

Bazy Danych - Projekt

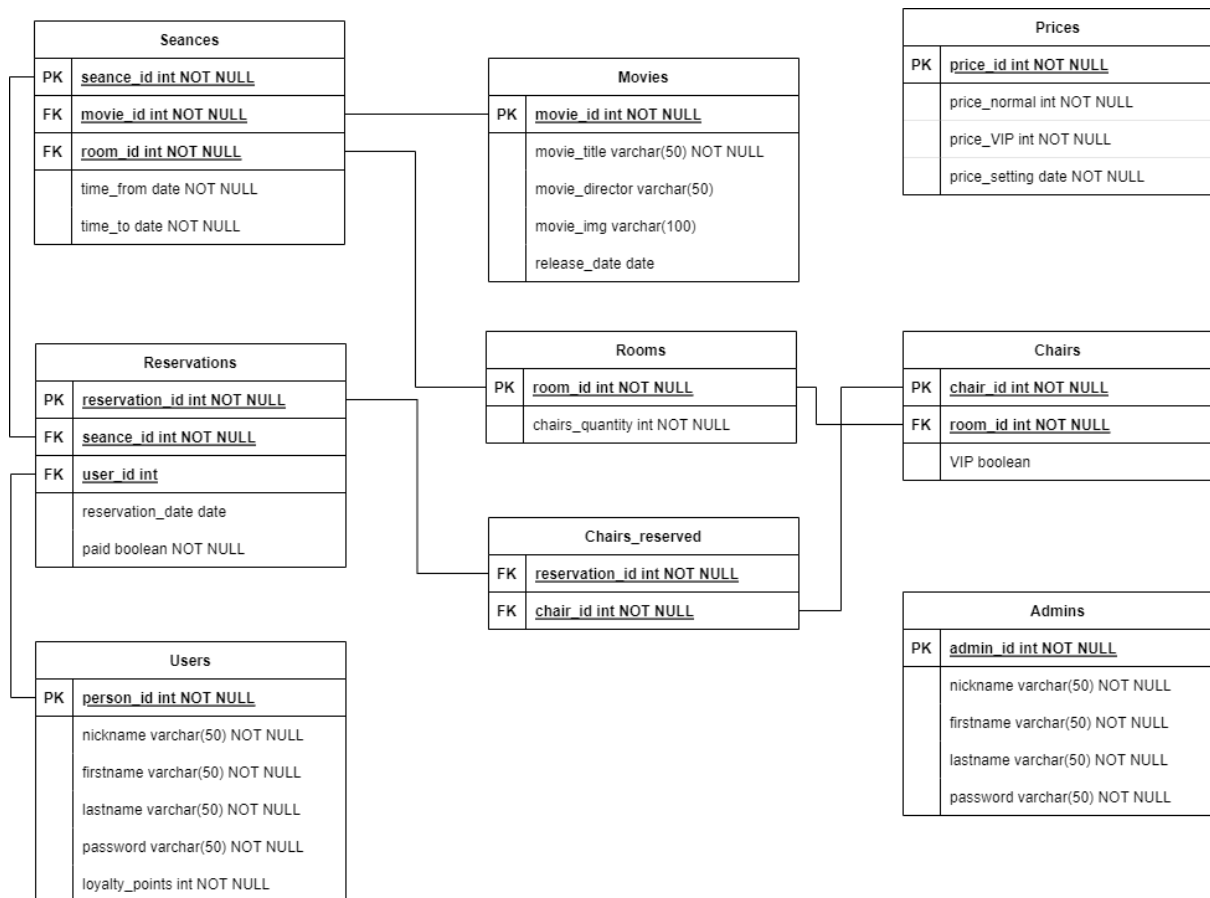
Temat: Rezerwacja i zakup biletów w kinie

Aleksandra Poskróbek i Kacper Kotkiewicz

1. Schemat bazy danych	4
2. Tabele	4
2.1. Tabela Seances	4
2.2. Tabela Movies	4
2.3. Tabela Reservations	5
2.4. Tabela Rooms	5
2.5. Tabela Chairs	5
2.6. Tabela ReservedChairs	6
2.7. Tabela Users	6
2.8. Tabela Admins	6
2.9. Tabela Prices	6
3. Obiekty i typy	6
3.1. Obiekt SEANCE_INFO	6
3.2. Obiekt SEAT_INFO	7
3.3. Typ NUM_ARRAY	7
4. Funkcje	7
4.1. Funkcja SEANCES_PER_MOVIE_ID	7
4.2. Funkcja SEATS_PER_SEANCE_ID	8
4.3. Funkcja IS_RESERVATION_PAID	9
4.4. Funkcja GET_CURRENT_PRICE	9
4.5. Funkcja GET_RESERVATION_PRICE	10
4.6. Funkcja RESERVATION_EXISTS	10
4.7. Funkcja USER_EXISTS	11
4.8. Funkcja SEANCE_EXISTS	11
4.9. Funkcja ROOM_EXISTS	12
4.10. Funkcja CHAIR_EXISTS	12
4.11. Funkcja MOVIE_EXISTS	13
5. Procedury	13
5.1. Procedura add_reserved_chairs	13
5.2. Procedura add_reservation	14
5.3. Procedura pay_for_reservation	14
5.4. Procedura ADD_USER	14
5.5. Procedura ADD_ADMIN	15
5.6. Procedura ADD_LOYALTY_POINTS	15
5.7. Procedura DELETE_RESERVATION	15
6. Triggery	16
6.1. Trigger SEAT_ADD	16
6.2. Trigger SEAT_DELETE	16
6.3. Trigger RESERVATION_ADD	17
6.4. Trigger CHAIRS_ADD	17
6.5. Trigger RESERVED_CHAIRS_CHECK_IF_TAKEN	18
6.6. Trigger RESERVATION_PAYMENT	18

7. Serwer (Express.js)	19
7.1. Połączenie z bazą	19
7.2. Pobranie dostępnych filmów	19
7.3. Pobranie informacji odnośnie dostępnych dla filmu seansów	20
7.4. Pobranie informacji o krzesłach dostępnych dla wybranego seansu	21
7.5. Dokonanie rezerwacji	22
8. Frontend (React.js)	23
8.1. Wygląd głównej strony kina	23
8.2. Wybór dostępnego dla klikniętego filmu seansu	23
8.3. Widok sali kinowej z dostępnymi miejscami	23
8.4. Wybór miejsc	24
8.5. Rezerwacja wybranych wcześniej miejsc	24

1. Schemat bazy danych



Boolean w bazie danych napisanej w Oracle SQL został zastąpiony typem Number (1, 0)

2. Tabele

2.1. Tabela Seances

Tabela odpowiedzialna za przechowywanie informacji dotyczących seansów, gdzie zebrane są dane na temat filmu, sali w której seans się odbywa oraz czasu rozpoczęcia i zakończenia seansu.

	SEANCE_ID	MOVIE_ID	ROOM_ID	TIME_FROM	TIME_TO
1	1	2	1	2023-05-20 10:30:00	2023-05-20 12:30:00
2	2	2	1	2023-05-20 13:00:00	2023-05-20 15:30:00
3	3	2	1	2023-05-20 16:00:00	2023-05-20 18:30:00

2.2. Tabela Movies

Przechowywane są tu informacje na temat filmu, takie jak tytuł, reżyser, obrazek reprezentujący dany film oraz datę wypuszczenia.

MOVIE_ID	MOVIE_TITLE	MOVIE_DIRECTOR	MOVIE_IMG	RELEASE_DATE
2	Little Mermaid	Rob Marshall	https://fwcdn.pl/fpo/60/56/836056/8066248.3.jpg	2023-05-26
3	Guardians of the Galaxy Vol. 3	James Gunn	https://fwcdn.pl/fpo/60/77/806877/8065161.3.jpg	2023-05-05
4	Fast & Furious X	Louis Leterrier	https://www.fastmovie.com/images/main/mobile-optimized-v2.png?id=8_7	2023-05-19
21	John Wick 4	Chad Stahelski	https://fwcdn.pl/fpo/15/20/831520/8061066.3.jpg	2023-03-24
22	Dungeons & Dragons: Honor Among Thieves	John Francis Daley	https://fwcdn.pl/fpo/14/95/751495/8056026.3.jpg	2023-04-14
23	Barbie	Greta Gerwig	https://fwcdn.pl/fpo/48/00/754800/8049756.3.jpg	2023-07-21
24	The Flash	Andres Muschietti	https://fwcdn.pl/fpo/45/18/494518/8057595.3.jpg	2023-06-14
25	Suzume	Makoto Shinkai	https://fwcdn.pl/fpo/13/68/1001368/8066510.3.jpg	2023-06-21
26	Super Mario Bros. Movie	Michael Jelenic	https://fwcdn.pl/fpo/05/81/850581/8057312.3.jpg	2023-05-26

2.3. Tabela Reservations

Każda rezerwacja zawiera informację o id seansu, id użytkownika (jeżeli rezerwuje zalogowany użytkownik) oraz datę rezerwacji i informację o płatności. Siedzenia zarezerwowane przez użytkownika, jako że dla danej rezerwacji może ich być więcej, znajdują się w osobnej tabeli.

	RESERVATION_ID	SEANCE_ID	USER_ID	RESERVATION_DATE	PAID
1	21	3	<null>	2023-05-11 11:38:04	0
2	22	2	<null>	2023-05-11 11:44:09	0
3	23	2	<null>	2023-05-11 11:52:33	0
4	24	2	<null>	2023-05-11 11:53:12	0
5	53	3	1	2023-05-12 16:24:34	0
6	1	1	<null>	2023-05-10 22:08:16	0
7	62	2	<null>	2023-05-13 19:27:09	0

2.4. Tabela Rooms

Są tu wymienione dostępne sale kinowe oraz ilość miejsc każdej z nich.

	ROOM_ID	CHAIRS_QUANTITY
1	1	30
2	2	0
3	3	0

2.5. Tabela Chairs

Ta tabela zawiera krzesła, którym przypisane są numery sal, w których się znajdują oraz informację czy są to krzesła VIP (wygodniejsze i droższe).

	CHAIR_ID	ROOM_ID	VIP
1	1	1	1
2	2	1	1
3	3	1	1
4	4	1	1
5	5	1	0
6	6	1	0
7	7	1	0
8	8	1	0
9	9	1	0
10	10	1	0

2.6. Tabela ReservedChairs






Tabela bezpośrednio powiązana z rezerwacjami. Zawiera zarezerwowane krzesła dla danej rezerwacji (a co za tym idzie dla konkretnego seansu oraz sali).

	RESERVATION_ID	CHAIR_ID
1	1	2
2	1	5
3	1	9
4	21	2
5	21	5
6	21	9
7	22	2
8	22	5
9	22	9

2.7. Tabela Users

	USER_ID	NICKNAME	FIRSTNAME	LASTNAME	PASSWORD	LOYALTY_POINTS
1	1	Mario	Marian	Kowalski	xxx	20

2.8. Tabela Admins

WHERE	ORDER BY
 ADMIN_ID	 NICKNAME
 FIRSTNAME	 LASTNAME
 PASSWORD	

2.9. Tabela Prices

Jest tu możliwość przechowywania cen dla normalnych i VIP-owskich biletów wraz z datą ustanowienia tych cen.

	PRICE_ID	PRICE_NORMAL	PRICE_VIP	PRICE_SETTING
1	41	25	35	2023-05-15 09:56:27
2	21	20	30	2023-05-12 09:59:44

3. Obiekty i typy

3.1. Obiekt SEANCE_INFO

```
create TYPE SEANCE_INFO AS OBJECT
```

```
(
    SEANCE_ID NUMBER,
    MOVIE_ID NUMBER,
    MOVIE_TITLE VARCHAR2(50),
    ROOM_ID NUMBER,
    TIME_FROM DATE,
    TIME_TO DATE
);
create type SEANCE_INFO_TABLE as table of SEANCE_INFO
```

3.2. Obiekt SEAT_INFO

```
create TYPE SEAT_INFO AS OBJECT
(
    SEANCE_ID NUMBER,
    SEAT_ID NUMBER,
    SEAT_TAKEN NUMBER(1,0),
    VIP NUMBER(1,0)
);
create type SEAT_INFO_TABLE as table of SEAT_INFO
```

3.3. Typ NUM_ARRAY

```
create type NUM_ARRAY as table of NUMBER(0)
```

4. Funkcje

4.1. Funkcja SEANCES_PER_MOVIE_ID

Zwraca dostępne seanse dla filmu o podanym ID

```
create or replace FUNCTION SEANCES_PER_MOVIE_ID( id
MOVIES.MOVIE_ID%TYPE)
RETURN SEANCE_INFO_TABLE
AS
    result SEANCE_INFO_TABLE;
BEGIN
    IF MOVIE_EXISTS(id) = 0 THEN
        RAISE_APPLICATION_ERROR(-20000 , 'Movie with chosen ID does not
exist' );
    END IF;
```

```

        SELECT SEANCE_INFO(s.SEANCE_ID, s.MOVIE_ID, m.MOVIE_TITLE,
s.ROOM_ID, s.TIME_FROM, s.TIME_TO)
        BULK COLLECT
        INTO result
        FROM SEANCES s
            JOIN MOVIES m
                ON m.MOVIE_ID = s.MOVIE_ID
        WHERE s.MOVIE_ID = id
        AND s.TIME_FROM > CURRENT_DATE;

    RETURN result;
END;
```

4.2. Funkcja SEATS_PER_SEANCE_ID

Funkcja zwraca dostępne miejsca dla danego seansu oraz to czy są zajęte.

```

create or replace FUNCTION SEATS_PER_SEANCE_ID( id
SEANCES.SEANCE_ID%TYPE)
    RETURN SEAT_INFO_TABLE
AS
    result SEAT_INFO_TABLE;
BEGIN
    IF SEANCE_EXISTS(id) = 0 THEN
        RAISE_APPLICATION_ERROR(-20000 , 'Seance with chosen ID does not
exist' );
    END IF;

    SELECT SEAT_INFO(s.SEANCE_ID, c.CHAIR_ID, (select count (*) as
reserved from RESERVATIONS r2
                                                    JOIN SEANCES p
                                                    ON p.SEANCE_ID =
r2.SEANCE_ID
                                                    JOIN CHAIRS_RESERVED C2
on r2.RESERVATION_ID = C2.RESERVATION_ID
                                                    WHERE p.SEANCE_ID = id AND
C2.CHAIR_ID = c.CHAIR_ID), c.VIP)
        BULK COLLECT
        INTO result
        FROM SEANCES s
            JOIN ROOMS r
                ON s.ROOM_ID = r.ROOM_ID
            JOIN CHAIRS c
                ON r.ROOM_ID = c.ROOM_ID
        WHERE s.SEANCE_ID = id;
```



```
    RETURN result;
END;
```

4.3. Funkcja IS_RESERVATION_PAID

```
create or replace FUNCTION IS_RESERVATION_PAID( id
RESERVATIONS.RESERVATION_ID%TYPE)
    RETURN RESERVATIONS.PAID%TYPE
AS
    result RESERVATIONS.PAID%TYPE;
BEGIN
    IF RESERVATION_EXISTS(id) = 0 THEN
        RAISE_APPLICATION_ERROR(-20000 , 'Reservation with chosen ID
does not exist' );
    END IF;

    SELECT RESERVATIONS.PAID
    INTO result
    FROM RESERVATIONS
    WHERE RESERVATIONS.RESERVATION_ID = id;

    RETURN result;
END;
```

4.4. Funkcja GET_CURRENT_PRICE

Pobiera aktualną cenę dla krzesła VIP/normalnego (podane w parametrach).

```
create or replace FUNCTION GET_CURRENT_PRICE( CHAIR_TYPE
CHAIRS.VIP%TYPE)
    RETURN NUMBER
AS
    result NUMBER;
BEGIN
    IF CHAIR_TYPE = 0 THEN
        SELECT PRICES.PRICE_NORMAL
        INTO result
        FROM PRICES
        ORDER BY PRICES.PRICE_SETTING DESC
        fetch first 1 row only;
    ELSE
        SELECT PRICES.PRICE_VIP
        INTO result
```

```

        FROM PRICES
        ORDER BY PRICES.PRICE_SETTING DESC
        fetch first 1 row only;
    END IF;

    RETURN result;
END;
```

4.5. Funkcja GET_RESERVATION_PRICE

Korzystając z funkcji GET_CURRENT_PRICE oblicza wartość podanej w parametrach rezerwacji.

```

create or replace FUNCTION GET_RESERVATION_PRICE( id
RESERVATIONS.RESERVATION_ID%TYPE)
    RETURN NUMBER
AS
    result NUMBER;
BEGIN
    IF RESERVATION_EXISTS(id) = 0 THEN
        RAISE_APPLICATION_ERROR(-20000 , 'Reservation with chosen ID
does not exist' );
    END IF;

    SELECT SUM(GET_CURRENT_PRICE(C2.VIP))
    INTO result
    FROM RESERVATIONS
    JOIN CHAIRS_RESERVED CR on RESERVATIONS.RESERVATION_ID =
CR.RESERVATION_ID
    JOIN CHAIRS C2 on C2.CHAIR_ID = CR.CHAIR_ID
    WHERE RESERVATIONS.RESERVATION_ID = id;

    RETURN result;
END;
```

4.6. Funkcja RESERVATION_EXISTS

```

create or replace FUNCTION RESERVATION_EXISTS( id
RESERVATIONS.RESERVATION_ID%TYPE)
    RETURN NUMBER
AS
    result NUMBER;
    reservation_exists INT;
BEGIN
```

```

        SELECT COUNT(*) INTO reservation_exists FROM RESERVATIONS WHERE
        RESERVATIONS.RESERVATION_ID = id;

        IF reservation_exists = 0 THEN
            result := 0;
        ELSE
            result := 1;
        END IF;

        RETURN result;
    END;

```

4.7. Funkcja USER_EXISTS

```

create or replace FUNCTION USER_EXISTS( id USERS.USER_ID%TYPE)
    RETURN NUMBER
AS
    result NUMBER;
    user_exists INT;
BEGIN
    SELECT COUNT(*) INTO user_exists FROM USERS WHERE USERS.USER_ID = id;

    IF user_exists = 0 THEN
        result := 0;
    ELSE
        result := 1;
    END IF;

    RETURN result;
END;

```

4.8. Funkcja SEANCE_EXISTS

```

create or replace FUNCTION SEANCE_EXISTS( id SEANCES.SEANCE_ID%TYPE)
    RETURN NUMBER
AS
    result NUMBER;
    seance_exists INT;
BEGIN
    SELECT COUNT(*) INTO seance_exists FROM SEANCES WHERE
    SEANCES.SEANCE_ID = id;

    IF seance_exists = 0 THEN
        result := 0;
    END IF;

    RETURN result;
END;

```

```

ELSE
    result := 1;
END IF;

RETURN result;
END;

```

4.9. Funkcja ROOM_EXISTS

```

create or replace FUNCTION ROOM_EXISTS( id ROOMS.ROOM_ID%TYPE)
RETURN NUMBER
AS
    result NUMBER;
    room_exists INT;
BEGIN
    SELECT COUNT(*) INTO room_exists FROM ROOMS WHERE ROOMS.ROOM_ID =
id;

    IF room_exists = 0 THEN
        result := 0;
    ELSE
        result := 1;
    END IF;

    RETURN result;
END;

```

4.10. Funkcja CHAIR_EXISTS

```

create or replace FUNCTION CHAIR_EXISTS( id CHAIRS.CHAIR_ID%TYPE)
RETURN NUMBER
AS
    result NUMBER;
    chairs_exist INT;
BEGIN
    SELECT COUNT(*) INTO chairs_exist FROM CHAIRS WHERE CHAIRS.CHAIR_ID
= id;

    IF chairs_exist = 0 THEN
        result := 0;
    ELSE
        result := 1;
    END IF;

```

```
        RETURN result;
END;
```

4.11. Funkcja MOVIE_EXISTS

```
create or replace FUNCTION MOVIE_EXISTS( id MOVIES.MOVIE_ID%TYPE)
RETURN NUMBER
AS
    result NUMBER;
    movie_exists INT;
BEGIN
    SELECT COUNT(*) INTO movie_exists FROM MOVIES WHERE MOVIES.MOVIE_ID
= id;

    IF movie_exists = 0 THEN
        result := 0;
    ELSE
        result := 1;
    END IF;

    RETURN result;
END;
```

5. Procedury

5.1. Procedura add_reserved_chairs

Dodaje podane w tablicy krzesła do tabeli CHAIRS_RESERVED z przypisanym podanym jako parametr id rezerwacji.

```
create procedure add_reserved_chairs ( chairs_reserved_ids num_array,
id_reservation RESERVATIONS.RESERVATION_ID%type)
AS
begin
    for i in 1..chairs_reserved_ids.count loop
        INSERT INTO CHAIRS_RESERVED(reservation_id, chair_id)
        VALUES(id_reservation, TO_NUMBER(chairs_reserved_ids(i)));
    end loop;
    COMMIT;
end;
```

5.2. Procedura add_reservation

Bezpośrednio połączona z poprzednią (add_reserved_chairs), dodaje rezerwację do tabeli RESERVATIONS oraz uruchamia procedurę dodającą zarezerwowane miejsca do tabeli CHAIRS_RESERVED.

```
create procedure add_reservation ( chairs_reserved_ids num_array,
                                id_seance
RESERVATIONS.SEANCE_ID%type,
                                id_user
RESERVATIONS.USER_ID%type,
                                paid RESERVATIONS.PAID%type)
AS
    id RESERVATIONS.RESERVATION_ID%TYPE;
begin
    INSERT INTO RESERVATIONS(seance_id, user_id, reservation_date, paid)
    VALUES(id_seance, id_user, CURRENT_DATE, paid) returning
RESERVATION_ID into id;
    COMMIT;
    ADD_RESERVED_CHAIRS(chairs_reserved_ids, id);
end;
```

5.3. Procedura pay_for_reservation

```
create procedure pay_for_reservation ( id_reservation number)
AS
begin
    UPDATE RESERVATIONS SET RESERVATIONS.PAID = 1 WHERE
RESERVATIONS.RESERVATION_ID = id_reservation;
    COMMIT;
end;
```

5.4. Procedura ADD_USER

```
create procedure ADD_USER (nick USERS.NICKNAME%TYPE, fn
USERS.FIRSTNAME%TYPE, ln USERS.LASTNAME%TYPE, ps USERS.PASSWORD%TYPE)
AS
begin
    INSERT INTO USERS(NICKNAME, FIRSTNAME, LASTNAME, PASSWORD,
LOYALTY_POINTS)
    VALUES(nick, fn, ln, ps, 0);
    COMMIT;
end;
```

5.5. Procedura ADD_ADMIN

```
create procedure ADD_ADMIN (nick USERS.NICKNAME%TYPE, fn
USERS.FIRSTNAME%TYPE, ln USERS.LASTNAME%TYPE, ps USERS.PASSWORD%TYPE)
AS
begin
    INSERT INTO ADMINS(NICKNAME, FIRSTNAME, LAST NAME, PASSWORD)
    VALUES(nick, fn, ln, ps);
    COMMIT;
end;
```

5.6. Procedura ADD_LOYALTY_POINTS

```
create procedure ADD_LOYALTY_POINTS (id USERS.USER_ID%TYPE, points
USERS.LOYALTY_POINTS%TYPE)
AS
begin
    IF USER_EXISTS(id) = 0 THEN
        RAISE_APPLICATION_ERROR(-20002 , 'User with chosen ID not
found. ');
    END IF;

    UPDATE USERS SET USERS.LOYALTY_POINTS = USERS.LOYALTY_POINTS +
points WHERE USERS.USER_ID = id;
    COMMIT;
end;
```

5.7. Procedura DELETE_RESERVATION

```
create or replace procedure DELETE_RESERVATION (id
RESERVATIONS.RESERVATION_ID%TYPE)
AS
begin
    IF IS_RESERVATION_PAID(id) = 1 THEN
        RAISE_APPLICATION_ERROR(-20000 , 'Reservation with chosen ID is
already paid' );
    end if;

    DELETE (SELECT * FROM CHAIRS_RESERVED
    JOIN RESERVATIONS R on CHAIRS_RESERVED.RESERVATION_ID =
R.RESERVATION_ID
    WHERE id = R.RESERVATION_ID);
    COMMIT;
```

```
DELETE FROM RESERVATIONS
WHERE RESERVATION_ID = id;
COMMIT;
end;
```

6. Triggery

6.1. Trigger SEAT_ADD

Przy dodawaniu krzeseł do tabeli CHAIRS sprawdza czy pokój do którego chcemy przypisać krzesło istnieje oraz nalicza ilość miejsc w tabeli ROOMS.

```
create or replace trigger SEAT_ADD
after insert
on CHAIRS
for each row
DECLARE
BEGIN
  IF ROOM_EXISTS(:new.ROOM_ID) = 0 THEN
    RAISE_APPLICATION_ERROR(-20002 , 'Room with chosen ID not
found. ');
  END IF;

  UPDATE ROOMS
    set rooms.CHAIRS_QUANTITY = ROOMS.CHAIRS_QUANTITY + 1
  where rooms.ROOM_ID = :new.ROOM_ID;
END;
```

6.2. Trigger SEAT_DELETE

Przy usuwaniu krzeseł do tabeli CHAIRS sprawdza czy pokój z którego chcemy usunąć krzesło istnieje oraz nalicza ilość miejsc w tabeli ROOMS.

```
create or replace trigger SEAT_DELETE
after delete
on CHAIRS
for each row
DECLARE
BEGIN
  IF ROOM_EXISTS(:new.ROOM_ID) = 0 THEN
    RAISE_APPLICATION_ERROR(-20002 , 'Room with chosen ID not
found. ');
```



```
END IF;
```

```
UPDATE ROOMS
  set rooms.CHAIRS_QUANTITY = ROOMS.CHAIRS_QUANTITY - 1
  where rooms.ROOM_ID = :new.ROOM_ID;
END;
```

6.3. Trigger RESERVATION_ADD

Przy dodawaniu rezerwacji sprawdza czy istnieje wybrany seans i jeżeli został podany użytkownik to czy takowy również istnieje.

```
create or replace trigger RESERVATION_ADD
  before insert
  on RESERVATIONS
  for each row
DECLARE
BEGIN
  IF SEANCE_EXISTS(:new.SEANCE_ID) = 0 or (USER_EXISTS(:new.USER_ID) = 0
and :new.USER_ID != NULL) THEN
    RAISE_APPLICATION_ERROR(-20002 , 'Seance or user with chosen ID not
found. ');
  END IF;
END;
```

6.4. Trigger CHAIRS_ADD

Przy dodawaniu krzeseł do rezerwacji sprawdza czy wybrane krzesła istnieją w puli krzeseł do wyboru oraz czy podana rezerwacja istnieje.

```
create or replace trigger RESERVED_CHAIRS_ADD
  before insert
  on CHAIRS_RESERVED
  for each row
DECLARE
BEGIN
  IF CHAIR_EXISTS(:new.CHAIR_ID) = 0 or
RESERVATION_EXISTS(:new.RESERVATION_ID) = 0 THEN
    RAISE_APPLICATION_ERROR(-20002 , 'Room or chair with chosen ID not
found. ');
  END IF;
END;
```

6.5. Trigger RESERVED_CHAIRS_CHECK_IF_TAKEN

Przy dodawaniu krzeseł sprawdza czy wybrane krzesło nie jest już przypadkiem zajęte.

```
create trigger RESERVED_CHAIRS_CHECK_IF_TAKEN
  before insert
  on CHAIRS_RESERVED
  for each row
DECLARE
  chair_exists int;
BEGIN
  SELECT COUNT(*) INTO chair_exists FROM CHAIRS_RESERVED
    JOIN RESERVATIONS R on CHAIRS_RESERVED.RESERVATION_ID =
R.RESERVATION_ID
    JOIN SEANCES S on S.SEANCE_ID = R.SEANCE_ID
  WHERE CHAIRS_RESERVED.CHAIR_ID = :new.CHAIR_ID
    and S.SEANCE_ID = (select SEANCE_ID FROM RESERVATIONS WHERE
RESERVATIONS.RESERVATION_ID = :new.RESERVATION_ID);

  IF chair_exists > 0 THEN
    RAISE_APPLICATION_ERROR(-20002 , 'This chair for chosen seance is
already taken');
  END IF;
END;
```

6.6. Trigger RESERVATION_PAYMENT

Sprawdza czy rezerwacja istnieje oraz czy nie była już wcześniej opłacona.

```
create or replace trigger RESERVATION_PAYMENT
  before update
  on RESERVATIONS
  for each row
DECLARE
  paid RESERVATIONS.PAID%TYPE;
BEGIN
  SELECT IS_RESERVATION_PAID(:old.PAID) INTO paid FROM dual;

  IF RESERVATION_EXISTS(:old.RESERVATION_ID) = 0 THEN
    RAISE_APPLICATION_ERROR(-20002 , 'Reservation with chosen ID not
found. ');
  END IF;
  IF paid = 1 THEN
    RAISE_APPLICATION_ERROR(-20002 , 'Reservation already paid');
  END IF;
```

```
END;
```

7. Serwer (Express.js)

7.1. Połączenie z bazą

```
const express = require("express");
const bodyParser = require('body-parser');
const cors = require('cors');
const port = 4000;

let app = express();
app.use(cors());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

const oracledb = require("oracledb");
oracledb.outFormat = oracledb.OUT_FORMAT_OBJECT;

const params = {
  user: "BD_411227",
  password: "st8735",
  connectionString: "(DESCRIPTION =(ADDRESS = (PROTOCOL = TCP)(HOST =
dbmanage.lab.ii.agh.edu.pl)(PORT = 1521))(CONNECT_DATA =(SID=
DBMANAGE)))"
};

const connect = async () => {
  try {
    const connection = await oracledb.getConnection({ ...params });
    return connection;
  } catch (error) {
    console.error(`[ERROR] ${error}`);
  }
};
```

7.2. Pobranie dostępnych filmów

Żądanie do bazy pobiera z niej wszystkie informacje dotyczące granych w kinie filmów.

```
app.get("/movies", async (req, res) => {
  const conn = await connect();
  const query = "select * from MOVIES";
```

```

conn?.execute(query, [], { autoCommit: true }, (error, result) => {
  if (error) {
    return res.status(500).json({
      message: error.message,
      error,
    });
  } else {
    return res.status(201).json(result.rows);
  }
});
});
})

```

Na localhost:4000/movies:

```

[{"MOVIE_ID":2,"MOVIE_TITLE":"Little Mermaid","MOVIE_DIRECTOR":"Rob
Marshall","MOVIE_IMG":"https://fwcdn.pl/fpo/60/56/836056/8066248.3.jpg","RELEASE_DATE":"2023-05-25T22:00:00
.000Z"},{"MOVIE_ID":3,"MOVIE_TITLE":"Guardians of the Galaxy Vol. 3","MOVIE_DIRECTOR":"James
Gunn","MOVIE_IMG":"https://fwcdn.pl/fpo/60/77/806077/8065161.3.jpg","RELEASE_DATE":"2023-05-04T22:00:00.00
0Z"},{"MOVIE_ID":4,"MOVIE_TITLE":"Fast & Furious X","MOVIE_DIRECTOR":"Louis
Leterrier","MOVIE_IMG":"https://www.fastxmovie.com/images/main/mobile-optimized-v2.png?id=8_7","RELEASE_DATE":"
2023-05-18T22:00:00.000Z"},{"MOVIE_ID":21,"MOVIE_TITLE":"John Wick 4","MOVIE_DIRECTOR":"Chad
Stahelski","MOVIE_IMG":"https://fwcdn.pl/fpo/15/20/831520/8061066.3.jpg","RELEASE_DATE":"2023-03-23T22:00:0
0.000Z"},{"MOVIE_ID":22,"MOVIE_TITLE":"Dungeons & Dragons: Honor Among Thieves","MOVIE_DIRECTOR":"John
Francis
Daley","MOVIE_IMG":"https://fwcdn.pl/fpo/14/95/751495/8056026.3.jpg","RELEASE_DATE":"2023-04-13T22:00:00.0
00Z"},{"MOVIE_ID":23,"MOVIE_TITLE":"Barbie","MOVIE_DIRECTOR":"Greta
Gerwig","MOVIE_IMG":"https://fwcdn.pl/fpo/48/00/754800/8049756.3.jpg","RELEASE_DATE":"2023-07-20T22:00:00.
000Z"},{"MOVIE_ID":24,"MOVIE_TITLE":"The Flash","MOVIE_DIRECTOR":"Andres
Muschietti","MOVIE_IMG":"https://fwcdn.pl/fpo/45/18/694518/8057595.3.jpg","RELEASE_DATE":"2023-06-13T22:00:
00.000Z"},{"MOVIE_ID":25,"MOVIE_TITLE":"Suzume","MOVIE_DIRECTOR":"Makoto
Shinkai","MOVIE_IMG":"https://fwcdn.pl/fpo/18/68/10011868/8066510.3.jpg","RELEASE_DATE":"2023-04-20T22:00:0
0.000Z"},{"MOVIE_ID":26,"MOVIE_TITLE":"Super Mario Bros. Movie","MOVIE_DIRECTOR":"Michael
Jelenic","MOVIE_IMG":"https://fwcdn.pl/fpo/05/81/850581/8057312.3.jpg","RELEASE_DATE":"2023-05-25T22:00:00.0
00Z"}]

```

7.3. Pobranie informacji o dostępnych dla filmu seansów

```

app.get("/movies/:id", async (req, res) => {
  const conn = await connect();
  const query = `select * from
BD_411227.SEANCES_PER_MOVIE_ID(${req.params.id})`;

  conn?.execute(query, [], { autoCommit: true }, (error, result) => {
    if (error) {
      return res.status(500).json({
        message: error.message,

```

```

        error,
    });
} else {
    return res.status(201).json(result.rows);
}
});
})

```

Na localhost:4000/movies/2:

```

[{"SEANCE_ID":1,"MOVIE_ID":2,"MOVIE_TITLE":"Little
Mermaid","ROOM_ID":1,"TIME_FROM":"2023-05-20T08:30:00.000Z","TIME_TO":"2023-05-20T10:30:00.000Z"},{"SEA
NCE_ID":2,"MOVIE_ID":2,"MOVIE_TITLE":"Little
Mermaid","ROOM_ID":1,"TIME_FROM":"2023-05-20T11:00:00.000Z","TIME_TO":"2023-05-20T13:30:00.000Z"},{"SEA
NCE_ID":3,"MOVIE_ID":2,"MOVIE_TITLE":"Little
Mermaid","ROOM_ID":1,"TIME_FROM":"2023-05-20T14:00:00.000Z","TIME_TO":"2023-05-20T16:30:00.000Z"}]

```

7.4. Pobranie informacji o krzesłach dostępnych dla wybranego seansu

```

app.get("/seances/:id", async (req, res) => {
    const conn = await connect();
    const query = `select * from
BD_411227.SEATS_PER_SEANCE_ID(${req.params.id})`;

    conn?.execute(query, [], { autoCommit: true }, (error, result) => {
        if (error) {
            return res.status(500).json({
                message: error.message,
                error,
            });
        } else {
            return res.status(201).json(result.rows);
        }
    });
});
})

```

Na localhost:4000/seances/1:

```

[{"SEANCE_ID":1,"SEAT_ID":1,"SEAT_TAKEN":1,"VIP":1}, {"SEANCE_ID":1,"SEAT_ID":2,"SEAT_TAKEN":1,"VIP":1}, {"SEANCE_I
D":1,"SEAT_ID":3,"SEAT_TAKEN":0,"VIP":1}, {"SEANCE_ID":1,"SEAT_ID":4,"SEAT_TAKEN":0,"VIP":1}, {"SEANCE_ID":1,"SEAT_I
D":5,"SEAT_TAKEN":1,"VIP":0}, {"SEANCE_ID":1,"SEAT_ID":6,"SEAT_TAKEN":0,"VIP":0}, {"SEANCE_ID":1,"SEAT_ID":7,"SEAT_
TAKEN":0,"VIP":0}, {"SEANCE_ID":1,"SEAT_ID":8,"SEAT_TAKEN":0,"VIP":0}, {"SEANCE_ID":1,"SEAT_ID":9,"SEAT_TAKEN":1,"
VIP":0}, {"SEANCE_ID":1,"SEAT_ID":10,"SEAT_TAKEN":0,"VIP":0}, {"SEANCE_ID":1,"SEAT_ID":11,"SEAT_TAKEN":0,"VIP":0}, {"
SEANCE_ID":1,"SEAT_ID":12,"SEAT_TAKEN":0,"VIP":0}, {"SEANCE_ID":1,"SEAT_ID":13,"SEAT_TAKEN":0,"VIP":0}, {"SEANCE_
ID":1,"SEAT_ID":14,"SEAT_TAKEN":0,"VIP":0}, {"SEANCE_ID":1,"SEAT_ID":15,"SEAT_TAKEN":0,"VIP":0}, {"SEANCE_ID":1,"SE
AT_ID":16,"SEAT_TAKEN":0,"VIP":0}, {"SEANCE_ID":1,"SEAT_ID":17,"SEAT_TAKEN":0,"VIP":0}, {"SEANCE_ID":1,"SEAT_ID":1

```

```
8,"SEAT_TAKEN":0,"VIP":0},{ "SEANCE_ID":1,"SEAT_ID":19,"SEAT_TAKEN":0,"VIP":0},{ "SEANCE_ID":1,"SEAT_ID":20,"SEAT_TAKEN":0,"VIP":0},{ "SEANCE_ID":1,"SEAT_ID":21,"SEAT_TAKEN":0,"VIP":0},{ "SEANCE_ID":1,"SEAT_ID":22,"SEAT_TAKEN":0,"VIP":0},{ "SEANCE_ID":1,"SEAT_ID":23,"SEAT_TAKEN":0,"VIP":0},{ "SEANCE_ID":1,"SEAT_ID":24,"SEAT_TAKEN":0,"VIP":0},{ "SEANCE_ID":1,"SEAT_ID":25,"SEAT_TAKEN":0,"VIP":0},{ "SEANCE_ID":1,"SEAT_ID":26,"SEAT_TAKEN":0,"VIP":0},{ "SEANCE_ID":1,"SEAT_ID":27,"SEAT_TAKEN":0,"VIP":0},{ "SEANCE_ID":1,"SEAT_ID":28,"SEAT_TAKEN":0,"VIP":0},{ "SEANCE_ID":1,"SEAT_ID":29,"SEAT_TAKEN":0,"VIP":0},{ "SEANCE_ID":1,"SEAT_ID":30,"SEAT_TAKEN":0,"VIP":0}]
```

7.5. Dokonanie rezerwacji

W poniższej funkcji tworzona jest tablica z siedzeniami wybranymi przez użytkownika (zamiana tablicy js-owej na oraclową), a następnie przesyłane jest żądanie do bazy o dodanie rezerwacji z otrzymanymi w body parametrami.

```
app.post("/reservation", async (req, res) => {
  const conn = await connect();

  let query = `declare
myarray num_array;
begin
myarray := num_array();`

  let i = 1;
  req.body.seats.forEach(seat => {
    query += `
myarray.extend(1);
myarray(${i}) := ${seat};`
    i++;
  });

  query += `add_reservation(myarray, ${req.body.seanceId},
${req.body.userId}, ${req.body.paid});
end;`;

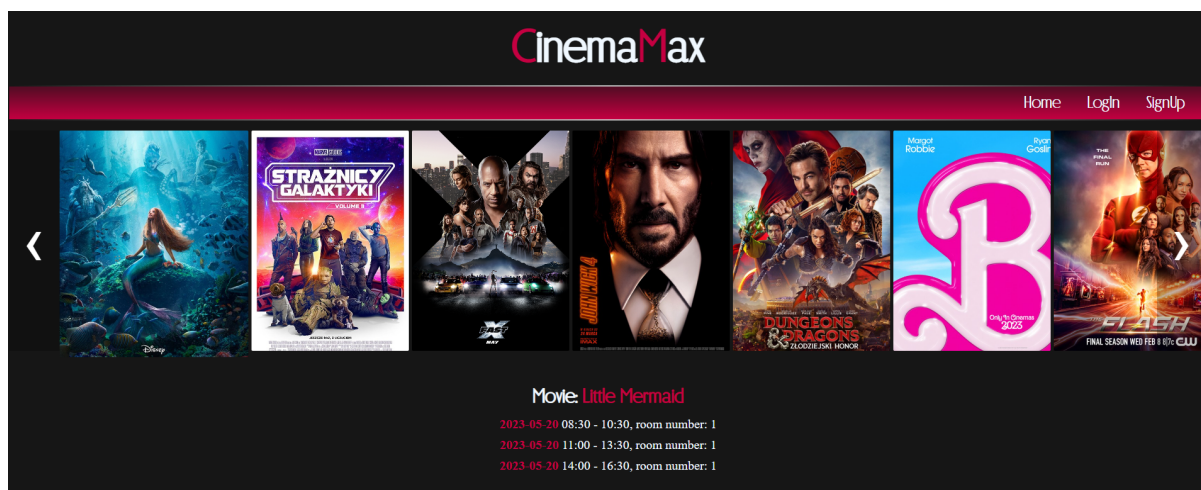
  conn?.execute(query, [], { autoCommit: true }, (error, result) => {
    if (error) {
      return res.status(500).json({
        message: error.message,
        error,
      });
    } else {
      return res.status(201);
    }
  });
});
```

8. Frontend (React.js)

8.1. Wygląd głównej strony kina

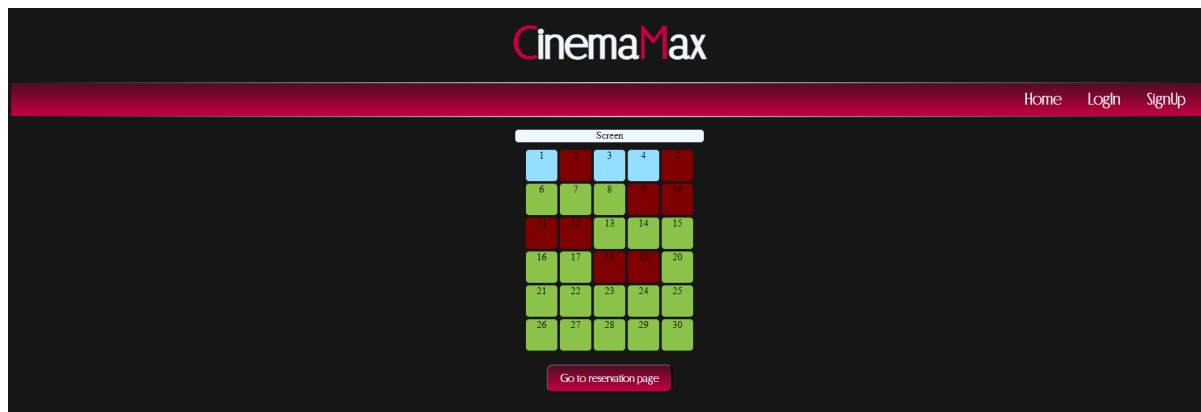


8.2. Wybór dostępnego dla klikniętego filmu seansu

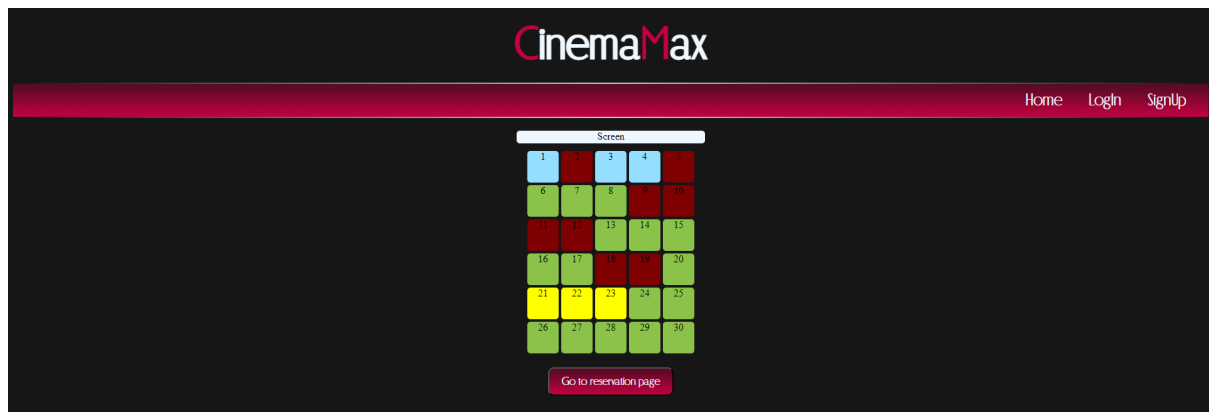


8.3. Widok sali kinowej z dostępnymi miejscami

Widoczne są zarówno siedzenia VIP, te już zarezerwowane jak i jeszcze dostępne



8.4. Wybór miejsc



8.5. Rezerwacja wybranych wcześniej miejsc

