Baza kina

Magdalena Potok

Dawid Kawalec

Spis treści

1 Schemat bazy

2 Opis bazy

Tabela "opis_sali" zawiera informacje o opisie każdej z sal. Każdy opis jest identyfikowany przez unikalne id_opisu, które jest generowane automatycznie. Tabela zawiera kolumny "id_opisu", "opis".

Tabela "klient" zawiera informacje o klientach, w tym ich imieniu, nazwisku, adresie e-mail, numerze telefonu i haśle. Każdy klient jest identyfikowany przez unikalne id_klienta, które jest generowane automatycznie. Adres e-mail jest wymagany i musi być unikalny. Tabela zawiera kolumny "id_klienta", "imie", "nazwisko", "e_mail", "nr_telefonu", "haslo".

Tabela "seanse" zawiera informacje o seansach filmowych, w tym tytułu filmu, formacie filmu (2D, 3D, 2D VIP, 3D VIP), godzinie i dacie seansu oraz informacji o sali. Każdy seans jest identyfikowany przez unikalne id_seansu, które jest generowane automatycznie. Sala jest powiązana z tabelą "sala" przez klucz obcy "sala", który określa, w której sali odbędzie się seans. Tabela zawiera kolumny "id_seansu", "tytul", "format_filmu", "godzina", "data_seans", "sala".

Tabela "statusy" zawiera informacje o statusach rezerwacji, w tym nazwę statusu. Każdy status jest identyfikowany przez unikalne id_status, które jest generowane automatycznie. Tabela zawiera kolumny "id_status", "nazwa_s".

Tabela "rodzaje_biletow" zawiera informacje o rodzajach biletów, w tym nazwę biletu. Każdy rodzaj biletu jest identyfikowany przez unikalne id_rb, które jest generowane automatycznie. Nazwa biletu musi być unikalna. Tabela zawiera kolumny "id_rb", "nazwa_b".

Tabela "rezerwacje" zawiera informacje o rezerwacjach biletów na seans filmowy. Kolumny "klient" i "seanse" są połączeniami z innymi tabelami przez klucze obce i tworzą relację wiele do wielu między tabelami "rezerwacje" i "seanse". Kolumna "status" jest połączeniem z tabelą "statusy" i określa status rezerwacji. W przypadku usunięcia krotki z tabeli "klient" lub "statusy" powiązane krotki w tabeli "rezerwacje" zostaną usunięte (ON DELETE CASCADE). W przypadku aktualizacji krotki w tabeli "statusy" powiązane krotki w tabeli "rezerwacje" również zostaną zaktualizowane (ON UPDATE CASCADE).

Tabela "sala" zawiera informacje o salach w kinie. Kolumna "opis" jest połączeniem z tabelą "opis_sali" i określa opis każdej z sal. W przypadku usunięcia krotki z tabeli "opis_sali" powiązane krotki w tabeli "sala" zostaną usunięte (ON DELETE CASCADE).

Tabela "cennik" zawiera informacje o cenach biletów w każdej z sal. Kolumna "nazwa_biletu" jest połączeniem z tabelą "rodzaje_biletów" i określa rodzaj biletu. Kolumna "sala" jest połączeniem z tabelą "sala" i określa salę, dla której dotyczy cena. W przypadku usunięcia krotki z tabel "rodzaje_biletów" lub "sala" powiązane krotki w tabeli "cennik" zostaną usunięte (ON DELETE CASCADE). W przypadku aktualizacji krotki w tabeli "rodzaje_biletów" powiązane krotki w tabeli "cennik" również zostaną zaktualizowane (ON UPDATE CASCADE).

3 Zapytania SQL

3.1 Zakładanie tabel

```
DROP TABLE IF EXISTS opis_sali CASCADE;
CREATE TABLE opis_sali (
    id_opisu SERIAL PRIMARY KEY,
    opis TEXT NOT NULL);

DROP TABLE IF EXISTS klient CASCADE;
CREATE TABLE klient (
    id_klienta SERIAL PRIMARY KEY,
    imie TEXT NOT NULL,
    nazwisko TEXT ,
    e_mail TEXT UNIQUE NOT NULL,
    nr_telefonu VARCHAR(10),
    haslo TEXT NOT NULL);

DROP TABLE IF EXISTS sala CASCADE;
CREATE TABLE sala (
    id_sali SERIAL PRIMARY KEY,
    ilosc_miejsc INT NOT NULL,
    opis INT REFERENCES opis_sali(id_opisu) ON DELETE CASCADE);

DROP TABLE IF EXISTS seanse CASCADE;
CREATE TABLE seanse (
    id_seansu SERIAL PRIMARY KEY,
    tytul TEXT NOT NULL,
    format_filmu TEXT CHECK (format_filmu in ('2D', '3D', '2D VIP', '3D VIP')),
    godzina TIME NOT NULL,
    data_seans DATE NOT NULL,
    sala INT REFERENCES sala(id_sali) ON DELETE CASCADE);

DROP TABLE IF EXISTS statusy CASCADE;
CREATE TABLE statusy (
    id_status SERIAL PRIMARY KEY,
    nazwa_s TEXT NOT NULL);

DROP TABLE IF EXISTS rodzaje_biletow CASCADE;
CREATE TABLE rodzaje_biletow (
    id_rb SERIAL PRIMARY KEY,
    nazwa_b TEXT UNIQUE NOT NULL);
```

```
DROP TABLE IF EXISTS rezerwacje CASCADE;
CREATE TABLE rezerwacje (
    id_rezerw SERIAL PRIMARY KEY,
    klient INT REFERENCES klient(id_klienta) ON DELETE CASCADE,
    ilosc_biletow INT NOT NULL,
    data_rezerw DATE NOT NULL,
    status INT REFERENCES statusy(id_status) ON UPDATE CASCADE ON DELETE CASCADE,
    seanse INT NOT NULL);
```

```
DROP TABLE IF EXISTS cennik CASCADE;
CREATE TABLE cennik (
    id_ceny SERIAL PRIMARY KEY,
    cena DECIMAL(7,2) NOT NULL,
    nazwa_biletu TEXT REFERENCES rodzaje_biletow(nazwa_b)
    ON UPDATE CASCADE ON DELETE CASCADE,
    sala INT REFERENCES sala(id_sali) ON DELETE CASCADE);
```

3.2 Widoki

```
DROP VIEW IF EXISTS ranking_tytulow;
CREATE VIEW ranking_tytulow AS
    SELECT tytul, count(*) as ilosc_rezerwacji, sum(ilosc_biletow) AS ilosc_biletow
    FROM rezerwacje
    JOIN seanse ON (seanse.id_seansu = rezerwacje.seanse)
    GROUP BY tytul
    ORDER BY ilosc_biletow DESC;

DROP VIEW IF EXISTS ranking_formatow;
CREATE VIEW ranking_formatow AS
    SELECT format_filmu, count(*) as ilosc_rezerwacji, sum(ilosc_biletow) AS ilosc_biletow
    FROM rezerwacje
    JOIN seanse ON (id_seansu = seanse)
    GROUP BY format_filmu
    ORDER BY ilosc_biletow DESC;

DROP VIEW IF EXISTS info_klient_sprzedaz;
CREATE VIEW info_klient_sprzedaz AS
    SELECT ilosc_rezerwacji, suma_biletow, klient AS id_klienta, e_mail
    FROM (SELECT COUNT(*) AS ilosc_rezerwacji, SUM(ilosc_biletow) AS suma_biletow, klient
    FROM rezerwacje GROUP BY klient) AS info
    JOIN klient ON (info.klient = klient.id_klienta);

DROP VIEW IF EXISTS wszystkie_rezerwacje;
CREATE VIEW wszystkie_rezerwacje AS
```

```
    SELECT id_rezerw, data_rezerw, e_mail, seanse AS id_seansu, ilosc_biletow, nazwa_s AS
    status
    FROM
    (SELECT id_rezerw, klient, ilosc_biletow, data_rezerw, seanse, nazwa_s FROM rezerwacje
    JOIN statusy ON (rezerwacje.status = id_status)) AS info
    JOIN klient ON (info.klient = klient.id_klienta);

DROP VIEW IF EXISTS widok_ceny_biletow;
CREATE VIEW widok_ceny_biletow AS
    SELECT id_ceny, cena, nazwa_biletu, ilosc_miejsc, opis_sali.opis
    FROM cennik
    JOIN rodzaje_biletow ON (nazwa_biletu = nazwa_b)
    JOIN sala ON (sala = id_sali)
    JOIN opis_sali ON (sala.opis = opis_sali.id_opisu);
```

3.3 Funkcje i wyzwalacze

```
DROP FUNCTION IF EXISTS usun_seans;
CREATE OR REPLACE FUNCTION usun_seans(id_arg INT) RETURNS VOID AS $$
    DECLARE
            krotka RECORD;
    BEGIN
            SELECT * INTO krotka FROM seanse WHERE id_seansu = id_arg;
            IF (NOT FOUND) THEN
                    RAISE EXCEPTION 'Nie ma takiego seansu';
            END IF;
            IF (FOUND) THEN
            DELETE FROM seanse WHERE id_seansu = id_arg;
            UPDATE rezerwacje
            SET status = 2
            WHERE seanse = id_arg;
            END IF;
     END;
    $$ LANGUAGE 'plpgsql';

    DROP FUNCTION IF EXISTS dodaj_seans CASCADE;
    CREATE OR REPLACE FUNCTION dodaj_seans()
    RETURNS TRIGGER AS $$
            DECLARE
                    krotka RECORD;
            BEGIN
                    SELECT * INTO krotka FROM seanse WHERE format_filmu =
                    NEW.format_filmu AND data_seans = NEW.data_seans;
                    IF (NOT FOUND) THEN
                            RETURN NEW;
                    END IF;
```

```
                IF ( (krotka.godzina > NEW.godzina + interval '2 hours') OR (NEW.godzina >
                krotka.godzina + interval '2 hours')) THEN
                RETURN NEW;
                ELSE
                        RAISE EXCEPTION 'nie da sie';
                END IF;
        END;
$$ LANGUAGE 'plpgsql';

CREATE OR REPLACE TRIGGER dodaj_s_trigger BEFORE INSERT ON seanse
        FOR EACH ROW EXECUTE PROCEDURE dodaj_seans();

DROP FUNCTION IF EXISTS zmien_cene;
CREATE OR REPLACE FUNCTION zmien_cene(procent DECIMAL(5,2)) RETURNS VOID AS $$
        BEGIN
                UPDATE cennik SET cena = cena*(1 + procent/100);
        END;
$$ LANGUAGE 'plpgsql';

DROP FUNCTION IF EXISTS capitalize_seans CASCADE;
CREATE OR REPLACE FUNCTION capitalize_seans()
RETURNS TRIGGER AS $$
        BEGIN
                        NEW.tytul := concat(upper(substring(NEW.tytul from 1 for 1)),
                        substring(NEW.tytul from 2));
                        RETURN NEW;
        END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER capitalize_seans_tr BEFORE INSERT OR UPDATE ON seanse
        FOR EACH ROW EXECUTE FUNCTION capitalize_seans();
```

3.4 Inserty

```
INSERT INTO opis_sali(opis) VALUES ('sala 2d'),('sala 3d'),('sala 2d VIP'),('sala 3d VIP');

INSERT INTO sala(ilosc_miejsc, opis) VALUES (80, 1),(90,2),(60,3),(80,4);

INSERT INTO rodzaje_biletow(nazwa_b) VALUES ('ulgowy'), ('normalny');

INSERT INTO seanse(tytul, format_filmu, godzina, data_seans, sala) VALUES
('Kot w butach', '2D', '13:00', '01-02-2023', 1), ('Kot w butach', '2D VIP', '15:00', '02-03-2023', 3),
('Kot w butach', '3D', '13:30', '01-02-2023', 2),('Kot w butach', '3D VIP', '13:00', '03-02-2023', 4),
('Kot w butach', '2D', '13:00', '05-02-2023', 1),('Kot w butach', '3D', '17:00', '05-02-2023', 2),
('Titans', '2D', '16:00', '01-02-2023', 1),('Titans', '2D VIP', '13:30', '01-02-2023', 3),
('Titans', '3D', '16:00', '01-02-2023', 2),('Titans', '3D VIP', '14:00', '01-02-2023', 4),
('Titans', '2D', '13:00', '02-02-2023', 1),('Titans', '3D', '10:00', '02-02-2023', 2),
```

('Shrek', '2D', '20:00', '01-02-2023', 1),('Shrek', '2D VIP', '18:00', '02-02-2023', 3),
('Shrek', '3D', '14:00', '02-02-2023', 2),('Shrek', '3D VIP', '17:00', '01-02-2023', 4),
('Shrek', '2D', '17:00', '02-02-2023', 1),('Shrek', '3D', '12:00', '03-02-2023', 2),
('Alicja w krainie czarow', '2D', '13:00', '03-02-2023', 1),('Alicja w krainie czarow', '2D VIP', '11:00', '03-02-2023', 3),
('Alicja w krainie czarow', '3D', '16:00', '03-02-2023', 2),('Alicja w krainie czarow', '3D VIP', '15:00', '02-02-2023', 4),
('Alicja w krainie czarow', '2D', '14:30', '07-02-2023', 1),('Alicja w krainie czarow', '3D', '14:00', '05-02-2023', 2),
('Avatar', '2D', '16:00', '03-02-2023', 1),('Avatar', '2D VIP', '15:00', '03-02-2023', 3),
('Avatar', '3D', '20:00', '03-02-2023', 2),('Avatar', '3D VIP', '18:00', '02-02-2023', 4),
('Avatar', '2D', '18:30', '07-02-2023', 1),('Avatar', '3D', '10:00', '05-02-2023', 2),
('Kosmonauci', '2D', '13:00', '04-02-2023', 1),('Kosmonauci', '2D VIP', '12:00', '04-02-2023', 3),
('Kosmonauci', '3D', '10:00', '04-02-2023', 2),('Kosmonauci', '3D VIP', '13:30', '05-02-2023', 4),
('Kosmonauci', '2D', '20:00', '04-02-2023', 1),('Kosmonauci', '3D', '15:00', '04-02-2023', 2),
('Kosmonauci', '2D', '17:30', '05-02-2023', 1),('Kosmonauci', '3D', '20:00', '05-02-2023', 2);

INSERT INTO statusy(nazwa_s) VALUES ('zaakceptowana'),('anulowana');

INSERT INTO klient(imie, nazwisko, e_mail, nr_telefonu, haslo) VALUES
('Magda', 'Potok', 'mpotok@gmail.com', 098765432, 'maxburgers1$'),
('Dawid', 'Padalec', 'dpadal@gmail.com', 123456789, 'haslo01'),
('Paulina', 'Adamczyk', 'padam@wp.pl', NULL, 'piesek1'),
('Ewelina', 'Osoka', 'eosok@onet.pl', 897654312, 'kotek2'),
('Arek', 'Kolec', 'akol123@gmail.com', 656565643, 'kucyk5'),
('Kacper', 'Malczyk', 'kmal@olo.sro', 201345621, 'konik8'),
('Konrad', 'Ludwicki', 'klud@gmail.com', NULL, 'shiba543'),
('Daniel', 'Kowalicki', 'dkowal@onet.pl', 090822364, 'cockerspaniel8'),
('Magdalena', 'Tokarczuk', 'mtokar@o2.pl', 111111111, 'maltanczyk65'),
('Dawid', 'Tusznio', 'dtusz@gmail.com', 212123456, 'york76'),
('Eliza', 'Majeranek', 'emaj@gmail.com', 564327865, 'roslinka93'),
('Kamil', 'Tymianek', 'ktym@wp.pl', 123498765, 'miska2');

INSERT INTO cennik (cena, nazwa_biletu, sala) VALUES
(22.22, 'normalny', 1), (15.60, 'ulgowy', 1),
(27.20, 'normalny', 2), (19.80, 'ulgowy', 2),
(25.50, 'normalny', 3), (17.20, 'ulgowy', 3),
(31.33, 'normalny', 4), (23.50, 'ulgowy', 4);

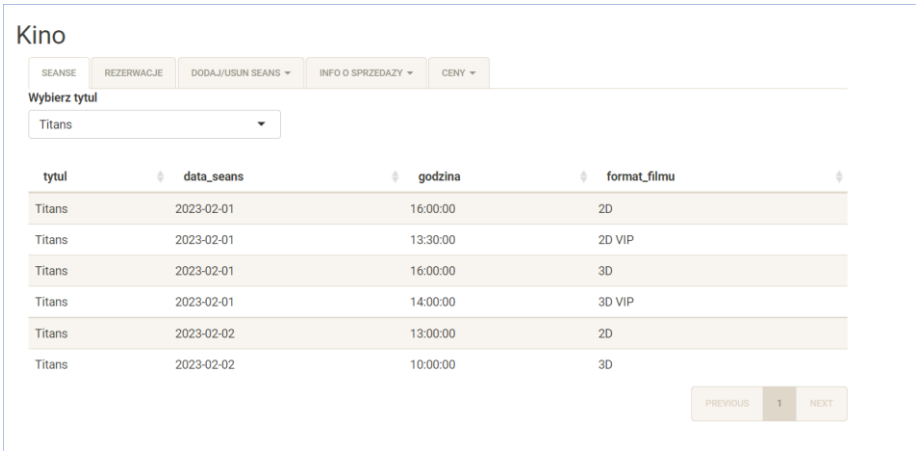
INSERT INTO rezerwacje(klient, ilosc_biletow, data_rezerw, status, seanse) VALUES
(1, 2, '26-01-2023', 1, 36),(2, 2, '26-01-2023', 2, 36),(3, 7, '27-01-2023', 1, 12),
(4, 3, '27-01-2023', 1, 11),(5, 4, '27-01-2023', 1, 10),(6, 2, '28-01-2023', 1, 13),
(7, 3, '28-01-2023', 2, 11),(8, 5, '28-01-2023', 1, 5),(9, 6, '29-01-2023', 2, 5),
(10, 7, '29-01-2023', 1, 21),(11, 2, '29-01-2023', 1, 15),(12, 3, '29-01-2023', 1, 10),
(1, 1, '30-01-2023', 2, 11),(2, 4, '30-01-2023', 1, 23),(3, 4, '30-01-2023', 1, 28),
(4, 1, '31-01-2023', 1, 7),(5, 2, '31-01-2023', 2, 32),(2, 1, '31-01-2023', 1, 31);

4 Interfejs graficzny bazy danych

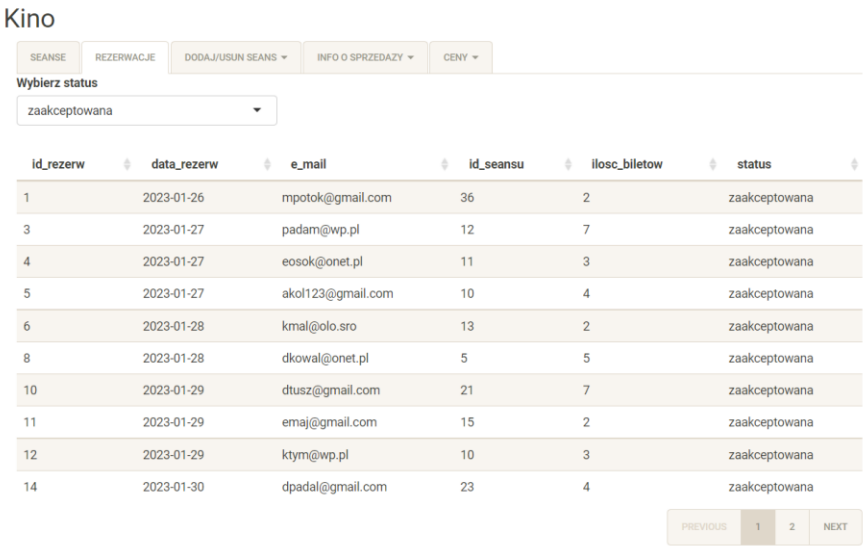
4.1 Interfejs graficzny

Do naszej bazy danych kina stworzyliśmy interfejs graficzny przy użyciu języka R pokazujący ją od strony administratora

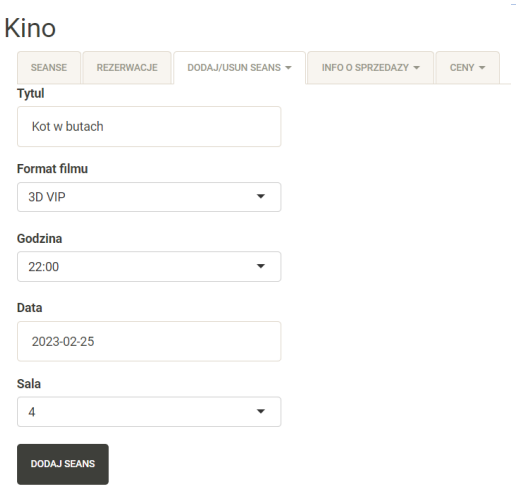
W zakładce „SEANSE” wybierając tytuł możemy zobaczyć informacje o danym tytule.

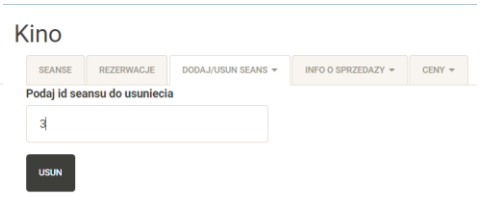


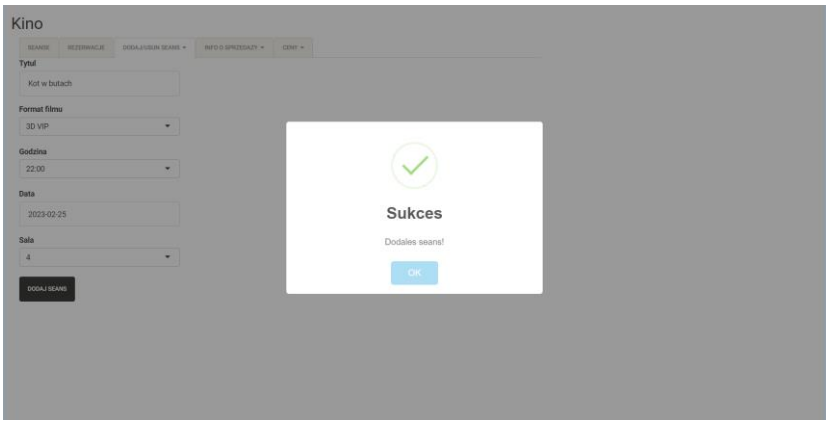
W zakładce „REZERWACJE” wybieramy jeden ze statusów, który nas interesuje i pokazuje nam wszystkie rezerwacje, które są zaakceptowane albo anulowane.

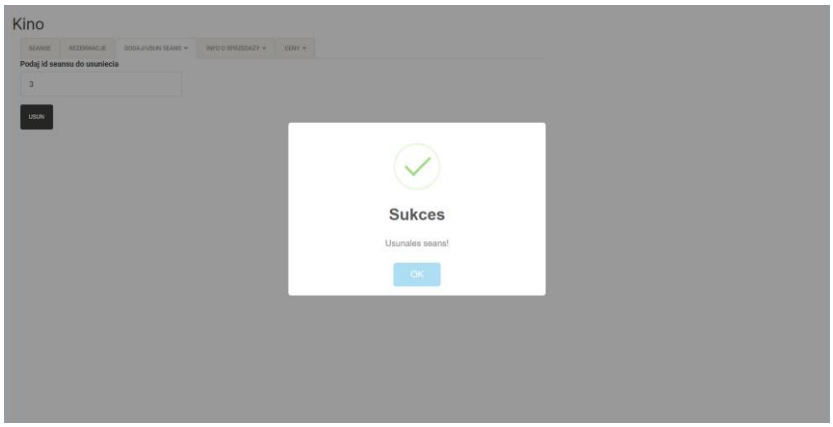


W kolejnej zakładce „DODAJ/USUN SEANS” możemy wybrać jedną z dostępnych opcji, czyli „DODAJ” lub „USUN”, gdzie odpowiednio opcja „DODAJ” dodaje nam seans do bazy danych, a opcja „USUN” usuwa seans z bazy danych. Po naciśnięciu przycisku „DODAJ SEANS” albo „USUN” pojawi się powiadomienie o wykonaniu tej czynności.



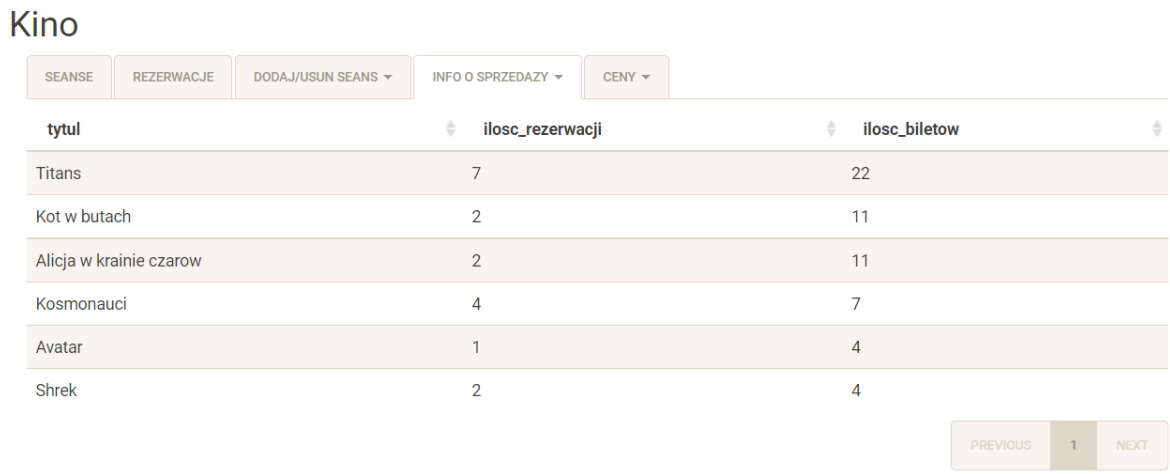






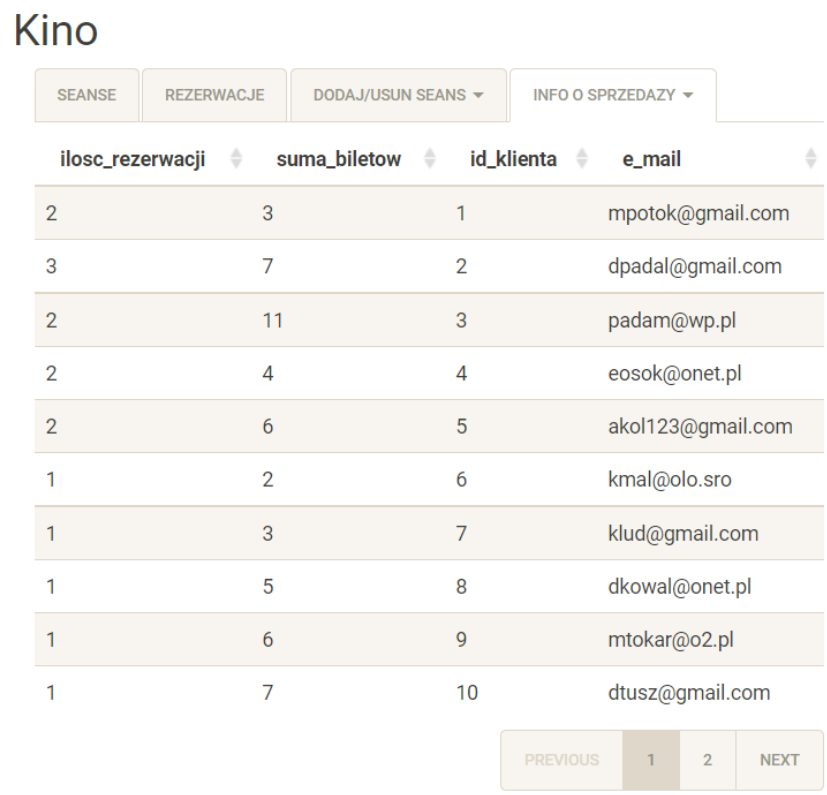
W następnej zakładce „INFO O SPRZEDAŻY” możemy zobaczyć 3 różne statystki, które mogą być istotne dla administratora.

Pierwszą z nich jest „RANKING TYTUŁOW” który pokazuje nam ,który tytuł ma najwięcej sprzedanych biletów.

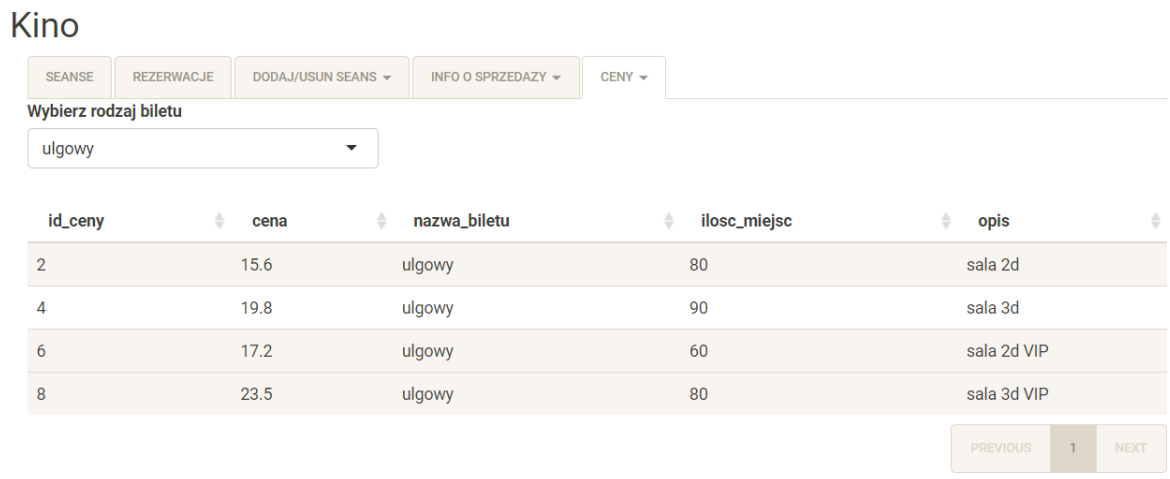


Drugą jest „RANKING FORMATOW” który tym razem pokazuje nam na jaki format filmu jest najwięcej sprzedanych biletów.

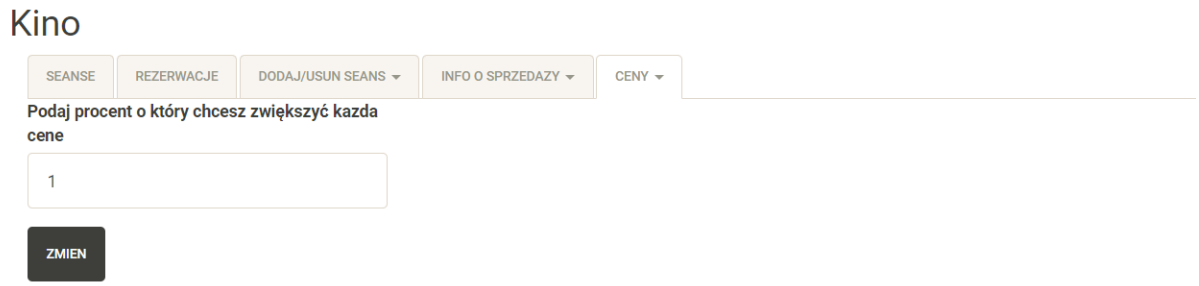
Ostatnią z nich jest „INFO O SPRZEDAZY’ które pokazuje nam ile biletów zakupił jaki klient.



W zakładce CENY możemy zobaczyć 2 informacje, jedną z nich jest „CENNIK”, w której możemy zobaczyć ceny dla wybranego przez nas rodzaju biletu.



Drugą jest „ZMIEN CENY’ w której możemy zmienić wszystkie ceny naraz o pewny procent.



4.2 Kod R

```
library("RPostgres")
library("shiny")
library("shinyalert")
library("shinythemes")

open.my.connection <- function() {
      con <- dbConnect(RPostgres::Postgres(),dbname = 'projekt6',
                  host = 'localhost',
                   port = 5432,
                  user = 'dawid',
                   password = 'Haslo098776')
       return (con)}

      close.my.connection <- function(con) {
      dbDisconnect(con)}

      load.tytul <- function() {
       query = "SELECT tytul FROM seanse"
       con = open.my.connection()
       res = dbSendQuery(con,query)
        tytuly = dbFetch(res)
        dbClearResult(res)
        close.my.connection(con)
        return(tytuly)}

      id_seansu <- function() {
        query = "SELECT id_seansu FROM seanse"
        con = open.my.connection()
        res = dbSendQuery(con,query)
        id_seansu = dbFetch(res)
        dbClearResult(res)
        close.my.connection(con)
        return(id_seansu)}
```

```
load.status <- function() {
query = "SELECT id_status FROM statusy"
 con = open.my.connection()
  res = dbSendQuery(con,query)
  statusy = dbFetch(res)
  dbClearResult(res)
  close.my.connection(con)
 return(statusy)}

load.rezerwacje <- function(status) {
 query = paste0("SELECT * FROM wszystkie_rezerwacje
     WHERE status = ", "'",status,"'")
  con = open.my.connection()
  res = dbSendQuery(con,query)
  rezerwacje = dbFetch(res)
  dbClearResult(res)
  close.my.connection(con)
  return(rezerwacje)}

load.r_t <- function() {
 query = paste0("SELECT * FROM ranking_tytulow")
 con = open.my.connection()
  res = dbSendQuery(con,query)
  r_t = dbFetch(res)
  dbClearResult(res)
  close.my.connection(con)
  return(r_t)}
load.r_f <- function() {
  query = paste0("SELECT * FROM ranking_formatow")
  con = open.my.connection()
  res = dbSendQuery(con,query)
  r_f = dbFetch(res)
  dbClearResult(res)
  close.my.connection(con)
  return(r_f)}

load.i_o_k <- function() {
  query = paste0("SELECT * FROM info_klient_sprzedaz")
  con = open.my.connection()
 res = dbSendQuery(con,query)
  i_o_k = dbFetch(res)
  dbClearResult(res)
  close.my.connection(con)
  return(i_o_k)}
```

```
load.seans <- function(tytul) {
  query = paste0("SELECT tytul, format_filmu, data_seans, godzina, sala
                         FROM seanse WHERE tytul = '", tytul,"'")
  con = open.my.connection()
  res = dbSendQuery(con,query)
  szczegoly = dbFetch(res)
  dbClearResult(res)
  close.my.connection(con)
  return(szczegoly)}

load.seanse.func <- function(title) {
  query = paste0("SELECT tytul, data_seans, godzina, format_filmu
                        FROM seanse WHERE tytul = '",title,"'")
  con = open.my.connection()
res = dbSendQuery(con,query)
 ratings = dbFetch(res)
  dbClearResult(res)
  close.my.connection(con)
 return(ratings)}

load.cennik.func <- function(rodzaj.biletu) {
  query = paste0("SELECT * FROM widok_ceny_biletow
              WHERE nazwa_biletu = '",rodzaj.biletu,"'")
  con = open.my.connection()
  res = dbSendQuery(con,query)
  ceny = dbFetch(res)
  dbClearResult(res)
  close.my.connection(con)
  return(ceny)}

add.or.update.seans <- function(tytul, format_filmu, godzina, data_seans, sala) {
  query = paste0("INSERT INTO seanse(tytul, format_filmu, godzina, data_seans, sala)
              VALUES('"
        ,tytul,"','",format_filmu,"','",godzina,"','",data_seans,"',",
          sala,")")
  con = open.my.connection()
  dbSendQuery(con,query)
  close.my.connection(con)}

u.seans <- function(id_seansu) {
  query = paste0("SELECT usun_seans(",id_seansu,")")
  con = open.my.connection()
  dbSendQuery(con,query)
  close.my.connection(con)}
```

```
zmien_cene_biletow <- function(procent) {
  query = paste0("SELECT zmien_cene(",procent,")")
  con = open.my.connection()
  dbSendQuery(con,query)
  close.my.connection(con)}

load.sala <- function() {
  query = "SELECT id_sali FROM sala"
  con = open.my.connection()
  res = dbSendQuery(con,query)
  sale = dbFetch(res)
  dbClearResult(res)
  close.my.connection(con)
  return(sale)}

shinyServer <- function(input, output, session) {

output$seanse.func <- renderDataTable(
  load.seanse.func(input$title),
options = list(
   pageLength = 10,
   lengthChange = FALSE,
   searching = FALSE,
   info = FALSE))

 output$cennik.func <- renderDataTable(
  load.cennik.func(input$rodzaj.biletu),
  options = list(
   pageLength = 10,
   lengthChange = FALSE,
   searching = FALSE,
   info = FALSE))

output$rezerwacje.func <- renderDataTable(
 load.rezerwacje(input$status),
 options = list(
  pageLength = 10,
  lengthChange = FALSE,
  searching = FALSE,
  info = FALSE))

output$r_t.func <- renderDataTable(
 load.r_t(),
options = list(
  pageLength = 10,
  lengthChange = FALSE,
  searching = FALSE,
  info = FALSE))
```

```
output$r_f.func <- renderDataTable(
  load.r_f(),
options = list(
 pageLength = 10,
 lengthChange = FALSE,
 searching = FALSE,
 info = FALSE))

output$i_o_k.func <- renderDataTable(
  load.i_o_k(),
options = list(
  pageLength = 10,
  lengthChange = FALSE,
  searching = FALSE,
  info = FALSE))

 observeEvent(input$add.seans,
          add.or.update.seans(
                  input$seanse.tytul,
                   input$seanse.format,
                   input$seanse.godzina,
                   input$seanse.data,
                   input$seanse.sala))


 observeEvent(input$zmien_cene_biletow,
              zmien_cene_biletow(input$procent))

 observeEvent(input$u.seans,
              u.seans(input$id_seansu))

observeEvent(input$add.seans, {
          shinyalert(title = "Sukces", type = "success", text = "Dodales seans!")})

observeEvent(input$u.seans, {
          shinyalert(title = "Sukces", type = "success", text = "Usunales seans!")})

 observeEvent(input$zmien_cene_biletow, {
          shinyalert(title = "Sukces", type = "success", text = "Zmieniles ceny biletow!")})
}
```

```
shinyUI <- fluidPage(
 theme = shinytheme("sandstone"),
 shinyalert::useShinyalert(),
 titlePanel("Kino"),
 mainPanel(
tabsetPanel(
tabPanel('Seanse',
 selectInput(inputId='title',
             label='Wybierz tytul',
              choices=load.tytul()),
            dataTableOutput('seanse.func'),
 textOutput('seanse')),


tabPanel('Rezerwacje',
            selectInput(inputId='status',
              label='Wybierz status',
              choices=list("zaakceptowana","anulowana")),
  dataTableOutput('rezerwacje.func'),
  textOutput('rezerwacje')),


navbarMenu("Dodaj/Usun seans",
            tabPanel('Dodaj',
                       textInput(inputId='seanse.tytul',
                       label='Tytul'),
                     selectInput(inputId='seanse.format',
                       label='Format filmu',
                       choices=list('2D', '3D', '2D VIP', '3D VIP')),
                     selectInput(inputId='seanse.godzina',
                       label='Godzina',
                       choices=list('8:30', '10:00', '11:30', '13:00', '14:30', '16:00', '17:30', '19:00',
                       '20:30', '22:00')),
                      dateInput(inputId='seanse.data',
                       label='Data'),
                       selectInput(inputId='seanse.sala',
                       label='Sala',
                       choices=load.sala()),
                     actionButton(inputId='add.seans',
                      label='Dodaj seans')),
             tabPanel('Usun',
                     textInput(inputId='id_seansu',
                       label='Podaj id seansu do usuniecia'),
                      actionButton(inputId='u.seans',
                       label='usun'))),
```

```
        navbarMenu("Info o sprzedazy",
                           tabPanel('Ranking tytulow',
                                    dataTableOutput('r_t.func'),
                                    textOutput('r_t')),
                           tabPanel('Ranking formatow',
                                    dataTableOutput('r_f.func'),
                                    textOutput('r_f')),
                           tabPanel('Info o klientach',
                                    dataTableOutput('i_o_k.func'),
                                    textOutput('i_o_k'))),

         navbarMenu("Ceny",
                           tabPanel("Cennik",
                                      selectInput(inputId='rodzaj.biletu',
                                      label='Wybierz rodzaj biletu',
                                      choices=list("ulgowy", "normalny")),
                                      dataTableOutput('cennik.func'),
                                      textOutput('cennik')),
                           tabPanel('Zmien ceny',
                                    numericInput(inputId='procent',
                                    label='Podaj procent o który chcesz zwiększyć kazda cene',
                                    value=1, min=1, max=100),
                                   actionButton(inputId='zmien_cene_biletow',
                                    label='Zmien'))))))

shinyApp(ui = shinyUI, server = shinyServer)
```