

Groapa Mihai, 343C5

Aplicatie Web – Gestiunea unei firme de transporturi

1. Descrierea temei

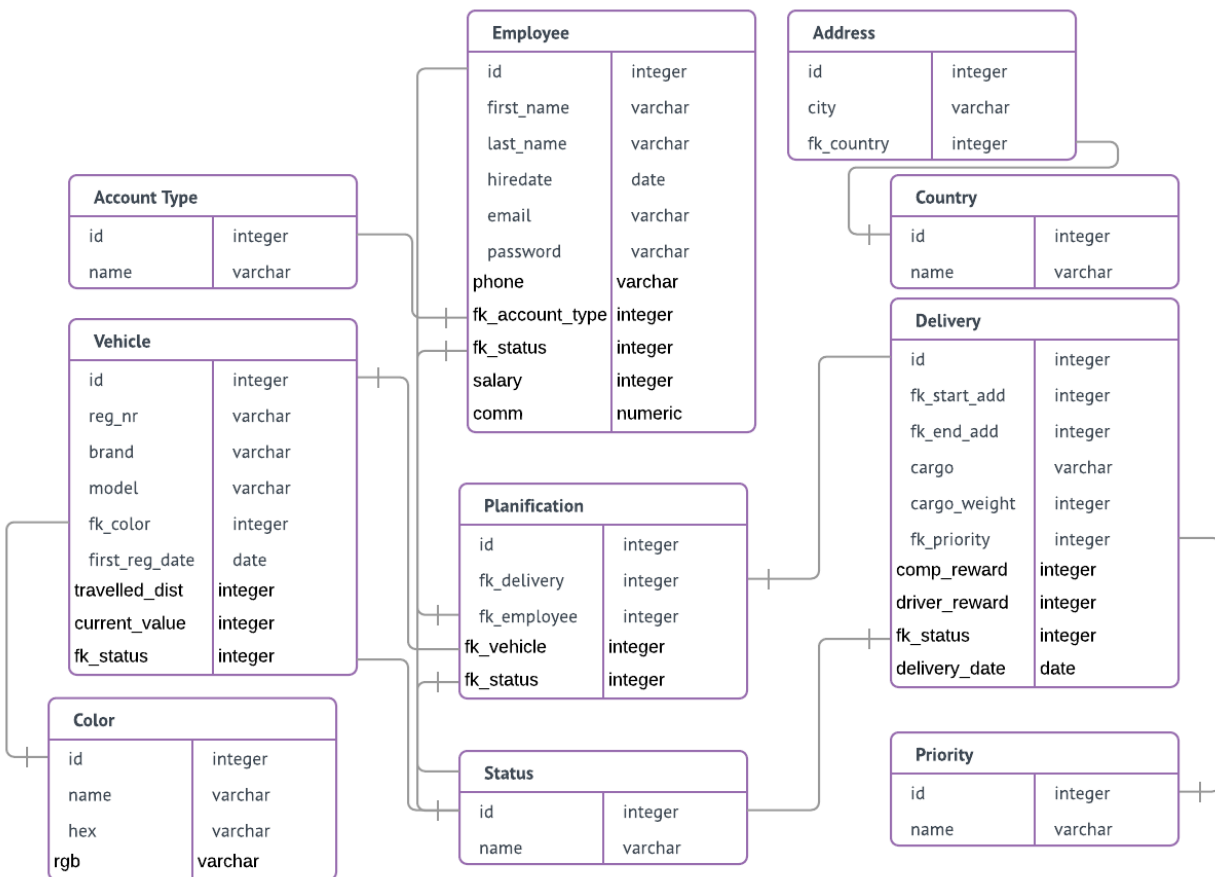
Sistemul permite gestiunea vehiculelor, a angajatilor si a livrarilor care trebuiesc efectuate. In sistem exista doua roluri de utilizator:

- **Coordinator:** inregistreaza vehicule, adauga angajati (le creeaza profil), adauga curse (specificand sursa, destinatia, continutul cargo-ului, valoarea cuvenita soferului ce va realiza transportul si altele), are posibilitatea de a realiza distribuirea curselor catre angajati si asignarea vehiculelor destinate transportului. O ultima actiune specifica coordonatorului este modificarea venitului soferilor, pe baza unor specificatii legate de performantele avute de acestia intr.o anumita perioada.
- **Driver:** soferul, dupa autentificare are posibilitatea de a accesa un istoric al livrarilor efectuate de el, precum si vizualizarea primei curse viitoare, asignata de unul dintre coordonatori lui. In momentul in care termina

livrarea, soferul marcheaza acest lucru in portal, el si camionul cu care a efectuat transportul devenind disponibili unor noi curse.

2. Baza de date

Pentru baza de date am folosit PgAdmin4 cu PostgreSQL.



Pentru toate operatiile executate am folosit functii stocate in baza, inclusive pentru operatiile de tip CRUD simple.

```

1 DECLARE
2     employee_row employee%rowtype;
3     deliveries_number_for_employee BIGINT;
4     company_revenue_from_employee BIGINT;
5 BEGIN
6     FOR employee_row IN SELECT * FROM get_all_employees()
7     LOOP
8         deliveries_number_for_employee := get_deliveries_nr_between_dates_for_employee(start_date, end_date,
9         company_revenue_from_employee := get_company_revenue_between_dates_from_employee(start_date, end_date,
10
11         IF operator_v = 'AND' THEN
12             IF deliveries_number_for_employee >= min_deliveries AND company_revenue_from_employee >= min_com
13                 RETURN NEXT employee_row;
14             END IF;
15         ELSIF operator_v = 'OR' THEN
16             IF deliveries_number_for_employee >= min_deliveries OR company_revenue_from_employee >= min_com
17                 RETURN NEXT employee_row;
18             END IF;
19         END IF;
20     END LOOP;
21     RETURN;
22 END
23

```

Funcție care întoarce lista angajaților ce într.un interval încadrat de două date [d1.m1.y1, d2.m2.y2] au realizat un minim de 'min_deliveries' curse și/sau (în funcție de caz), au adus în firmă un venit minim de 'min_company_revenue' în această perioadă.

```

1 DECLARE
2     employee_row employee%rowtype;
3     new_salary INTEGER;
4     new_comm NUMERIC;
5 BEGIN
6     FOR employee_row IN SELECT * FROM get_employees_eligible_for_revenue_inc(min_deliveries, operator_v, min
7     LOOP
8         IF salary_increase_type = 'Fixed Value' THEN
9             new_salary := employee_row.salary + salary_increase;
10        ELSIF salary_increase_type = 'Percentage' THEN
11            new_salary := employee_row.salary * salary_increase;
12        END IF;

```

```

14      IF comm_increase_type = 'Fixed Value' THEN
15          new_comm := employee_row.comm + comm_increase;
16      ELSIF comm_increase_type = 'Percentage' THEN
17          new_comm := employee_row.comm * comm_increase;
18      END IF;
19
20      employee_row.salary := new_salary;
21      employee_row.comm := new_comm;
22
23      UPDATE employee
24      SET salary = new_salary, comm = new_comm
25      WHERE id = employee_row.id;
26
27      RETURN NEXT employee_row;
28  END LOOP;
29  RETURN;
30 END

```

Procedura care aplica o crestere a salariului si a comisionului (procentuala sau cu valoare fixata) pentru angajatii intorsi de functia precedenta.

```

1 BEGIN
2     IF (TG_OP='INSERT') THEN
3         INSERT INTO planification_audits
4         VALUES((select nextval('seq_planification_audits')),
5     END IF;
6
7     RETURN NEW;
8 END;
9

```

Trigger declansat la inserarea in baza a unei planificari, ce salveaza intr.o tabela suplimentara numita

'planification_audits' id-ul planificarii introduce si timestamp-ul current.

Constrangeri

O planificare este compusa din: un vehicul cu care se realizeaza transportul, o entitate delivery, ce la randul ei are o sursa si o destinatie (fk catre tabela address), un status al livrarii (Available, In Transit, Delivered, valori preluate prin fk catre tabela status) si legatura catre soferul ce face deplasarea (fk_employee). In afara de acestea, fiecare tabela are constrangere de tip cheie primara, prin campul id.

3. Descrierea aplicatiei

Aplicatia urmareste modelul Model-View-Controller. Partea de Backend este separata de modulul pentru Frontend.

In Backend, este urmat fluxul:

RestController -> Service -> Dao -> Entity

Asadar, pentru fiecare tip de entitate din aplicatie am 4 clase.

RestController-ul face maparea pe url-uri. Exemplu:

```

@CrossOrigin(allowedHeaders="*",allowCredentials="true")
@PostMapping(value = "/employee/driversRevenueIncrease")
@ResponseBody
public List<Employee> increaseRevenues(@RequestBody RevenueIncrease jsonRevenueIncrease) {
    return employeeService.increaseRevenues(jsonRevenueIncrease);
}

```

Service-ul are rol de intermediar intre controller si Dao:

```

@Override
public List<Employee> increaseRevenues(RevenueIncrease revenueIncrease) {
    return employeeDao.increaseRevenues(revenueIncrease);
}

```

In componenta Dao fac apelul catre procedurile/functiile stocate:

```

public List<Employee> increaseRevenues(RevenueIncrease revenueIncrease) {
    CallableStatement callableStatement;
    ResultSet resultSet = null;

    try {
        callableStatement = connection.prepareCall( sql: "{call apply_revenue_inc(?,?,?,?,?,?,?,?)}" );
        callableStatement.setBigDecimal( parameterIndex: 1, revenueIncrease.getSalaryIncrease());
        callableStatement.setString( parameterIndex: 2, revenueIncrease.getSalaryIncreaseType());
        callableStatement.setBigDecimal( parameterIndex: 3, revenueIncrease.getCommIncrease());
        callableStatement.setString( parameterIndex: 4, revenueIncrease.getCommIncreaseType());
        callableStatement.setInt( parameterIndex: 5, revenueIncrease.getMinDeliveries());
        callableStatement.setString( parameterIndex: 6, revenueIncrease.getOperator());
        callableStatement.setInt( parameterIndex: 7, revenueIncrease.getMinCompanyRevenueFromDriver());
        callableStatement.setDate( parameterIndex: 8, revenueIncrease.getStartDate());
        callableStatement.setDate( parameterIndex: 9, revenueIncrease.getEndDate());
        callableStatement.executeUpdate();
        resultSet = callableStatement.getResultSet();
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return extractEmployeesFromResultSet(resultSet);
}

```

In Frontend, se urmeaza tiparul:

View -> Controller -> Service[s]

Asadar, fiecare ecran are asociat un controller angular in care, prin dependency injection accesez serviciile. Acestea din urma sunt responsabile cu requesturile de tip GET, POST, PUT catre server.

Conexiunea la baza de date

Am folosit Spring, astfel ca pentru conexiune a fost necesar sa declar dependenta pentru postgres in pom.xml, in fisierul application.properties sa declar parametrii bazei de date si nu in ultimul rand, in clasa de configurare a proiectului sa declar un bean de tip DataSource, din care se poate extrage obiectul connection.

```
23 @Configuration
24 @EnableAutoConfiguration
25 @EnableTransactionManagement
26 @ComponentScan(basePackages = {"ro.transport.demo.controllers"})
27 @EntityScan(value = "ro.transport.demo.domain")
28 public class ProjectConfiguration {
29
30     private static final String PACKAGES_TO_SCAN = "ro.transport.demo.domain";
31     private static final String USERNAME_DATABASE = "postgres";
32     private static final String PASSWORD_DATABASE = "postgres";
33     private static final String URL_DATABASE = "jdbc:postgresql://localhost:5432/Transport_Logistics_DB";
34     private static final String ENTITY_MANAGER_DIALECT = "org.hibernate.dialect.PostgreSQLDialect";
35
36     @Bean
37     public DataSource dataSource() {
38         SimpleDriverDataSource dataSource = new SimpleDriverDataSource();
39         dataSource.setDriverClass(org.postgresql.Driver.class);
40         dataSource.setUsername(USERNAME_DATABASE);
41         dataSource.setPassword(PASSWORD_DATABASE);
42         dataSource.setUrl(URL_DATABASE);
43
44         return dataSource;
45     }
46 }
```

```
41 <dependency>
42     <groupId>org.postgresql</groupId>
43     <artifactId>postgresql</artifactId>
44     <scope>runtime</scope>
45 </dependency>
```

```

1 spring.datasource.url=jdbc:postgresql://localhost:5432/Transport_Logistics_DB
2 spring.datasource.username=postgres
3 spring.datasource.password=postgres
4 spring.jpa.generate-ddl=false
5 spring.jpa.hibernate.ddl-auto=none
6 spring.jpa.properties.hibernate.id.new_generator_mappings=false
7 liquibase.change-log=classpath:/liquibase/demo.xml
8 server.port=9090

```

Transport Logistics Appli X

localhost:9089/#/coordinator/planification/all

Menu Home Deliveries Employees Planifications Vehicles gmlhai9549@gmail.com 2018-01-08 04:08:41

Status:

All

Id	Starting Address	Destination Address	Cargo	Company Reward (euro)	Driver Reward (euro)	Vehicle	Driver	Status	Delivery Date	Action
9	[BZ] Belmopan	[AG] Potters Village	Chocolate	6700	1500	Mercedes Actros	Paulo Dybala	In transit	N/A	View
8	[AM] Aneghahkot	[RO] Mihăilești	Salt	5602	1600	Scania R450	Kevin De Bruyne	Delivered	08-01-2018	View
10	[MC] Monaco	[OM] Sufâlat Samâ'il	Weed	14500	4000	Volvo PA-STEER	Kevin De Bruyne	In transit	N/A	View

Windows 10 Type here to search

Transport Logistics Appli X

localhost:9089/#/coordinator/employee/driversRevenueIncrease

Increase salary by: 200 Increase type: Fixed Value

Increase commission by: 20 Increase type: Percentage

Increase conditions

Minimum number of deliveries: 1 Operator: AND Minimum company revenue from driver's deliveries: 3000

Start date: 02/02/2000 End date: 02/02/2019

[List eligible drivers](#) [Apply increase](#) [Reset](#)

Eligible Drivers

Id	First Name	Last Name	Hire Date	Email	Phone Number	Salary	Commission	Status
13	Kevin	De Bruyne	01-01-2013	kdb@city.com	902-352-678	6700	0.3	In transit

Windows 10 Type here to search

Transport Logistics Appl

localhost:9089/#/coordinator/vehicle/all

Menu Home Deliveries Employees Planifications Vehicles gmihai9549@gmail.com 2018-01-08 04:10:39

Status: Available

Id	Registration Number	Brand	Model	Color	First Registration Date	Travelled Distance (km)	Current Value (euro)	Status	Action
8	BV-90-CRS	Volvo	FH	Blue Bell	01-01-2014	445000	47500	Available	View
10	VS-46-MNA	Scania	R420	Red Orange	15-08-2012	620000	28000	Available	View
1	B-76-DAF	DAF	XF 105.460	Periwinkle	01-01-2012	783874	21700	Available	View
4	SV-202-MAN	MAN	33.400	Jazzberry Jam	01-06-2014	50200	88200	Available	View
13	TL-168-FRD	Ford	L 9000	Glitter Spack	25-02-1979	2456902	12500	Available	View
2	B-37-DAF	DAF	AE75PC	Raw Sienna	01-01-2005	1107464	14900	Available	View
3	GR-100-BOS	Iveco	AS440S45T	Bittersweet	01-03-2011	785609	15500	Available	View
12	AG-15-RPD	Renault	Magnum 460	Brown	15-04-2007	769838	7900	Available	View
11	OT-69-SCN	Scania	R450	Asparagus	12-08-2014	526000	46800	Available	View

Concluzii

- Proiect foarte interesant, am invatat multe lucruri despre baze de date, relatii intre tabele (de exemplu, e foarte important sa stabilesti de la inceput ce coloane sunt necesare pentru o entitate, daca o faci ulterior modifici mult cod).
- Foarte mult de lucru, mai ales ca a trebuit sa scriu proceduri/functii pentru fiecare operatie cu baza (in final au iesit undeva in jur de 45).

Bibliografie: niciuna, am aplicat cunostinte acumulate la locul de munca, cam asta e modul de lucru acolo, mai putin partea de procedure stocate, folosim JPA.