującej nas osoby po loginie. Odbywa się to w następujący sposób, trzeba wpisać login, reprezentujący daną osobę z bazy w przeznaczone do tego pole i nacisnąć przycisk szukaj. Tabela na stronie wyświetli informacje dotyczące danej osoby: tytuł filmu, datę, godzinę projekcji, numer miejsca i sali. Dane w tabeli można porządkować, od najmniejszych do największych wartości lub odwrotnie, klikając na nagłówek kolumny z podkreśleniem.

| id_seansu | login | tytuł filmu | data | czas | nr_sali | nr_miejsca | id_miejsca |
|-----------|-------|-----------------------|------------|-------|---------|------------|------------|
| 385 | jan | Cieńka czerwona linia | 01-05-2010 | 11:00 | 2 | | 31 |
| 386 | jan | Cieńka czerwona linia | 01-05-2010 | 11:00 | 2 | | 32 |
| 387 | jan | Cieńka czerwona linia | 01-05-2010 | 11:00 | 2 | | 33 |
| 388 | jan | Cieńka czerwona linia | 01-05-2010 | 11:00 | 2 | | 34 |
| 406 | jan | Karmazynowy przyływ | 01-05-2010 | 12:10 | 1 | | 116 |
| | | Karmazynowy przyływ | | | | | |
| id_seansu | login | tytuł filmu | data | czas | nr_sali | nr_miejsca | id_miejsca |
| 375 | | Karmazynowy przyływ | 01-05-2010 | 12:10 | 1 | | 1 |
| 376 | | Karmazynowy przyływ | 01-05-2010 | 12:10 | 1 | | 2 |
| 377 | | Karmazynowy przyływ | 01-05-2010 | 12:10 | 1 | | 3 |
| 378 | | Karmazynowy przyływ | 02-05-2010 | 15:15 | 1 | | 16 |
| 379 | | Karmazynowy przyływ | 02-05-2010 | 15:15 | 1 | | 17 |

Rysunek 5.15. przedstawia służący do rezerwacji.

6. Testy aplikacji

Aplikację sprawdzano używając przeglądarek internetowych: Mozilla Firefox 3.5, Internet Explorer 7 oraz Google Chrome 5.0.

Zostały przeprowadzone testy, czy aplikacja działa prawidłowo pod względem spełniania swoich funkcji, oraz poprawności wprowadzania danych, wypełniania pól obowiązkowych czy nieprzewidzianych wypadków obsługiwanych przez globalną obsługę wyjątków, zadeklarowaną w pliku konfiguracyjnym projektu Web.config.

```
<customErrors mode="On" defaultRedirect="Bład.aspx">  
</customErrors>
```

Po wystąpieniu nieprzewidzianego błędu użytkownik zostaje przekierowany na specjalnie stworzoną stronę z komunikatem informacyjnym.



Rysunek 6.1. przedstawia stronę informacyjną wyświetlaną po wystąpieniu błędu.

Działanie panelu logowania zostało sprawdzone pod kontem właściwego przekierowania na strony użytkownika i administratora po podaniu odpowiedniego loginu i hasła. W przypadku braku wypełnienia któregoś z tych pól, osoba wprowadzająca zostanie poproszona o korektę. Podobnie dzieje się wtedy gdy login lub hasło nie odpowiadają tym które, zapisane są w bazie.

Została również sprawdzona poprawność wprowadzania, edycji czy usuwania danych przy pomocy komponentów, dostarczanych przez środowisko programistyczne Visual Studio, takich jak GridView i DetailsView.

Ponadto został modyfikowany kod źródłowy komponentu i jego pola są sprawdzane pod kontem wprowadzania poprawności danych, według wzorca.

```

        </asp:TemplateField>
        <asp:TemplateField HeaderText="nazwisko"
SortExpression="nazwisko">
            <EditItemTemplate>
                <asp:TextBox ID="TextBox2" runat="server"
Text='<%# Bind("nazwisko") %>'></asp:TextBox>
            </EditItemTemplate>
            <InsertItemTemplate>
                <asp:TextBox ID="TextBox2" runat="server"
Text='<%# Bind("nazwisko") %>'></asp:TextBox>
                <asp:RegularExpressionValidator
ID="RegularExpressionValidator2" runat="server"
ControlToValidate="TextBox2" ErrorMessage="Proszę
wprowadzić litery"
ValidationExpression="\w+[a-zA-
Z]"></asp:RegularExpressionValidator>
            </InsertItemTemplate>
            <ItemTemplate>
                <asp:Label ID="Label2" runat="server"
Text='<%# Bind("nazwisko") %>'></asp:Label>
            </ItemTemplate>

```

Fragment kodu zmodyfikowanej kontrolki DetailsView, zajmującego się walidacją pola nazwisko.

Dzieje się to za pomocą kontrolki obsługującej walidację danych, w której jest zaopatrzone Visual Studio.

CompareValidator – Porównuje odczytaną wartość pola kontrolki formularza z określoną wartością lub z wartością innego pola formularza.

CustomValidator – Pobiera i weryfikuje poprawność danych, gdy inne kontrolki walidacji nie udostępniają odpowiednich metod sprawdzania danych.

RangeValidator – Sprawdza czy odczytana wartość mieści się w podanym zakresie.

RegularExpressionValidator – Dopasowuje odczytaną wartość pola do określonego wyrażenia regularnego.

RequiredFieldValidator – Weryfikuje istnienie i typ wprowadzanej wartości.

ValidationSummary – Zbiera i łączy w komunikat wszystkie informacje o błędach, pochodzące ze wszystkich pozostałych kontrolki walidacji [2].

Ze względu na swoją funkcjonalność dokładnie został przeanalizowany panel administratora. Sprawdzone funkcje dodawania, kasowania i edycji użytkowników, wyszukiwania seansów jak i funkcje wysyłania poczty do osób znajdujących się w bazie.

Szczegółnej kontroli została poddana część modułu użytkownika, odpowiedzialna za

wybór filmu i rezerwację miejsc na wybrany seans. Pole login wyświetla właściwe nazwy zalogowanych użytkowników, odpowiednio działają też listy wyboru, w których widnieją właściwe pozycje. Program w prawidłowy sposób dokonuje zapisu wybranych przez użytkownika wartości tzn.: tytułu filmu, daty i godziny. Numer sali zostaje wygenerowany automatycznie i zgadza się z faktyczny przyporządkowanym w bazie danemu filmowi. Przy próbie zaznaczenia danych dotyczących wcześniej wybranego seansu, automatycznie generowany podgląd miejsc na sali, pokazuje w prawidłowy sposób, że wybrane we wcześniejszej sesji miejsca są już zajęte.

7. Instalacja

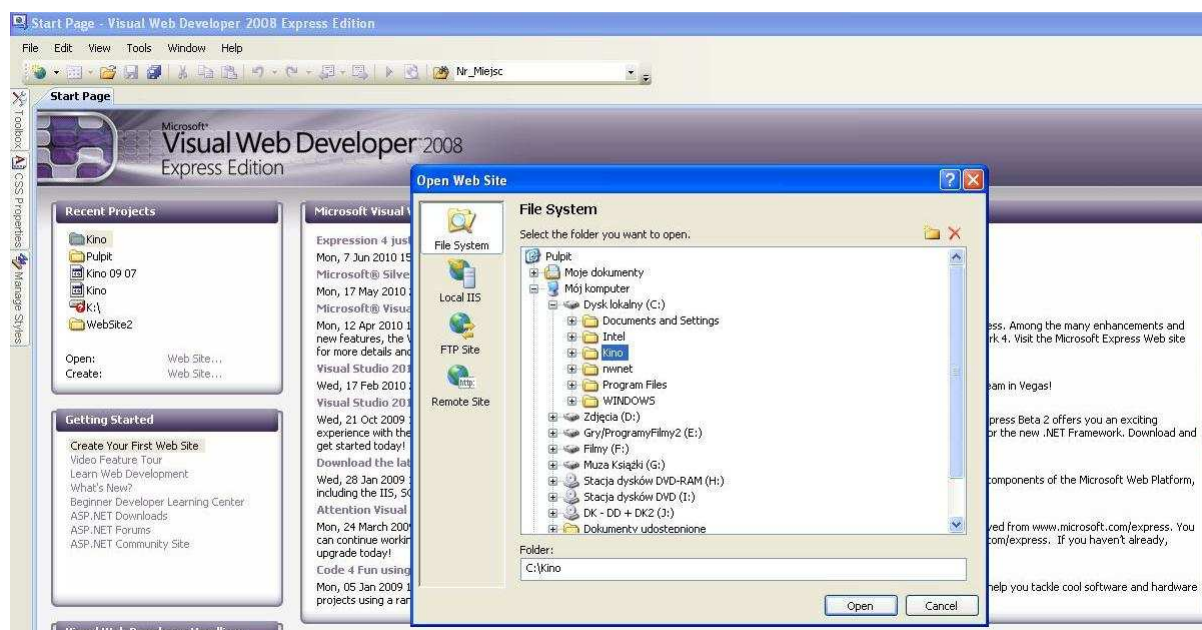
Aplikację można zainstalować na serwerze ASP, postępując zgodnie z instrukcją danego serwera. Jest to konieczne w przypadku, gdy oprogramowanie ma być używane w celach komercyjnych, jako oprogramowanie obsługujące kino. Serwery obsługujące strony ASP, i bazy danych SQL, są płatne. Na potrzeby sprawdzenia funkcji oprogramowania wystarczy posiadać darmowe środowisko programistyczne Microsoft Visual Web Developer 2008 Express, które można pobrać za darmo ze stron Microsoftu, lub ze stron zajmujących się rozpowszechnianiem darmowego oprogramowania. Visual Studio posiada wbudowany serwer IIS ([ang. Internet Information Services](#)), który jest wystarczającym do uruchamiania i testowania programów.

Aby uruchomić aplikację wystarczy skopiować Katalog kino, bezpośrednio na dysk C, lub kliknąć na płycie CD na samorozpakowywalne archiwum kino.exe. Wyświetli się okno dialogowe, w którym należy wybrać miejsce instalacji, wpisując ręcznie nazwę dysku C lub wybierając z listy. Nastąpi wypakowanie katalogu na dysk C, do utworzonego katalogu *Kino*.



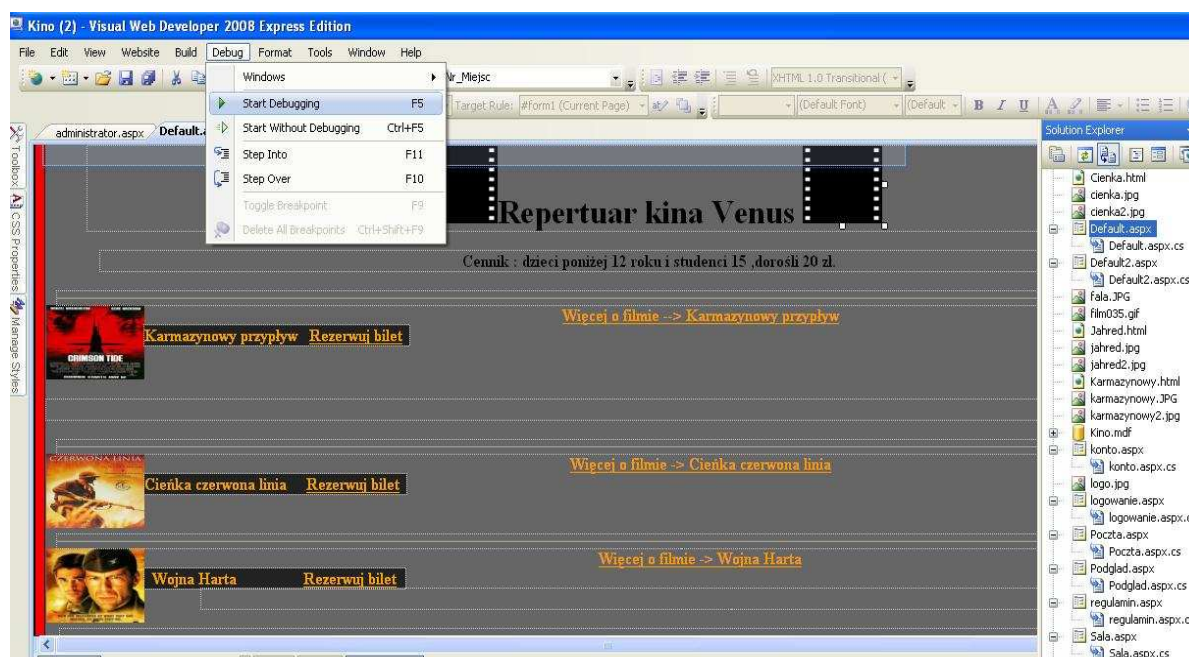
Rysunek 7.1. przedstawia okno dialogowe programu WinRAR.

Po uruchomieniu Visual Studio należy wybrać z zakładki *File*, *Open Web Site*. Zaznaczamy katalog Kino i klikamy na przycisk *Open*.



Rysunek 7.2. przedstawia okno dialogowe programu Visual Studio.

Jeśli nie otworzy się automatycznie, należy w bocznej zakładki *Solution Explorer*, wybrać plik *Default.asp* następnie z górnego menu *Debug* i *Start Debugging* lub wcisnąć klawisz F5, który również uruchamia aplikację. Po uruchomieniu nadaje się już do testowania.



Rysunek 7.3. przedstawia okno dialogowe programu Visual Studio podczas procesu debugowania.

8. Podsumowanie

W ramach pracy omówiono funkcjonalność witryn Multikina i kina Cinema City. Witryny te należą do najbardziej popularnych i najdłużej istniejących. Dokonano przeglądu technologii ASP, języka programowania C#, oraz rozbudowanych narzędzi programistycznych takich, jak Visual Studio czy Microsoft SQL Server 2005, krótko je opisując. Przemysłano i scharakteryzowano potrzeby osób wykorzystujących aplikację, wykonano diagramy przypadków użycia dla poszczególnych użytkowników. Starano się zaprojektować bazę danych która miała by być funkcjonalna, spełniać swoje zadania a zarazem ograniczyć zbędną nadmiarowość danych, dążąc do jej jak najmniejszego zwiększania swojej objętości. Jej budowę przedstawiono na schematach.

Zaprojektowano i zrealizowano w projekcie serwis kina. Zgodnie z zamierzonym celem uzyskano proste w obsłudze, funkcjonalne rozwiązanie, aplikację do której obsługi wystarczy dostęp do internetu, niepotrzebny jest proces instalacji. Użytkownik, za pośrednictwem stron internetowych, ma możliwość zapoznać się z repertuarem, cennikiem i promocjami oferowanymi przez kino, po założeniu konta uzyskuje dostęp do systemu rezerwacji, dzięki któremu, jest w stanie wybrać sobie datę, czas i miejsca na sali na interesujący go film. Wybór ten ułatwia wizualizacja podglądu dostępnych miejsc na sali kinowej, oraz zdjęcie, plakat z filmu, który aktualnie jest wybierany. Dla wygody użytkowników wprowadzono możliwość realizacji płatności przez internet za pośrednictwem strony eCard, lidera na rynku jeśli chodzi o płatności on-line w internecie.

Część aplikacji przeznaczona dla administratora wspomaga jego prace w systemie, daje mu narzędzie do sprawowania kontroli nad kontami użytkowników, możliwość podglądu stanu zajętości sali. Jedna z zakładek wspomaga wyszukiwanie osób, które dokonywał rezerwacji i śledzenie ich aktywności w systemie. Administrator ma również dostęp do panelu wysyłania poczty, dzięki któremu może kontaktować się z klientami kina drogą poczty elektronicznej.

Elastyczność aplikacji sprawia, że może ona być stosowana w kinach, do rezerwacji miejsc na imprezy multimedialne, widowiska sportowe, koncerty.

W przyszłości, aplikacji można by poszerzyć funkcjonalność o możliwość obsługi sieci kin, dodatkowych sal, dodać nowe funkcje w poszczególnych panelach, wedle zaleceń zleceńodawcy, czy na jego życzenie zmienić szatę graficzną. Można by również, zastanowić się nad pomysłem informowania użytkowników o promocjach za pośrednictwem telefonii komórkowej.

Literatura

- [1] Cinema City,
<http://www.cinemacity.pl/index.php?module=cinema&action=info&cid=1078&id=4>,
Poznań, 2009
- [2] Chłosta P. : ASP.NET i kolekcje C#. Mikom, Warszawa, 2004
- [3] Chuchra G.: Kurs SQL, http://www.centrumxp.pl/dotNet/21,1,kategoria,Kurs_SQL.aspx,
2005
- [4] Gaszewski P.: Kurs ASP.NET 2,
http://www.centrumxp.pl/dotNet/22,1,kategoria,Kurs_ASPNET_2.aspx 2005
- [5] Lis M.: C# ćwiczenia. Helion, Gliwice, 2003
- [6] Multikino, <http://www.multikino.pl/poznan/tekst,pokaz,18.html>, Poznań, 2007
- [7] Stefańczyk A.: Sekrety języka C#. Złote Myśli, Gliwice, 2005
- [8] Waymire R.: Sawtell R.: MS SQL Server 2000 dla każdego. Helion, Gliwice, 2002
- [9] Zychła W.: Programowanie pod Windows. Instytut informatyki Uniwersytetu
Wrocławskiego, Wrocław, 2003

Załączniki

Do pracy załączono w postaci elektronicznej następujące dokumenty:

- Pliki programu z kodem źródłowym
- Bazę danych Kino.mdf
- Pracę dyplomową w formacie Microsoft Word

Wszelkie potrzebne do obsługi programu hasła, znajdują się w tabeli 5.13.