WebFuel

Alkalmazás dokumentáció

Fejlesztői és felhasználói kézikönyv

Készítették:

Cserkó József

Szabó János

Szalánczi Róbert

Neumann János Egyetem - GAMF Kar

Informatikai Tanszék

Informatikai projektfeladat - 2018. május

Tartalom

[Fejlesztői dokumentáció 3](#_Toc513796548)

[Web alapú üzemanyag nyilvántartó rendszer 3](#_Toc513796549)

[A konkrét feladat 3](#_Toc513796550)

[Felhasznált technológiák 3](#_Toc513796551)

[Modulok 3](#_Toc513796552)

[Futtatási környezet, telepítési információk 4](#_Toc513796553)

[Az alkalmazás elérhető: 4](#_Toc513796554)

[Belépés adatok 4](#_Toc513796555)

[Az alkalmazás forráskódja elérhető 4](#_Toc513796556)

[Az alkalmazás gyökér – komponense 4](#_Toc513796557)

[Adatbázis entitások modelljei 5](#_Toc513796558)

[A be és kijelentkeztetésért felelős komponens osztály 6](#_Toc513796559)

[A navigációért felelős komponens. 7](#_Toc513796560)

[A Driver (sofőr) entitásokat kezelő komponens. 7](#_Toc513796561)

[A tankolásokat kezelő komponens. 8](#_Toc513796562)

[A Vehicle entitás kezelése. 10](#_Toc513796563)

[A statisztikák megjelenítéséért felelős komponens. 11](#_Toc513796564)

# Fejlesztői dokumentáció

## Web alapú üzemanyag nyilvántartó rendszer

A feladat web alapú alkalmazás készítése, mely lehetővé teszi a felhasznált üzemanyag mennyiségének tárolását és későbbi lekérdezését.

Megvalósítandó feladatok:

* Ki és bejelentkezés, felhasználók kezelése.
* Járművek felvitele, kezelése, módosítása.
* Tankolások felvitele és listázása egyszerű és átlátható felületen.
* Adott időszakban történt üzemanyag felhasználás lekérése és összesítése.

## A konkrét feladat

Egyre inkább terjednek a Web alapú adattároló és elemző rendszerek, mivel ezeket könnyen el lehet érni minden eszközről amely böngészővel rendelkezik.

A konkrét feladat a felhasználók, járművek és tankolások megfelelő felvitele és tárolása adatbázisban, illetve webes felületen történő megjelenítése és kezelése.

## Felhasznált technológiák

A következő technológiákat használjuk fel a kivitelezés során:

* **HTML5**
* **CSS3**
* **Bootstrap** CSS keretrendszer
* **Angular5** JAVASCRIPT keretrendszer
* **Firebase** felhő alapú adatbázis kezelő rendszer

## Modulok

A projekt keretében történő megvalósítás egy lehetséges felbontási lehetősége az alábbi:

* Adatbázis tervezés, minta adatok feltöltése
* Felhasználóbarát front end (GUI) tervezése, kivitelezése
  + Program felület, grafikai elemek
  + Menürendszer
  + Statisztikák
  + Beállítások

## Futtatási környezet, telepítési információk

Nincs szükség telepítésre, az alkalmazás elérhető szinte bármely szabványos böngészőből.

## Az alkalmazás elérhető:

<http://cherryapps.hu/webfuel>

## Belépés adatok

Felhasználó név: info@webfuel.com

Jelszó: webfuel

## Az alkalmazás forráskódja elérhető

<https://github.com/cherryApp/web-fuel>

## Az alkalmazás gyökér – komponense

app/app.component.ts

user: any = false; // Aktuális felhasználó.

authSubscribe: Subscription; // Feliratkozás az authentikációs állapot módosulásaira.

onlineSubscribe: Subscription; // Feliratkozás az online állapot módosulásaira.

onlineStatus: boolean = true; // Az online státusz tárolása.

constructor(

private authService: AuthService, // authentikációs szolgáltatás.

private router: Router, // az útválasztó szolgáltatás.

private config: ConfigService // a konfigurációs szolgáltatás.

) {

}

ngOnInit() // Inicializálás során feliratkozik az bejelentkezett státusz módosulásaira. Figyeli az online státusz módosulásait.

ngOnDestroy() // Megszűnés előtt törli a feliratkozásokat.

## Adatbázis entitások modelljei

*A Driver entitás modellje.*

app/model/driver.ts

address: string

age: number

birthDate: Date

email: string

name: string

phone: string

*A Fueling entitás modellje.*

app/model/fueling.ts

amount: number

driverId: string

time: Date

vehicleId: string

vehicleLp: string

*A Vehicle entitás modellje.*

app/model/vehicle.ts

consumption: string

fuelType: string

lp: string

manufacturer: string

type: string

year: string

## A be és kijelentkeztetésért felelős komponens osztály

*Az osztály konstruktora:*

constructor(

private authService: AuthService, // az authentikációért felelős szolgálatás.

private router: Router //az oldalak dinamikus betöltéséért felelős szolgáltatás.

) {

}

*A bejelentkezéshez szükséges adatokat tároló objektum:*

export class LoginComponent implements OnInit {

auth: any;

loginData: {email: string, password: string} = {

email: "",

password: ""

};

*Bejelentkezés email cím és jelszó használatával:*

signInWithEmail() {

this.authService.signInRegular(this.loginData.email, this.loginData.password)

.then( (res) => this.router.navigate(['/']) )

.catch( (err) => console.log('error: ' + err) );

}

## A navigációért felelős komponens.

myConfig: any; // A konfigurációs beállításokra mutató referencia.

authSubscribe: Subscription; // Az authentikációs állapotot figyelő feliratkozás.

user: any; // A bejelentkezett felhasználó.

collapsed: boolean = true; // A menü vizuális állapota, ha true akkor összecsukott állapotban van.

ngOnInit() // Inicializáló metódus. Feliratkozunk az authentikációs állapot figyelésére.

ngOnDestroy() // A példány törlése előtt leiratkozunk a figyelésről, hogy memóriát szabadítsunk fel.

onSignOut() // Kijelentkezés. Az authService signOut metódusát hívja meg.

toggleNav() // Összecsukja, vagy kinyitja a mobil navigációt.

## A Driver (sofőr) entitásokat kezelő komponens.

app/page/driver/driver.component.ts

cols = [

{key: "name", type: "text"},

{key: "email", type: "text"},

{key: "phone", type: "text"},

{key: "birthDate", type: "date"},

{key: "address", type: "text"}

]; // A táblázatban megjelenő oszlopok fejlécei és típusa.

constructor(

private dService: DriverService, // a Driver -eket kezelő szolgáltatás.

toaster: ToastsManager, // toast üzenetek szolgáltatása.

vcr: ViewContainerRef // a megjelenítő konténer referenciája.

) {

super(toaster, vcr);

this.dataService = dService; // Az adatokat biztosító szolgáltatás

this.entityName = 'sofőr'; // Az entitás neve.

this.list = dService.list; // Az adatok általános listája, generikus típussal.

}

ngOnInit() { // Inicializálás során feliratkozunk a Driver lista figyelésére.

this.listSubscribe = this.dService.all.subscribe(

list => this.list = this.dService.list

);

}

ngOnDestroy() { // Törlés előtt töröljük a figyelőket.

this.listSubscribe.unsubscribe();

}

}

## A tankolásokat kezelő komponens.

app/page/fueling/fueling.component.ts

constructor(

private fService: FuelingService, // tankolások szolgáltatása.

private vService: VehicleService, // járművek szolgáltatása.

private dService: DriverService, // sofőrök szolgáltatása.

toaster: ToastsManager, // toast üzenetek megjelenítése.

vcr: ViewContainerRef // a megjelenítő konténer.

) {

super(toaster, vcr);

this.dataService = fService; // Az adatokat biztosító szolgáltatás, amely mindig eltérő.

this.entityName = 'tankolás'; // Az entitás neve.

this.list = fService.list;

this.cols = [

{key: "vehicleId", type: "select", options: this.vService.vOptions},

{key: "driverId", type: "select", options: this.dService.vOptions},

{key: "time", type: "date"},

{key: "amount", type: "number"}

]; // Oszlopok definíciója a táblázatos megjelenítéshez.

}

ngOnInit() { // Inicializáció során iratkozunk fel az entitások módosulásaira.

this.listSubscribe = this.fService.all.subscribe(

list => {

this.list = this.fService.list;

console.log(list);

}

);

this.vSubscribe = this.vService.all.subscribe(

list => {

console.log(this.vService.vOptions);

this.cols[0] = {

key: "vehicleId", type: "select", options: this.vService.vOptions

};

}

);

this.dService.all.subscribe(

list => {

this.cols[1] = {

key: "driverId", type: "select", options: this.dService.vOptions

};

}

)

}

ngOnDestroy() { // Leiratkozunk a figyelőkről.

this.listSubscribe.unsubscribe();

this.vSubscribe.unsubscribe();

}

## A Vehicle entitás kezelése.

app/page/vehicle/vehicle.component.ts

cols = [

{key: "lp", type: "text"},

{key: "manufacturer", type: "text"},

{key: "type", type: "text"},

{key: "consumption", type: "number"},

{key: "fuelType", type: "select", options: Fuels},

{key: "year", type: "number"}

]; // A táblázat oszlopainak beállítása.

constructor(

private vService: VehicleService, // a járművek szolgáltatása.

toaster: ToastsManager, // toast üzenetek szolgáltatása.

vcr: ViewContainerRef // a megjelenítő kontíner referenciája.

) {

super(toaster, vcr);

this.dataService = vService; // Az adatokat biztosító szolgáltatás

this.entityName = 'sofőr'; // Az entitás neve.

this.list = vService.list;

}

ngOnInit() {

this.listSubscribe = this.vService.all.subscribe(

list => this.list = this.vService.list

);

}

ngOnDestroy() {

this.listSubscribe.unsubscribe();

}

## A statisztikák megjelenítéséért felelős komponens.

app/page/stat/stat.component.ts

constructor(

private cService: ChartDataService, // a chart adatainak szolgálatása.

private differs: KeyValueDiffers, // a különbségek figyelését végző szolgálatás.

toaster: ToastsManager,

vcr: ViewContainerRef

) {

super(toaster, vcr);

this.allData = this.cService.cachedData; // Az összes adatforrás.

this.createChartData();

this.filterDiffer = this.differs.find(this.filterObject).create();

}

ngOnInit() {

this.zipSubscription = this.cService.dataSubject

.map( values => {

values.drivers.forEach( item => {

this.driverList.push({

key: item.key,

label: item.driver.name

});

});

return values;

})

.subscribe(

zipData => {

this.allData = zipData;

this.createChartData();

}

);

}

ngDoCheck() {

var changes = this.filterDiffer.diff(this.filterObject); // check for changes

if (changes) {

this.chartType =

this.filterObject.driverID != 'notset' || this.filterObject.startDate ||

this.filterObject.endDate

? 'ColumnChart'

: 'PieChart';

this.createChartData(); // A feldolgozott adatotk a chart számára.

}

}

ngOnDestroy() {

this.zipSubscription.unsubscribe(); // Összesített feliratkozás a listák változásaira.

}

createChartData(): void { // Legenerálja a chart adatforrását. Piechart és ColumnChart esetén különböző adatstruktúrát készít.

if (!this.allData) return;

let chartData = {

chartType: this.chartType, // Az alapértelmezett chart típus.

dataTable: this.chartType == 'PieChart'

? this.getConsumptionPerDriver() // Egy sofőr tankolásait generálja listába.

: this.getFuelingsByDriver(),

options: {

'title': 'Tankolás',

'is3D': true,

'legend': 'bottom',

'width': 800,

'height': 600

}

};

this.pieChartData = chartData;

}

applyFilters(fuelings: Array<any>): Array<any> { // az összes tankolást tartalmazó tömb.

return fuelings.filter( item => {

let out = true;

if (this.filterObject.driverID != 'notset' &&

this.filterObject.driverID != item.fueling.driverId) {

out = false;

}

if (this.filterObject.startDate &&

this.filterObject.startDate > item.fueling.time) {

out = false;

}

if (this.filterObject.endDate &&

this.filterObject.endDate < item.fueling.time) {

out = false;

}

return out;

});

}

getConsumptionPerDriver(): Array<any> {

let compared: Array<any> = [

['Names', 'Fuel Consumption']

];

this.allData.drivers.forEach(item => {

compared.push(

[item.driver.name, this.getFueling(item.key)]

);

});

return compared;

}

getFuelingsByDriver(): Array<any> { // Az összes tankolás sofőrönkénti listáját generálja le.

let fueling: Array<any> = [

['Időpont', 'Üzemanyag mennyisége']

];

let fuelingData: Array<any> = this.applyFilters(this.allData.fuelings);

fuelingData.forEach(item => {

fueling.push(

[

item.fueling.time,

parseInt(item.fueling.amount)

]

);

});

return fueling;

}

getFueling(driverKey): any { // Visszaadja egy sofőr összes tankolását.

let fueling: any = 0;

let fuelingData: Array<any> = this.applyFilters(this.allData.fuelings);

fuelingData.forEach(item => {

if (item.fueling.driverId == driverKey) {

fueling += parseInt(item.fueling.amount);

}

});

return parseInt(fueling);

}