2023 gpt

Oppgave 1

```
class Fugl {
            constructor(navn, vekt){
                this._navn = navn;
                this._vekt = vekt;
            }

            get navn(){
                return this._navn;
            }

            set navn(navn) {
                this._navn = navn;
            }

            get vekt() {
                return this._vekt;
            }

            set vekt(vekt) {
                this._vekt = vekt;
            }
        }

        class Papegøye extends Fugl{
            constructor(navn, vekt, ord = []){
                super(navn, vekt);

                this._ord = ord;
            }

            lærOrd(nyttOrd){
                this._ord.push(nyttOrd);
            }

            siOrd(){
                const tilfeldigOrd = Math.floor(Math.random()* this._ord.length);
                return this._ord[tilfeldigOrd];
            }
        }

        const papegøye = new Papegøye("Angry bird", 15);
        papegøye.lærOrd("øving");
        papegøye.lærOrd("hei");
```

```
        papegøye.lærOrd("ord");
        console.log(papegøye.siOrd());
```

Oppgave 3

Den funker ikke helt, men funker sånn ish. Så det er liksom greit nok. Sletent.

```
CREATE TABLE TeaterShows (
      id INT NOT NULL,
      title TEXT,
      date_time DATETIME,
      PRIMARY KEY(id)
    )

CREATE TABLE Seats (
    id INT NOT NULL AUTO_INCREMENT,
    row INT,
    columnArea INT,
    show_id INT NOT NULL,
    available BOOLEAN,
    PRIMARY KEY(id)
  )
```

Index

```
import { pool } from './mysql-pool';
import type { RowDataPacket, ResultSetHeader } from 'mysql2';

export class TeaterShows {
  id!: number;
  title: string = "";
  date_time!: Date;
}

export class Seats {
  id!: number;
  row!: number;
  antSeter!: number;
  show_id!: number;
  available: boolean = true;
}

class TeaterService {
  getShows(success: (shows: TeaterShows[]) => void){
    pool.query(
      'SELECT * FROM  TeaterShows', (error: Error, results: any) => {
        if(error) return console.error(error);
```

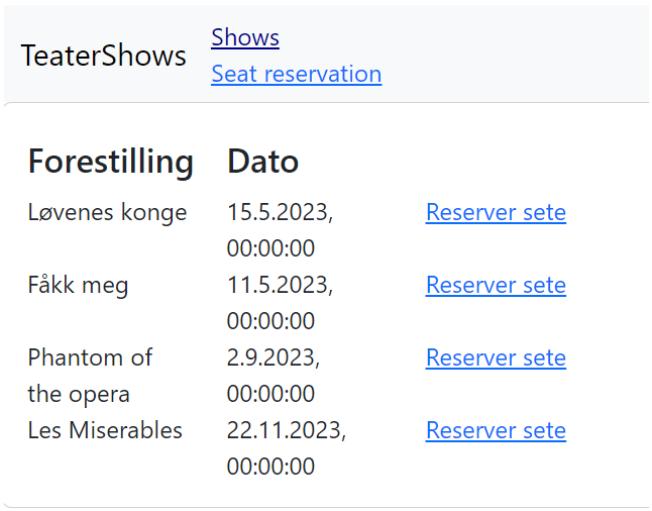
```
        success(results);
      }
    )
  }

  getShow(id: number, success: (show: TeaterShows)=> void){
    pool.query(
      'SELECT * FROM TeaterShows WHERE id=?', [id], (error: any, results: any) => {
        if(error) return console.error(error);

        success(results[0]);
      }
    )
  }

  getSeats(id: number, success: (seats: Seats[])=> void){
    pool.query(
      'SELECT * FROM  Seats s INNER JOIN TeaterShows ts ON s.show_id = ts.id',
          [id], (error: any, results: any) => {
        if(error) return console.error(error);

        success(results);
      }
    )
  }



  saveReservation(id:number, success: ()=> void){
    pool.query(
      'UPDATE Seats SET available=0 WHERE id=?', [id], (error: any, results: any) => {
        if(error) return console.error(error);

        success();
      }
    )
  }
}
```

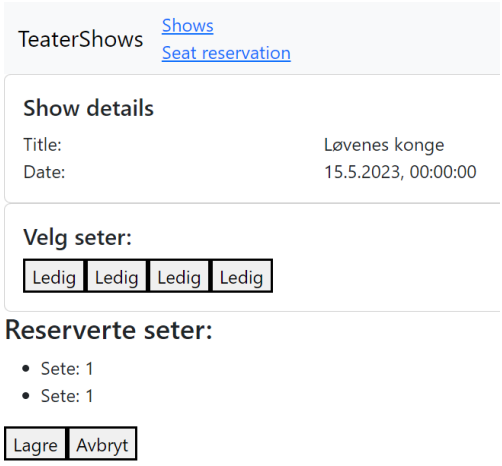
Services

```
import { pool } from './mysql-pool';
import type { RowDataPacket, ResultSetHeader } from 'mysql2';

export class TeaterShows {
  id!: number;
  title: string = "";
  date_time!: Date;
}

export class Seats {
  id!: number;
```

```
  row!: number;
  antSeter!: number;
  show_id!: number;
  available: boolean = true;
}

class TeaterService {
  getShows(success: (shows: TeaterShows[]) => void){
    pool.query(
      'SELECT * FROM  TeaterShows', (error: Error, results: any) => {
        if(error) return console.error(error);

        success(results);
      }
    )
  }

  getShow(id: number, success: (show: TeaterShows)=> void){
    pool.query(
      'SELECT * FROM TeaterShows WHERE id=?', [id], (error: any, results: any) => {
        if(error) return console.error(error);

        success(results[0]);
      }
    )
  }

  getSeats(id: number, success: (seats: Seats[])=> void){
    pool.query(
      'SELECT * FROM  Seats s INNER JOIN TeaterShows ts ON s.show_id = ts.id',
[id], (error: any, results: any) => {
        if(error) return console.error(error);

        success(results);
      }
    )
  }



  saveReservation(id:number, success: ()=> void){
    pool.query(
      'UPDATE Seats SET available=0 WHERE id=?', [id], (error: any, results: any) => {
        if(error) return console.error(error);

        success();
      }
    )
  }
}

export let teaterService = new TeaterService();
```

OUTPUT


TeaterShows
Shows
Seat reservation
Forestilling
Dato
Løvenes konge
15.5.2023, 00:00:00
Reserver sete
Fåkk meg
11.5.2023, 00:00:00
Reserver sete
Phantom of the opera
2.9.2023, 00:00:00
Reserver sete
Les Miserables
22.11.2023, 00:00:00
Reserver sete


TeaterShows
Shows
Seat reservation
Show details
Title:
Løvenes konge
Date:
15.5.2023, 00:00:00
Velg seter:
Ledig
Ledig
Ledig
Ledig
Reserverte seter:
Sete: 1
Sete: 1
Lagre
Avbryt