Web technológiák 2

Féléves feladat
Jegyzőkönyv

Készítette: **Torma Antal**

Neptunkód: **YK11Q1**

Tartalomjegyzék

1. Backend

A backend-et egy mysql-t futtató szerveren helyeztem el, ahol létrehoztam a webtech2 nevű adatbázist.

1.1 Adatmodell

Egy entitást használ a program, amely a Worker.ts fájlban van leírva. Tulajdonságai a következők:

- id: number (automatikusan generálódó elsődleges kulcs)
- name: string (a munkás neve)
- competence: string (képzettségeket tárolja)
- wage:number (fizetési igény)
- status:string (aktuális státusza a munkásnak)

Az adatbázisban a táblát a TypeORM dekorátoraival hozom létre a backend futtatásakor.

```
@Entity()
export class Worker {

    @PrimaryGeneratedColumn()
    id: number;

    @Column()
    name: string;

    @Column()
    competence: string;

    @Column()
    wage: number;

    @Column()
    status: string;

}
```

1.2 Controller

A base.controllerben található az SQL lekérdezések, műveletek.

- összes lekérdezés
- egy elem lekérdezése
- létrehozás
- törlés
- hibakezelés

Az adatbázis egy mySQL szerveren fut, a táblát a program hozza létre. Az inser.txt fájlban elérhető az sql parancsok, amikkel a kezdeti adatokat felvihető.

2. Frontend

A frontend két komponensből, egy adatmodellből és egy szerviz rétegből áll össze.

2.1 Modell

Egy modell osztályt létrehozva megadtam a kezelendő adatok struktúráját.

```
export interface Worker{
    id: number;
    name: string;
    competence: string;
    wage: number;
    status: string;
}
```

2.2 Service

Itt kapcsolódik össze a backend és a frontend, a http kérések jóvoltából.

```
export class WorkerService {

  constructor(private http: HttpClient) { }

  async getAll(): Promise<Worker[]> {
    return await lastValueFrom(this.http.get<Worker[]>('api/workers'));
  }

  async getWorker(id: number) {
    return await lastValueFrom(this.http.get<Worker>('api/workers/'+ id));
  }

  async create(worker: Worker) {
    return await lastValueFrom(this.http.post<Worker>('/api/workers', worker));
  }

  async delete(id: number){
    return await lastValueFrom(this.http.delete<Worker>('/api/workers/' + id))
  }
}
```

2.3 Worker-list

Ez a komponens jelenik meg egyben a főoldal is. Az oldalak között az app-component.html biztosítja az átjárást egy menü segítségével.

Ezen a felületen látható az adatbázisban szereplő objektumok, amelyeket egy táblázatban listázza a program.

Láthatóak a dolgozók tulajdonságai, továbbá lehetőség van arra, hogy töröljük a meglévő objektumokat a Delete gomb segítségével.

```
<tbody>
    <tr *ngFor="let w of workers">
        <td>{{w.id}}</td>
        <td>{{w.name}}</td>
        <td>{{w.competence}}</td>
        <td>{{w.wage}}</td>
        <td>{{w.status}}</td>
        <td>
            <button type="button" class="btn btn-danger" (click)="deleteWorker(w.id)">Delete</button>
        </td>
    </tr>
</tbody>
```

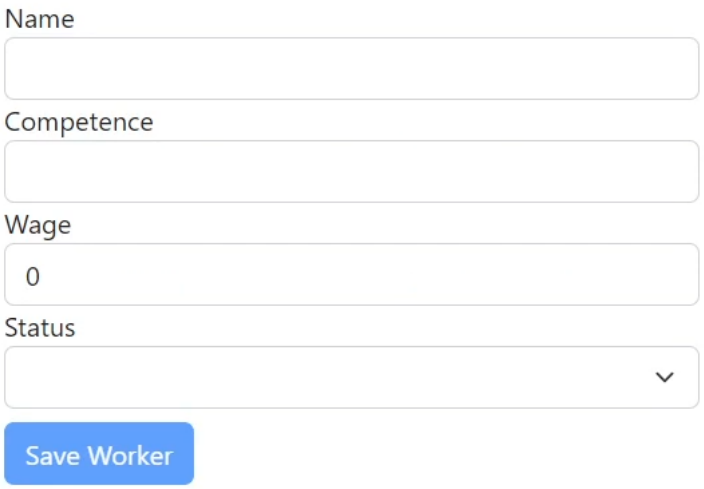
A táblázatban egy for ciklus segít a munkások listázásában, illetve a törlés gomb kattintáskor tovább adja az adott ember ID-át a meghívott függvénynek.

A worker-list.component.ts fájlban kezeltem le az komponens által használt eseményeket. Az oldal betöltésekor a workers tömbbe elmentem az aktuálisan tárolt adatokat az adatbázisban. A törlés művelet után frissítem a workers tömb adatait.

```
export class WorkerListComponent {
  workers!: Worker[];

  constructor(private router: Router,
    private workerService: WorkerService){ }

  async ngOnInit() {
    this.workers = await this.workerService.getAll();
  }

  async deleteWorker(id: number){
    await this.workerService.delete(id);
    this.workers = await this.workerService.getAll();
  }
}
```

2.4 Worker-form

A komponens arra szolgál, hogy új dolgozókat adjunk hozzá az adatbázisunkhoz. Ezt egy form segítségével oldottam meg.



Lekezelésre került a duplikáció elkerülése nevek alapján. Kötelező kitölteni a név, a fizetés és a státusz mezőket. A fizetésnek legalább 1-nek kell lennie. Amíg ezek invalidak addig a mentés gomb inaktív. A kompetencia mezőn nincs megkötés.


```
async create() {
  this.errorMessage = 'ok';
  const worker = this.workerForm.value;
  this.workers = await this.workerService.getAll();
  for (let index = 0; index < this.workers.length; index++) {
    const element = this.workers[index];
    if(element.name === worker.name) {
      this.errorMessage = 'Worker already in database!'
      index = this.workers.length;
    }
  }
  if(this.errorMessage === 'ok'){
    this.workerService.create(worker)
    this.router.navigateByUrl("/worker-list");
  }
}
```