

POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

PROJEKT Z BAZ DANYCH

BAZA DANYCH ORAZ APLIKACJA WEBOWA DO ZARZĄDZANIA KINEM

Termin zajęć: Piątek, 9:15–11:00

AUTORZY:

Ewa Zajdel, 226494

Monika Wójcik, 226446

Monika Skoczylas, 195348

PROWADZĄCY ZAJĘCIA:

dr inż. Roman Ptak, W4/K9

Wrocław, 2018 r.

1. Wstęp

1.1. Cel projektu

Celem projektu jest stworzenie bazy danych oraz aplikacji webowej służących do sprzedaży i rezerwacji biletów kinowych oraz do zarządzania kinem.

1.2. Zakres projektu

- Analiza wytycznych podanych przez klienta
- Pozyskanie, analiza i implementacja danych
- Stworzenie odpowiedniej bazy danych
- Wdrożenie systemu - dostosowanie oprogramowania do wymagań klienta, testowanie, uruchomienie systemu

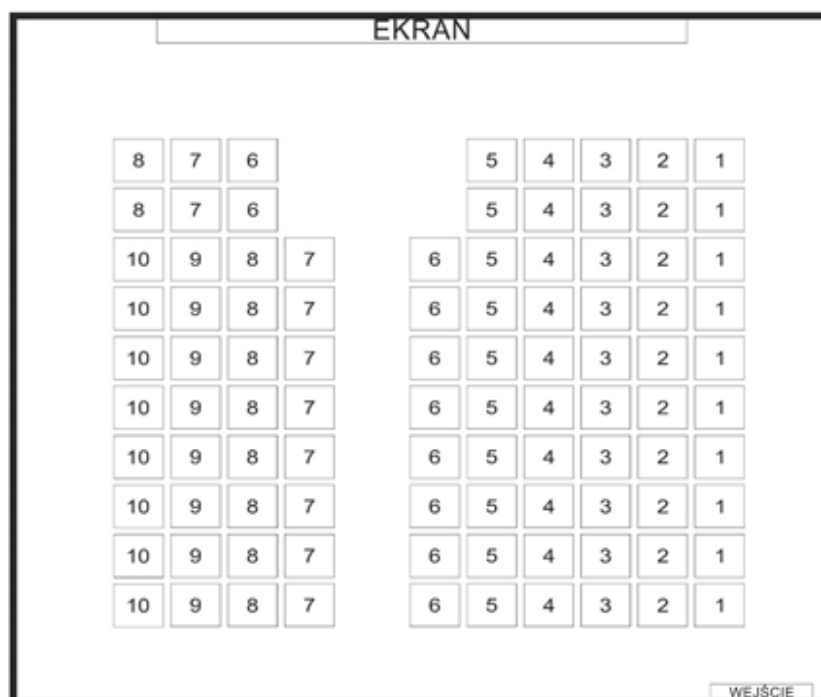
2. Analiza wymagań

2.1. Opis działania i schemat logiczny systemu

Struktura składa się z sieci kin znajdujących się w dwóch lokalizacjach. W każdym z nich znajdują się 3 sale, posiadające po około 100 miejsc. Na jedno kino przypada około 20 pracowników, którzy odpowiedzialni są za sprzedaż biletów, nadzór oraz czystość kina. Sale kinowe znajdują się na jednym piętrze. Klient wchodząc do kina przechodzi kolejno przez: kasy biletowe, strefę biletową oraz udaje się do sali. W kasach biletowych znajdują się 3 stanowiska kasowe, których system sprzedaży biletów jest połączony ze sobą tak, aby w tym samym czasie nie można było sprzedać tego samego miejsca na sali. W strefie biletowej znajduje się 2 pracowników, którzy kontrolują klientom bilety oraz nakierowują do określonej sali z seansem. Klient ma możliwość wyboru kina, seansu, rodzaju biletu, a także określonego wolnego miejsca w danej sali. Istnieją różne rodzaje biletów – podzielone ze względu na wiek (ulgowy/studencki/normalny) lub z rabatem w zależności od dnia tygodnia (Tania Środy, Piątkowy Duet). Pracownik nadzoruje sprzedaż biletów, kontroluje rezerwacje miejsc na sali oraz sprawdza bilety przed salą kinową. Kierownik ma dostęp do systemu - kontroluje czas pracy pracownika, może zarządzać użytkownikami systemu oraz salami i repertuarem.



Rys. 1 Rozkład kina w jednej z lokalizacji.



Rys. 2 Przykładowy rozkład miejsc w sali kinowej.

2.2. Wymagania funkcjonalne

Film opisany jest przez cały szereg własności, m.in. reżyser, gatunek, grupa wiekowa, dla jakiej jest przeznaczony oraz krótki opis filmu. Podstawową funkcjonalnością jest rezerwacja miejsc na dane seanse. Rezerwacja ta odbywa się pod warunkiem istnienia wolnych miejsc w sali, w której odbędzie się seans. Po sfinalizowaniu operacji zakupu bądź rezerwacji zmienia się dostępność wybranego przez użytkownika miejsca. Podczas każdego seansu będzie obecny jeden z pracowników (przypisany do seansu).

Do wszystkich zamówień zrealizowanych przez klientów ma dostęp pracownik kina po zalogowaniu się do systemu indywidualnym loginem i hasłem. Pracownik może również tworzyć zamówienia klientów. Kierownik ma możliwość śledzić czas pracy pracowników - od momentu zalogowania do systemu do momentu wylogowania. Ma dostęp do repertuaru kina i ma możliwość jego modyfikacji. Aby mieć dostęp do danych funkcji, również musi być zalogowany do systemu.

Wymagania są zależne od użytkownika aplikacji. Klient/pracownik/kierownik mają różny dostęp do możliwych funkcji.

2.2.1. Dla klienta

- Wyszukiwanie repertuaru
- Wybór miejsca
- Rezerwacja biletu na wybrany seans bez potrzeby logowania się do systemu

2.2.2. Dla pracownika

- Możliwości klienta
- Logowanie się do systemu za pomocą loginu i hasła
- Sprzedaż biletów
- Zarządzanie biletami - anulowanie
- Dostęp do informacji o zarezerwowanych przez klientów biletach
- Dostęp do informacji o sprzedanych biletach

2.2.3. Dla kierownika

- Możliwości pracownika
- Dodawanie filmów do repertuaru kina
- Usuwanie filmów z repertuaru kina
- Dodawanie seansów z repertuaru
- Zarządzanie kontami pracowników

2.3. Wymagania нефunkcjonalne

2.3.1. Wykorzystywane technologie i narzędzia

Program do rezerwacji biletów do kina, będzie działał w sieci na serwerze Apache. System operacyjny, który będzie obsługiwać to Windows 10. Do implementacji bazy danych

zostanie użyte środowisko MySQL, a do programowania język PHP. Baza danych będzie bazą relacyjną. Na początek zostanie dodanych do bazy 20 użytkowników.

2.3.2. Wymagania dotyczące rozmiaru bazy danych

- Dwie lokalizacje kina
- 2 sale kinowe w obu lokalizacjach
- Ok. 100 miejsc w każdej z sal
- Średnio 20 pracowników na kino
- Maksymalnie 30 filmów w repertuarze każdego z kin dziennie
- Seanse od 10:00 do 22:00 przez 7 dni w tygodniu
- Średnio 4 seanse dziennie w każdej z sal kinowych
- Maksymalnie 1200 sprzedanych biletów dziennie
- Ok. 5 rodzajów biletów

2.3.3. Wymagania dotyczące bezpieczeństwa systemu

- Nie będzie możliwości rejestracji przez klientów
- Możliwość logowania do systemu będą mieli tylko pracownicy kina (login+hasło)
- Szyfrowanie haseł pracowników

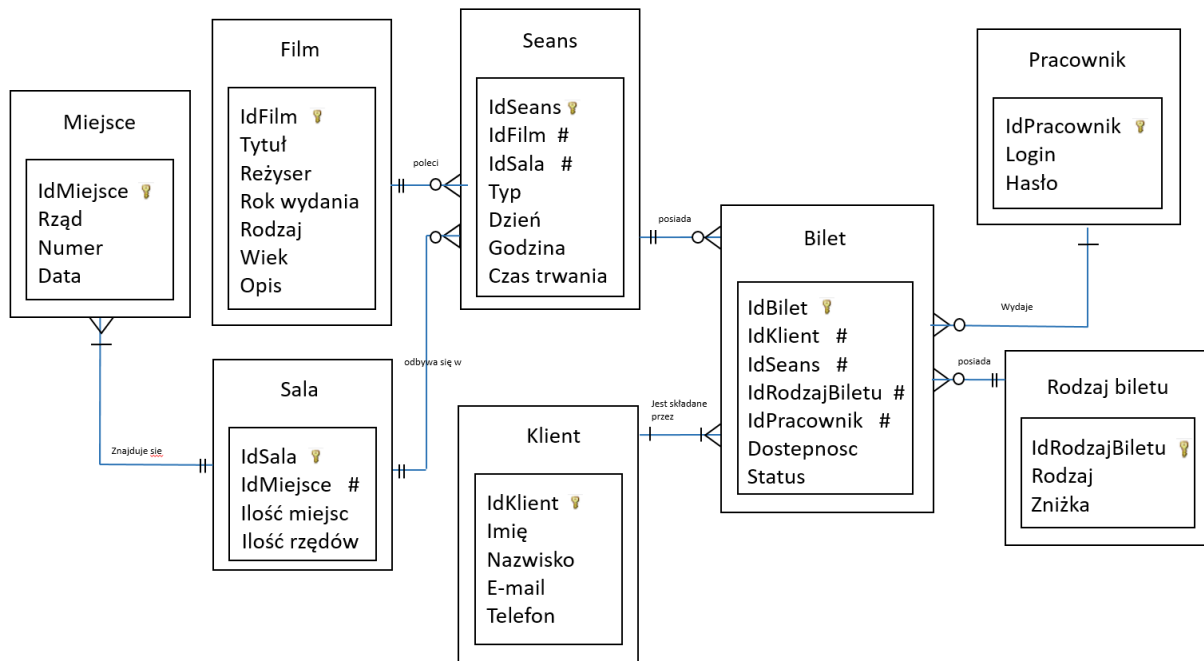
2.4. Przyjęte założenia projektowe

- Stworzenie wygodnego systemu do zakupu lub rezerwacji biletów kinowych.
- Stworzenie systemu do zarządzania siecią kin.
- System z możliwością zalogowania się przez pracowników kina.

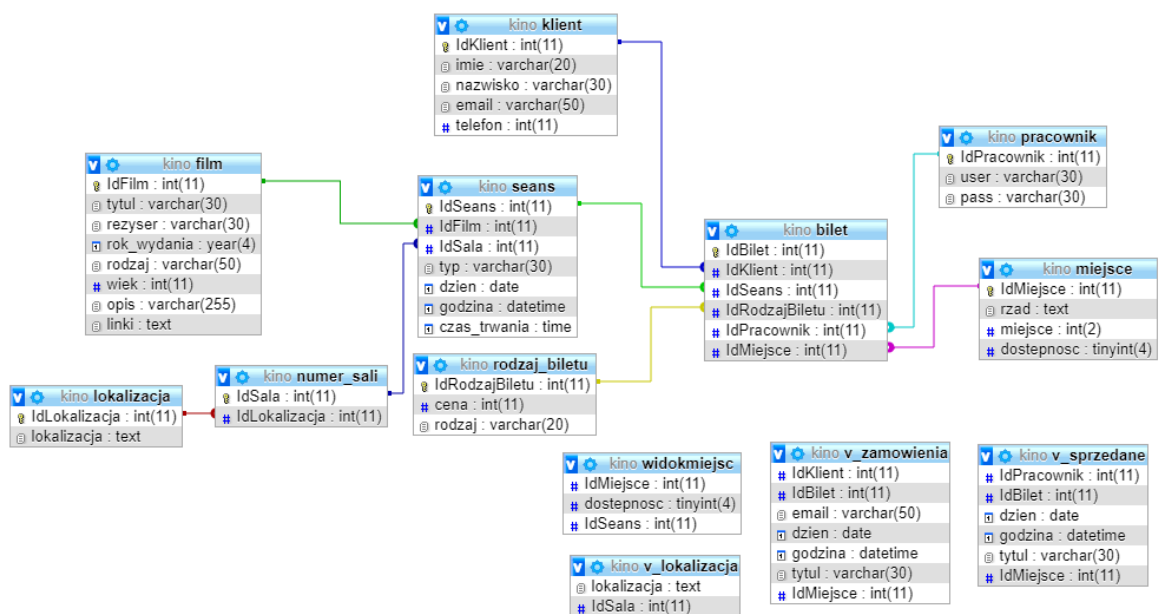
3. Projekt systemu

3.1. Projekt bazy danych

3.1.1. Model konceptualny bazy danych



3.1.2. Model fizyczny oraz logiczny bazy danych



#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Komentarze	Dodatkowo
1	IdBilet 🗝️	int(11)			Nie	Brak		AUTO_INCREMENT
2	IdKlient 🗝️	int(11)			Nie	Brak		
3	IdSeans 🗝️	int(11)			Nie	Brak		
4	IdRodzajBiletu 🗝️	int(11)			Nie	Brak		
5	IdPracownik 🗝️	int(11)			Nie	Brak		
6	dostepnosc	tinyint(1)			Nie	Brak		
7	status	tinyint(1)			Nie	Brak		

Tab. 1 Tabela Bilet

#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Komentarze	Dodatkowo
1	IdFilm 🗝️	int(11)			Nie	Brak		AUTO_INCREMENT
2	tytul	varchar(50)	utf8_polish_ci		Nie	Brak		
3	rezyser	varchar(30)	utf32_polish_ci		Tak	NULL		
4	rok_wydania	year(4)			Tak	NULL		
5	rodzaj	varchar(30)	utf8_polish_ci		Tak	NULL		
6	wiek	int(11)			Nie	Brak		
7	opis	varchar(255)	utf8_polish_ci		Tak	NULL		

Tab. 2 Tabela Film

#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Komentarze	Dodatkowo
1	IdKlient 🗝️	int(11)			Nie	Brak		AUTO_INCREMENT
2	imie	varchar(20)	utf32_polish_ci		Nie	Brak		
3	nazwisko	varchar(20)	utf32_polish_ci		Nie	Brak		
4	email	varchar(40)	utf32_polish_ci		Nie	Brak		
5	telefon	int(11)			Nie	Brak		

Tab. 3 Tabela Klient

#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Komentarze	Dodatkowo
1	IdMiejsce 🗝️	float			Nie	Brak		
2	miejsce	int(11)			Nie	Brak		
3	rzad	int(11)			Nie	Brak		
4	data	date			Nie	Brak		

Tab. 4 Tabela Miejsce

#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Komentarze	Dodatkowo
1	IdPracownik 🗝️	int(11)			Nie	Brak		AUTO_INCREMENT
2	login	varchar(30)	utf32_polish_ci		Nie	Brak		
3	haslo	varchar(30)	utf32_polish_ci		Nie	Brak		
4	imie	varchar(30)	utf8_polish_ci		Nie	Brak		
5	nazwisko	varchar(30)	utf8_polish_ci		Nie	Brak		

Tab. 5 Tabela Pracownik

#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Komentarze	Dodatkowo
1	IdRodzajBiletu 🗝️	int(11)			Nie	Brak		AUTO_INCREMENT
2	znizka	varchar(20)	utf32_polish_ci		Nie	Brak		
3	rodzaj	varchar(20)	utf32_polish_ci		Nie	Brak		

Tab. 6 Tabela Rodzaj biletu

#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Komentarze	Dodatkowo
1	IdSala 🗝️	int(11)			Nie	Brak		AUTO_INCREMENT
2	IdMiejsce 🗑️	int(11)			Nie	Brak		
3	miejsce	int(11)			Nie	Brak		
4	rzad	int(11)			Nie	Brak		

Tab. 7 Tabela Sala

#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Komentarze	Dodatkowo
1	IdSeans 🗝️	int(11)			Nie	Brak		AUTO_INCREMENT
2	IdFilm 🗑️	int(11)			Nie	Brak		
3	IdSala 🗑️	int(11)			Nie	Brak		
4	typ	varchar(30)	utf8_polish_ci		Nie	Brak		
5	dzien	date			Nie	Brak		
6	godzina	datetime			Nie	Brak		
7	czas_trwania	time			Nie	Brak		

Tab. 8 Tabela Seans

3.1.3. Inne elementy schematu – mechanizmy przetwarzania danych

- Indeks złożony występujący na dwóch kolumnach w tabeli Klient, pozwoli na szybkie wyszukiwanie konkretnej osoby oraz podglądu jego rezerwacji, czy zamówienia (imię + nazwisko)
- Poprzez indeks tytuł w tabeli Film lub godzina w tabeli Seans będziemy mogli wyszukać interesujący nas film, czy sprawdzić jakie seanse są dostępne w danej godzinie
- Indeks wiek z tabeli Film pozwoli na wyszukiwanie filmów z bieżącego repertuaru odpowiednich dla wybranej grupy wiekowej.
- Indeks rodzaj z tabeli Film pozwoli na wyszukiwanie filmów z bieżącego repertuaru wybranego gatunku filmu.
- Widok *Aktualny repertuar*:
tytuł (Film), godzina (Seans)
- Widok *Rodzaje biletów*:
imię (Klient), nazwisko (Klient), rodzajbiletu (Rodzaj Biletu)
- Widok *Miejsca*:
idSala (Sala), miejsce (Miejsce), rząd (Miejsce), dostepnosc (Bilet)
- Widok *Kupna/Rezerwacji*
idBiletu (Bilet), status (Bilet)
- Widok *Gatunek filmu*:
Tytuł(Film), Rodzaj(Film)
- Widok *Seanse w sali*:
idSala(Seans), Dzień(Seans),Godzina(Seans)
- Widok *v_zamowienia*:
IdKlient(Klient), IdBilet(Bilet), email(Klient), dzien(Seans), godzina(Seans),
tytuł(Film), IdMiejsce(Miejsce)
- Widok *v_sprzedane*:
IdPracownik(Pracownik), dzien(Seasn), godzina(Seans), tytuł(Filmt),
IdMiejsce(Miejsce)
- Widok *v_lokalizacja*:
lokalizacja(Lokalizacja), IdSala(Sala),
- W przypadku usunięcia z harmonogramu konkretnego seansu, do Klientów, którzy wykupili bilet lub zarezerwowali miejsce będzie wysyłane powiadomienie o zaistniałym fakcie
- Po finalizacji zakupu/rezerwacji biletu przez klienta, będzie wysyłane mailowe powiadomienie o udanym zakupie/udanej rezerwacji wraz z informacją o numerze biletu.

3.1.4. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych

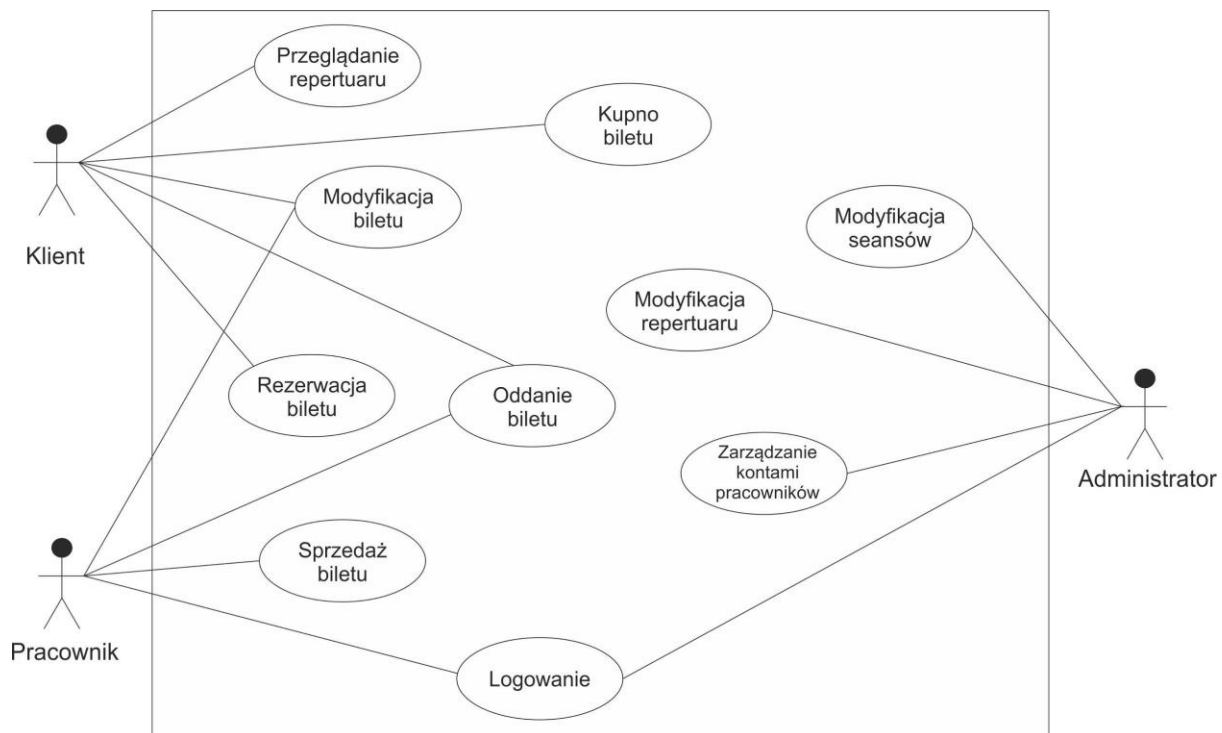
	Klient	Pracownik	Administrator
Film	R	R	CRUD
Seans	R	R	CRUD
Sala	R	R	CRUD
Klient	CRU	R	CRUD
Bilet	CRU	CRUD	CRUD
Miejsce	RU	RU	CRUD
Rodzaj biletu	R	R	CRUD
Pracownik	-	RU	CRUD

C- create utwórz
R-read odczytaj
U-update aktualizuj
D-delete usuń

Tab. 9 Uprawnienia użytkowników do poszczególnych tabel z bazy danych kina

3.2. Projekt aplikacji użytkownika

3.2.1. Architektura aplikacji i diagramy projektowe



Rys. 3 Diagram przypadków użycia

3.2.2. Interfejs graficzny i struktura menu

Na głównej stronie będzie menu, w którym będą dostępne następujące opcje do wyboru:

REPERTUAR - wyświetlenie aktualnego repertuaru na najbliższy tydzień oraz możliwość rezerwacji biletów na wybrany seans

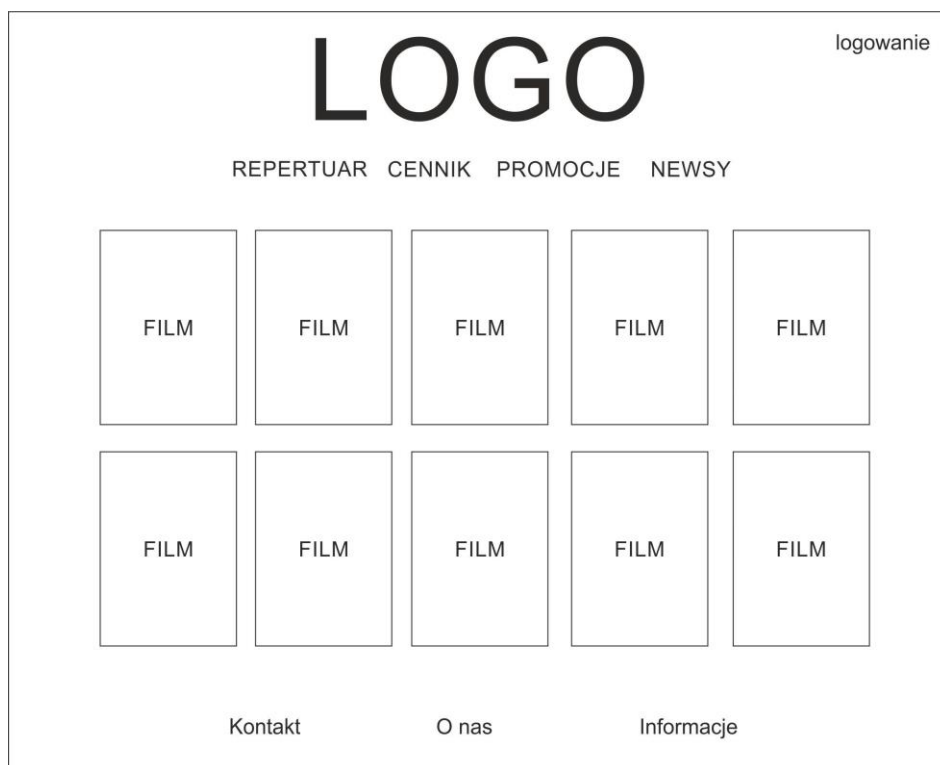
PROMOCJE - wyświetlenie listy aktualnych ofert, rabatów, pakietów dotyczących biletów kinowych

CENNIK - aktualne ceny biletów na seanse

NEWS - bieżące wiadomości o sieci kina, o premierach filmowych, komunikaty od administratora

Pod menu nawigacji będą widoczne tytuły aktualnych hitów kinowych wraz z krótkim opisem fabuły każdego filmu i najważniejszymi informacjami, które wyświetlą się po najechaniu kursorem na wybraną okładkę filmu.

W stopce aplikacji będą informacje o kinie oraz dane kontaktowe.



Rys. 4 Poglądowy rozkład strony głównej aplikacji



Rys. 5 Poglądowy wygląd strony logowania dla pracowników

LOGO

logowanie

REPERTUAR

CENNIK

PROMOCJE

NEWSY

NAZWA WYBRANEGO FILMU

Wybierz lokalizację

▼

Wybierz datę

▼

Wybierz seans

▼

REZERWUJĘ

KUPUJĘ

FILM

FILM

Kontakt

O nas

Informacje

Rys. 6 Poglądowy wygląd strony wyboru seansu do rezerwacji/kupna biletu

LOGO

logowanie

REPERTUAR

CENNIK

PROMOCJE

NEWSY

TYTUŁ FILMU

DATA

SEANS

Wybierz miejsca:

twój wybór

wolne

zajęte

EKRAN

rzęd

I

8

7

6

5

4

3

2

1

II

8

7

6

5

4

3

2

1

III

10

9

8

7

6

5

4

3

2

1

IV

10

9

8

7

6

5

4

3

2

1

V

10

9

8

7

6

5

4

3

2

1

VI

10

9

8

7

6

5

4

3

2

1

VII

10

9

8

7

6

5

4

3

2

1

VIII

10

9

8

7

6

5

4

3

2

1

IX

10

9

8

7

6

5

4

3

2

1

X

10

9

8

7

6

5

4

3

2

1

DALEJ

Kontakt

O nas

Informacje

Rys. 7 Poglądowy wygląd wyboru miejsc na seans

3.2.3. Projekt wybranych funkcji systemu

- Funkcja rezerwacji/kupna biletu przez klienta:

Po wybraniu odpowiedniego filmu z danego w aplikacji repertuaru, wyświetli się okno z możliwością wyboru daty i godziny seansu, a następnie okno z wykazem miejsc w sali, w której wybrany seans się odbywa. Klient może wybrać jedno lub kilka miejsc w sali poprzez zaznaczenie wybranych miejsc. Klient wybiera również rodzaje biletów przypisanych do danego miejsca (normalny/ulgowy itd.) Następnie pojawiają się opcje do wyboru: rezerwacja/kupno. Aplikacja przechodzi do formularza danych klienta, w której wymagane jest podanie m.in. adresu email oraz numeru telefonu, a następnie, w zależności od wybranej opcji: rezerwacja/kupno wyświetla się odpowiednio komunikat o udanej rezerwacji/płatności.

- Funkcja sprzedaży biletów przez pracownika:

Ze strony głównej po wybraniu opcji “logowanie” wyświetla się okno jak w rysunku poglądowym nr 4. Po udanym zalogowaniu system udostępnia funkcję sprzedaży biletów na seanse. Pracownik ma możliwość rezerwacji biletów dla klientów oraz ich sprzedaży. Ma również dostęp do wszystkich zamówień klientów. Pracownik może modyfikować zamówienia - anulować, zmienić rodzaj biletu itd. poprzez wybranie odpowiedniego rekordu z listy i za pomocą opcji “usuń” lub “zmień” może wykonać odpowiednią operację.

- Funkcja modyfikacji repertuaru dla administratora

Po zalogowaniu do systemu administrator będzie miał możliwość dodawania i usuwania filmów w repertuarze kina lub zmiany informacji dotyczących filmów. Będzie możliwość zaznaczenia wybranego filmu i za pomocą opcji “zmień” lub “usuń” będzie można wykonać odpowiednią modyfikację repertuaru. Opcja “dodaj” pozwala na dodanie do repertuaru filmu wraz z odpowiednimi informacjami (np. opis fabuły, minimalny wiek widza, gatunek itp.).

3.2.4. Metoda podłączania do bazy danych – integracja z bazą danych

Nawiązanie połączenia z bazą danych odbywa się poprzez utworzenie instancji klasy MySQLi (rozszerzenie PHP). Do konstruktora należy przekazać cztery wartości: nazwę hosta, nazwę użytkownika, hasło użytkownika i nazwę bazy danych, której będziemy używali.

3.2.5. Projekt zabezpieczeń na poziomie aplikacji

- Brak możliwości rejestracji nowych użytkowników (pracowników) przez aplikację
- Implementacja zabezpieczenia Captcha przy wypełnianiu formularza
- Odpowiednie określenie indywidualnych dla każdej grupy zasobów, do jakich będą mieli dostęp klienci, pracownicy i administrator w systemie
- Ograniczenie uprawnień dla klientów
- Przy wypełnianiu formularza przez klienta adres email musi mieć poprawną formę, a numer telefonu wyznaczoną liczbę cyfr
- Monitorowanie logowania do systemu oraz ich nieudane próby
- Wybór miejsca jest możliwy tylko wtedy, gdy dane miejsce nie jest zajęte
- Ograniczenia długości wyrazów i poszczególnych znaków wpisywanych w formularze
- Tylko administrator może dodać pracownika
- Zabezpieczenia przed niepożądanym otwarciem strony wymagające uwierzytelnienia
- Recaptcha przy dodawaniu klienta do bazy

4. Implementacja systemu baz danych

Implementacja i testy bazy danych w wybranym systemie zarządzania bazą danych.

4.1. Tworzenie tabel i definiowanie ograniczeń

Stworzono tabele jak w punkcie 3.1.2 (Tab.1 - Tab.8).

Stworzenie tabeli *bilet*:

```
CREATE TABLE `bilet` (  
  `IdBilet` int(11) NOT NULL,  
  `IdKlient` int(11) NOT NULL,  
  `IdSeans` int(11) NOT NULL,  
  `IdRodzajBiletu` int(11) NOT NULL,  
  `IdPracownik` int(11) NOT NULL,  
  `dostepnosc` tinyint(1) NOT NULL,  
  `status` tinyint(1) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf32 COLLATE=utf32_polish_ci;
```

Stworzenie tabeli *seans*:

```
CREATE TABLE `seans` (  
  `IdSeans` int(11) NOT NULL,  
  `IdFilm` int(11) NOT NULL,  
  `IdSala` int(11) NOT NULL,  
  `typ` varchar(30) COLLATE utf8_polish_ci NOT NULL,  
  `dzien` date NOT NULL,  
  `godzina` datetime NOT NULL,  
  `czas_trwania` time NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_polish_ci;
```

Indeksy dla tabeli *bilet*:

```
ALTER TABLE `bilet`  
  ADD PRIMARY KEY (`IdBilet`),  
  ADD UNIQUE KEY `klient_id` (`IdKlient`),  
  ADD UNIQUE KEY `IdSeans` (`IdRodzajBiletu`, `IdPracownik`),  
  ADD KEY `wydaje` (`IdPracownik`),  
  ADD KEY `ma` (`IdSeans`);
```

AUTO_INCREMENT dla tabeli *bilet*:

```
ALTER TABLE `bilet`  
  MODIFY `IdBilet` int(11) NOT NULL AUTO_INCREMENT;
```

Ograniczenia dla tabeli *seans*:

```
ALTER TABLE `seans`  
  ADD CONSTRAINT `odbywa sie w` FOREIGN KEY (`IdSala`) REFERENCES `sala`  
  (`IdSala`),  
  ADD CONSTRAINT `poleci` FOREIGN KEY (`IdFilm`) REFERENCES `film`  
  (`IdFilm`) ON UPDATE CASCADE;
```

4.2. Implementacja mechanizmów przetwarzania danych

Stworzenie indeksów:

- `CREATE INDEX indeks_imie_nazw ON klient (imie, nazwisko)`
- `CREATE INDEX indeks_tytul ON film (tytul)`
- `CREATE INDEX indeks_godzina_seansu ON seans (godzina)`
- `CREATE INDEX indeks_ogr_wieku ON film (wiek)`
- `CREATE INDEX indeks_gatunek ON film (rodzaj)`

Implementacja widoków (przykłady) :

- Widok pokazujący lokalizację sal:

```
CREATE VIEW v_lokalizacja AS
SELECT l.lokalizacja, ns.IdSala FROM numer_sali ns
LEFT JOIN lokalizacja l on l.IdLokalizacja=ns.IdLokalizacja
```

- Widok wyszukujący gatunek filmu:

```
CREATE VIEW Gatunek filmu
AS
SELECT tytul, rodzaj
FROM film;
```

gdzie możemy się później odwołać do widoku, wyszukując gatunek 'Animacja':

```
SELECT *
FROM Gatunek filmu
WHERE title LIKE 'Animacja';
```

- Widok zamówień dokonanych przez klientów:

```
CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER
VIEW v_zamowienia AS select k.`IdKlient` AS IdKlient,`b`.`IdBilet` AS
IdBilet,`k`.`email` AS email,`s`.`dzien` AS dzien,`s`.`godzina` AS
godzina,`f`.`tytul` AS tytul,`m`.`IdMiejsce` AS IdMiejsce from (((klient k
left join bilet b on((b.`IdKlient` = k.`IdKlient`))) left join seans s
on((s.`IdSeans` = b.`IdSeans`))) left join film f on((f.`IdFilm` =
s.`IdFilm`))) left join miejsce m on((m.`IdMiejsce` = b.`IdMiejsce`))) ;
```

- Widok zamówień dokonanych przez pracowników został stworzony analogicznie do powyższego.

4.3. Implementacja uprawnień i innych zabezpieczeń

Na podstawie tabeli Tab.9 nadano uprawnienia użytkownikom:

Nadanie wszystkich uprawnień w całej bazie danych *kino* dla administratora:

```
GRANT ALL PRIVILEGES ON `kino`.* TO 'administrator'@'localhost';
```

Przykładowe uprawnienia do tabel *bilet* i *miejsce* dla pracowników

```
GRANT ALL PRIVILEGES ON bilet TO 'pracownik'@'localhost';
GRANT SELECT, UPDATE ON miejsce TO 'pracownik'@'localhost';
```


Przykładowe uprawnienia do tabeli *bilet* dla klientów:

```
GRANT SELECT, UPDATE ON bilet TO 'klient'@'localhost';
```

4.4. Testowanie bazy danych na przykładowych danych

- Dodawanie rekordów do bazy:

Dodano przykładowe rekordy do wybranych tabel, np. w tabeli *film* dodano 7 filmów.

Przykład dodawania filmu:

```
INSERT INTO `film` (`IdFilm`, `tytul`, `rezyser`, `rok_wydania`, `rodzaj`, `wiek`, `opis`)
VALUES
```

(1, 'Kształt wody', 'Guillermo del Toro', 2017, 'Fantasy', 12, 'Elisa Esposito pracuje jako woźna w tajnym rządowym laboratorium. Jej życie ulega nagłej zmianie, gdy odkrywa pilnie strzeżony sekret.'),

IdFilm	tytul	rezyser	rok_wydania	rodzaj	wiek	opis
1	Kształt wody	Guillermo del Toro	2017	Fantasy	12	Elisa Esposito pracuje jako woźna w tajnym rządowy...
2	Czas mroku	Joe Wright	2017	Biograficzny	12	Winston Churchill zostaje premierem Wielkiej Bryta...
3	Czwarta władza	Steven Spielberg	2017	Dramat	0	Działania prezydenta USA doprowadzają do konfliktu...
4	Tamte dni, tamte noce	Luca Guadagnino	2017	Melodramat	12	Nastoletni chłopak zakochuje się w gościu, który p...
5	Jaskiniowiec	Nick Park	2018	Animacja	0	W czasach, kiedy po ziemi chodziły dinozaury i mam...
6	Twój Vincent	Dorota Kobiela	2017	Animacja	6	Bohaterowie obrazów Vincenta van Gogha przedstawia...
7	Tomb Raider	Roar Uthaug	2018	Przygodowy	12	Lara Croft wyrusza w swoją pierwszą ekspedycję, ab...

Tab. 10 Tabela *film* z bazy *kino*

- Wyświetlanie widoku tabeli pracownik

```
SELECT * FROM pracownik;
```

IdPracownik	login	haslo
1	annakowalska	*2C0CAF33D242118E36D9D363F41B0
2	jankowalski	*7F74DA13EDAC1E8AB3FCD7E690839
3	piotrnowak	*F7476159976A0FBC599CB16E87335

Tab. 11 Tabela *pracownik* z bazy *kino*

Dodawanie wartości do tabel przebiega prawidłowo. Dane są poprawnie odczytywane wraz z napisami w języku polskim.

Gdy kolumna jest oznaczona jako NOT NULL w trakcie zapisu występuje błąd, który należy poprawić, bądź do tabeli są wpisywane 4 zera (0000).

Hasło pracownika jest zmienną Password, więc jest zaszyfrowane.

- Wybieranie rekordów z bazy

Przykład wybrania wartości z tabeli *film*:

```
SELECT tytul, rezyser, wiek FROM film;
```

tytuł	reżyser	wiek
Kształt wody	Guillermo del Toro	12
Czas mroku	Joe Wright	12
Czwarta władza	Steven Spielberg	0
Tamte dni, tamte noce	Luca Guadagnino	12
Jaskiniowiec	Nick Park	0
Twój Vincent	Dorota Kobiela	6

Tab. 12 Wybrane rekordy z bazy

SELECT * FROM film WHERE rodzaj = 'sensacyjny'

IdFilm	tytuł	reżyser	rok_wydania	rodzaj	wiek	opis
8	Pitbull. Ostatni pies	Władysław Pasikowski	2018	Sensacyjny	15	Po śmierci Soczka policjanci z warszawskiej komend...
10	Kobiety mafii	Patryk Vega	2018	Sensacyjny	15	Bela, była funkcjonariuszka policji, dostaje od AB...

Tab. 13 Wynik zapytania SELECT

5. Implementacja i testy aplikacji

5.1. Instalacja i konfigurowanie systemu

Aby aplikacja działała poprawnie należy pobrać oraz zainstalować na komputerze serwer lokalny xampp. Po uruchomieniu panelu kontrolnego xampp należy załączyć moduł: Apache oraz MySQL. W przeglądarkę internetową należy wpisać localhost/phpmyadmin, następnie utworzyć nową bazę danych o nazwie kino i wgrać bazę danych aplikacji kino.sql. Wszystkie pliki powinny znajdować się w folderze htdocs. Aby się tam dostać, trzeba wejść w dysk, na którym został zainstalowany xampp, następnie w folder xampp, oraz htdocs (przykładowa ścieżka do folderu: C:\xampp\htdocs) Gdy to już zostanie wykonane, w nowym oknie przeglądarki wpisać adres localhost/kino/mem.php. Aplikacja została poprawnie uruchomiona.

Aby wgrać pliki na serwer ftp potrzebny jest program, który to umożliwi oraz dane do serwera ftp (adres hosta, login i hasło). W programie (np. FileZilla) należy połączyć się z serwerem podając adres serwera, login oraz hasło, a po udanym połączeniu “przenieść” wybrane pliki (*.php oraz kino.sql) z okna programu z zawartością dysku do wybranego katalogu z okna z zawartością serwera ftp. Po udanym przesłaniu plików można wylogować się z serwera i zamknąć program. Aplikacja powinna poprawnie działać na wybranym serwerze.

5.2. Instrukcja użytkowania aplikacji

5.2.1. Instrukcja obsługi zakupu biletu dla klienta

1. Otworzyć plik mem.php
2. Wejść w repertuar
3. Wybrać film
4. Wybrać dzień oraz godzinę seansu w wybranej lokalizacji
5. Wybrać miejsca
6. Wybrać ilość i rodzaje biletów
7. Wypełnić formularz danymi klienta

5.2.2. Instrukcja obsługi sprzedaży biletu dla pracownika

1. Otworzyć plik mem.php
2. Zalogować się
3. Wybrać film
4. Wybrać dzień oraz godzinę seansu
5. Wybrać miejsce
6. Wybrać ilość i rodzaje biletów
7. Zatwierdzić wybór

5.2.3. Instrukcja obsługi dodawania filmu dla pracownika

1. Otworzyć plik mem.php
2. Zalogować się
3. Wejść w dodawanie filmu
4. Uzupełnić dane filmu

5.2.4. Instrukcja obsługi dodawania seansu dla pracownika

1. Otworzyć plik mem.php
2. Zalogować się
3. Wejść w dodawanie seansu
4. Uzupełnić dane seansu

5.3. Testowanie opracowanych funkcji systemu

5.3.1. Przejście do stron niedostępnych dla klienta

Wpisując nazwy stron w przeglądarkę, które są dostępne po uwierzytelnieniu tylko dla pracowników aplikacja przenosi nas do strony logowanie.php, gdzie należy podać dane pracownika, aby przejść dalej.

5.3.2. Próba zarezerwowania zajętego miejsca

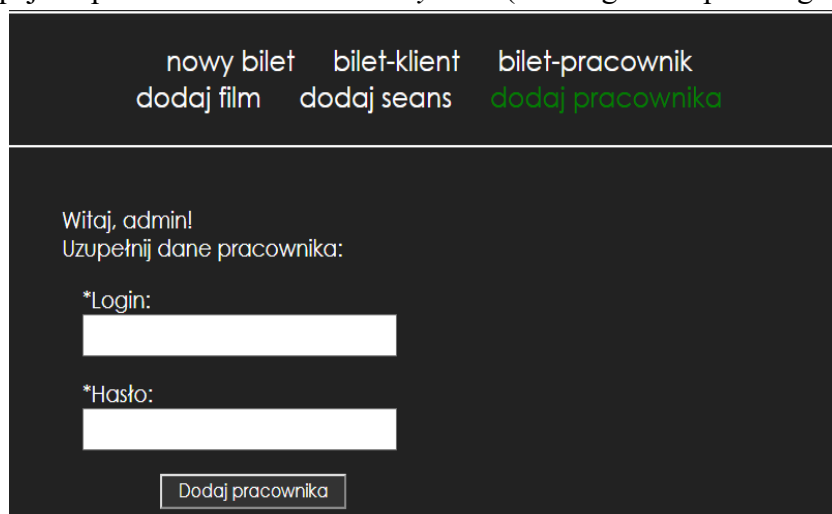
Miejsce, które zostało już zajęte zamiast checkboxa wyświetla ikonę która nie jest interaktywna - brak możliwości ponownego wybrania tego miejsca.



Rys. 8 Przykładowe miejsca wolne (checkboxy) i zarezerwowane (ikony).

5.3.3. Dodanie pracownika przez administratora

Do programu zostało dodane zabezpieczenie, które ma za zadanie sprawdzać, kto jest zalogowany, a do strony *dodaj pracownika* dać dostęp tylko administratorowi. Dla pracownika opcja ta przenosi do zakładki *nowy bilet* (strona główna po zalogowaniu).



Rys. 9 Zrzut z aplikacji – dodawanie pracownika

5.3.4. Dodawanie seansu

Pracownik oraz administrator ma możliwość dodania seansu. Aby dodać seans należy wejść na stronę *dodanie seansu*, gdzie pojawi się formularz. Aby dodawanie było łatwiejsze, dane zostały pobrane z bazy oraz przedstawione za pomocą listy rozwijanej. Typ pól do wpisywania danych zgadza się z typem danych w bazie.

Dodaj dane seansu:

Film:

Sala:

Typ:

Wybierz datę:

Wybierz godzinę:

Czas trwania:

Rys. 10 Zrzut z aplikacji – dodawanie seansu

5.3.5. Anulowanie zamówienia

Na stronie *baza-klient* dostępnej tylko po uwierzytelnieniu jest wyświetlona baza wszystkich biletów zakupionych przez klienta. Pracownik ma możliwość anulowania biletu. Aby anulować bilet należy wpisać odpowiedni numer biletu, który chcemy usunąć oraz zatwierdzić wybór. Bilet automatycznie usuwa się z bazy.

*Numer zamówienia:

Wszystkie prawa zastrzeżone Kino Mem 2018 ©

Numer biletu	Tytuł	Data	Numer miejsca	Email
130	Coco	2018-06-16 09:00:00	34	monika.wojcik5@gmail.com
131	Coco	2018-06-16 09:00:00	35	monika.wojcik5@gmail.com
132	Coco	2018-06-16 09:00:00	36	monika.wojcik5@gmail.com

Rys.11 Zrzut z aplikacji – usuwanie zamówienia (widok przed usunięciem)

*Numer zamówienia:

Wszystkie prawa zastrzeżone Kino Mem 2018 ©

Numer biletu	Tytuł	Data	Numer miejsca	Email
130	Coco	2018-06-16 09:00:00	34	monika.wojcik5@gmail.com
132	Coco	2018-06-16 09:00:00	36	monika.wojcik5@gmail.com
133	Coco	2018-06-16 09:00:00	37	monika.wojcik5@gmail.com

Rys.12 Zrzut z aplikacji – usuwanie zamówienia (widok po usunięciu)

5.4. Omówienie wybranych rozwiązań programistycznych

5.4.1. Implementacja interfejsu dostępu do bazy danych

Aby nawiązać połączenie z bazą danych należy skorzystać z rozszerzenie PHP o nazwie MySQLi.

Listing 1: Nawiązanie połączenia z bazą danych kino

```
<?php
$con=mysqli_connect("localhost","root", "", "kino");
?>
```

Na powyższym listingu “localhost” jest nazwą serwera mysql, “root” to login administratora bazy danych, kolejnym elementem jest hasło (tutaj brak), natomiast “kino” jest nazwą bazy danych, z którą nawiązujemy połączenie.

Aby odczytać wybrane dane z bazy, z którą nawiązano połączenie, należy podać odpowiednie zapytanie SQL. Przykładowe zapytania:

Listing 2: Zapytanie SQL w kodzie aplikacji - cena biletu normalnego

```
$cena = mysqli_fetch_array(mysqli_query($con, 'SELECT `cena` FROM `rodzaj_biletu` WHERE `rodzaj`="normalny"'));
```

Listing 3: Zapytanie SQL w kodzie aplikacji - dodanie danych klienta do bazy

```
//Wszystkie testy zaliczone, dodanie danych klienta do bazy
if ($polaczenie->query("INSERT INTO klient VALUE (NULL, '$imie', '$nazwisko', '$email', '$tel')"))
```

5.4.2. Implementacja wybranych funkcjonalności systemu

Listing 4: Pobieranie danych z bazy i wyświetlanie w tabeli – repertuar

```
<form method="GET">
  <h1>Aktualnie gramy:</h1>
  <table width="600px" border = "1" cellpadding = "1" cellspacing = "1">
    <tr><td><b>Tytuł</b></td><td><b>Ograniczenie wiekowe</b></td><td><b>Gatunek</b></td></tr>
    <?php
    $seanse = mysqli_query($con, 'SELECT * FROM `film`');
    while ($rec = mysqli_fetch_array($seanse))
    {
      echo '<tr>

      <td><a href="repertuar_wybor_daty.php?n1='.$rec['tytul'].'">'.$rec['tytul'].'</a></td>

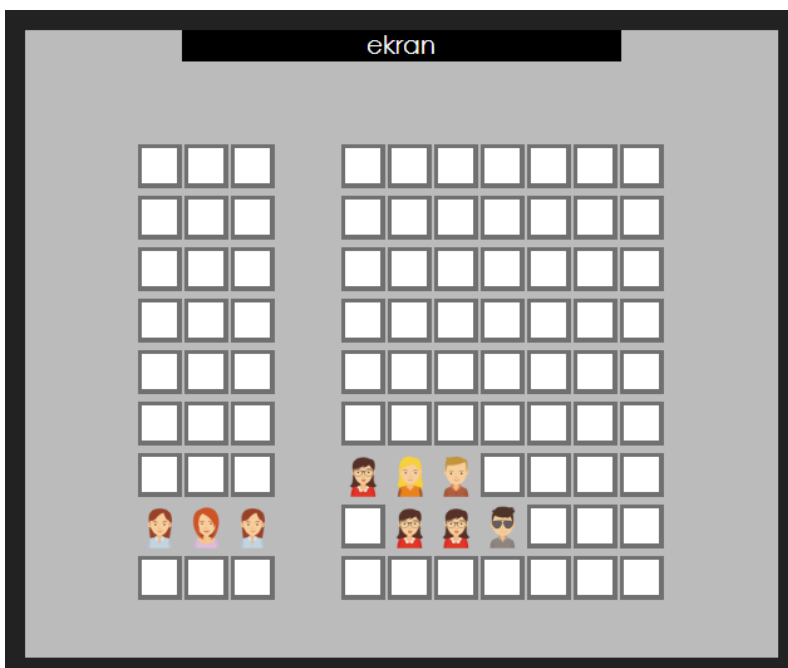
      <td>'.$rec['wiek'].'</td>
      <td>'.$rec['rodzaj'].'</td>

      </tr>';
    }
    ?>
  </table>
</form>
```

Aktualnie gramy:		
Tytuł	Ograniczenie wiekowe	Gatunek
Kształt wody	12	Fantasy
Deadpool 2	12	Sci-Fi
Czwarta władza	0	Dramat
Tamte dni, tamte noce	12	Melodramat
Coco	0	Animacja
Twój Vincent	6	Animacja
Tomb Raider	12	Przygodowy
Pitbull. Ostatni pies	15	Sensacyjny
Czerwona jaskółka	15	Thriller
Kobiety mafii	15	Sensacyjny
Cudowny chłopak	6	Familijny
Luis i obcy	0	Animacja
Lady Bird	12	Dramat

Rys. 13 Repertuar - wynik implementacji kodu z Listingu 4

Listing 5: Pobranie danych z bazy i wyświetlenie - miejsca w sali kinowej

[illegible]

Rys. 14 Miejsca w sali kinowej - wynik implementacji kodu z Listingu 5

Listing 6: Pobieranie danych z widoku i wyświetlanie - bilety sprzedane

```
<?php
echo '<center><table width="800px" border = "1" cellpadding = "1" cellspacing = "1">
<tr><td><b>Numer biletu</b></td><td><b>Tytuł</b></td><td><b>Data</b></td><td><b>Numer
miejsca</b></td><td><b>IdPracownika</b></td></tr>';
$bilety = mysqli_query($con, 'SELECT * FROM v_sprzedane WHERE IdBilet IS NOT NULL');
while ($rec = mysqli_fetch_array($bilety))
{
    echo '<tr>
<td>'.$rec['IdBilet'].'</td>
<td>'.$rec['tytul'].'</td>
<td>'.$rec['godzina'].'</td>
<td>'.$rec['IdMiejsce'].'</td>
<td>'.$rec['IdPracownik'].'</td>
</tr>';
} '</center>';
?>
```


Numer biletu	Tytuł	Data	Numer miejsca	IdPracownika
165	Coco	2018-06-16 15:00:00	75	1
166	Coco	2018-06-16 15:00:00	76	1
173	Deadpool 2	2018-06-18 21:00:00	71	1
174	Deadpool 2	2018-06-18 21:00:00	72	1
175	Deadpool 2	2018-06-18 21:00:00	73	1

Rys. 15 Sprzedane bilety - wynik implementacji kodu z Listingu 6

5.4.3. Implementacja mechanizmów bezpieczeństwa

- Logowanie do systemu

Na stronie *logowanie.php* użytkownik proszony jest o wypełnienie pól odpowiednimi danymi - loginem oraz hasłem przypisanym do konta. Po wpisaniu następuje identyfikacja użytkownika - czy użytkownik o takiej nazwie istnieje w bazie oraz uwierzytelnienie - sprawdzenie, czy podane hasło jest prawidłowe, pasujące do podanego loginu. Jeśli nie podano hasła/loginu bądź są one błędne, zostaje wyświetlony formularz wraz z informacjami o błędach. W przypadku, gdy dane są prawidłowe, następuje autoryzacja, rozpoczyna się sesja i zostaje zarejestrowana zmienna *\$zalogowany*. Następuje przekierowanie pracownika do chronionej strony *repertuar_pracownik.php*. Funkcja *session_start()* będzie wznawiała sesję rozpoczętą podczas logowania. Próby nieautoryzowanego dostępu do strony będą niepowodzeniem, ponieważ nie będą zawierały ustawionej zmiennej *\$zalogowany*, a więc niezalogowany użytkownik zostanie przekierowany do skryptu uwierzytelniającego (formularz logowania na stronie *logowanie.php*).

Aby wylogować użytkownika z systemu należy wznović sesję (*session_start*), a następnie ją zakończyć (*session_unset*). Wylogowanie następuje po naciśnięciu przycisku *wyloguj się*.

Listing 7: Implementacja formularza do logowania dla pracowników

```
<form action="zaloguj.php" method="post" >
    Login:<br><input type="text" name="login" required="required"/><br><br>
    Haslo:<br><input type="password" name="haslo" required="required"/><br><br>
    <input type="submit" value="Zaloguj się" style="margin-left: 75px; width: 150px; height: 30px; font-size: 16px; color: #ffffff; background-color: #333333;"/>
</form>
```

Listing 8: Sprawdzenie, czy istnieje zmienna sesyjna

```
session_start();

if ((isset($_SESSION['zalogowany'])) && ($_SESSION['zalogowany']==true))
{
    header('Location: repertuar_pracownik.php');
    exit();
}
```

- Zabezpieczenia przy wypełnianiu formularza

Wypełnienie formularza z danymi odbywa się po poprawnym wybraniu seansu oraz miejsc w sali. W formularzu należy podać imię, nazwisko, email oraz numer telefonu, zaakceptować regulamin i zatwierdzić zabezpieczenie Recaptcha. Wszystkie dane muszą być poprawnie wypełnione. Dane zostają przesłane do bazy do tabeli klient oraz bilet. W tabeli klient zostają wpisane dane klienta, a w tabeli bilet tworzona jest odpowiednia ilość wierszy w zależności od ilości zakupionych biletów.

The image shows a registration form with the following fields and error messages:

- *Imię:** [Empty text box] - Error: *Zawiera niedozwolone znaki*
- *Nazwisko:** [Empty text box] - Error: *Zawiera niedozwolone znaki*
- *E-mail:** [Empty text box] - Error: *Podaj poprawny adres e-mail.*
- *Telefon:** [Empty text box] - Error: *Zawiera niedozwolone znaki*
- ☒ **Akceptuję regulamin.***
- ☐ **Nie jestem robotem** - Includes a reCAPTCHA logo and the text "reCAPTCHA Prywatność - Warunki". Below this is the error message: *Potwierdź, że nie jesteś botem.*

Rys. 16 Widok ostrzeżeń przy błędnym wypełnieniu formularza (brak wprowadzonych danych)

Listing 9: Zabezpieczenia poprawności wprowadzonych danych

```
//Sprawdzenie długości imienia i nazwiska
if ((strlen($imie)<3) || (strlen($imie)>25))
{
    $wszystko_OK=false;
    $_SESSION['e_imie']="Od 3 do 25 znaków!";
}

if ((strlen($nazwisko)<2) || (strlen($nazwisko)>50))
{
    $wszystko_OK=false;
    $_SESSION['e_nazwisko']="Od 2 do 50 znaków!";
}

//Sprawdzenie poprawności imienia i nazwiska
if (!preg_match('/^[a-zA-ĆĘŁŃÓŚŻ]+$/ui', $imie))
{
    $wszystko_OK=false;
    $_SESSION['e_imie']="Zawiera niedozwolone znaki";
}

if (!preg_match('/^[a-zA-ĆĘŁŃÓŚŻ\~]+$/ui', $nazwisko))
{
    $wszystko_OK=false;
    $_SESSION['e_nazwisko']="Zawiera niedozwolone znaki";
}
```

Listing 10: Sprawdzenie poprawności email

```
$emailB=filter_var($email, FILTER_SANITIZE_EMAIL);
if ((filter_var($emailB, FILTER_VALIDATE_EMAIL)==false) || ($emailB!=$email))
{
    $wszystko_OK=false;
    $_SESSION['e_email']="Podaj poprawny adres e-mail.";
}
```

Listing 11: Sprawdzenie poprawności numeru telefonu - musi się składać z 9 cyfr:

```
if (strlen($tel)!=9)
{
    $wszystko_OK=false;
    $_SESSION['e_tel']="Podaj 9 cyfr bez prefiksu.";
}
if (!preg_match('/^[0-9]+$/i', $tel))
{
    $wszystko_OK=false;
    $_SESSION['e_tel']="Zawiera niedozwolone znaki";
}
```

- Sprawdzenie, czy ilość miejsc zgadza się z ilością wybranych biletów (brak możliwości przejścia do formularza w przypadku błędu)

Listing 12: Sprawdzenie poprawności wprowadzonej ilości miejsc

```
if (($normalny+$ulgowy+$senior+$student)==$iloscmiejsc)
{
    //echo 'super';

    $scenan = mysqli_fetch_array(mysqli_query($con, 'SELECT `cena` FROM `rodzaj_biletu` WHERE
`rodzaj`="normalny"'));
    $scenau = mysqli_fetch_array(mysqli_query($con, 'SELECT `cena` FROM `rodzaj_biletu` WHERE `rodzaj`="ulgowy"
));
    $scenast = mysqli_fetch_array(mysqli_query($con, 'SELECT `cena` FROM `rodzaj_biletu` WHERE
`rodzaj`="student"'));
    $scenase = mysqli_fetch_array(mysqli_query($con, 'SELECT `cena` FROM `rodzaj_biletu` WHERE `rodzaj`="senior"
));

    $suma = $normalny*$scenan['cena']+$ulgowy*$scenau['cena']+$student*$scenast['cena']+$senior*$scenase['cena'];
    echo 'PODSUMOWANIE ZAMÓWIENIA<br>Do zapłaty: '.$suma.'.zł<br><br>';

    echo '<input type="submit" name="dalej" style=" margin-left: 300px; width: 150px; height: 30px; font-size:
16px; color: #ffffff; background-color: #333333;" value="Potwierdź">';
}
else
{
    echo 'Ilość wybranych miejsc nie zgadza się z ilością biletów. Cofnij stronę i popraw ilość.';
}
```

6. Podsumowanie i wnioski

Projekt został wykonany zgodnie z wcześniej ustalonymi założeniami. Baza danych MySQL pozwala na płynną współpracę z zaimplementowaną aplikacją webową. Została stworzona aplikacja w technologii PHP i MySQL z interfejsem wykonanym w technologii HTML oraz CSS, która umożliwia w prosty sposób zarezerwowanie miejsca na wybrany seans, a także umożliwia zarządzanie kinem, np. dodawanie seansów. Zostały zaimplementowane zabezpieczenia ułatwiające łączenie aplikacji z bazą. Zaimplementowane funkcje działają poprawnie - brak błędów podczas działania (rezerwacji biletów, dodania pracownika itp.).

Zostały wykonane testy sprawdzające poprawność działania aplikacji oraz jej połączenia z bazą danych. Wynik testów jest pozytywny.

Projekt umożliwił nam zapoznanie się z technologią HTML oraz CSS, a także dał możliwość poznania nowego języka programowania PHP w ciągu niecałych dwóch miesięcy. W trakcie realizacji projektu nauczyliśmy się podstawowych oraz bardziej zaawansowanych komend MySQL.

Bibliografia

- 1) http://roman.ptak.staff.iiar.pwr.wroc.pl/BD_wyklad_nr1_ver6.pdf
- 2) <http://it.dth.pl/create-update-alter-view-tworzenie-i-modyfikacja-widokow-kurs-jezyka-sql/>
- 3) <https://stackoverflow.com>
- 4) Duckett Jon “HTML i CSS”, HELION, 2014
- 5) (Pasja Informatyki, Kurs PHP. Programowanie backendowe)
(<https://youtu.be/tD0Q5QwoQJI>)

Spis rysunków

- Rys. 1 Rozkład kina w jednej z lokalizacji
- Rys. 2 Przykładowy rozkład miejsc w sali kinowej
- Rys. 3 Diagram przypadków użycia
- Rys. 4 Poglądowy rozkład strony głównej aplikacji
- Rys. 5 Poglądowy wygląd strony logowania dla pracowników
- Rys. 6 Poglądowy wygląd strony wyboru seansu do rezerwacji/kupna biletu
- Rys. 7 Poglądowy wygląd wyboru miejsc na seans
- Rys. 8 Zrzut z aplikacji – widok zajętych miejsc
- Rys. 9 Zrzut z aplikacji – dodawanie pracownika
- Rys. 10 Zrzut z aplikacji – dodawanie seansu
- Rys. 11 Zrzut z aplikacji – usuwanie zamówienia (widok przed usunięciem)
- Rys. 12 Zrzut z aplikacji – usuwanie zamówienia (widok po usunięciu)
- Rys. 13 Repertuar - wynik implementacji kodu z Listingu 4
- Rys. 14 Miejsca w sali kinowej - wynik implementacji kodu z Listingu 5
- Rys. 15 Sprzedane bilety - wynik implementacji kodu z Listingu 6
- Rys. 16 Widok ostrzeżeń przy błędnym wypełnieniu formularza (brak wprowadzonych danych)

Spis tabel

- Tab. 1 Tabela Bilet
- Tab. 2 Tabela Film
- Tab. 3 Tabela Klient
- Tab. 4 Tabela Miejsce
- Tab. 5 Tabela Pracownik
- Tab. 6 Tabela Rodzaj biletu
- Tab. 7 Tabela Sala
- Tab. 8 Tabela Seans
- Tab. 9 Uprawnienia użytkowników do poszczególnych tabel z bazy danych kina

Tab. 10 Tabela film z bazy kino
Tab. 11 Tabela pracownik z bazy kino
Tab. 12 Wybrane rekordy z bazy
Tab. 13 Wynik zapytania SELECT

Spis listingów

Listing 1: Nawiązanie połączenia z bazą danych kino
Listing 2: Zapytanie SQL w kodzie aplikacji - cena biletu normalnego
Listing 3: Zapytanie SQL w kodzie aplikacji - dodanie danych klienta do bazy
Listing 4: Pobieranie danych z bazy i wyświetlanie w tabeli – repertuar
Listing 5: Pobranie danych z bazy i wyświetlenie - miejsca w sali kinowej
Listing 6: Pobieranie danych z widoku i wyświetlanie - bilety sprzedane
Listing 7: Implementacja formularza do logowania dla pracowników
Listing 8: Sprawdzenie, czy istnieje zmienna sesyjna
Listing 9: Zabezpieczenia poprawności wprowadzonych danych
Listing 10: Sprawdzenie poprawności email
Listing 11: Sprawdzenie poprawności numeru telefonu - musi się składać z 9 cyfr
Listing 12: Sprawdzenie poprawności wprowadzonej ilości miejsc

Spis treści

1. Wstęp	2
1.1. Cel projektu	2
1.2. Zakres projektu.....	2
1. Analiza wymagań.....	2
2.1. Opis działania i schemat logiczny systemu	2
2.2. Wymagania funkcjonalne.....	3
2.2.1. Dla klienta.....	4
2.2.2. Dla pracownika.....	4
2.2.3. Dla kierownika	4
2.3. Wymagania niefunkcjonalne.....	4
2.3.1. Wykorzystywane technologie i narzędzia	4
2.3.2. Wymagania dotyczące rozmiaru bazy danych	5
2.3.3. Wymagania dotyczące bezpieczeństwa systemu.....	5
3. Projekt systemu	6
3.1. Projekt bazy danych	6
3.1.1. Model konceptualny bazy danych	6
3.1.2. Model fizyczny oraz logiczny bazy danych	6
3.1.3. Inne elementy schematu – mechanizmy przetwarzania danych.....	9
3.1.4. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych	10
3.2. Projekt aplikacji użytkownika.....	11
3.2.1. Architektura aplikacji i diagramy projektowe	11
3.2.2. Interfejs graficzny i struktura menu.....	11
3.2.3. Projekt wybranych funkcji systemu	14
3.2.4. Metoda podłączania do bazy danych – integracja z bazą danych	14
3.2.5. Projekt zabezpieczeń na poziomie aplikacji	14
4. Implementacja systemu baz danych.....	15
4.1. Tworzenie tabel i definiowanie ograniczeń	15
4.2. Implementacja mechanizmów przetwarzania danych.....	16
4.3. Implementacja uprawnień i innych zabezpieczeń.....	16
4.4. Testowanie bazy danych na przykładowych danych	17

5. Implementacja i testy aplikacji	18
5.1. Instalacja i konfigurowanie systemu	18
5.2. Instrukcja użytkowania aplikacji.....	19
5.2.1. Instrukcja obsługi zakupu biletu dla klienta	19
5.2.2. Instrukcja obsługi sprzedaży biletu dla pracownika.....	19
5.2.3. Instrukcja obsługi dodawanie filmu dla pracownika.....	19
5.2.4. Instrukcja obsługi dodawania seansu dla pracownika	19
5.3. Testowanie opracowanych funkcji systemu.....	19
5.3.1. Przejście do stron niedostępnych dla klienta	19
5.3.2. Próba zarezerwowania zajętego miejsca	20
5.3.3. Dodanie pracownika przez administratora	20
5.3.4. Dodawanie seansu	20
5.3.5. Anulowanie zamówienia	21
5.4. Omówienie wybranych rozwiązań programistycznych	22
5.4.1. Implementacja interfejsu dostępu do bazy danych.....	22
5.4.2. Implementacja wybranych funkcjonalności systemu	23
5.4.3. Implementacja mechanizmów bezpieczeństwa	25
6. Podsumowanie i wnioski	28
Bibliografia	29
Spis rysunków	29
Spis tabel	29
Spis listingów.....	30