

# Beadandó vizsgafeladat:

## Személygépkocsi bérlő oldal

Bodnár László D1H8VP

A weboldalam 2 részből áll. Egy fronted és egy backend részből. Ezeket külön kell indítani a hozzájuk tartozó utasítások segítségével.

Frontend: ng serve

Backend npm start

Teszteléshez email cím: [test1@gmail.com](mailto:test1@gmail.com)

Jelszó: 12345678

A backendemen egy saját nodemon szerver fut.

Az adatbázisom pedig MongoDB.

Modelljeim így néznek ki:

```
export const GepSchema = new Schema<Gep>({ no usages
{
  name: {type: String, required: true},
  marka: {type: String, required: true},
  tipus: {type: String, required: true},
  teljesitmeny: {type: Number, required: true},
  suly: {type: Number, required: true},
  berletidij: {type: Number, required: true},
  letet: {type: Number, required: true},
}, {
  toJSON: {
    virtuals: true
  },
  toObject: {
    virtuals: true
  },
  timestamps: true
})
```

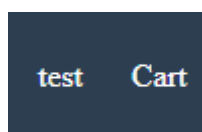
```
export const UserSchema = new Schema<User>({ no usages
  name: {type: String, required: true},
  ambassador: {type: String, required: true},
  email: {type: String, required: true},
  password: {type: String, required: true},
  taxnum: {type: Number, required: true},
  address: {type: String, required: true},
  isAdmin: {type: Boolean, default: false},
  balance: {type: Number, required: true},
}, {
  timestamps: true,
  toJSON: { virtuals: true },
  toObject: { virtuals: true },
}
);
```

```
const orderSchema = new Schema<order>({
  name: {type:String, required:true},
  address: {type:String, required:true},
  addressLatLng: {type:LatLngSchema, required:true},
  paymentId: {type:String},
  paymentMethod: {type:String},
  items: {type:[OrderSchema], required:true},
  status: {type:String, default: OrderStatus.NEW},
  user: {type:Schema.Types.ObjectId, required:true }
}, {
  timestamps:true,
  toJSON:{
    virtuals: true,
  },
  toObject:{
    virtuals: true,
  }
});
```

A weboldalam egy homepage-el kezdődik, ahol a kölcsönözhető gépek listáját lehet látni.

Gep Kölcsönzes			
Gep1	Gep2	gep3	audi2
Márka: Audi	Márka: BMW	Márka: peugeot	Márka: audi
Típus: Szemelyauto	Típus: Szemelyauto	Típus: személygépkocsi	Típus: személyauto
Teljesítmény: 200 W	Teljesítmény: 220 W	Teljesítmény: 200 W	Teljesítmény: 200 W
Súly: 1400 kg	Súly: 1500 kg	Súly: 1200 kg	Súly: 1500 kg
Bérelti díj/Nap: €20,000	Bérelti díj/Nap: €50,000	Bérelti díj/Nap: €20	Bérelti díj/Nap: €60
Biztonsági letét: €90,000	Biztonsági letét: €140,000	Biztonsági letét: €120	Biztonsági letét: €150

Itt helyett kapott meg 2 gomb is, ahol a saját profilt lehet szemügyre venni, valamint a kosarat:



Az egyik gép információjára kattintva egy megjelenik a csak az az egy gép fókuszban, valamint egy kölcsönzés gomb.

### Gep1

Marka:	Tipus:
Audi	Szemelyauto
Teljesitmeny:	Suly:
200 W	1400 kg
Bérleti Díj:	Letet:
€20,000	€90,000

Kolcsonzes

A kölcsönzés gomb megnyomása után, a Cart Pagere kerülünk ahol kiválaszthatjuk, hogy hány napra szeretnénk az adott járműbet bérelni.

Cart Page

Gep1

1 ~ 110000 EUR

Total Items:

Total Price:

110000 EUR

Proceed to Checkout →

Továbblépve a checkout page éri hogy adjuk meg nevunket és számlázási cimunket majd egy térképen be lehet jelölni, hogy hova szeretnénk a gépjármű kiszállítását

## Order Form

Name

test

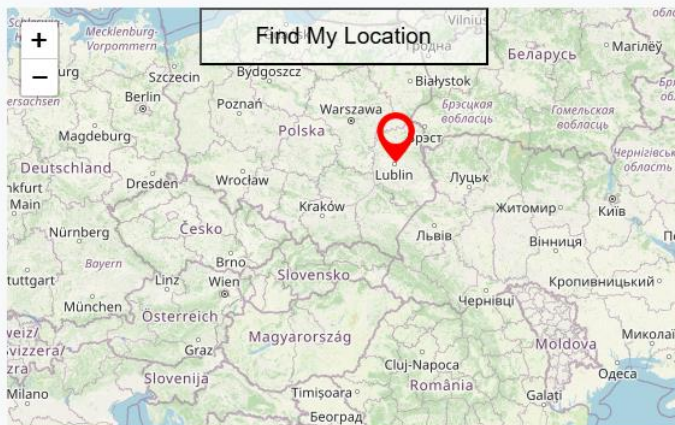
Address

test utca 3

### Order Items:

Gep1	€20,000	€90,000	1	€110,000
Total:				€110,000

## Choose Your Address



Ezután a payment pagehez érkezve, kiválaszthatunk két féle fizetési opciót, az egyik egy paypal felületét használó fizetési mód, a másik pedig a felhasználó

feltöltött egyelegét használja

Order Summary

Name:

test

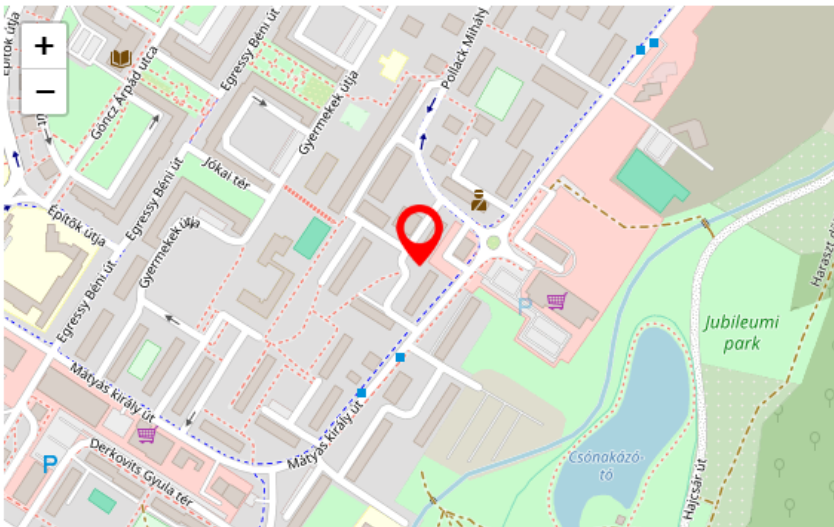
Address:

test utca 3

Order Items:

Gep1

Shipping Address



PayPal



Bank- vagy hitelkártya

Üzemeltető: PayPal

Fizetés egyenleggel

JD

PayPal


110 000,00 EUR

Szállítási cím: John Doe

Módosítás

Budapest, Erzsébet Ter 9-10, 1051

Fizetés a következővel:

 Visa

19 214 536 JPY

Jóváírás \*\*\*\*3394

☐ Állítsa be ezt elsődleges fizetési módként

A PayPal átváltási árfolyamát használva fizet: 1 JPY = 0,00572 EUR

Vagy válassza a kártyakibocsátó árfolyamát

+ Adjon hozzá bank- vagy hitelkártyát

Vásárlás befejezése

Paypal developer sandbox segítségével való fizetés

Ha a fizetés sikeresen megtörtént akkor az order track pagere fog vinni az oldal, ahol van lehetőségünk lezárni a rendelést.

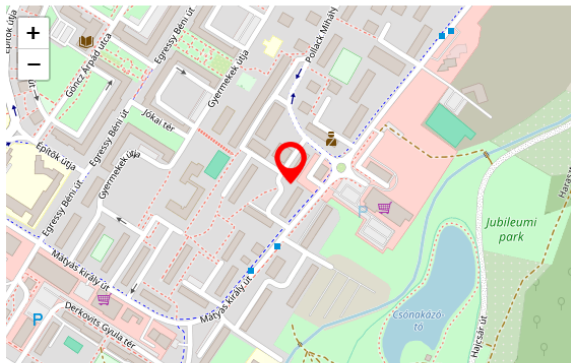
### Order #6863c0e88bce2c5799e535f1

Date	Name	Address	Status
	test	test utca 3	PAYED

#### Order Items:

Gep1	€20,000	€90,000	1	€110,000
Total:				

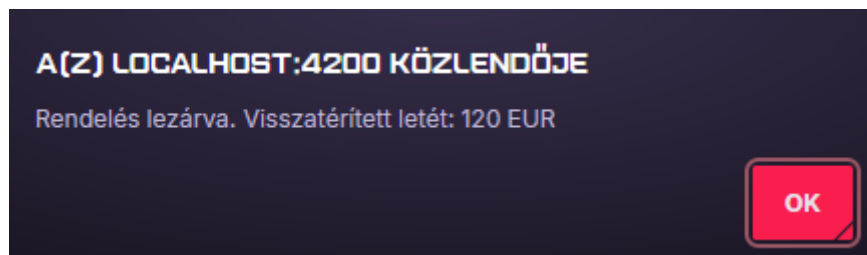
### Shipping Address



Rendelés lezárása

Egyenleggel fizetés hasonló végeredménnyel zárul, annyi különbséggel, hogy jelzi ha nincsen elegendő pénz a fiókon.

Rendelés lezárása után, a kaució ára visszajár:



Néhány fontosabb információ a kódról.

Authguardot használok, hogy token alapú felhasználó hitelesítést érjek el:

```

@Inject() no usages
export class AuthInterceptor implements HttpInterceptor {
  constructor(private userService: UserService) { } no usages

  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> { no usages

    const token :string = this.userService.currentUser?.token;
    console.log('Interceptor running. Token:', token);
    if (token) {
      const cloned :HttpRequest<any> = req.clone({
        setHeaders: {
          Authorization: `Bearer ${token}`
        }
      });
      return next.handle(cloned);
    }
    return next.handle(req);
  }
}

```

```

import { verify } from "jsonwebtoken";
import { HTTP_UNAUTHORIZED } from "../constants/http_status";

export default (req: any, res: any, next: any) :any => { no usages laszlobodnar14

  const token :any = req.headers['authorization']?.split(' ')[1];
  if (!token) {
    console.error('No token found in request headers');
    return res.status(HTTP_UNAUTHORIZED).send('No Token');
  }

  try {
    console.log('Token found, verifying...');
    const decodedUser :string | JwtPayload = verify(token, process.env.JWT_SECRET!);
    console.log('Decoded user:', decodedUser);
    req.user = decodedUser;
    next();
  } catch (error) {
    console.error('Token verification failed:', error);
    return res.status(HTTP_UNAUTHORIZED).send('Invalid Token');
  }
}

```

Az adatbázisba írást/olvasást routerek segítségével oldom meg, amikben get és post metódusok vannak



```

router.get('/:gepId', asyncHandler(async (req : Request<ParamsDictionary, any, any, Parse... , res : Response<any, Record<string, any>, number... ) : Promise<void> => {
  const gepts : (Document<unknown, {}, Gep, {}> & Gep & {...) = await GepModel.findById(req.params.gepId);
  res.send(gepts);
}));

router.post('/gepregister', asyncHandler(async (req : Request<ParamsDictionary, any, any, Parse... , res : Response<any, Record<string, any>, number... ) : Promise<void> => {
  const { name, marka, tipus, teljesitmeny, suly, berletidij, letet } = req.body;

  const newGep = new GepModel({
    name,
    marka,
    tipus,
    teljesitmeny,
    suly,
    berletidij,
    letet
  });

```

Saját input validatort használlok:

```

export class InputValidationComponent implements OnInit, OnChanges {
  @Input()
  control!: AbstractControl;
  @Input()
  showErrorsWhen: boolean = true;
  errorMessages: string[] = [];
  constructor() {} // no usages

  ngOnChanges(changes: SimpleChanges): void { // no usages
    this.checkValidation();
  }

  ngOnInit(): void { // no usages
    this.control.statusChanges.subscribe(() :void => {
      this.checkValidation();
    });
    this.control.valueChanges.subscribe(() :void => {
      this.checkValidation();
    })
  }

  checkValidation() :void { // no usages
    const errors : ValidationErrors | null = this.control.errors;
    if(!errors){
      this.errorMessages = [];;
      return;
    }

    const errorKeys : string[] = Object.keys(errors);
    this.errorMessages = errorKeys.map(key : string => VALIDATORS_MESSAGES[key]);
  }
}

```

```

export const PasswordMatchValidator : (passwordControlName: string, confirmPasswordControlName: string) : (form: AbstractControl<any, any>) => void => { // no us
  const validator : (form: AbstractControl<any, any>) => void = (form: AbstractControl) :void => { // Show usages
    const passwordControl : AbstractControl<any, any> | null = form.get(passwordControlName);
    const confirmPasswordControl : AbstractControl<any, any> | null = form.get(confirmPasswordControlName);
    if(!passwordControl || !confirmPasswordControl) return;
    if(passwordControl.value !== confirmPasswordControl.value){
      confirmPasswordControl.setErrors({notMatch: true});
    } else {
      const errors : ValidationErrors | null = confirmPasswordControl.errors;
      if(!errors) return;

      delete errors['notMatch'];
      confirmPasswordControl.setErrors(errors);
    }
  }
  return validator;
}

```

A térképkezelést a leaflet könyvtárral oldom meg:

```

initializeMap() :void { no usages
    if(this.map) return;

    this.map = map(this.mapRef.nativeElement,{
        attributionControl:false
    }).setView(this.DEFAULT_LATLNG,1);

    tileLayer('https://{s}.tile.osm.org/{z}/{x}/{y}.png').addTo(this.map);
    this.map.on('click', (event:LeafletMouseEvent) :void => {
        this.setMarker(event.latlng);
    })
}

findMyLocation() :void { no usages
    this.locationService.getCurrentLocation().subscribe({
        next: (latlng :LatLngLiteral ) :void => {
            this.map.setView(latlng,this.MARKER_ZOOM_LEVEL);
            this.setMarker(latlng);
        }
    })
}

setMarker(latlng:LatLngExpression) :void { no usages
    this.addressLatLng= latlng as LatLng;
    if(this.currentMarker){
        this.currentMarker.setLatLng(latlng);
        return;
    }
    this.currentMarker = marker(latlng,{
        draggable: true,
        icon: this.MARKER_ICON,
    }).addTo(this.map);

    this.currentMarker.on('dragend', () :void =>{
        this.addressLatLng = this.currentMarker.getLatLng();
    })
}

```