Aplikacja do zakupu i rezerwacji biletów pociągowych.

Zamysł Aplikacji: Możliwość udostępnienia rezerwacji biletu użytkownikowi oraz zmiany zarezerwowanych biletów (zmiana ilości/usunięcie).
Aplikacja została zrealizowana w frontendzie react, backend express oraz z bazą danych oracle.
Członkowie zespołu: Adam Ćwikła.

Baza danych

Schemat naszej bazy wygląda następująco:

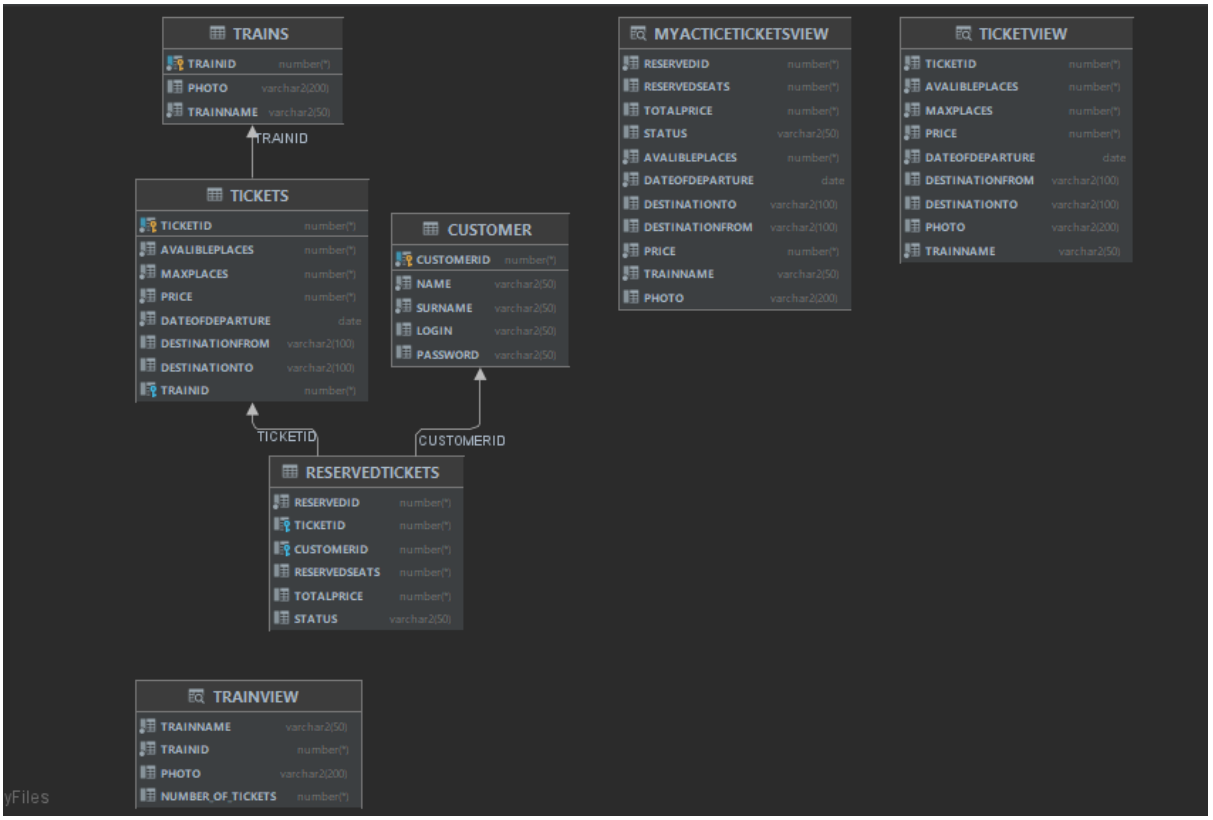

Tabela Trains - Przechowuje podstawowe dane dla pociągów, zdjęcie dla estetyczniejszego wyświetlania. Każdy bilet ma jeden pociąg ale wiele biletów może być dla jednego pociągu.
Tabela Tickets - odpowiada za wszystkie bilety w obiegu. Tutaj znajduję się pula dla biletów na dany kurs. Tutaj mamy reprezentanta każdego biletu, z ilością biletów możliwą do rezerwacji oraz z ilością biletów na cały kurs.
Tabela Reserved Tickets - Tutaj przechowywane są rezerwację na bilet. Ilość zarezerwowanych miejsc dla pojedynczego biletu dla danego klienta.
Tabela Client - podstawowe dane dla klienta.

Wszystkie widoki są używane jako endpointy serwera z których pobieramy dane.

Widoki:

1. **TrainView** -> używany jako możliwość łatwego pobrania danych o danym pociągu.
2. **MyActiveTicketsView** -> wyświetla wszystkie zarezerwowane bilety przez klienta.
3. **TicketView** -> używany do wyświetlania możliwych biletów do rezerwacji.

Przebieg zakupu biletu:

1. Klient ogląda bilety wyświetlone za pomocą frontendu react w zakładce Bilety, który pobiera dane z backendu express widoku MyActiveTicketsView.
2. Wybiera konkretny bilet klikając na obrazek pociągu.
3. Za pomocą suwaka wybiera ilość biletów. Posiadamy tutaj zabezpieczenie ze strony frontendu.
4. Po zatwierdzeniu zostaje wysłana wiadomosc do serwera o zakupie biletu.

Przebieg rezygnacji z biletu/modyfikacji ilości zakupionych miejsc:

1. Po wciśnięciu przycisku usuń bilet/zrezygnuj z biletu do serwera zostanie wysłana wiadomość. Jeśli usuwamy bilet z zakładki Twoje bilety zostanie uruchomiony odpowiedni trigger, który doda bilety z których zrezygnowaliśmy z powrotem do możliwych do zakupu. Jeśli modyfikujemy bilet trigger odpowiednio doda bądź usunie bilety z powrotem do możliwych do zakupu

Dodanie biletu:

1. Jesteśmy w stanie dodać nowy rodzaj biletu do puli za pomocą formularza w zakładce Dodaj bilet.
2. Po zatwierdzeniu zostaje wysłana wiadomosc do serwera o wprowadzeniu w obieg nowego biletu.

Serwer

Serwer został zrealizowany za pomocą expressa.
Funkcja pomocnicza do tworzenia endpointów GET:

```
async function getFrom(req,res,query){
  try{
    connection = await oracledb.getConnection({
      user: "demonode",
      password: "OracleAdam6",
      connectionString: "localhost/xepdb1" });

    result2 = await connection.execute(query);
    result=JSON.stringify(result2.rows);
  }catch (err) {
    return res.send(err.message);
  } finally {
    if (connection) {
        try {
          await connection.close();
          console.log('close connection success');
        } catch (err) {
          console.error(err.message);
        }
    }
    return res.send(result);
  }
}
```

Endpointy zostały zrealizowane jako odpowiednie selecty z widoków GET:

```
app.get('/tickets', function (req, res) {
  //getTickets(req, res);
  getFrom(req,res,`SELECT * FROM TicketView`);
})
app.get('/trains', function (req, res) {
  //getTrains(req, res);
  getFrom(req,res,`SELECT * FROM trainView`);
})
app.get('/tickets/:id',function(req,res){
  getFrom(req,res,`select * from TICKETVIEW
  where TicketID=${req.params.id}`);
});
app.get('/reservedTickets',function(req,res){
  getFrom(req,res,`select * from myActiceTicketsView`)
})
```

- /tickets -> odpowiada za wyświetlanie danych dotyczących biletów
- /trains -> odpowiada za wyświetlanie informacji o pociągach
- /tickets/:id -> odpowiada za wyświetlanie informacji o bilecie z konkretnym id
- /reservedTickets -> odpowiada za wyświetlanie informacji o zarezerwowanych biletach dla klienta

Aby dodać nowy bilet do naszej bazy danych użyjemy endpointów POST

```
app.post('/addTicket',(req,res)=>{
  console.log(req.body);
  addTicket(req,res)
});
app.post('/reserveTicket',(req,res)=>{
  console.log(req.body);
  addReserveTicket(req,res);
})
```

gdzie odpowiednie funkcje wykonują odpowiedniego inserta do naszej bazy
Aby zmodyfikować zarezerwowany bilet (ilość) możemy użyć PUT

```
async function chengeReservedTicket(req,res){
  console.log(req.body)
  const sql=`update reservedTickets set ReservedSeats=${req.body.RESERVEDSEATS},TotalPrice=${req.body.TOTALPRICE} where reservedID=${req.body.RESERVEDID}
  console.log(sql)

  try {
    connection = await oracledb.getConnection({
      user: "demonode",
      password: "OracleAdam6",
      connectionString: "localhost/xepdb1" });


    let result = await connection.execute(sql)
    console.log(result.rowsAffected, "Rows Inserted");
    connection.commit();
  }catch(err){
    console.log(err)
  }finally {
    if (connection) {
        try {
          await connection.close();
          console.log('close connection success');
        } catch (err) {
          console.error(err.message);
        }
    }
  }
}
```

Aby usunąć bilet z rezerwacji bądź z puli dostępnych biletów używamy DELETE.

```
/*
DELETES
*/
app.delete('/tickets/:id',function(req,res){
  deleteFrom(req,res,`DELETE FROM TICKETS WHERE TICKETID=`+String(req.params.id))
})
app.delete('/reservedTicketsCancel/:id',function(req,res){
  console.log(req.params.id)
  deleteFrom(req,res,`DELETE FROM RESERVEDTICKETS WHERE RESERVEDID=`+String(req.params.id))
})
```