

## Dokumentacja projektu wykonywanego w ramach zajęć BAZY DANYCH I

### Projekt koncepcji założenia

#### Zdefiniowanie tematu projektu

Celem projektu jest zrealizowanie aplikacji internetowej umożliwiającej zarządzanie kinem. Użytkownik ma mieć możliwość przeglądania kadry pracowniczej, filmów, seansów, sprzedanych biletów, sal i reklam puszcanych w kine. Możliwe jest również dodawanie, edycja, usuwanie rekordów oraz ich wyszukiwanie i sortowanie na podstawie wybranych parametrów. System oferuje też możliwość tworzenia raportów przychodów i sprzedaży biletów dotyczących każdego filmu z kolei w określonym przez użytkownika przedziale czasu.

Wszyscy zalogowani użytkownicy będą mieli możliwość dodawania nowych rekordów oraz ich przeglądania.

#### Analiza wymagań użytkownika

Istnieją dwa poziomy dostępu do serwisu: dostęp jako administrator oraz jako zwykły pracownik. Pracownik ma ograniczone prawa jeśli chodzi o wykonywanie akcji w serwisie.

#### Administrator:

- login: *admin* hasło: *admin*
- przeglądanie informacji o pracownikach, filmach, seansach, sprzedanych biletach, salach i reklamach
- edycja zawartości każdego z rekordów
- usuwanie rekordów
- dodawanie nowych rekordów
- widok ogólnego raportu wszystkich przychodów z podziałem na filmy
- generowanie raporów i sprzedaży biletów z podziałem na filmy w określonym przez użytkownika przedziale czasu

#### Pracownik:

- login: *demo* hasło: *demo*
- przeglądanie informacji o pracownikach, filmach, seansach, sprzedanych biletach, salach i reklamach
- edycja zawartości każdego z rekordów
- dodawanie nowych rekordów
- widok ogólnego raportu wszystkich przychodów z podziałem na filmy

#### Zaprojektowanie funkcji

Poniżej określono podstawowe funkcje realizowane w bazie danych. Celem ich wszystkich jest walidacja wprowadzanych do tabel danych. Każda z przedstawionych poniżej funkcji zwraca *boolean*. Zaimplementowane funkcje:

- *sprawdz\_miejsca()* - funkcja służy sprawdza czy nie zakupiono wcześniej biletu na to samo miejsce i ten sam seans
- *sprawdz\_zajecia()* - funkcja służy sprawdzeniu dyspozycyjności pracownika danego dnia i o danej godzinie (czy nie zajmuje się on przypadkiem w tym czasie obsługą

innego seansu). Skuteczna przy założeniu, że seanse w kinie odbywają się w trzech różnych godzinach: 13:30, 16:15 i 19:10. Ponadto przy okazji tego punktu warto wspomnieć, że funkcje pełnione przez każdego z pracowników mogą się zmieniać. Tzn. jedna osoba może być raz kontrolerem biletu, a za chwilę operatorem sprzętu

- `sprawdz_sale()` - sprawdza przy tworzeniu nowego seansu czy sala, w której chcemy wyświetlić film jest wtedy wolna. Założenia takie same jak z podpunktu wyżej
- `sprawdz_film()` - sprawdza przy dodaniu do bazy nowego filmu czy nie istnieje już w bazie taki film (unikalność jest sprawdzana na podstawie tytułu i reżysera)
- `sprawdz_reklame()` - przy dodawaniu reklamy sprawdza czy nie istnieje już w bazie taka sama reklama puszczana przed takim samym filmem
- `sprawdz_stanowisko()` - funkcja zapewnia unikalność nowych stanowisk w kinie. Tworzenie nowych stanowisk z poziomu aplikacji nie jest możliwe

## Projekt diagramów (konceptualny)

### Zdefiniowanie encji i słowników

W ramach projektu zdefiniowano następujące encje:

- Film – zawiera numer identyfikujący, pole tytuł, premiera, gatunek, reżyser oraz informację o czasie trwania
- Reklama – posiada numer id, zawiera informacje o reklamowanym produkcie, wydawcy reklamy oraz numer identyfikacyjny filmu, przed którym taka reklama jest puszczana
- Sala – posiada numer id – będący jednocześnie numerem sali, informacje o numerze piętra, na którym się znajduje, całkowitej liczbie miejsc oraz liczbie rzędów i kolumn siedzeń
- Seans – zawiera wszystkie informacje dotyczące poszczególnych seansów: numer identyfikacyjny, numer sali, grany film, datę, godzinę oraz informację czy film jest grany w 3D
- Pracownik – posiada numer identyfikujący oraz pola zawierające imię i nazwisko
- Stanowiska – posiada pole zawierające nazwę stanowiska, gromadzi listę wszystkich możliwych stanowisk
- Kadra – encja pełni funkcję przyporządkowania pracownika i pełnionej przez niego funkcji do seansu. Zawiera pole id, numer identyfikacyjny seansu, id pracownika oraz nazwę przyporządkowanego mu stanowiska
- Transakcja – przechowuje informacje o numerze transakcji i numerze pracownika przeprowadzającego daną transakcję. W ramach jednej transakcji teoretycznie może zostać sprzedane wiele biletów
- Bilet – zawiera informacje o sprzedanych biletach: id biletu, jego cenę, informację czy był kupiony ze zniżką, id seansu, informacje o numerze siedzenia czyli rząd i miejsce oraz numer transakcji

### Zaprojektowanie relacji pomiędzy encjami i słownikami

Poniżej przedstawiono dokładny schemat wszystkich encji wraz z atrybutami oraz kluczami.

Film
film_id: INTEGER NOT NULL [PK]
tytuł: VARCHAR(40) NOT NULL
premiera: DATE NOT NULL
gatunek: VARCHAR(40) NOT NULL
reżyser: VARCHAR(40) NOT NULL

czas_trwania: INTEGER NOT NULL
--------------------------------

Aby uniknąć ręcznego wprowadzania id do tej i wszystkich kolejnych tabel wprowadzono sekwencje, zwiększające się o wartość 1 z dodaniem każdego kolejnego rekordu. Sekwencja dla tej tabeli nazywa się *bilet\_id\_seq*. Tak jak już wcześniej zostało wspomniane, że podczas dodawania rekordu do tabeli uruchamia się trigger, który pilnuje czy nie ma już w bazie filmu o takim samym tytule i tego samego reżysera. Pole *film\_id* stanowi klucz główny.

Reklama
reklama_id: INTEGER NOT NULL [PK]
reklamowany_produkt: VARCHAR(40) NOT NULL
wydawca: VARCHAR(40) NOT NULL
film_id: INTEGER NOT NULL [FK]

Tabela korzysta z sekwencji *reklama\_id\_seq*. Klucz główny znowu stanowi numer identyfikacyjny. Tabela posiada również klucz obcy, który pozwala przyporządkować daną reklamę do konkretnego filmu. Reklama jest zatem powiązana z filmem relacją 1:n. Każda reklama jest przyporządkowana do jednego filmu, natomiast przed każdym z filmów może być wyświetlonych wiele reklam. Podczas dodawania reklamy do tabeli sprawdzane jest czy nie ma już takiej reklamy przyporządkowanej do takiego samego filmu.

Sala
sala_id: INTEGER NOT NULL [PK]
pietro: INTEGER NOT NULL
liczba_siedzen: INTEGER NOT NULL
rzedy: INTEGER NOT NULL
kolumny: INTEGER NOT NULL

Kluczem głównym w tej tabeli jest id sali, będące jednocześnie jej numerem. Przy dodawaniu elementów do tabeli nie jest wykorzystywana tym razem sekwencja. Zgodnie z koncepcją pierwsza cyfra numeru sali symbolizuje piętro, na jakim ta sala się znajduje. Dlatego przy dodawaniu sali należy ręcznie wprowadzić jej nowy numer.

Seans
seans_id: INTEGER NOT NULL [PK]
sala_id: INTEGER NOT NULL [FK]
film_id: INTEGER NOT NULL [FK]
godzina: TIME NOT NULL
data: DATE NOT NULL
3D: BOOLEAN NOT NULL

Ponownie, przy dodawaniu rekordów do tabeli wykorzystuje się sekwencję *seans\_id\_seq*. Tabela posiada dwa klucze obce. Pole *sala\_id* wskazuje na to, w której sali ma się odbywać seans. Seans i sala również są powiązane relacją 1:n – każdy seans może odbywać się tylko w jednej sali, natomiast w każdej sali może być wiele seansów. Analogicznie do tego przypadku film i seans są powiązane

taką samą relcją. Podczas dodawania nowego seansu sprawdzane jest czy wybrana na seans sala jest wolna w terminie tego seansu.

Pracownik
pracownik_id: INTEGER NOT NULL [PK]
imie: VARCHAR(40) NOT NULL
nazwisko: VARCHAR(40) NOT NULL

Podczas dowawania nowych pracowników do tabeli wykorzystywana jest sekwencja *pracownik\_id\_seq*. Kluczem głównym w tej tabeli jest id pracownika.

Stanowiska
nazwa_stanowiska: VARCHAR(40) NOT NULL [PK]

Jest to encja słownikowa zawierająca nazwy wszystkich potencjalnych funkcji jakie można pełnić pracując w kinie. Podczas dodawania rekordu do tabeli sprawdzana jest jego unikalność.

Kadra
kadra_id: INTEGER NOT NULL [PK]
seans_id: INTEGER NOT NULL [FK]
nazwa_stanowiska: VARCHAR(40) NOT NULL [FK]
pracownik_id: INTEGER NOT NULL [FK]

Kadra to encja asocjacyjna realizująca powiązanie m:n pomiędzy tabelami pracownik i seans. Kluczem głównym jest tu numer id, wszystkie pozostałe parametry to klucze obce. Funkcje pełnione przez pracownika mogą się zmieniać. Ta sama osoba raz może być kontrolerem, a w razie potrzeby sprzątaczem. Do każdego z seansów powinny być przyporządkowane przynajmniej trzy osoby odpowiedzialne za kontrolowanie, obsługę sprzętu i sprzątanie. Ponad to oczywiste jest to, że ten sam pracownik może być kadra na różnych seansach. Przed dodaniem nowego rekordu sprawdzana jest dyspozycyjność pracownika, a przy dodawaniu wykorzystywana jest sekwencja *kadra\_id\_seq*.

Transakcja
transakcja_id: INTEGER NOT NULL [PK]
pracownik_id: INTEGER NOT NULL [FK]

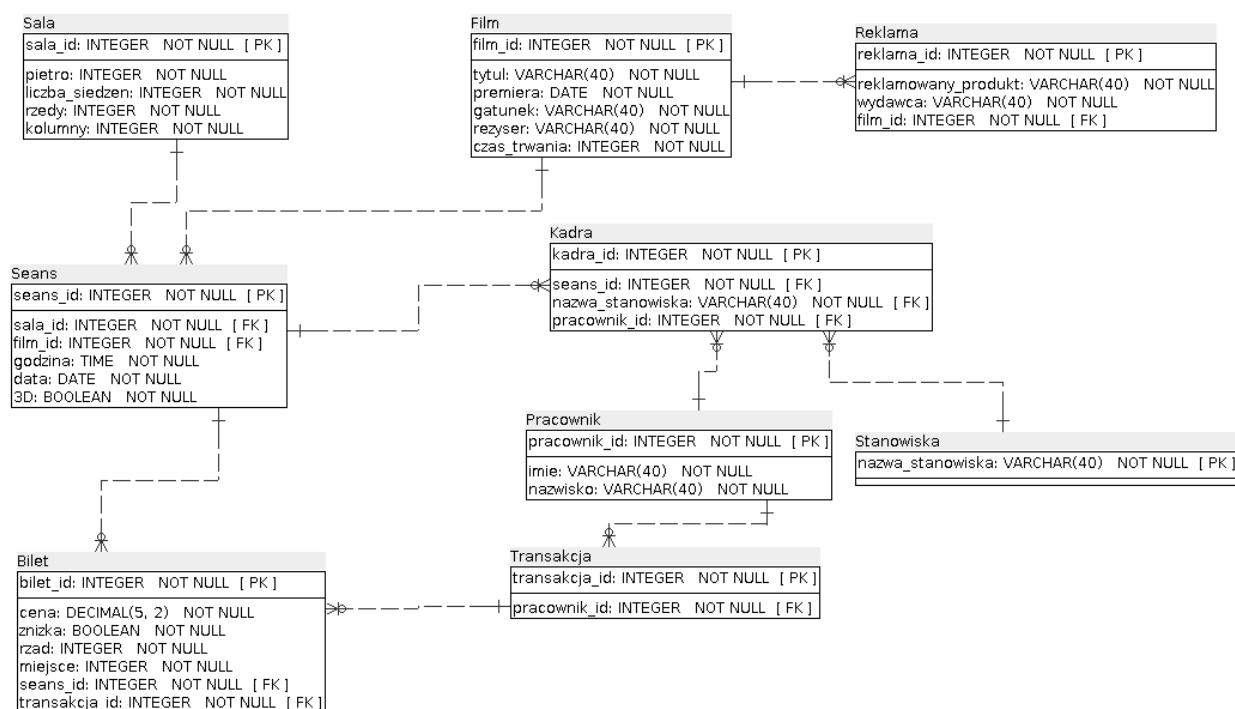
Encja ta łączy tabele bilet i pracownik. Kluczem głównym jest numer id transakcji, inkrementowany przy wykorzystaniu sekwencji *transakcja\_id\_seq*. Kluczem obcym jest natomiast numer id pracownika.

Bilet
bilet_id: INTEGER NOT NULL [PK]
cena: DECIMAL(5, 2) NOT NULL
znizka: BOOLEAN NOT NULL
rzad: INTEGER NOT NULL

miejsce: INTEGER NOT NULL
seans_id: INTEGER NOT NULL [FK]
transakcja_id: INTEGER NOT NULL [FK]

Tabela reprezentuje sprzedane już bilety. Kluczem głównym w tej tabeli jest numer id biletu. Przy dodawaniu biletu wykorzystywana jest sekwencja *bilet\_id\_seq*. Dla uproszczenia przyjęto, że cena biletu bez zniżki wynosi 20,00 zł, zaś ze zniżką 13,00 zł. Klucze obce w tej tabeli to numery id seansu i transakcji. Bilet jest połączony z transakcją relacją 1:n, w ramach jednej transakcji możliwy jest zakup wielu biletów, ale dany bilet jest przypisany tylko do jednej transakcji. Seans jest połączony z biletem również relacją 1:n, bilet upoważnia nas do wstępu na jeden seans, zaś biletów na jeden seans jest wiele. Przed wstawieniem nowego rekordu do tabeli sprawdzane jest czy na wybranym seansie nie jest już zajęte dane miejsce.

Do stowrzenia diagramu ERD wykorzystano program SQL Power Architect, który automatyzuje proces tworzenia skryptu SQL tworzącego tabele i zdefiniowane klucze. Poniżej przedstawiono wygenerowany diagram:



## Projekt logiczny

### Projektowanie tabel, kluczy, indeksów

Struktura bazy danych wraz ze szczegółami technicznymi została zdefiniowana w poprzednim punkcie. To samo tyczy się słowników danych.

### Analiza zależności funkcyjnych i normalizacja tabel

#### Pierwsza postać normalna

Wszystkie tabele składają się z liczb, ciągów znaków, dat i wartości logicznych, nie mają złożonej wewnętrznej struktury, zatem są to kolumny o wartościach atomowych. Nie występują tu kolekcje. Potencjalnie powtarzające się atrybut przechowujący nazwę pełnionej funkcji, wyodrębniono do

osobnej tabeli. Każda z tabel ma swój klucz główny.

### Druga postać normalna

Aby relacja spełniała drugą postać normalną oprócz spełnienia kryteriów na pierwszą postać normalną, żadna jej kolumna nie będąca kluczem nie może być częściowo funkcyjnie zależna od jakiegokolwiek klucza potencjalnego. Analiza atrybutów tabel pod kątem spełnienia tych wymagań:

- Film – żaden z parametrów niekluczowych nie jest potencjalnie zależny od jakiegokolwiek wymyślnego klucza potencjalnego
- Reklama – parametry niekluczowe są niezależne od reszty
- Sala – parametry nie będące kluczami to piętro, liczba siedzeń, rzędy i kolumny siedzeń. Niestety tabela ta nie spełnia kryteriów 2NF, gdyż 3 ostatnie wartości są zależne od siebie. Potencjalnym rozwiązaniem tego problemu byłoby usunięcie kolumny „liczba siedzeń”
- Seans – parametry niekluczowe, czyli godzina, data i 3D są całkowicie niezależne od siebie jak i od reszty kluczowych parametrów
- Pracownik – atrybuty znajdujące się poza kluczem to imię i nazwisko, są one niezależne od klucza potencjalnego, gdyż same mogą go tworzyć
- Stanowiska – składa się tylko z klucza głównego, dlatego spełnia 2NF
- Kadra – jest encją asocjacyjną, w której nie występują elementy niebędące kluczami
- Transakcja – wszystkie jej elementy są kluczami
- Bilet – parametry nie będące kluczami to: cena, zniżka, rząd i miejsce. Przy założeniu, że ceny biletów zarówno ulgowych jak i normalnych mogą się zmieniać w zależności od seansu tabela spełnia 2NF. W tym projekcie przyjęto jednak dla uproszczenia, że biletom ulgowym i normalnym odpowiada stała, niezmienna cena. Sprawia to, że tabela nie spełnia tego kryterium. Rozwiązanie takie daje nam jednak większe możliwości jeśli chodzi o potencjalny rozwój systemu (wprowadzenie w przyszłości bardziej zróżnicowanego cennika)

### Trzecia postać normalna

Mamy z nią do czynienia wtedy, gdy tabela jest 2NF oraz gdy wszystkie pola niebędące polami klucza potencjalnego są od niego zależne bezpośrednio.

Większość z tabel teoretycznie spełnia ten warunek, niemniej jednak kilka z nich zostało już odrzuconych na etapie sprawdzenia wymagań dla 2NF. Dlatego nie można powiedzieć o bazie danych, że została wykonana zgodnie z trzecią postacią normalną. Jestem jednak pewna, że przy wydedukowaniu odpowiednio sprytnych i naciąganych założeń dałoby się nazwać tę bazę zgodną z trzecią postacią (można by na przykład powiedzieć, że liczba miejsc na sali będzie zawsze o kilka większa niż iloczyn ilości rzędów i kolumn, gdyż potrzebne jest dodatkowe miejsce dla pracownika pilnującego porządku na sali).

### Zaprojektowanie operacji na danych

Wszystkie wyspecyfikowane wcześniej funkcje w bazie danych:

- sprawdz\_miejsca()
- sprawdz\_zajecia()
- sprawdz\_sale()
- sprawdz\_film()
- sprawdz\_reklame()
- sprawdz\_stanowisko()

Nie przyjmują żadnych argumentów. Każda z nich na podstawie parametrów aktualnie wstawianych wartości szuka w tabeli rekordu, który by wykluczał ten nowy rekord. Jeśli go znajdzie to każda z tych funkcji zwraca wartość pustą *null* uniemożliwiając wstawienie do tabeli nowej wartości. Jeśli sprzeczność nie zostanie znaleziona to zwracany jest rekord *new*.

W ramach projektu stworzono też widoki:

- **bilet\_sprzedawca** – przedstawia wszystkie informacje na temat biletu w przystępny sposób. Wyświetla informacje o:
  - id biletu
  - czy bilet jest ulgowy
  - tytule filmu
  - dacie i godzinie seansu
  - numerze sali
  - rzędzie i miejscu siedzenia
  - nazwisko sprzedawcy biletu
 Informacje te zostały wyciągnięte z tabel film, seans, pracownik i transakcja.
- **kadra\_seans** – zawiera uporządkowane w czytelny sposób wszystkie informacje o seansie oraz o obsłudze na tym seansie:
  - data i godzina
  - numer sali
  - tytuł filmu
  - czy film jest 3D
  - nazwisko osoby sprzątajacej po tym seansie
  - nazwisko kontrolera biletów
  - nazwisko operatora sprzętu
 Informacje te pochodzą z tabel seans, film, kadra i pracownik.
- **reklama\_film** – również prezentuje ważne informacje w czytelniejszy sposób, niż tabela reklama. Widoczne kolumny to:
  - tytuł filmu, na którym reklama jest puszczana
  - nazwa reklamowanego produktu
  - wydawca reklamy
- **przychody\_film** – zawierająca podstawowy raport przedstawiający przychód wygenerowany przez każdy z filmów. Do stworzenia tego widoku wykorzystano funkcję agregującą.

## Projekt funkcjonalny

### Interfejsy do prezentacji, edycji i obsługi danych

Po wejściu na stronę najpierw ukazuje się formularz logowania.

## Formularz logowania

Pola z \* są wymagane.

Login \*

Hasło \*

☐ Zapamiętaj mnie następnym razem

Istnieją dwa poziomy autoryzacji i dla każdego z nich ustalono „na sztywno” pewien login i hasło:

- Administrator – login: *admin*, hasło: *admin*
- Pracownik – login: *demo*, hasło: *demo*

Poniżej przedstawiono struktury poszczególnych formularzy do wprowadzania danych.  
Formularz umożliwiający wprowadzenie nowego pracownika:

## Dodaj pracownika

*Pola z \* są wymagane.*

Imie \*

Nazwisko \*

Create

Formularz umożliwiający dodanie filmu:

## Dodaj film

*Pola z \* są wymagane.*

Tytuł \*

Premiera \*

Gatunek \*

Reżyser \*

Czas trwania \*

Create



Formularz umożliwiający dodanie nowego seansu:

## Dodaj seans

*Pola z \* są wymagane.*

Data \*

Godzina \*

Sala \*

Tytuł \*

3d \*

Sprzątacznik \*

Kontroler \*

Operator \*

Create

Znajomość struktury bazy danych może sprawiać wrażenie, że dodajemy element do widoku. W rzeczywistości wstawiamy jednak jeden rekord do tabeli seans i trzy rekordy do tabeli kadra.

Formularz dodawania sprzedanego biletu:

## Dodaj bilet

*Pola z \* są wymagane.*

Zniżka \*

Seans \*

Rząd \*

Miejsce \*

Sprzedawca \*

Create

Podobnie jak w powyższym przypadku. Tym razem wstawiamy po rekordzie do tabeli transakcja i bilet. Formularz korzysta z założenia, że ceny każdego biletu ze zniżką i bez niej są takie same.

Formularz dodawania reklamy:

## Dodaj reklamę

*Pola z \* są wymagane.*

Tytuł \*

(Wybierz film) ▼

Reklamowany produkt \*

Wydawca \*

Create

Formularz dodania sali:

## Dodaj salę

*Pola z \* są wymagane.*

Sala \*

Pietro \*

Liczba Siedzen \*

Rzedy \*

Kolumny \*

Create

Formularz generowania raportu:

## Generuj raport

Od

Do

Generuj

Użycie aplikacji w przeglądarce Google Chrome ułatwi użytkownikowi wypełnianie raportów – w polach związanych z datą uruchomi się okno dialogowe umożliwiające interaktywny wybór daty.

## Wizualizacja danych

Poniżej przedstawiono wygląd typowego raportu aplikacji dla widoku kadra\_seans:

### Seanse

Wyświetlanie 1 - 10 z 184

**Data:** 2017-01-10  
**Godzina:** 13:30:00  
**Sala:** 11  
**Tytuł:** Aplik@cja  
**3d:** Nie  
**Sprzątacznik:** Nowak  
**Kontroler:** Zawadzka  
**Operator:** Poczta

**Data:** 2017-01-10  
**Godzina:** 13:30:00  
**Sala:** 12  
**Tytuł:** Assassins Creed  
**3d:** Tak  
**Sprzątacznik:** Polit  
**Kontroler:** Szmidt  
**Operator:** Bartłomiejczyk

Schemat ten powtarza się w obrębie pozostałych tabel i widoków.

Po wejściu na stronę ogólnego zarządzania danymi na stronie pojawia się tabela:

## Zarządzaj seansami

Możliwe jest opcjonalne wprowadzenie operatora porównania (<, <=, >, >=, <> lub =) na początku każdej z wprowadzonych wartości wyszukiwania w celu określenia jak porównanie powinno zostać wykonane.

[Zaawansowane wyszukiwanie](#)

Wyświetlanie 1 - 10 z 184

Data	Godzina	Sala	Tytuł	3d	Sprzątacznik	Kontroler	Operator	
2017-01-10	13:30:00	11	Aplik@cja	Nie	Nowak	Zawadzka	Poczta	  
2017-01-10	13:30:00	12	Assassins Creed	Tak	Polit	Szmidt	Bartłomiejczyk	  
2017-01-10	13:30:00	21	Dusigrosz	Nie	Sadowska	Włodarczyk	Grun	  
2017-01-10	16:15:00	11	Fantastyczne zwierzęta i jak je znaleźć	Nie	Chmielewski	Sikorski	Nowak	  
2017-01-10	16:15:00	12	Firmowa gwiazdka	Nie	Zawadzka	Poczta	Polit	  
2017-01-10	16:15:00	21	Jak zostać kotem	Nie	Szmidt	Bartłomiejczyk	Sadowska	  
2017-01-10	19:10:00	11	Królowa śniegu 3: Ogień i lód	Tak	Włodarczyk	Grun	Chmielewski	  
2017-01-10	19:10:00	12	Kubo i dwie struny	Nie	Sikorski	Nowak	Zawadzka	  
2017-01-10	19:10:00	21	Pasażerowie	Nie	Poczta	Polit	Szmidt	  
2017-01-11	13:30:00	11	Paterson	Nie	Bartłomiejczyk	Sadowska	Włodarczyk	  

Idź do strony: [< Poprzednia](#) [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [Następna >](#)

Umożliwia ona przeglądanie seansów, ich sortowanie, edycję, usuwanie oraz wyszukiwanie. Po kliknięciu opcji zaawansowanego wyszukiwania pojawia się formularz:

[Zaawansowane wyszukiwanie](#)

Data	<input type="text"/>
Godzina	<input type="text"/>
Sala	<input type="text"/>
Tytuł	<input type="text"/>
3d	<input type="text"/>
Sprzątac	<input type="text"/>
Kontroler	<input type="text"/>
Operator	<input type="text"/>
<input type="button" value="Szukaj"/>	

Wypełnienie wszystkich pól nie jest konieczne. Należy zwracać uwagę na format wpisywanych parametrów. Wartość boolean w rubryce 3d oznaczamy jako 1 lub 0.

## Przychody

Wyświetlanie 1 - 10 z 13

Tytuł: Aplik@cja  
Przychód: 2281.00

Tytuł: Firmowa gwiazdka  
Przychód: 2281.00

Poniżej przedstawiono podstawowy widok zakładki raporty:

Po wybraniu zakładki generuj raporty i zaznaczeniu interesującej nas daty otrzymujemy bardziej szczegółowy raport:

## Generuj raport

Od	<input type="text" value="2017-01-10"/>
Do	<input type="text" value="2017-01-11"/>
<input type="button" value="Generuj"/>	

Wyświetlanie 1 - 10 z 13

Tytuł ▲	Sprzedane bilety	Przychód
Aplik@cja	80	1523.00
Assassins Creed	80	1523.00

Poniżej przedstawiono też przykładowe okno aktualizacji rekordu:

## Aktualizuj reklamę

Pola z \* są wymagane.

Tytuł \*  
Aplikacja

Reklamowany produkt \*  
Majonez Kielecki

Wydawca \*  
Społem

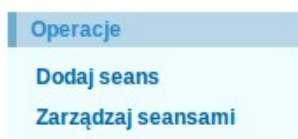
Save

## Zdefiniowanie panelu sterowania aplikacji

Po zalogowaniu się do aplikacji pojawia się na ekranie pasek menu aplikacji.



Umożliwia on szybkie nawigowanie po stronie i przenoszenie do aktualnie interesujących nas zakładki. Po wejściu do każdej z zakładek pojawia się z prawej strony pasek z opcjami operacji jakie możemy wykonać na danych:



Dostępne operacje aktualizują się wraz z eksploracją zakładki.

## Makropolecenia

?? TODO ?!?!?

## Dokumentacja

### Wprowadzanie danych

Jak już wcześniej to przedstawiono, dane wprowadzane są ręcznie poprzez formularze.

### Dokumentacja użytkownika

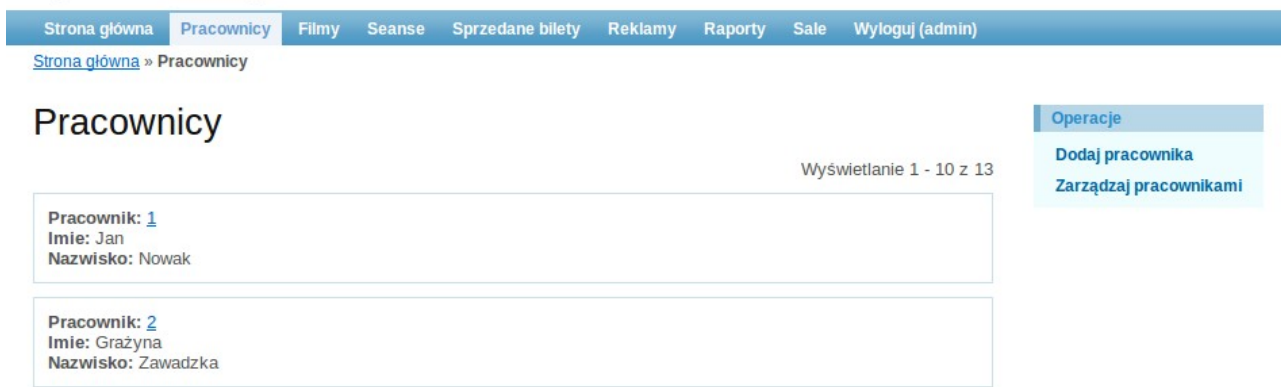
1. Aplikacja jest dostępna na serwerze pascal pod adresem:  
<http://pascal.fis.agh.edu.pl/~4weglinska/BDI/kino/index.php>  
Sugeruję otwierać ją w przeglądarce Google Chrome, zapewni to łatwiejsze uzupełnianie formularzy.
2. Należy się zalogować poprzez wypełnienie formularza logowania. Dostępne są dwa poziomy uwierzytelnienia:
  - Administrator - login: *admin*, hasło: *admin*
  - Pracownik – login: *demo*, hasło: *demo*

3. Z głównego paska menu należy wybrać interesujące nas dane



4. Po wejściu do przykładowej zakładki „Pracownicy” ujrzymy następujący widok

## System zarządzania kinem



5. Na dole strony znajdują się strzałki do przeglądania kolejnych rekordów



6. W rubryce operacje znajdują się dostępne opcje manipulacji danymi



7. Wybranie opcji „Dodaj pracownika” będzie skutkowało pojawieniem się odpowiedniego formularza

## Dodaj pracownika

Pola z \* są wymagane.

Imie \*

Nazwisko \*

Create

Po wpisaniu wszystkich danych i po przejściu przez nie walidacji rekord zostanie dodany do bazy. Pola związane z datą w przeglądarkach innych niż chrome należy wpisywać w formacie: RRRR-MM:DD, godzinę: GG:MM:SS, wartości typu Tak – 1, Nie – 0 (przy

zaawansowanym wyszukiwaniu). Po dodaniu rekordu należy sprawdzić czy pojawił się on na liście. Jeśli tak się nie stało, oznacza to że dana, którą próbowano dodać była niezgodna logicznie z dotychczasową zawartością bazy.

- Po wejściu do zakładki „Zarządzami pracownikami” otrzymamy aktualną listę pracowników w formie tabeli. Ikony z prawej strony każdego rekordu umożliwiają jego podgląd, edycje lub usunięcie. Dane w tabeli można też filtrować – poprzez użycie „Zaawansowanego wyszukiwania” lub przy użyciu pól znajdujących się bezpośrednio nad rekordami.

## Zarządzaj pracownikami










Możliwe jest opcjonalne wprowadzenie operatora porównania (<, <=, >, >=, <> lub =) na początku każdej z wprowadzonych wartości wyszukiwania w celu określenia jak porównanie powinno zostać wykonane.

### Zaawansowane wyszukiwanie

Wyświetlanie 1 - 8 z 8

Pracownik	Imie	Nazwisko	
>5			
6	Janusz	Bartłomiejczyk	  
7	Małgorzata	Sadowska	  
8	Włodzimierz	Włodarczyk	  

Z tego poziomu możliwe jest też sortowanie rekordów. Należy kliknąć interesujący nas atrybut według, którego chcemy sortować. Np. napis „Nazwisko” na niebieskim tle:

Pracownik	Imie	Nazwisko ▲	
>5			
14	Ala	Abacka	  
6	Janusz	Bartłomiejczyk	  
10	Andrzej	Chmielewski	  

- Wejście w pogląd rekordu:

### Widok pracownika #10

Pracownik	10
Imie	Andrzej
Nazwisko	Chmielewski

#### Operacje

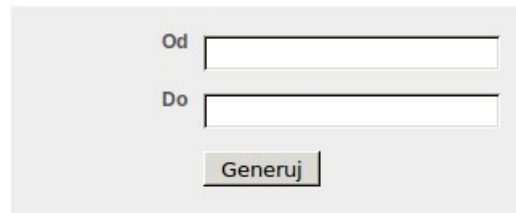
[Lista pracowników](#)  
[Dodaj pracownika](#)  
[Edytuj pracownika](#)  
[Usuń pracownika](#)  
[Zarządzaj pracownikami](#)

Z tego poziomu również jest możliwe przejście do wykonywania różnych operacji na rekordzie oraz na samej bazie.

- Zakładką najbardziej różniącą się od pozostałych są „Raporty”. W zakładce tej niemożliwe jest podejrzenie jednego z rekordów. Po wciśnięciu z menu bocznego napisu „Generuj raport” ukaże się formularz, proszący o wpisanie przedziału czasu, dla którego interesuje nas raport.



## Generuj raport



Form for generating a report. It contains two input fields: 'Od' (From) and 'Do' (To), and a button labeled 'Generuj' (Generate).

Po jego wypełnieniu ukaze nam się lista sprzedanych biletów i uzyskanych z nich przychodów z podziałem na każdy film.

11. Wciśnięcie przycisku „Wyloguj” - wylogowuje użytkownika z systemu i uniemożliwia mu wykonywanie jakichkolwiek akcji.

### UWAGI:

- polecaną przeglądarką do uruchamiania aplikacji jest Google Chrome
- przykładowe dane dotyczące sprzedaży biletów zostały wygenerowane tylko dla dni od 2017-01-10 do 2017-01-13 włącznie
- po każdym dodaniu rekordu do bazy należy się upewnić czy znalazł się on na liście, jeśli tak się nie stało oznacza to, że wprowadzone dane były niezgodne z rzeczywistym stanem bazy, np. to miejsce na danym seansie jest już zajęte lub pracownik o tej porze pracuje na innym seansie
- nie wszystkie przedstawione powyżej operacje są dozwolone na koncie pracownika
- należy zwracać uwagę na format wprowadzanych danych
- przyjęto założenie, że w kinie seanse są grane w godzinach: 13:30, 16:15 i 19:10.

### Opracowanie dokumentacji technicznej

Do wykonania projektu wykorzystano framework PHP Yii, wykorzystujący wzorzec MVC. Z tego powodu aplikacja jest podzielona na trzy główne części:

- Model – reprezentuje logikę aplikacji. Kluczowe klasy w aplikacji reprezentujące model:
  - BiletSprzedawca
  - Film
  - KadraSeans
  - LoginForm
  - Pracownik
  - PrzychodyFilm
  - ReklamaFilm
  - Sala
- Widok – opisuje jak wyświetlić część modelu w ramach interfejsu użytkownika. W ramach każdej z zakładek stworzono po kilka plików odpowiedzialnych za GUI.
- Kontroler – odpowiedzialny jest za otrzymywanie danych od użytkownika i reakcje na nie.

W tym projekcie odpowiadają za to klasy:

- BiletSprzedawcaController
- FilmController
- KadraSeansController
- SiteController
- PracownikController
- PrzychodyFilmController
- ReklamaFilmController



- SalaController

Bardziej szczegółowa dokumentacja znajduje się pod adresem:  
<http://pascal.fis.agh.edu.pl/~4weglinskaBDI/do/kumentacja/api/>

Wykaz źródeł

<http://www.yiiframework.com/doc-2.0/guide-index.html>

<https://pl.wikipedia.org>