אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

הפקולטה למדעי ההנדסה
המחלקה להנדסת מערכות מידע

Faculty of Engineering Sciences
Dept. of Information Systems Engineering

Ben-Gurion University of the Negev

“Superly”

Specification Document

Roi Tiefenbrunn 209829340
Ofek Nov 206618175
Chai Shalev Hadad 209268135
Eldad Vasker 325082311

Use Case Diagram:

Use Case Diagram detailing the following use cases:

a. Registering new employee

b. Updating employee details and constraints

c. Inventory updating and warning on low stock

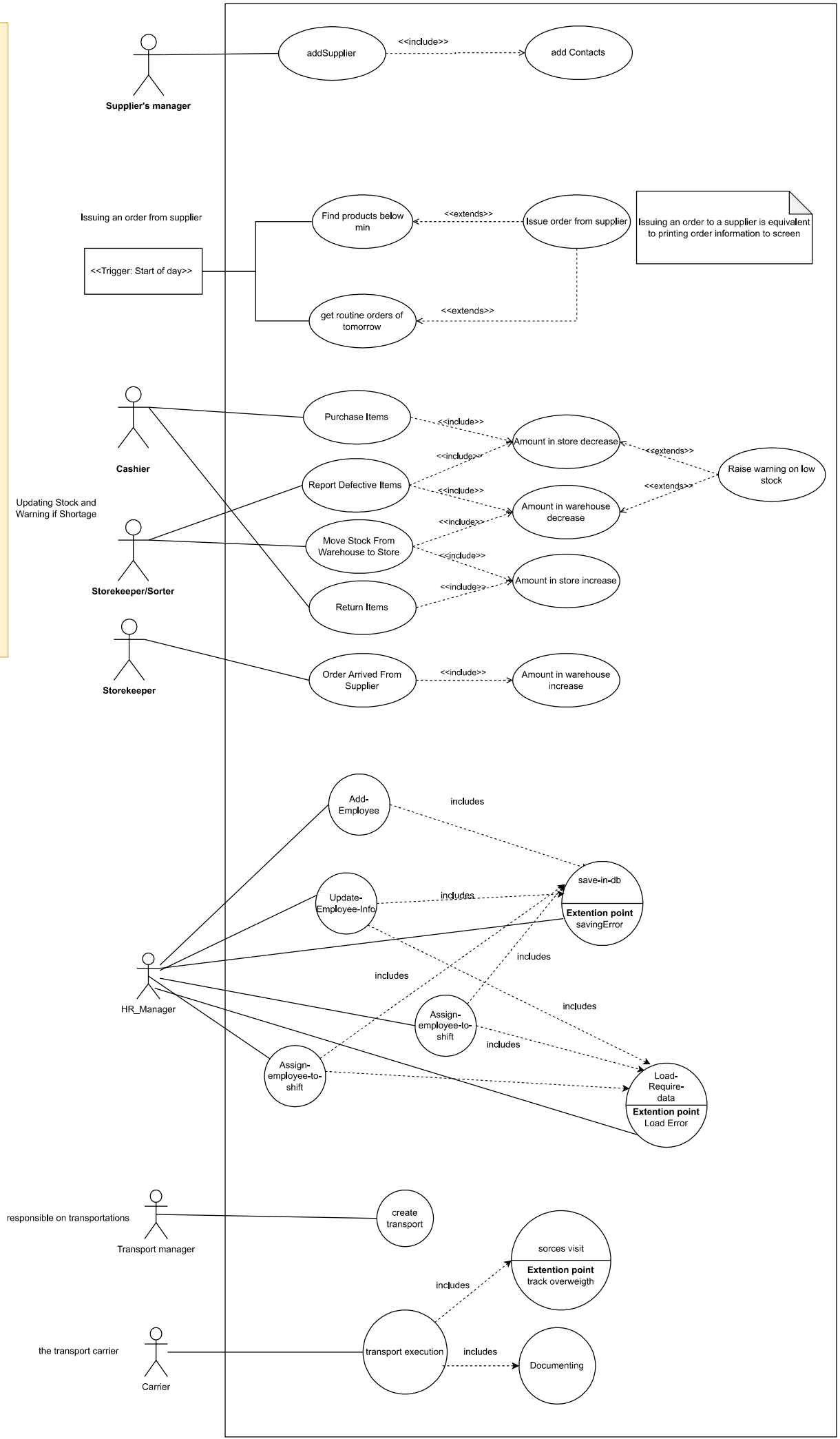
d. Adding new supplier

e. Carrying out routine order from supplier

f. Carrying out order due to low stock

g. Assigning employee to shift

h. Carrying out a transport

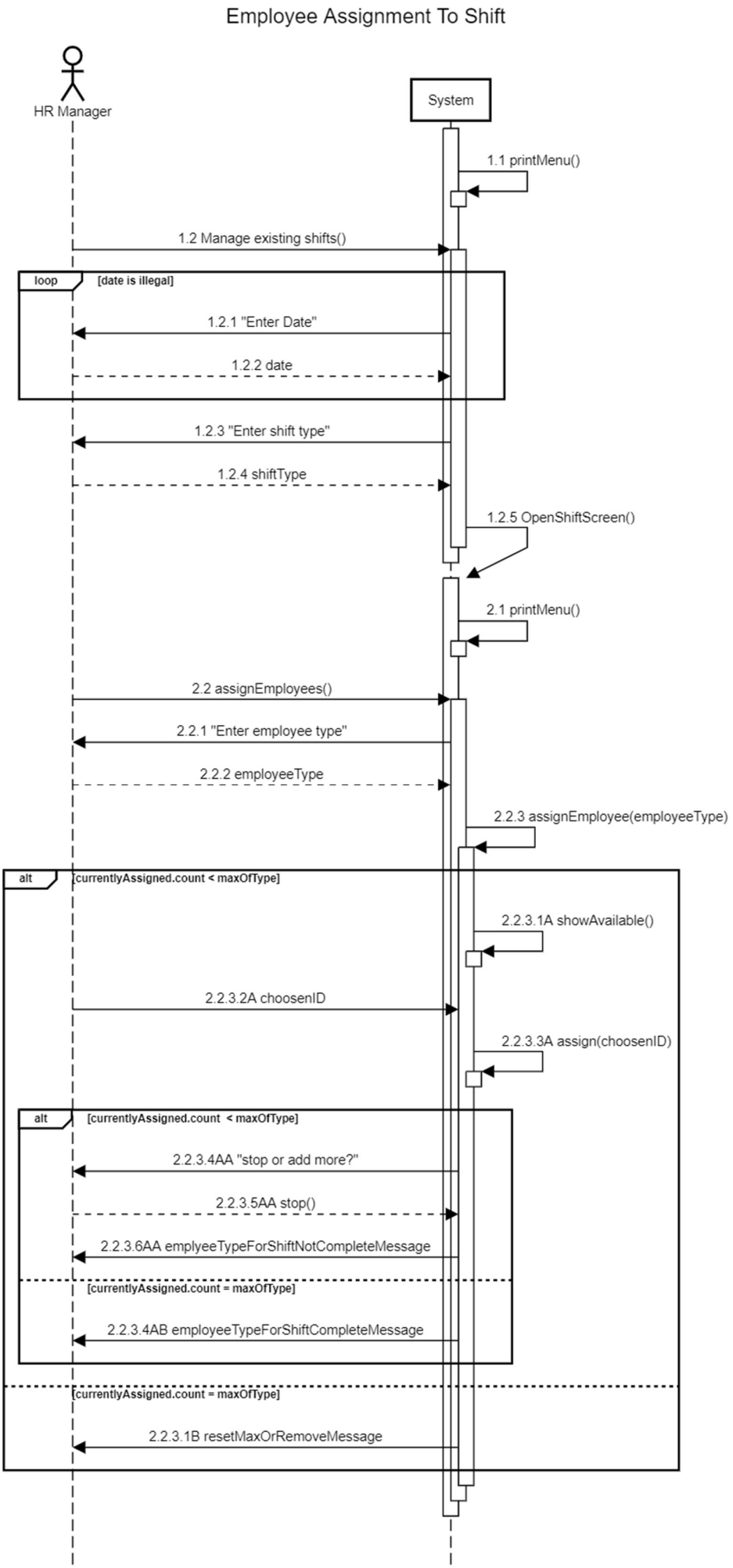


Detailed Use Cases:

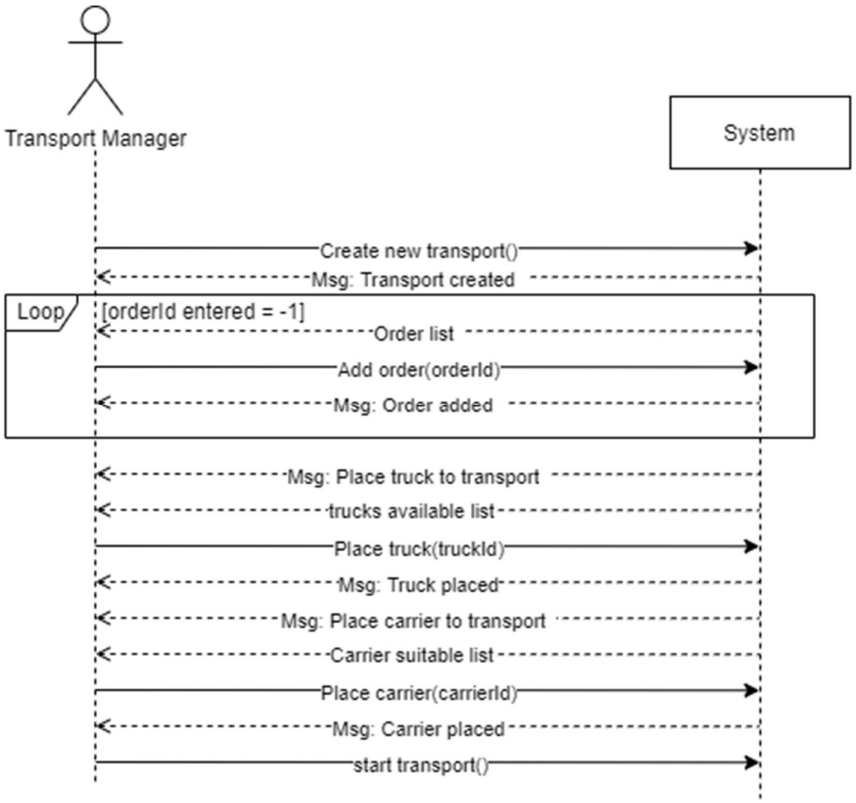
Use case name	Assigning-Employee-to-Shift
Textual Description	The HR manager assigns a specific employee to a specific shift.
List of Actor	HR Manager
Pre-Conditions	The HR manager is in the ShiftMenu, The shift must exist in the system; The employee has to have a constraint matching shift date and type; The employee must not be assigned to the shift
Post-Conditions	The shift has the employee assigned to it; the database includes a record which details the assignment
Main success scenario	1) The HR manager chooses the shift through the ShiftMenu 2) The HR manager chooses to add to the shift employees of the type of the wanted employee 3) System validates number of already assigned employees of chosen type is less than set max number for this type 4) System shows available employees for chosen shift of chosen type 5) HR Manager chooses the employee out of the list of available employees 6) System checks new number of currently assigned to set max 7) HR Manager stops the assignment process
Alternative/Extensions	3') Number of assigned employees matches set number 4') System notifies HR manager of the situation and offers to reset the number of employees for the shift or remove already assigned employee
	7'') System stops the assignment process

Use case name	Carrying out transport
Textual Description	Creating and carrying out a transport.
List of Actor	Transport manager and Carrier
Pre-Conditions	There is an order for transportation. An available truck exists for the time of the transportation. A carrier with a suitable license for this truck and with a constraint for the shift time. There are sources and destinations of transportation
Post-Conditions	In the completed transports archive the order exist. The truck and driver will be available for other transports
Main success scenario	**1)** Transport Manager create new transport. **2)** Transport manager adds transport-orders to the transport. **3)** Transport manager chooses truck for the transport. **4)** Transport manager chooses carrier suitable for this transport. **5)** The carrier updates the destination-document for each destination visited, the change is saved in the archive. 6) When the carrier finishes his ride the transport is marked as finished in the transport-document, the change is saved in the archive.
Alternative/Extensions	In case the truck enters overweight state an alert will be sent to the carrier and a redesign of the transport will be performed. The transport will end and start over.

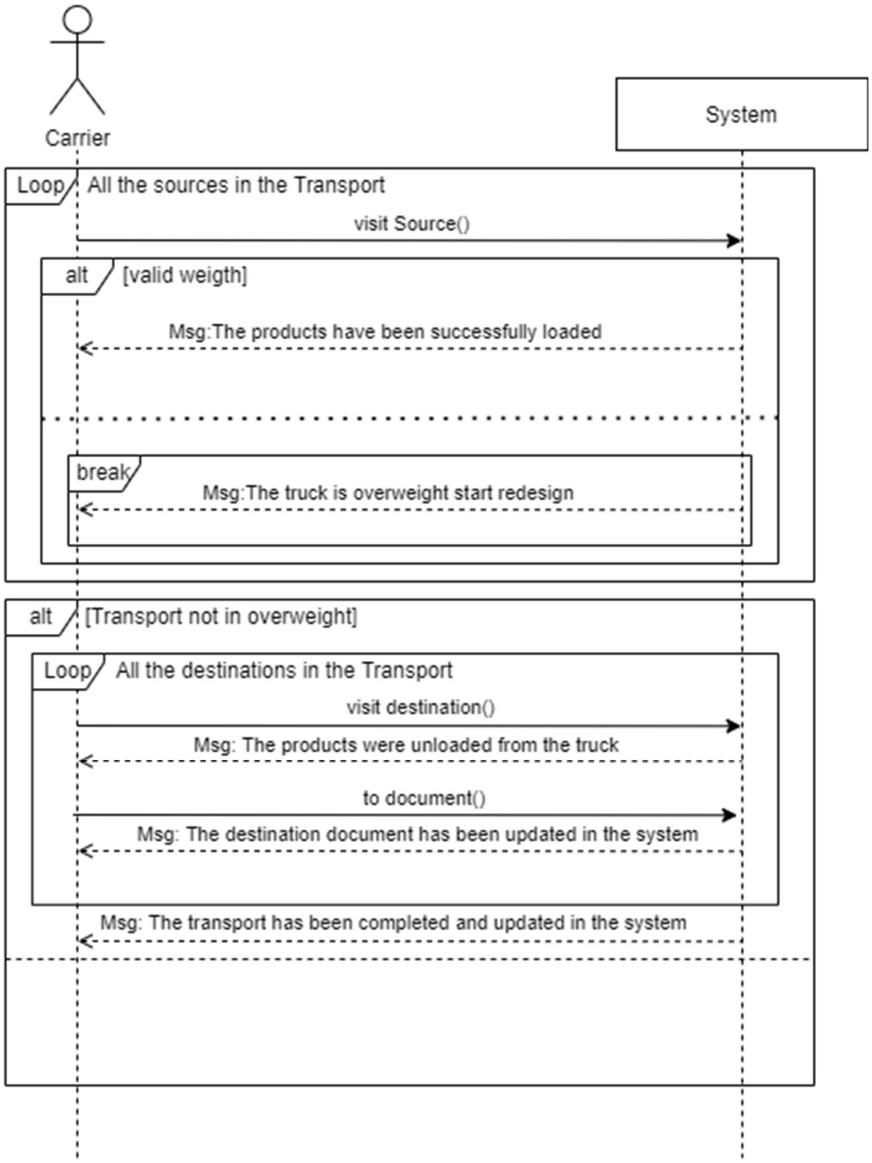
When assigning an employee to a shift the system will interact with the manager in order to produce identifying data for the shift and the employee. The system will assign and save the assignment in the system if conditions are met.



When we want to create transport, the transport manager need to assign the orders of the transport, a truck and a carrier for the transport, and then he can start the transport

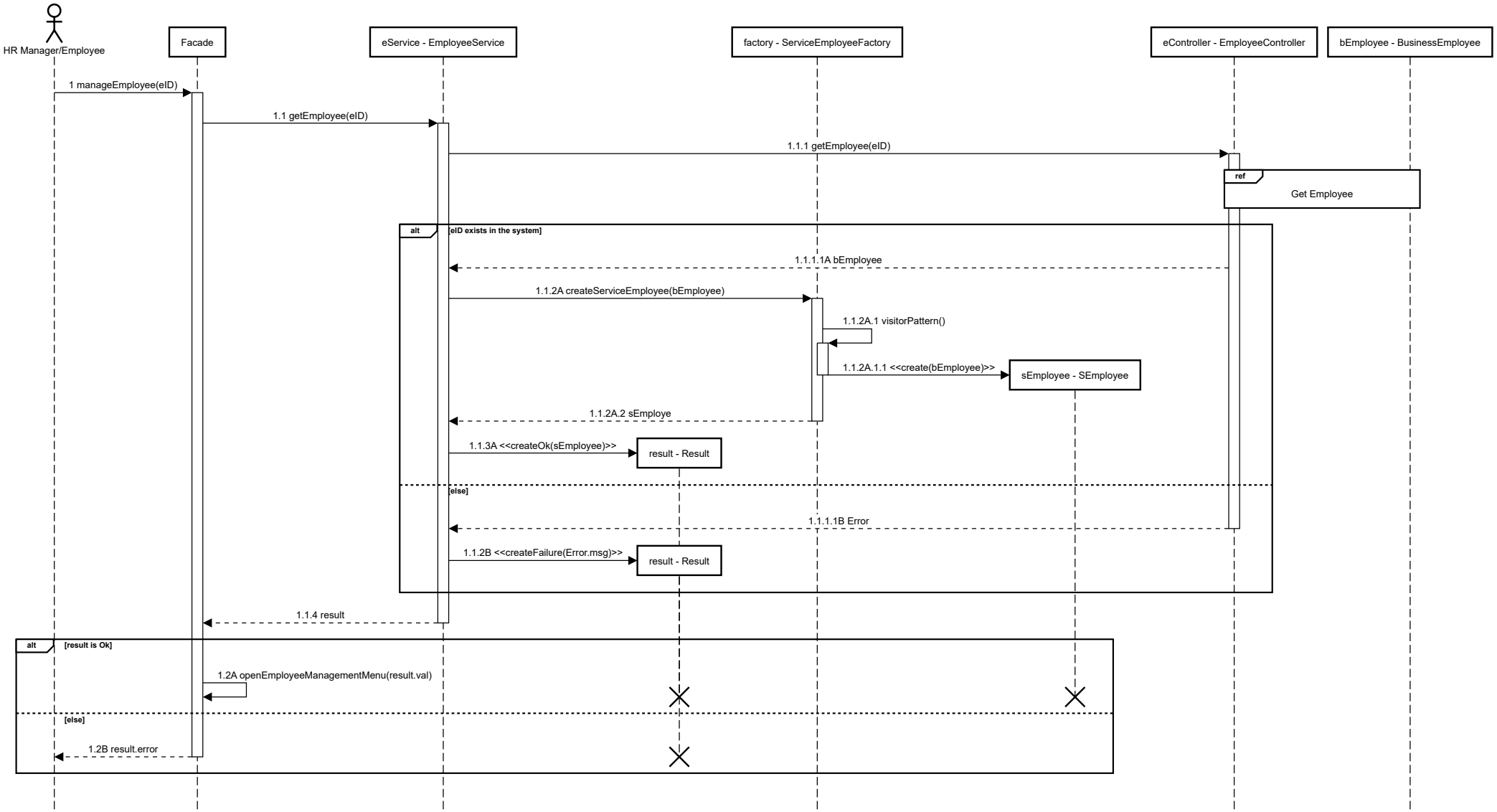


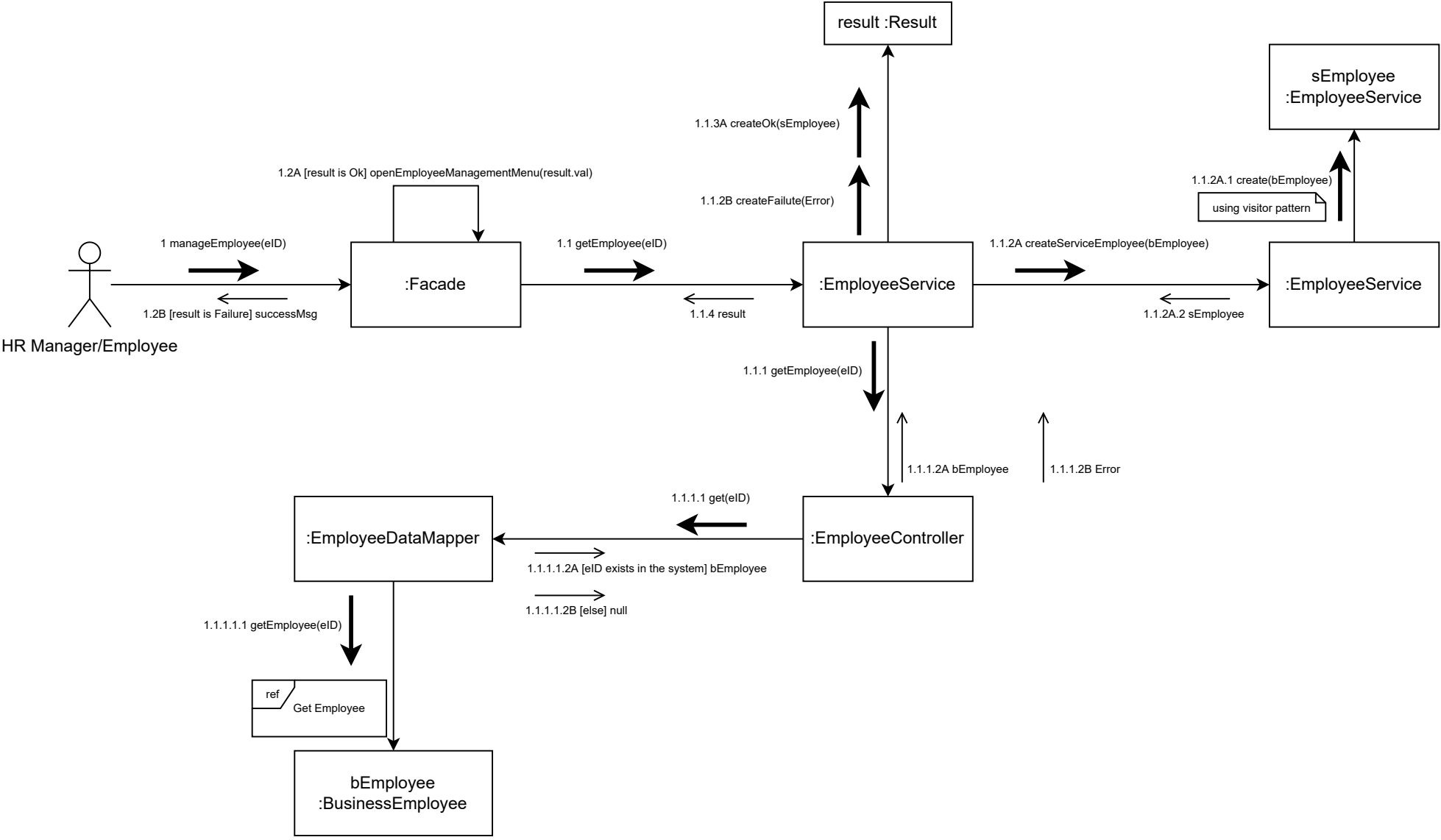
When the transport starts the carrier updates when he arrives to the next location. If he arrive to source and the truck is in overweight we start redesign and stop the transport. For every destination that the carrier arrives the driver get destination document for this destination and at the end gets the transport document.



Opening the Employee-Management-Menu is a key process in most employee-management use-cases. Which is why it’s included here in great detail.
Each Actor whishing to open an Employee-Management-Menu, be it the HR manager or the employee itself, would only need to input the ID of the wanted employee.
The system will open the menu if the ID exists in the system, or print an error message if it does not.
Since the system uses lazy-load way of work a Get request is sent to the DAO.

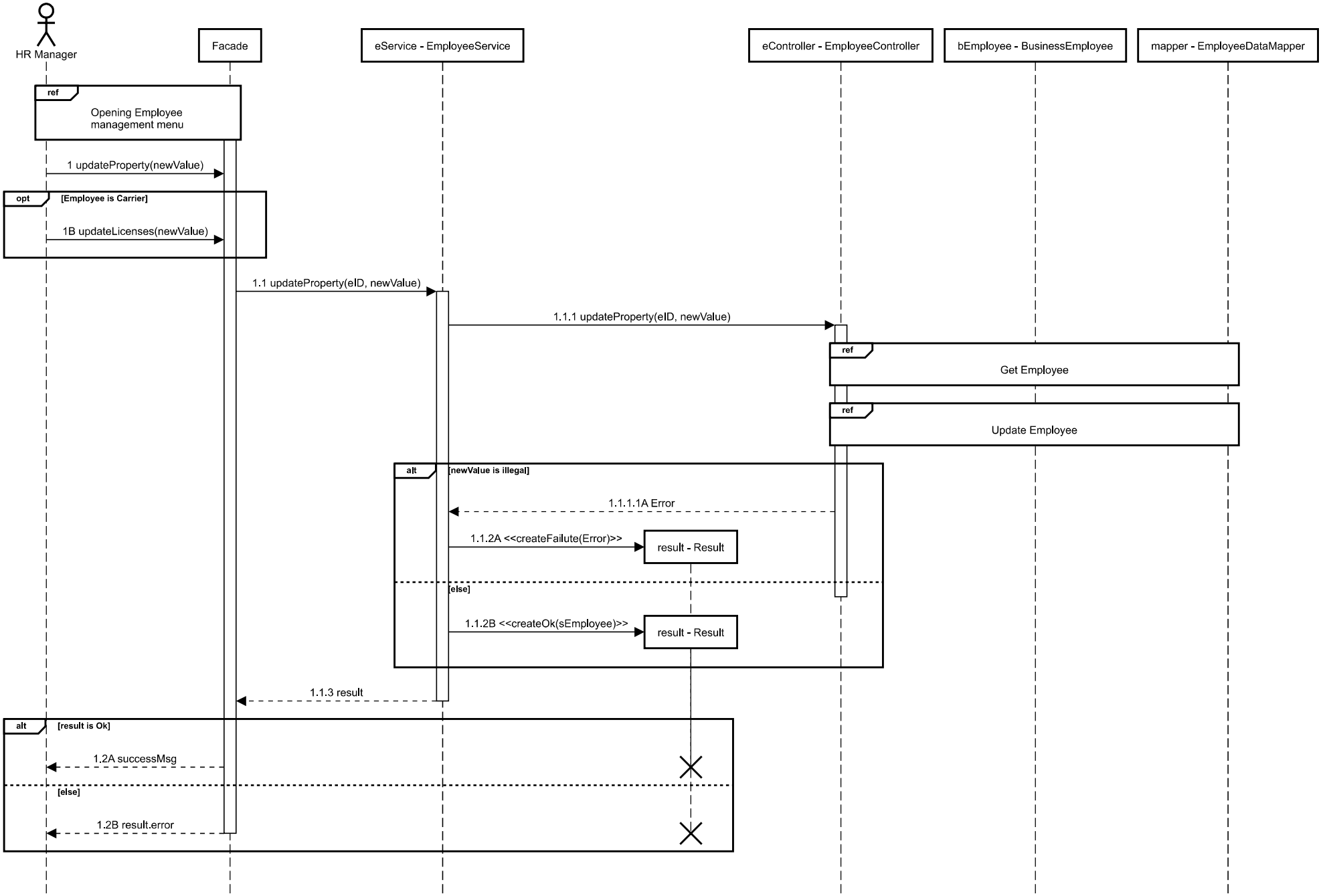
Opening Employee Management Menu

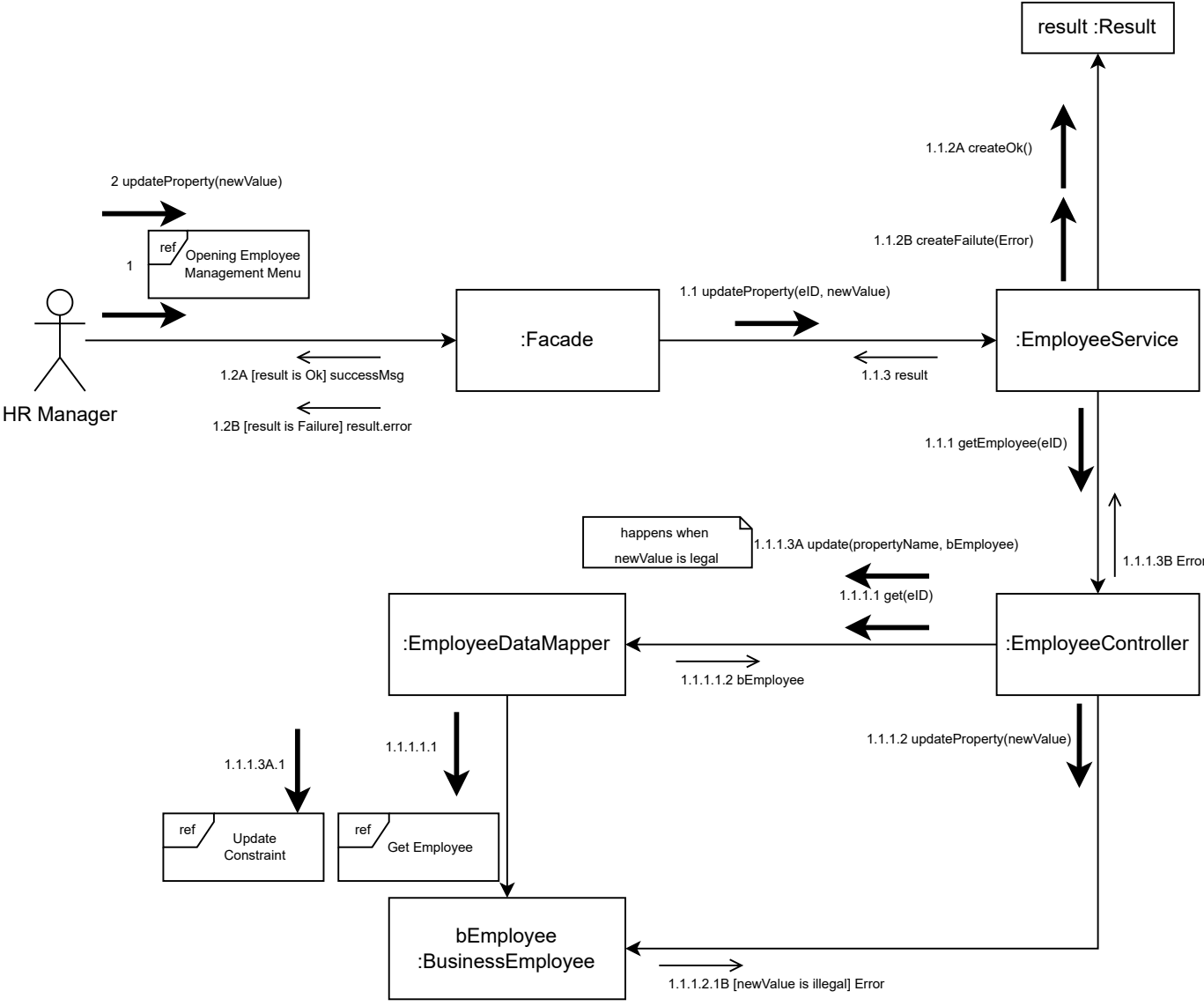



result :Result
sEmployee
:EmployeeService
1.1.3A createOk(sEmployee)
1.2A [result is Ok] openEmployeeManagementMenu(result.val)
1.1.2A.1 create(bEmployee)
using visitor pattern
1.1.2B createFailute(Error)
1 manageEmployee(eID)
1.1 getEmployee(eID)
1.1.2A createServiceEmployee(bEmployee)
:Facade
:EmployeeService
:EmployeeService
1.2B [result is Failure] successMsg
1.1.4 result
1.1.2A.2 sEmployee
HR Manager/Employee
1.1.1 getEmployee(eID)
1.1.1.2A bEmployee
1.1.1.2B Error
1.1.1.1 get(eID)
:EmployeeDataMapper
:EmployeeController
1.1.1.1.2A [eID exists in the system] bEmployee
1.1.1.1.2B [else] null
1.1.1.1.1 getEmployee(eID)
ref
Get Employee
bEmployee
:BusinessEmployee

When the HR manager requests to update an existing employee in the system he will first open the Employee-Management-Menu for that employee as shown above. After which he will choose which property to update and to what value. If the employee is a carrier the licenses for said carrier may also be updated.
The process requires use of multiple DAO processes.

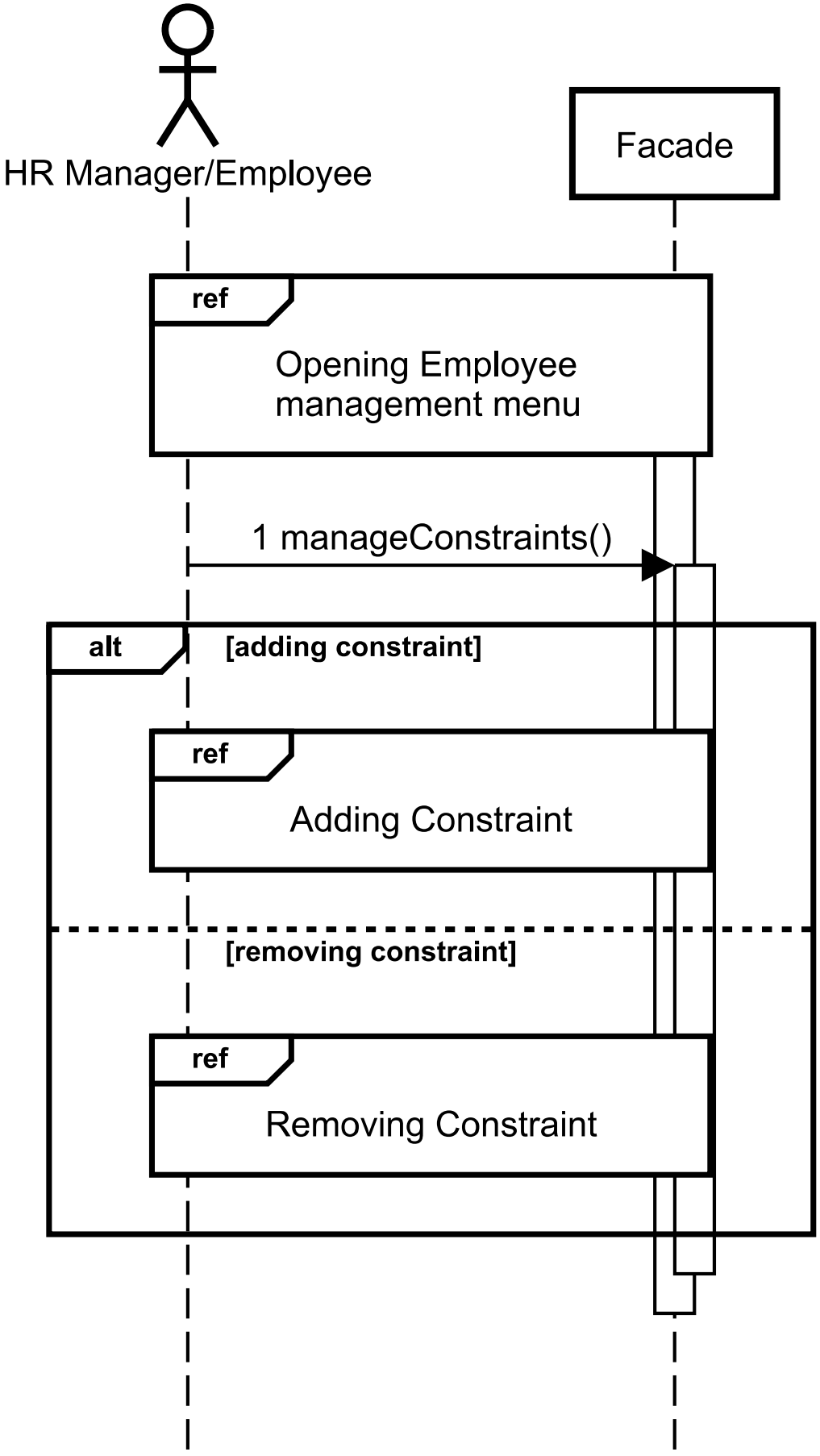
Updating Employee Details



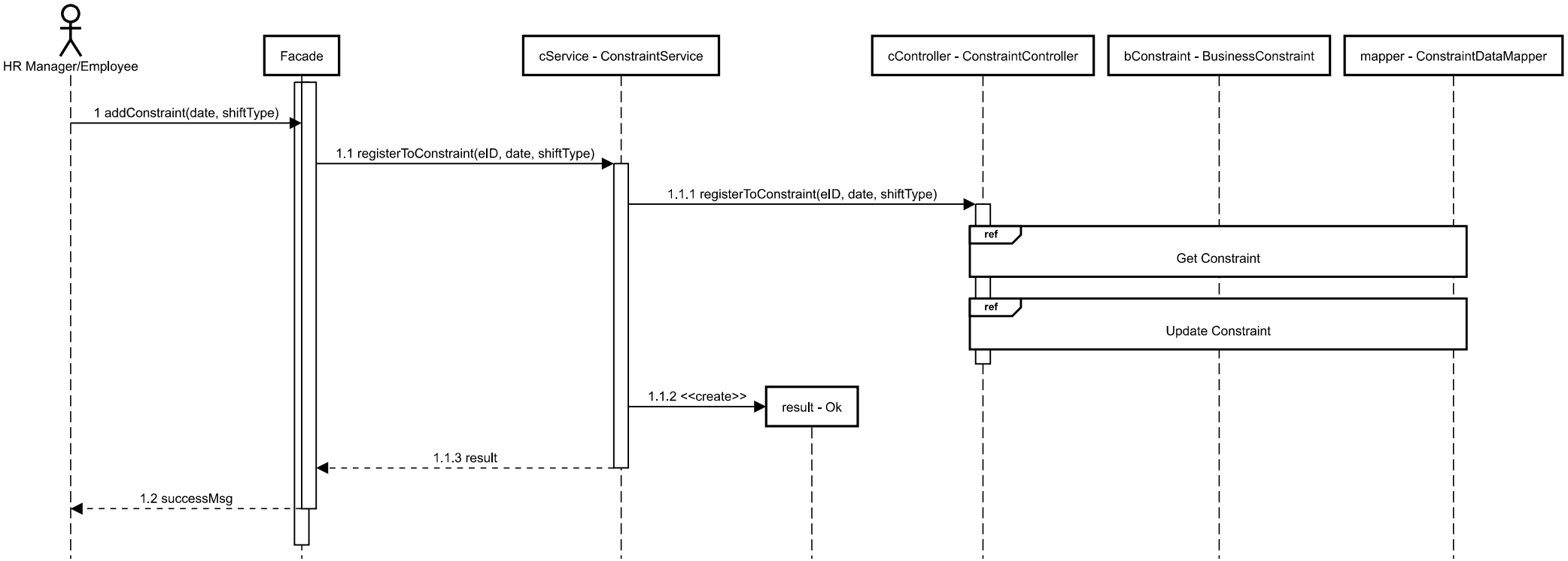

result :Result
1.1.2A createOk()
1.1.2B createFailute(Error)
2 updateProperty(newValue)
1
ref
Opening Employee Management Menu
HR Manager
:Facade
1.1 updateProperty(eID, newValue)
:EmployeeService
1.2A [result is Ok] successMsg
1.2B [result is Failure] result.error
1.1.3 result
1.1.1 getEmployee(eID)
happens when newValue is legal
1.1.1.3A update(propertyName, bEmployee)
1.1.1.3B Error
1.1.1.1 get(eID)
:EmployeeDataMapper
:EmployeeController
1.1.1.1.2 bEmployee
1.1.1.2 updateProperty(newValue)
1.1.1.3A.1
1.1.1.1.1
ref
Update Constraint
ref
Get Employee
bEmployee
:BusinessEmployee
1.1.1.2.1B [newValue is illegal] Error

In order to update employee constraints the HR manager or the employee itself will first have to open the Employee-Management-Menu and then choose whether to add or remove constraints. Both these processes run in a similar fashion – the user inputs the date and shift-type they can or can't work and will be notified with process success. This process doesn't fail unless an IO exception occurs when data is saved in the DB.

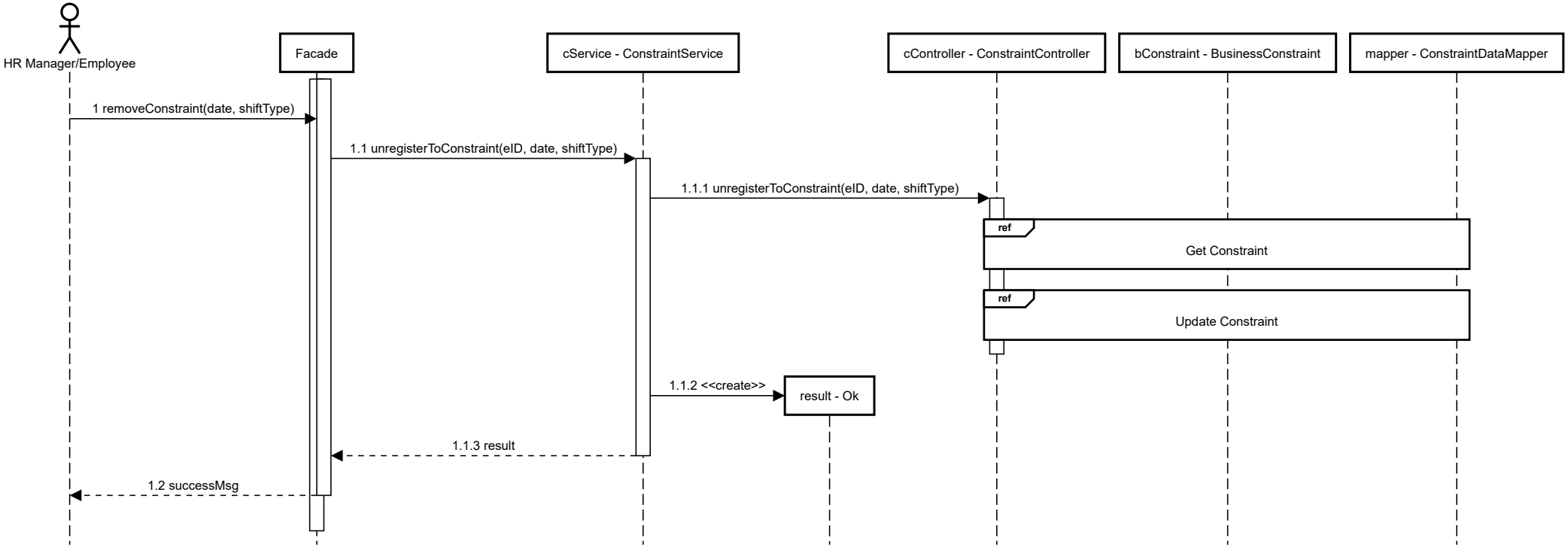
Update Employee Constraint

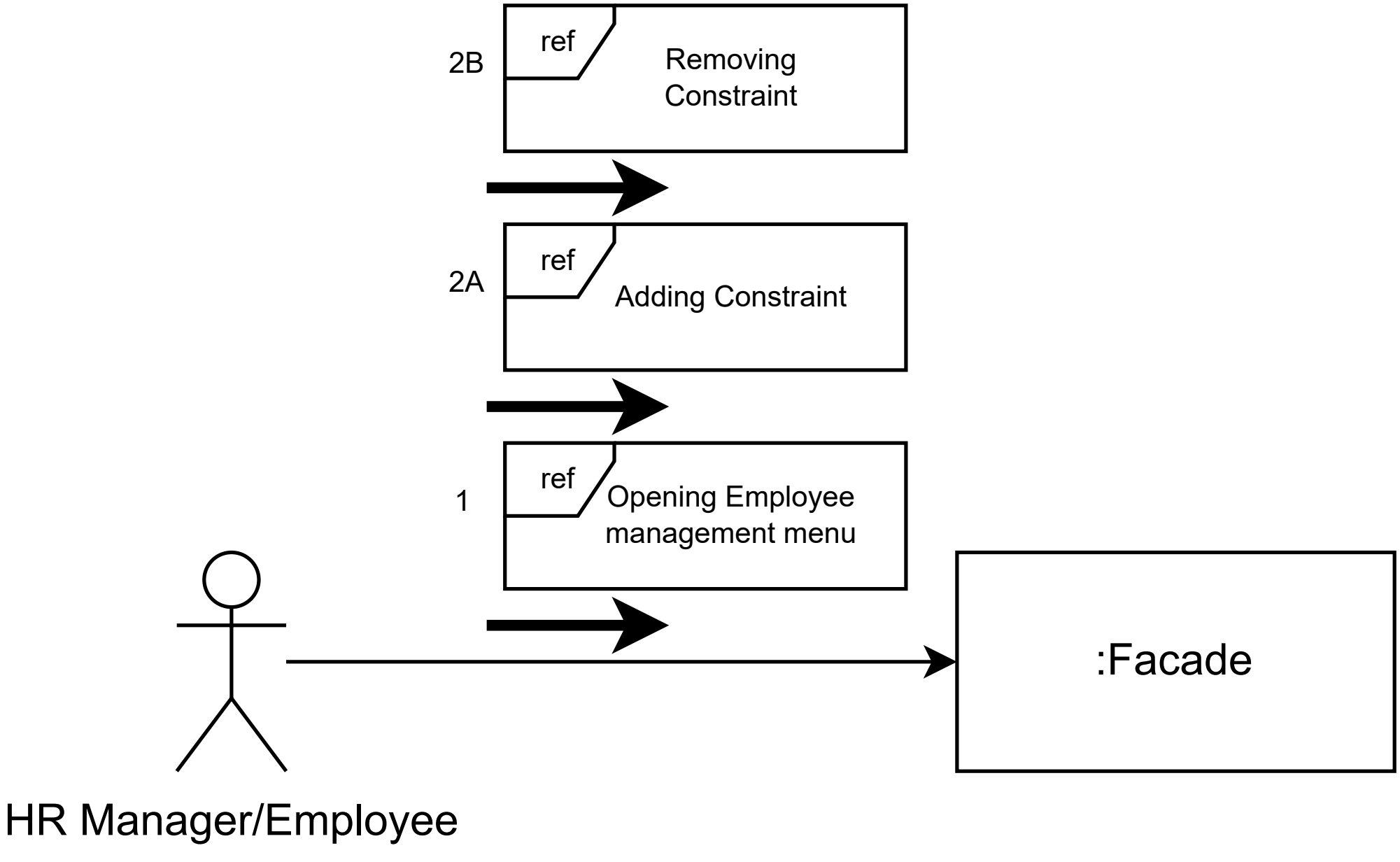


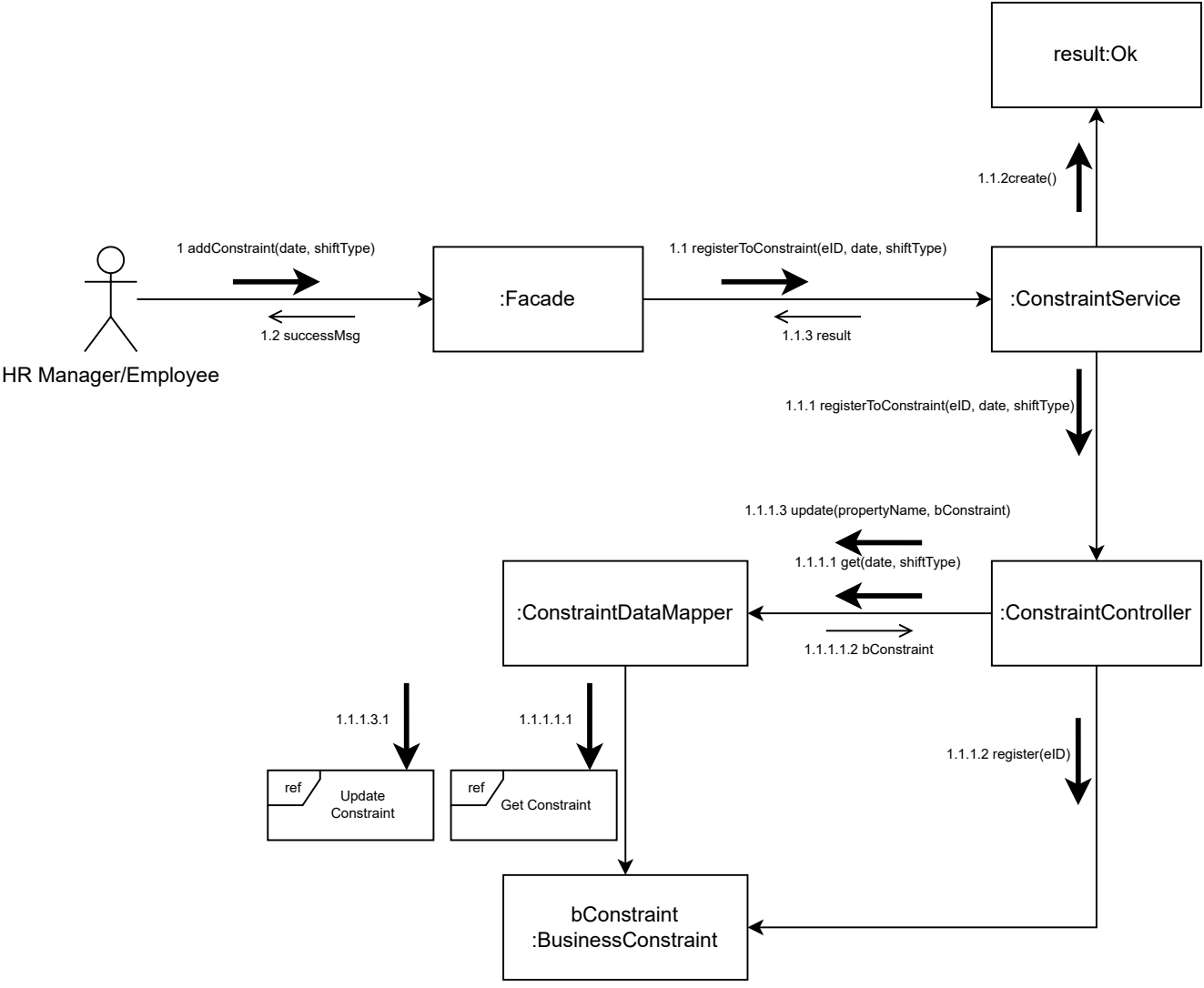
Adding Constraint

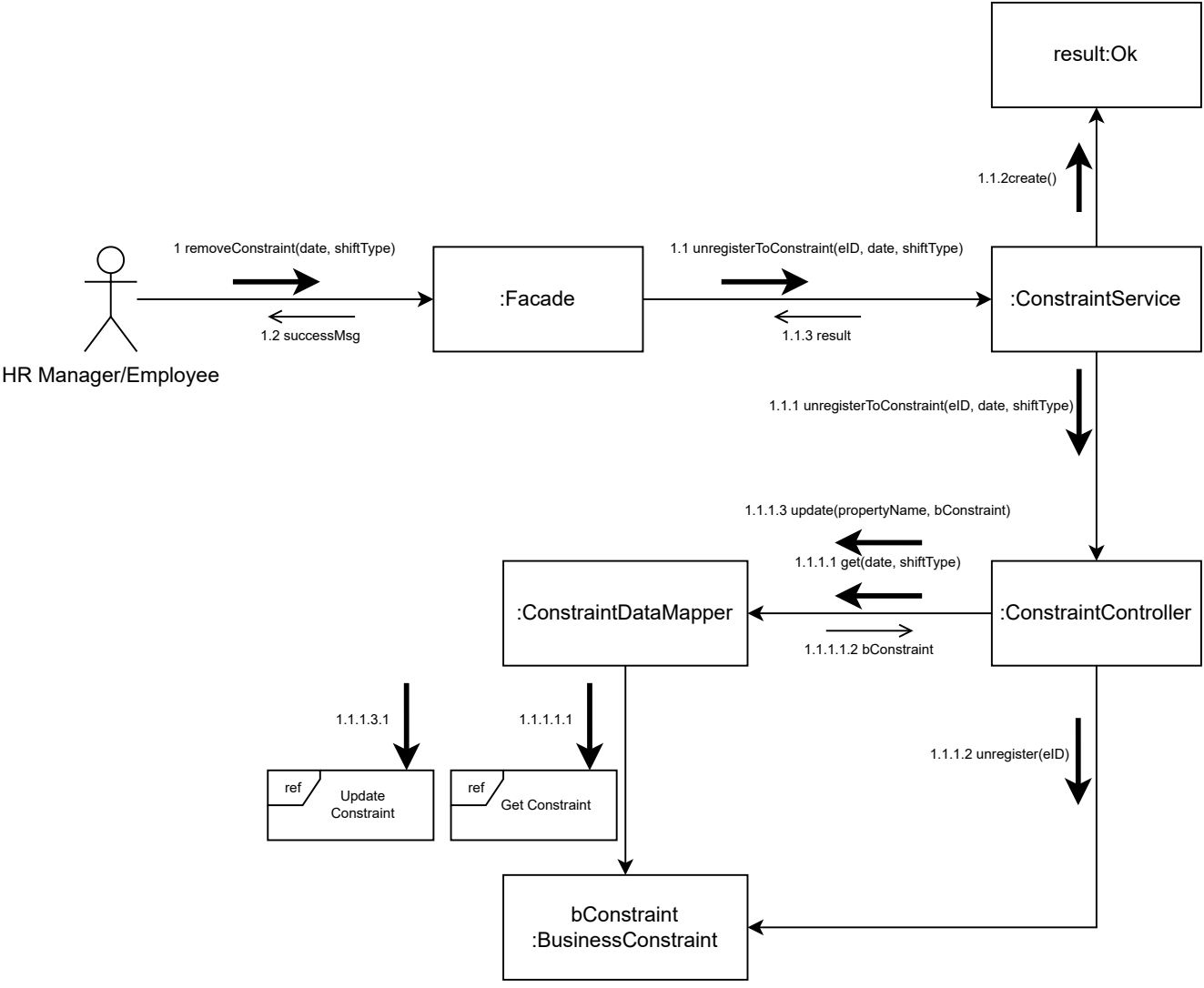


Removing Constraint



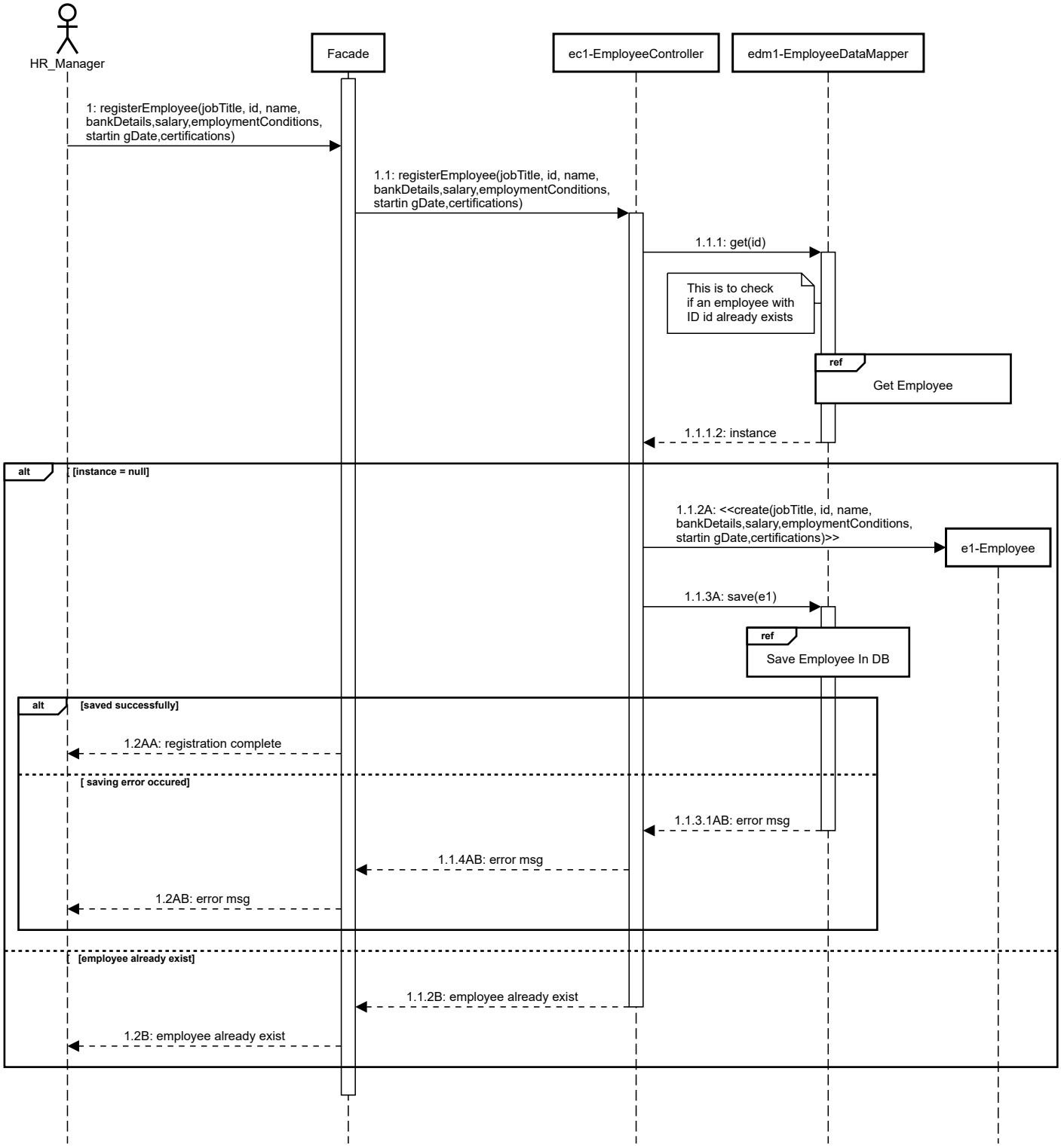

2B
ref
Removing Constraint
2A
ref
Adding Constraint
1
ref
Opening Employee management menu
:Facade
HR Manager/Employee

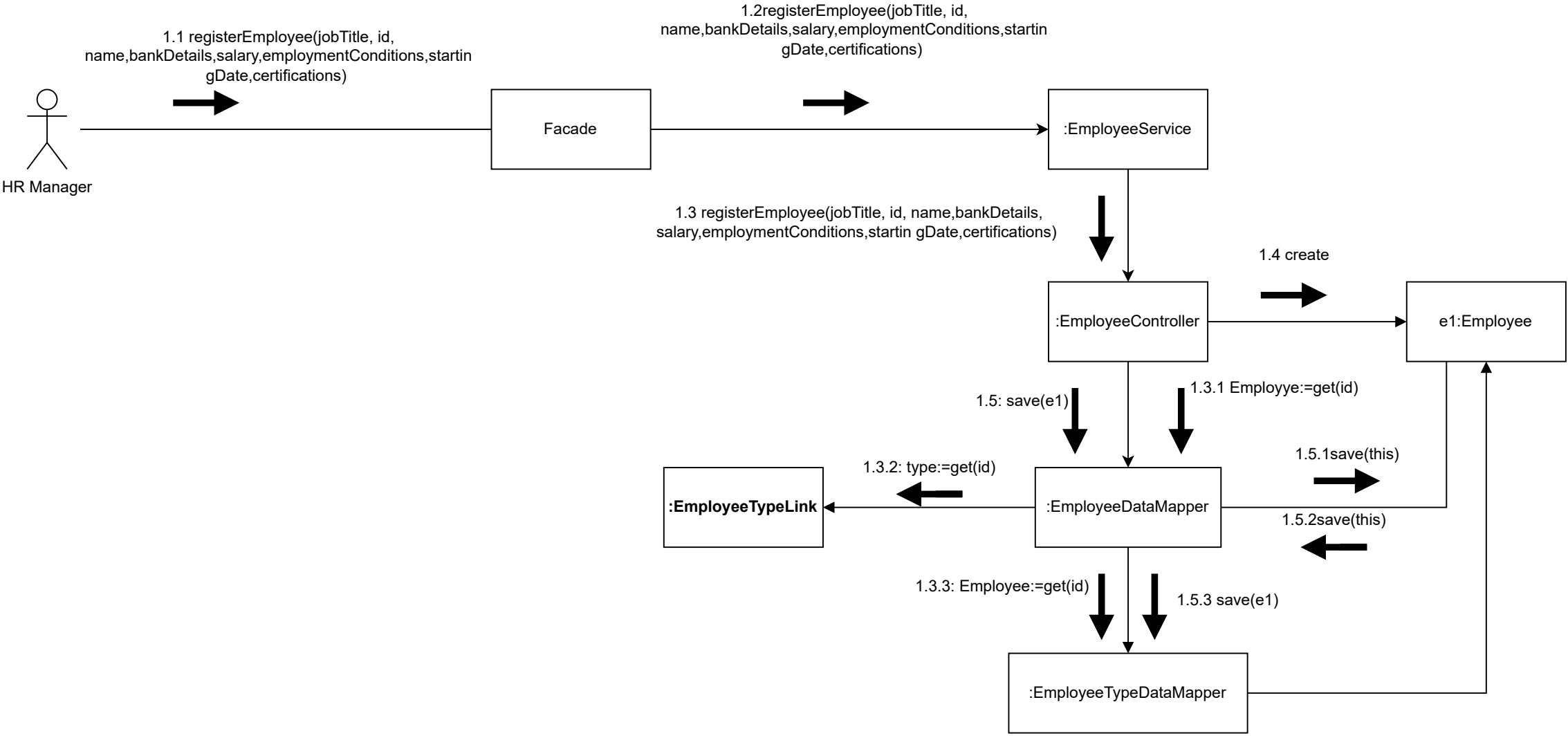

result:Ok
1.1.2create()
1 addConstraint(date, shiftType)
1.2 successMsg
:Facade
1.1 registerToConstraint(eID, date, shiftType)
1.1.3 result
:ConstraintService
HR Manager/Employee
1.1.1 registerToConstraint(eID, date, shiftType)
1.1.1.3 update(propertyName, bConstraint)
1.1.1.1 get(date, shiftType)
:ConstraintDataMapper
:ConstraintController
1.1.1.1.2 bConstraint
1.1.1.3.1
1.1.1.1.1
1.1.1.2 register(eID)
ref
Update Constraint
ref
Get Constraint
bConstraint
:BusinessConstraint


result:Ok
1.1.2create()
1 removeConstraint(date, shiftType)
1.2 successMsg
:Facade
1.1 unregisterToConstraint(eID, date, shiftType)
1.1.3 result
:ConstraintService
HR Manager/Employee
1.1.1 unregisterToConstraint(eID, date, shiftType)
1.1.1.3 update(propertyName, bConstraint)
1.1.1.1 get(date, shiftType)
:ConstraintDataMapper
:ConstraintController
1.1.1.1.2 bConstraint
1.1.1.3.1
1.1.1.1.1
1.1.1.2 unregister(eID)
ref
Update Constraint
ref
Get Constraint
bConstraint
:BusinessConstraint

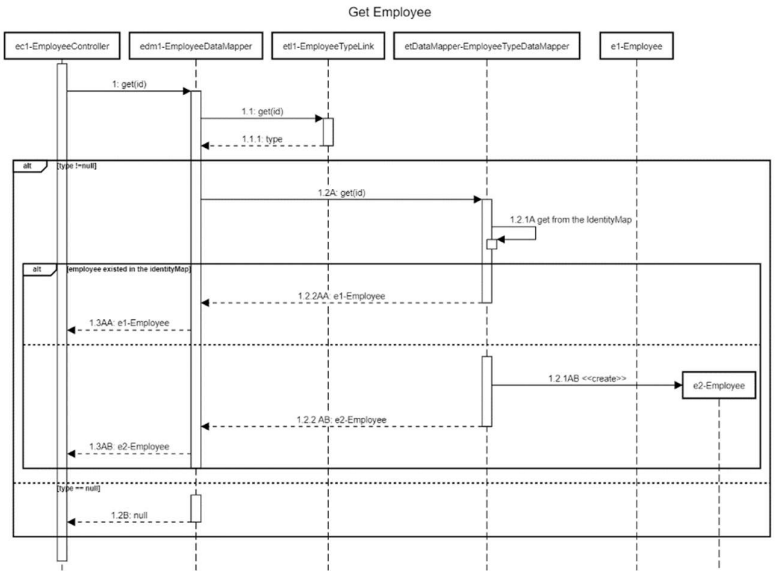
In order to register a new employee into the system the HR manager will input all details through dialog with the system. If the given ID exists in the system the process will fail, else the new employee will be saved in the system and in the DB.

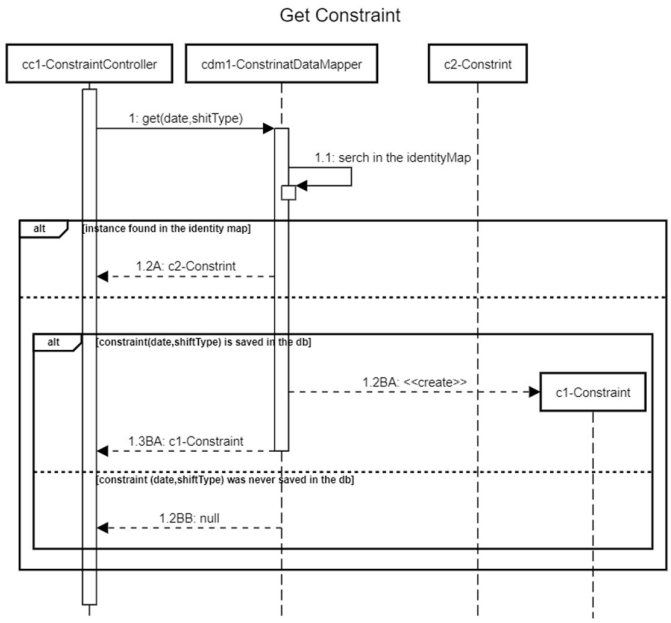
New Employee Registration

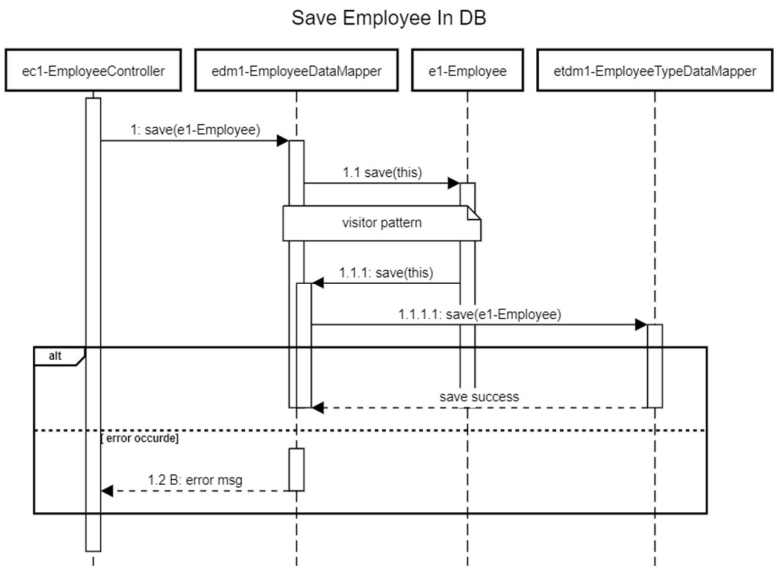


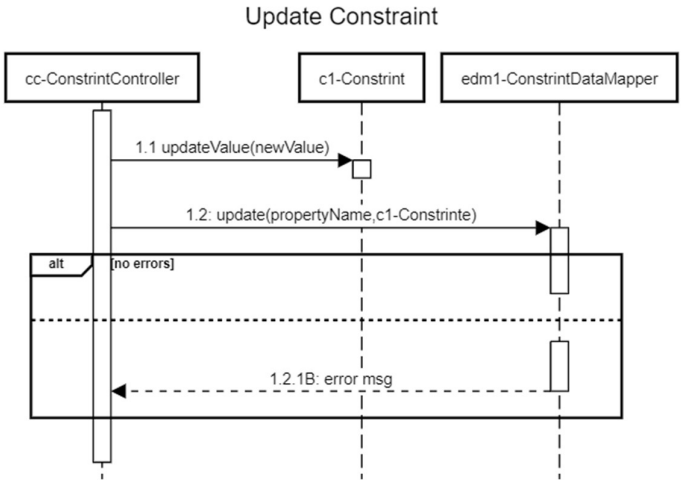

1.1 registerEmployee(jobTitle, id, name,bankDetails,salary,employmentConditions,startin gDate,certifications)
1.2registerEmployee(jobTitle, id, name,bankDetails,salary,employmentConditions,startin gDate,certifications)
HR Manager
Facade
:EmployeeService
1.3 registerEmployee(jobTitle, id, name,bankDetails, salary,employmentConditions,startin gDate,certifications)
1.4 create
:EmployeeController
e1:Employee
1.5: save(e1)
1.3.1 Employye:=get(id)
1.5.1save(this)
1.3.2: type:=get(id)
:EmployeeTypeLink
:EmployeeDataMapper
1.5.2save(this)
1.3.3: Employee:=get(id)
1.5.3 save(e1)
:EmployeeTypeDataMapper

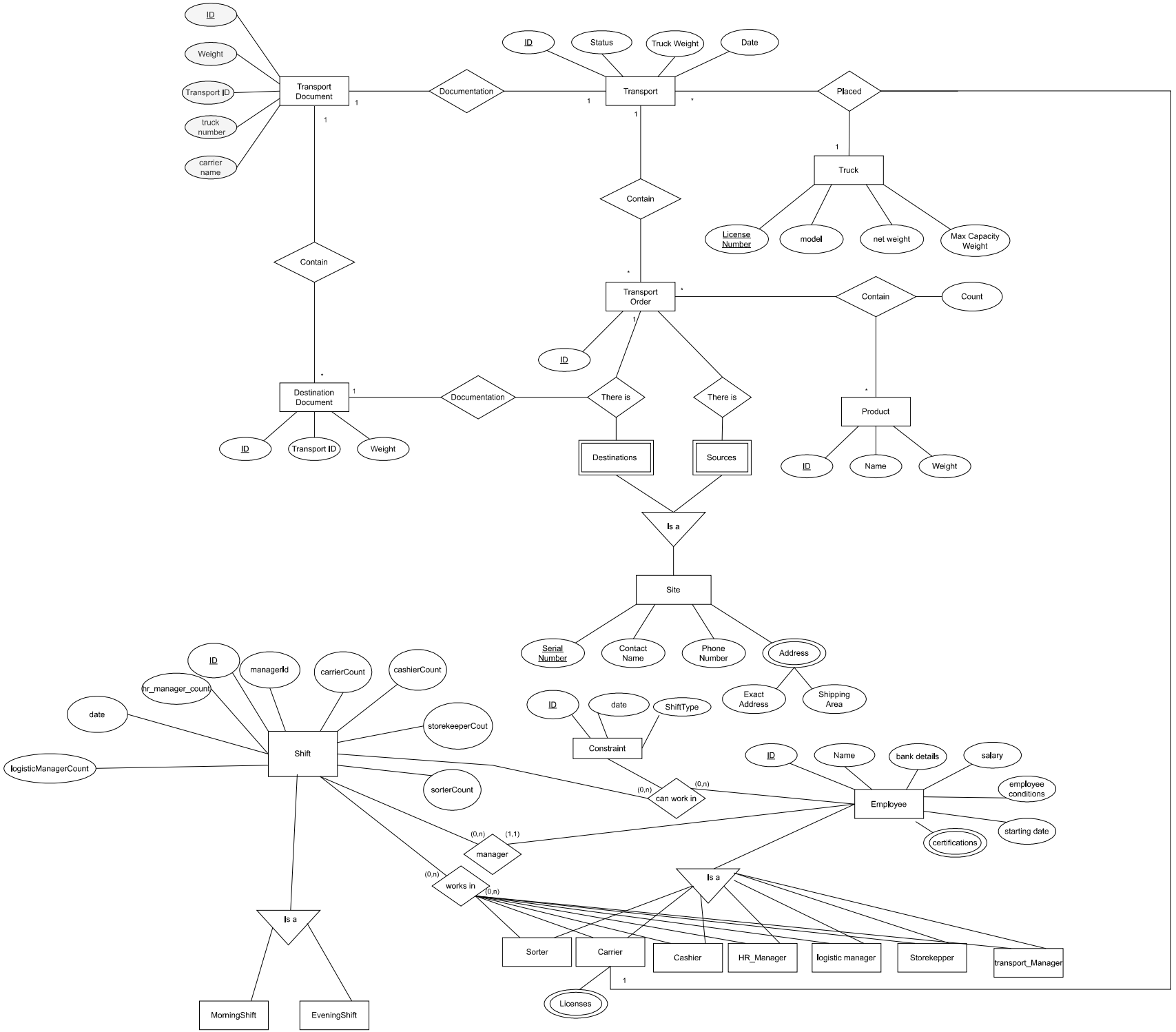
DAO processes:










ID
Weight
Transport ID
truck number
carrier name
Transport Document
1
Documentation
1
ID
Status
Truck Weight
Date
Transport
*
Placed
1
Truck
License Number
model
net weight
Max Capacity Weight
1
Contain
*
Transport Order
*
Contain
Count
1
ID
Contain
*
Destination Document
1
Documentation
There is
There is
*
Product
ID
Transport ID
Weight
Destinations
Sources
ID
Name
Weight
Is a
Site
Serial Number
Contact Name
Phone Number
Address
Exact Address
Shipping Area
ID
date
ShiftType
Constraint
ID
managerId
carrierCount
cashierCount
hr_manager_count
date
storekeeperCout
logisticManagerCount
Shift
sorterCount
(0,n)
(0,n)
can work in
ID
Name
bank details
salary
employee conditions
Employee
starting date
certifications
(0,n)
(1,1)
manager
(0,n)
works in
(0,n)
Is a
Is a
Sorter
Carrier
Cashier
HR_Manager
logistic manager
Storekepper
transport_Manager
1
Licenses
MorningShift
EveningShift

1. Functional Requirements

ID	Module	Description	Priority	Risk	Status	Feasable
1.	HR	The system **MUST** support management of employees. Creation: Employee can be 1 of any employee type Updatable information: Name, Salary, Bank Details, Certifications.	MH	Low	Done	Yes
2.	HR	The system **CANNOT** allow creation of 2 employees with the same ID.	MH	Low	Done	Yes
3.	HR	The system **MUST** support management of licenses for carriers.	MH	Low	Done	Yes
4.	HR	The system **MUST** store employee information on a DB	MH	Low	Done	Yes
5.	HR	The system **MUST** support management of shifts. Updatable information: shift manager, counts, assigned employee IDs	MH	High	Done	Yes
6.	HR	The system **CANNOT** allow creation of shifts with no shift manager	MH	Low	Done	Yes
~~7.~~	~~HR~~	~~The system **SHOULD** be allow setting of how many workers of each type each shift needs~~	~~NTH~~	~~Low~~	~~Done~~	~~Yes~~
8.	HR	The system **CANNOT** allow creation of shifts with no carriers, cashiers, storekeepers or sorters	MH	High	In Progress	Yes
9.	HR	The system **MUST** show which employees are available for each shift when scheduling	MH	High	Done	Yes
10.	HR	The system **SHOULD** show how many shifts each available employee has done the past month for each shift type	NTH	High	Done	Yes
11.	HR	The system **MUST** store shift history in DB	MH	Low	Done	Yes
12.	HR	The system **SHOULD** notify about upcoming incomplete shifts	NTH	Low	In Progress	Yes
13.	Suppliers	The system **MUST** support 3 kinds of suppliers (routine, by order, not transporting)	MH	Low	Done	Yes
14.	Suppliers	The system **MUST** save the supplier's information (id, bank account, paying agreement, contacts and contact's information)	MH	Low	Done	Yes
15.	Suppliers	The system **MUST** save if the supplier has fixed supplying days and when they are.	MH	Low	Done	Yes
16.	Suppliers	The system **MUST** save details about the supplied items (cost per item, cost when buying in bulk)	MH	Low	Done	Yes
17.	Suppliers	The system **MUST** show what items are purchased from each supplier	MH	Low	Done	Yes
18.	Suppliers	The system **MUST** give the corporate management the ability to edit the agreement's details (supplying days, prices, changing items)	NTH	Low	Done	Yes

19.	Inventory	The system **MUST** store the following product info: product ID, item name, category, price, weight, and manufacturer	MH	Low	Done	Yes
20.	Inventory	The system **MUST** store the following information about stock per each product: product ID, amount in each storefront, amount in each store warehouse, and each store's min amount	MH	Low	Done	Yes
21.	Inventory	The system **MUST** store the following information about sales per each product and category: active sales, passed sales, future sales, percent off in sale	MH	Low	Done	Yes
22.	Inventory	The system **MUST** send an alert when product is getting under the minimum amount in a certain branch. specifying product ID and store ID	MH	Low	Done	Yes
23.	Inventory	The system **MUST** allow employees to change the minimum and maximum value of each product in a branch.	MH	Low	Done	Yes
24.	Inventory	The system **MUST** be able to produce a report of all products in all branches that are below their minimum amount. The report should include the following: store ID, product ID, product name, product current amount in store, product current amount in warehouse, product current amount in the branch (store+warehouse), product minimum amount in the branch, product maximum amount in the branch	MH	Low	Done	Yes
25.	Inventory	The system **MUST** track amounts of product left in each location (each store and each warehouse), including: 1) buying product 2) removing damaged/expired products 3) moving products from the warehouse to the store 4) product has been returned to the store from a customer 5) product has been arrived from supplier.	MH	Low	Done	Yes
26.	Inventory	The system **MUST** be able to produce a report of all products in a chosen branch based on categories. The report should include the following: store ID, product ID, product name, category, product current amount in store, product current amount in warehouse, product current amount in the branch (store+warehouse), product minimum amount in the branch, product maximum amount in the branch.	MH	Low	Done	Yes
27.	Inventory	The system **SHOULD** be able to produce the history of the completed purchases from the suppliers, for a certain product. The history should include the following: StoreID, Supplier, Date, Amount, before discount price, after discount price, description.	NTH	Low	Done	Yes
28.	Inventory	The system **MUST** be able to produce the history of the completed purchases from the suppliers, for a certain product that was bought with a discount. The history should include the following: StoreID, Supplier, Date, Amount, before discount price, after discount price, description.	MH	Low	Done	Yes
29.	Inventory	The system **MUST** be able to create/cancel sales that apply to certain products and/or categories to specified dates. Sales include: sale ID, percentage, start date, end date, Products IDs, categories IDs.	MH	Low	Done	Yes
30.	Inventory	The system **MUST** be able to produce the history of the sales to customers by product or by category. The history should include the following: sale ID, percentage, start date,	MH	Low	Done	Yes

		end date, Products IDs, categories IDs.				
31.	Inventory	The system **MUST** be able to manage category (create, read, update, delete): update: change parent category, change name, change products in category	MH	Low	Done	Yes
32.	Inventory	The system **MUST** be able to manage product (create, read, update, delete): update: change category, change name, change price, add to a certain store, remove from a certain store, add new supplier to product, remove supplier from product	MH	Low	Done	Yes
33.	Inventory	The system **MUST** allow employees to report about damaged/expired products. The user will insert the following information: store ID, product ID, amout, user ID, description.	MH	Low	Done	Yes
34.	Inventory	The system **MUST** be able to supply a report of damaged/expired products needs to be able to be sorted by: store, product, category.	MH	Low	Done	Yes
35.	Inventory	The system **MUST** be able to produce a report of damaged products/expired products/both products which were reported on certain range of dates. The report must include the following information: product ID, store ID, amount, user ID, description, date, expired/damaged.	MH	Low	Done	Yes
36.	Inventory	The system **MUST** be able to add and to remove branches to and from the chain.	MH	Low	Done	Yes
37.	Suppliers & Inventory	The system **MUST** support automatic orders, initiated whenever stock running low.	MH	Low	Done	Yes
38.	Suppliers & Inventory	The system **MUST** support automatic orders in fixed times (routine).	MH	Low	Done	Yes
39.	Suppliers & Inventory	The system **MUST** support that orders cannot be edited in the last 24 hours before the order's planned deliveries.	MH	Low	Done	Yes
40.	Suppliers & Inventory	The system **MUST** support choosing the best supplier for a specific order, choosing the cheapest one.	MH	Low	Done	Yes
41.	Suppliers & Inventory	The system **MUST** store the following order information: store id, Supplier id, Products ids, Products amounts, arrival time, original price, discount and final price.	MH	Low	Done	Yes
42.	HR & Transport	The system **MUSN'T** allow carrying out transport when no storekeepers are present at the transport arrival time.	MH	High	In Progress	Yes
43.	Transport	The system **MUST** support management of a transports documents in the transport management database.	MH	Low	Done	Yes
44.	Transport	The system **MUST** provide to the carrier a destination document for each destination.	MH	Low	In Progress	Yes
45.	Transport	The system **MUST** warn if the actual weight exceeds the maximum weight.	MH	Low	Done	Yes
46.	HR & Transport	The system **MUSN'T** allow placement of a carrier without a proper license for the truck selected for transportation.	MH	Low	Done	Yes

47.	Transport	The system **MUST** store the following transport info: transport ID, date, time of departure, truck number, driver name, weight of the truck, source and destinations.	MH	Low	Done	Yes
48.	Transport	The system **MUST** store the following destination document info: document ID, transport ID, list of products provided in this destination.	MH	Low	Done	Yes
49.	Transport	The system **MUST** store the following site info: site ID, address, phone number and name of the contact person.	MH	Low	Done	Yes
50.	Transport	The system **MUST** store the following truck info: license number, model, net weight, maximum capacity weight.	MH	Low	Done	Yes
51.	Transport	The system **MUST** support redesign the route in case of weight deviation.	MH	High	In Progress	Yes
52.	Transport	The system **MUST** update the transport document about the redesign of the route.	MH	Low	Done	Yes
53.	Transport	The system **SHOULD** support management of a trucks system.	NTH	Low	Done	Yes
54.	Transport	The system **MUST** store the following transport document info: date, time of departure, truck number, driver name, weight of the truck, source, destinations and about redesign if done.	MH	Low	Done	Yes
55.	Transport	The system **MUST** support the redesign of downloading or replacing one of the destinations, or replacing a truck, or removing some of the products from the truck.	MH	High	In Progress	Yes
56.	Transport	The system **MUST** provide delivery of a product when there is a shortage.	MH	High	Backlog	No

2. Non-Functional Requirements

ID	Module	Description	Priority	Risk	Status	Feasible
1.	Suppliers	The system **SHOULD** have the ability to add/remove/edit/watch a supplier	MH	Low	Done	Yes
2.	Inventory	The system **MUST** support each product belonging to exactly one category, and each category having exactly one parent category or none, but can have any number of sub-categories. (sub-category and parent category are in a two-way connection)	MH	Low	Done	Yes
3.	Transport	The system **MUST** support weigh of the truck before the truck leaving the source.	MH	Low	Done	Yes

Appendix A Terms

Table 1: Terms

#	Term	Description
1.	Management	Set of action which include creation, reading, editing (information) and removal of defined class.
2.	Employee	Person with work contract for working for ‘Superly’
3.	Employee information	ID, Name, Bank details, Salary, Employment conditions, Starting date, Certifications
4.	Employee types	Cashier, Storekeeper, Carrier, Sorter, HR Manager, Logistics Manager, Transport Manager
5.	Shift	Concept used to describe a fixed work period.
6.	Shift Information	Date, Shift Type, Shift Manager’s ID, Count of Employees Needed for each Employee Type, Assigned Employee ID’s for each Employee Type
7.	Shift types	Morning shift, Evening shift
8.	Constraint	Concept used to describe when an Employee is available for work
9.	Constraint Information	Date, Shift Type, Available Employees for the Shift
10.	Target amount	The number of items needed for certain product in a specific store after it gets under its minimum amount.
11.	License types	B, C, C1, C + E
12.	Truck model	Van, Semi-Trailer, Double-Trailer, Full-Trailer.
13.	Shipping areas	South, North, East, West, Southwest, Southeast, Northwest, Northeast.
14.	Transport	Delivering products from supplier to destination.
15.	Redesign	When the transport exceeds the possible weight for it the transport returns to redesign.
16.	Transport Document	A document describing the transport after it is completed.
17.	Destination Document	A document describing the products delivered in a particular shipment to the destination.
18.	Shortage	When inventory runs out and exceeds the minimum quantity set for the product.

Appendix B Open Questions

Table 3: Questions

#	Topic	Issue
1.	Document	Will there be an additional need for the documents?
2.		
3.		

Appendix C Assumptions

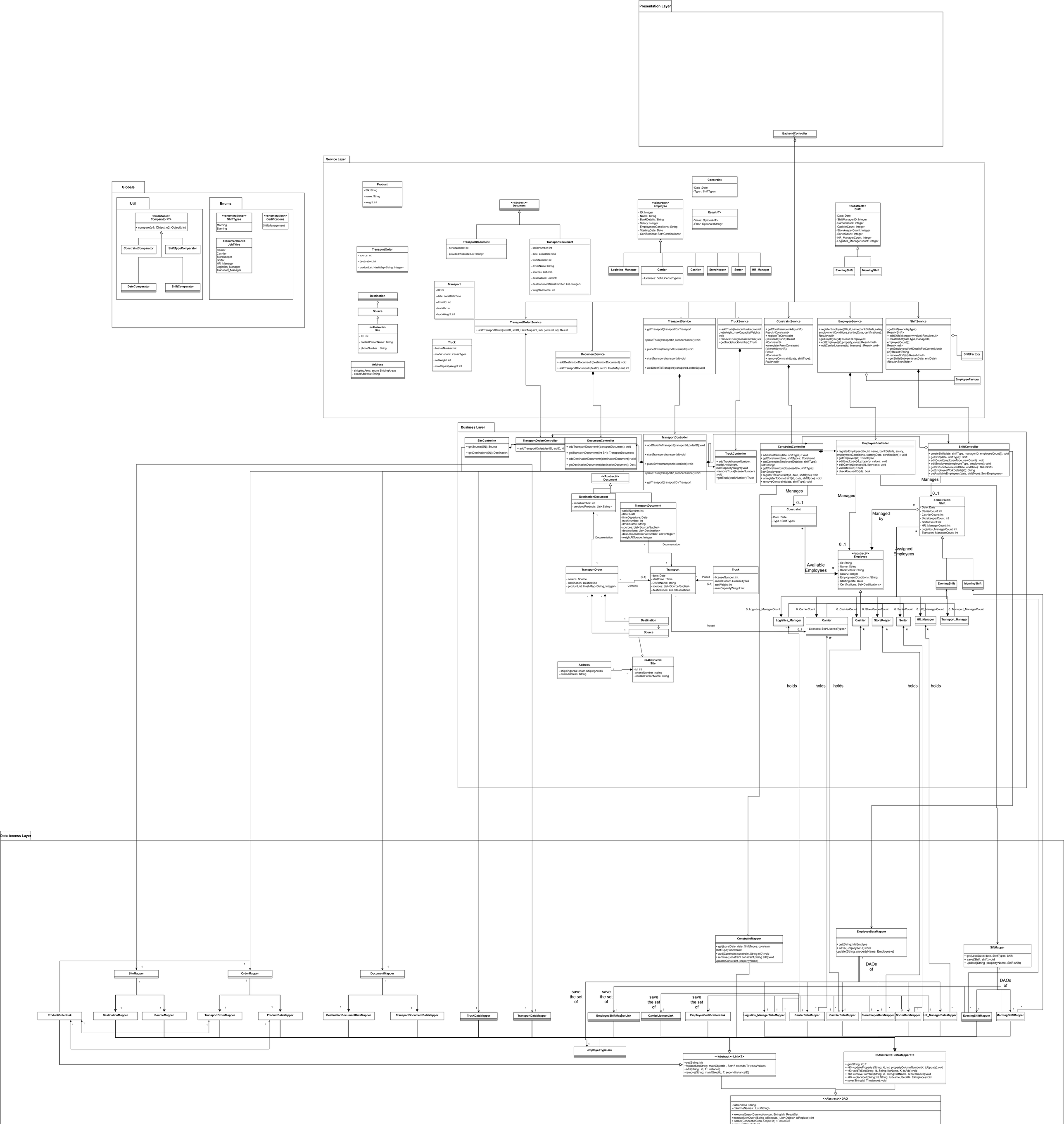
Table 2: Assumptions

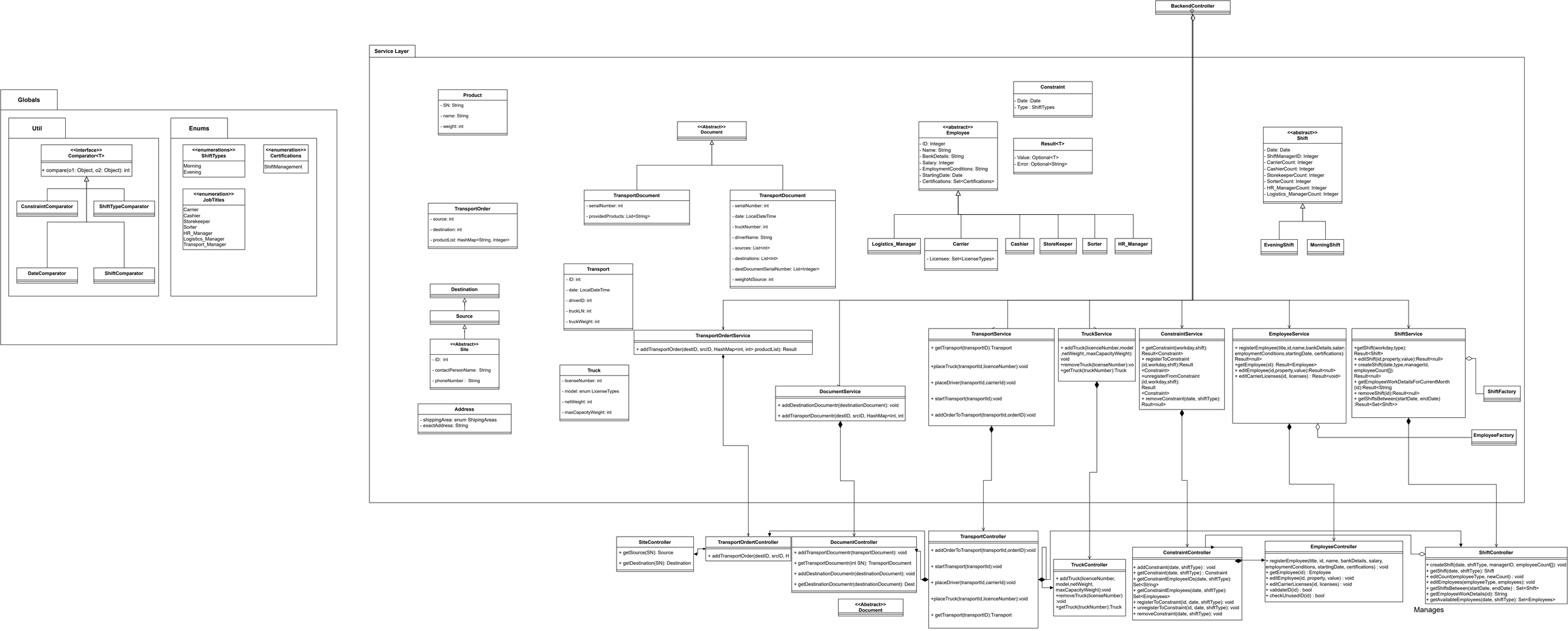
#	Topic	Assumption
1.	Suppliers	The contact's info is only phone number and name.
2.	Suppliers	We can't change orders from by order suppliers in the last 24 hours before the shipment.
3.	Suppliers	A supplier has only one type of agreement, from : by order, routine, not transporting.
4.	Suppliers	When stocks running low, we choose the cheapest supplier to order from, not considering his arrival time.
5.	Suppliers	We can add new supplier for the system without agreement. (Maybe we want just to save his information for future deals).
6.	Suppliers	We assume that when adding an item to agreement, the supplier manager knows both the id this item has at the supplier catalog and the id this item has at inventory.
7.	Inventory	There is predefined target amount per product per store which is above the min amount, such that when creating an order because of low stock the order amount should be target-min
8.	Suppliers+Inventory	The System is shut down at the end of each workday, and turned on at the beginning of the day. At the beginning of each day we send the suppliers the updated amounts for the next day's orders if needed.
9.	Suppliers+Inventory	When a delivery arrives at the warehouse, the orderID and supplierID is known (attached to the driver's receipt)
10.	Suppliers+Inventory	Each supplier can have different ID for the same product (For example: Bamba can have ID 1 in Superly, ID 5 in supplier 1 and ID 1 in supplier 2)
11.	Suppliers	The "best" supplier is the cheapest one regardless to arriving time.
12.	Suppliers+Inventory	Printing an order on screen equals to sending the order to the corresponding supplier.
13.	Suppliers+Inventory	Sending an order that has already been sent to a supplier equals to informing the supplier about updating the order.

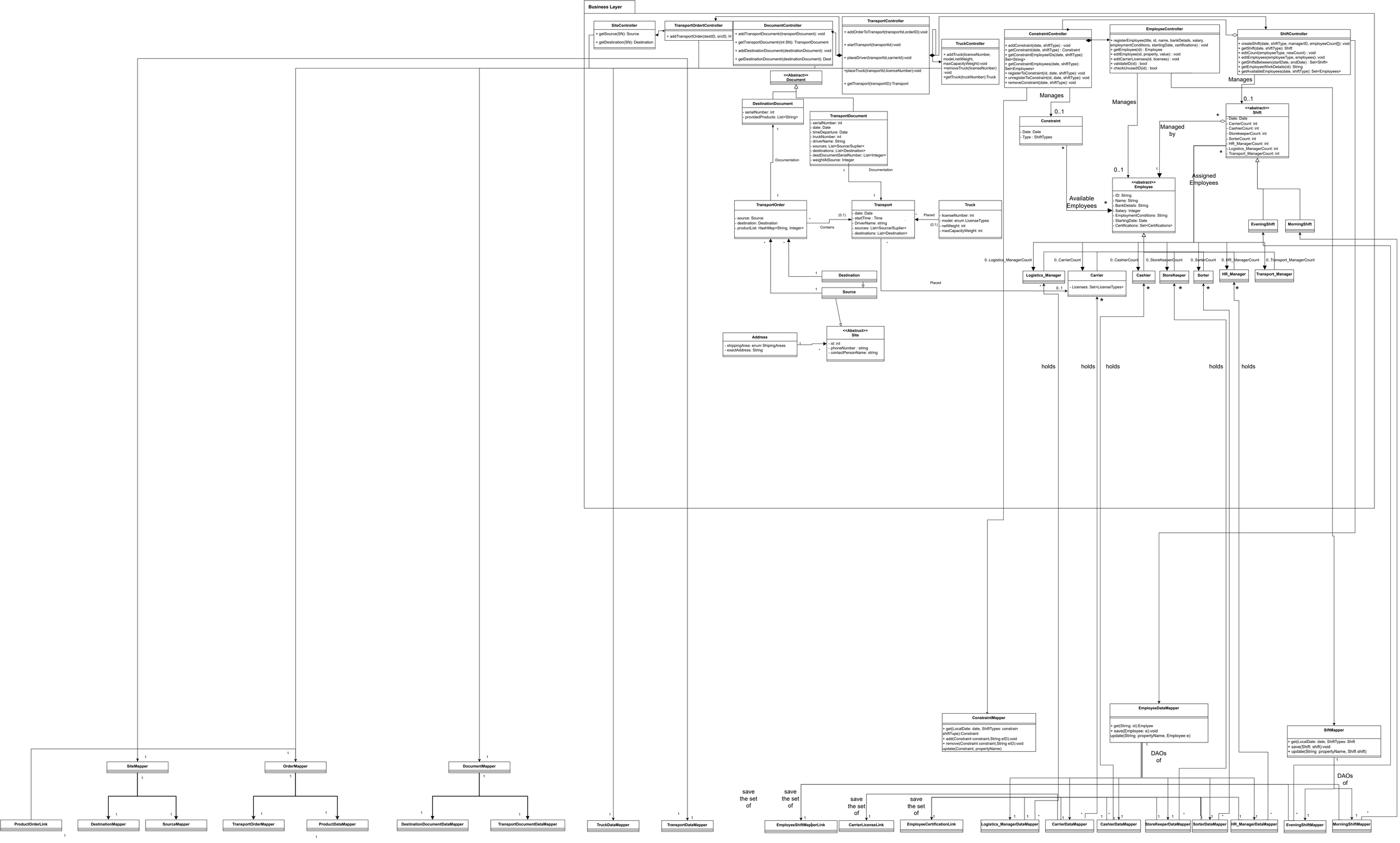
Version History

Table 2: Version History

Date	Author	Remarks
25/04/2022	Roi Tiefenbrunn	First draft
09/05/2022	Yonatan Diga	Added suppliers & Inventory requirements and suppliers assumptions.
11/05/2022	Yonatan Diga	Improved requirements
14/05/2022	Tomer Ravkaie	Improved Requirements and Assumptions
15/05/2022	Roi Tiefenhbrunn	Deleting of non-functional reqs for HR-module Adding term ‘Employee’, ‘Employee Information’, ‘Employee types’, ‘Shift’, ‘Shift Information’, ‘Shift Types’ Adding requirements 12, 42
15/05/2022	Roi Tiefenbrunn	Completing Requirements 1, 4, 5, 11 Adding Requirement 2 Adding the terms ‘Constraint’, ‘Constraint Information’ Removed Requirement 7
15/05/2022	Chai-Shalev Hadad	Add transport requirements. Fix the incorrect requirements. Add terms and open question.


Presentation Layer
BackendController
Service Layer
Globals
Util
<<interface>>
Comparator<T>
+ compare(o1: Object, o2: Object): int
ConstraintComparator
ShiftTypeComparator
DateComparator
ShiftComparator
Enums
<<enumerations>>
ShiftTypes
Morning
Evening
<<enumeration>>
Certifications
ShiftManagement
<<enumeration>>
JobTitles
Carrier
Cashier
Storekeeper
Sorter
HR_Manager
Logistics_Manager
Transport_Manager
Product
- SN: String
- name: String
- weight: int
<<Abstract>>
Document
TransportDocument
- serialNumber: int
- providedProducts: List<String>
TransportDocument
- serialNumber: int
- date: LocalDateTime
- truckNumber: int
- driverName: String
- sources: List<int>
- destinations: List<int>
- destDocumentSerialNumber: List<Integer>
- weightAtSource: int
TransportOrder
- source: int
- destination: int
- productList: HashMap<String, Integer>
Transport
- ID: int
- date: LocalDateTime
- driverID: int
- truckLN: int
- truckWeight: int
Destination
Source
<<Abstract>>
Site
- ID: int
- contactPersonName: String
- phoneNumber : String
Address
- shippingArea: enum ShipingAreas
- exactAddress: String
Truck
- licenseNumber: int
- model: enum LicenseTypes
- netWeight: int
- maxCapacityWeight: int
<<abstract>>
Employee
- ID: Integer
- Name: String
- BankDetails: String
- Salary: Integer
- EmploymentConditions: String
- StartingDate: Date
- Certifications: Set<Certifications>
Logistics_Manager
Carrier
- Licenses: Set<LicenseTypes>
Cashier
StoreKeeper
Sorter
HR_Manager
Constraint
- Date :Date
- Type : ShiftTypes
Result<T>
- Value: Optional<T>
- Error: Optional<String>
<<abstract>>
Shift
- Date: Date
- ShiftManagerID: Integer
- CarrierCount: Integer
- CashierCount: Integer
- StorekeeperCount: Integer
- SorterCount: Integer
- HR_ManagerCount: Integer
- Logistics_ManagerCount: Integer
EveningShift
MorningShift
TransportOrdertService
+ addTransportOrder(destID, srcID, HashMap<int, int> productList): Result
DocumentService
+ addDestinationDocumentr(destinationDocument): void
+ addTransportDocumentr(destID, srcID, HashMap<int, int
TransportService
+ getTransport(transportID):Transport
+placeTruck(transportId,licenceNumber):void
+ placeDriver(transportId,carrierId):void
+ startTransport(transportId):void
+ addOrderToTransport(transportId,orderID):void
TruckService
+ addTruck(licenceNumber,model,netWeight,maxCapacityWeight):void
+removeTruck(licenseNumber):vo
+getTruck(truckNumber):Truck
ConstraintService
+ getConstraint(workday,shift): Result<Constraint>
+ registerToConstraint (id,workday,shift):Result
<Constraint>
+unregisterFromConstraint (id,workday,shift): Result<Constraint>
+ removeConstraint(date, shiftType): Rsult<null>
EmployeeService
+ registerEmployee(title,id,name,bankDetails,salary,employmentConditions,startingDate, certifications): Result<null>
+getEmployee(id): Result<Employee>
+ editEmployee(id,property,value):Result<null>
+ editCarrierLicenses(id, licenses) : Result<void>
ShiftService
+getShift(workday,type): Result<Shift>
+ editShift(id,property,value):Result<null>
+ createShift(date,type,manager!d, employeeCount[]): Result<null>
+ getEmployeeWorkDetailsForCurrentMonth (id):Result<String
+ removeShift(id):Result<null>
+ getShiftsBetween(startDate, endDate) :Result<Set<Shift>>
ShiftFactory
EmployeeFactory
Business Layer
SiteController
+ getSource(SN): Source
+ getDestination(SN): Destination
TransportOrderController
+ addTransportOrder(destID, srcID, H
DocumentController
+ addTransportDocumentr(transportDocument): void
+ getTransportDocumentr(int SN): TransportDocument
+ addDestinationDocumentr(destinationDocument): void
+ getDestinationDocumentr(destinationDocument): Dest
TransportController
+ addOrderToTransport(transportId,orderID):void
+ startTransport(transportId):void
+ placeDriver(transportId,carrierId):void
+placeTruck(transportId,licenceNumber):void
+ getTransport(transportID):Transport
TruckController
+ addTruck(licenceNumber, model,netWeight, maxCapacityWeight):void
+removeTruck(licenseNumber) :void
+getTruck(truckNumber):Truck
ConstraintController
+ addConstraint(date, shiftType) : void
+ getConstraint(date, shiftType) : Constraint
+ getConstraintEmployeeDa(date, shiftType): Set<String>
+ getConstraintEmployees(date, shiftType): Set<Employees>
+ registerToConstraint(id, date, shiftType): void
+ unregisterToConstraint(id, date, shiftType): void
+ removeConstraint(date, shiftType): void
EmployeeController
+ registerEmployee(title, id, name, bankDetails, salary, employmentConditions, startingDate, certifications) : void
+ getEmployee(id) : Employee
+ editEmployee(id, property, value) : void
+ editCarrierLicenses(id, licenses) : void
+ validateID(id) : bool
+ checkUnusedID(id) : bool
ShiftController
+ createShift(date, shiftType, managerID, employeeCount[]): void
+ getShift(date, shiftType): Shift
+ editCount(employeeType, newCount) : void
+ editEmployees(employeeType, employees): void
+ getShiftsBetween(startDate, endDate) : Set<Shift>
+ getEmployeeWorkDetails(id): String
+ getAvailableEmployees(date, shiftType): Set<Employees>
Manages
0..1
Manages
0..1
Manages
Managed by
*
Assigned Employees
0..1
1
Available Employees
*
Placed
(0,1)
Contains
{0,1}
Documentation
1
Documentation
1
Placed
0..Logistics_ManagerCount
0..CarrierCount
0..CashierCount
0..StoreKeeperCount
0..SorterCount
0..HR_ManagerCount
0..Transport_ManagerCount
holds
holds
holds
holds
holds
Data Access Layer
SiteMapper
OrderMapper
DocumentMapper
ProductOrderLink
DestinationMapper
SourceMapper
TransportOrderMapper
ProductDataMapper
DestinationDocumentDataMapper
TransportDocumentDataMapper
TruckDataMapper
TransportDataMapper
ConstraintMapper
+ get(LocalDate: date, ShiftTypes: constrain shiftTupe):Constraint
+ add(Constraint constraint,String eID):void
+ remove(Constraint constraint,String eID):void
update(Constraint, propertyName)
EmployeeDataMapper
+ get(String: id):Emplyee
+ save(Employee: e):void
update(String :propertyName, Employee e)
SiftMapper
+ get(LocalDate: date, ShiftTypes: Shift
+ save(Shift: shift):void
+ update(String :propertyName, Shift shift)
DAOs of
DAOs of
save the set of
save the set of
save the set of
save the set of
EmployeeShiftMapperLink
CarrierLicenseLink
EmployeeCertificationLink
Logistics_ManagerDataMapper
CarrierDataMapper
CashierDataMapper
StoreKeeperDataMapper
SorterDataMapper
HR_ManagerDataMapper
EveningShiftMapper
MorningShiftMapper
employeeTypeLink
<<Abstract>> Link<T>
+get(String: id)
+replaceSet(String: mainObjectId , Set<? extends T>) :newValues
+add(String : id, T : instance)
+remove(String: mainObjectId, T: secondInstanceID)
<<Abstract>> DataMapper<T>
+ get(String: id):T
+ <K> updateProperty (String: id, int: propertyColumnNumber,K: toUpdate):void
+ <K> addToSet(String: id, String: listName, K: toAdd):void
+ <K> removeFromSet(String: id, String: listName, K: toRemove):void
+ <K> replaceSet(String: id, String: listName, Set<K>: toReplace):void
+ save(String id, T instance): void
<<Abstract>> DAO
- tableName :String
- columnsNames : List<String>
+ executeQuery(Connection con, String id): ResultSet
+executeNonQuery(String toExecute, List<Object> toReplace) :int
+ select(Connection con, Object id) : ResultSet
+remove(Object id): int
+update(List<Integer> columnsLocationToUpdate,List<Object> valuesToUpdate,List<Integer> conditionColumnLocation,List<Object> conditionValues)


Globals
Util
<<interface>>
Comparator<T>
+ compare(o1: Object, o2: Object): int
ConstraintComparator
ShiftTypeComparator
DateComparator
ShiftComparator
Enums
<<enumerations>>
ShiftTypes
Morning
Evening
<<enumeration>>
Certifications
ShiftManagement
<<enumeration>>
JobTitles
Carrier
Cashier
Storekeeper
Sorter
HR_Manager
Logistics_Manager
Transport_Manager
Service Layer
BackendController
Product
- SN: String
- name: String
- weight: int
<<Abstract>>
Document
TransportDocument
- serialNumber: int
- providedProducts: List<String>
TransportDocument
- serialNumber: int
- date: LocalDateTime
- truckNumber: int
- driverName: String
- sources: List<int>
- destinations: List<int>
- destDocumentSerialNumber: List<Integer>
- weightAtSource: int
Constraint
- Date :Date
- Type : ShiftTypes
<<abstract>>
Employee
- ID: Integer
- Name: String
- BankDetails: String
- Salary: Integer
- EmploymentConditions: String
- StartingDate: Date
- Certifications: Set<Certifications>
Result<T>
- Value: Optional<T>
- Error: Optional<String>
<<abstract>>
Shift
- Date: Date
- ShiftManagerID: Integer
- CarrierCount: Integer
- CashierCount: Integer
- StorekeeperCount: Integer
- SorterCount: Integer
- HR_ManagerCount: Integer
- Logistics_ManagerCount: Integer
TransportOrder
- source: int
- destination: int
- productList: HashMap<String, Integer>
Logistics_Manager
Carrier
- Licenses: Set<LicenseTypes>
Cashier
StoreKeeper
Sorter
HR_Manager
EveningShift
MorningShift
Transport
- ID: int
- date: LocalDateTime
- driverID: int
- truckLN: int
- truckWeight: int
Destination
Source
<<Abstract>>
Site
- ID: int
- contactPersonName: String
- phoneNumber : String
Address
- shippingArea: enum ShipingAreas
- exactAddress: String
Truck
- licenseNumber: int
- model: enum LicenseTypes
- netWeight: int
- maxCapacityWeight: int
TransportOrdertService
+ addTransportOrder(destID, srcID, HashMap<int, int> productList): Result
DocumentService
+ addDestinationDocumentr(destinationDocument): void
+ addTransportDocumentr(destID, srcID, HashMap<int, int
TransportService
+ getTransport(transportID):Transport
+placeTruck(transportId,licenceNumber):void
+ placeDriver(transportId,carrierId):void
+ startTransport(transportId):void
+ addOrderToTransport(transportId,orderID):void
TruckService
+ addTruck(licenceNumber,model ,netWeight,,maxCapacityWeight): void
+removeTruck(licenseNumber):vo
+getTruck(truckNumber):Truck
ConstraintService
+ getConstraint(workday,shift): Result<Constraint>
+ registerToConstraint (id,workday,shift):Result <Constraint>
+unregisterFromConstraint (id,workday,shift): Result <Constraint>
+ removeConstraint(date, shiftType): Rsult<null>
EmployeeService
+ registerEmployee(title,id,name,bankDetails,salary, employmentConditions,startingDate, certifications): Result<null>
+getEmployee(id): Result<Employee>
+ editEmployee(id,property,value):Result<null>
+ editCarrierLicenses(id, licenses) : Result<void>
ShiftService
+getShift(workday,type): Result<Shift>
+ editShift(id,property,value):Result<null>
+ createShift(date,type,managerId, employeeCount[]): Result<null>
+ getEmployeeWorkDetailsForCurrentMonth ((id):Result<String
+ removeShift(id):Result<null>
+ getShiftsBetween(startDate, endDate) :Result<Set<Shift>>
ShiftFactory
EmployeeFactory
SiteController
+ getSource(SN): Source
+ getDestination(SN): Destination
TransportOrdertController
+ addTransportOrder(destID, srcID, H
DocumentController
+ addTransportDocumentr(transportDocument): void
+ getTransportDocumentr(int SN): TransportDocument
+ addDestinationDocumentr(destinationDocument): void
+ getDestinationDocumentr(destinationDocument): Dest
<<Abstract>>
Document
TransportController
+ addOrderToTransport(transportId,orderID):void
+ startTransport(transportId):void
+ placeDriver(transportId,carrierId):void
+placeTruck(transportId,licenceNumber):void
+ getTransport(transportID):Transport
TruckController
+ addTruck(licenceNumber, model,netWeight, maxCapacityWeight):void
+removeTruck(licenseNumber) :void
+getTruck(truckNumber):Truck
ConstraintController
+ addConstraint(date, shiftType) : void
+ getConstraint(date, shiftType) : Constraint
+ getConstraintEmployeeIDs(date, shiftType): Set<String>
+ getConstraintEmployees(date, shiftType): Set<Employees>
+ registerToConstraint(id, date, shiftType): void
+ unregisterToConstraint(id, date, shiftType): void
+ removeConstraint(date, shiftType): void
EmployeeController
+ registerEmployee(title, id, name, bankDetails, salary, employmentConditions, startingDate, certifications) : void
+ getEmployee(id) : Employee
+ editEmployee(id, property, value) : void
+ editCarrierLicenses(id, licenses) : void
+ validateID(id) : bool
+ checkUnusedID(id) : bool
ShiftController
+ createShift(date, shiftType, managerID, employeeCount[]): void
+ getShift(date, shiftType): Shift
+ editCount(employeeType, newCount) : void
+ editEmployees(employeeType, employees): void
+ getShiftsBetween(startDate, endDate) : Set<Shift>
+ getEmployeeWorkDetails(id): String
+ getAvailableEmployees(date, shiftType): Set<Employees>
Manages


Business Layer
SiteController
+ getSource(SN): Source
+ getDestination(SN): Destination
TransportOrdertController
+ addTransportOrder(destID, srcID, H
DocumentController
+ addTransportDocumentr(transportDocument): void
+ getTransportDocumentr(int SN): TransportDocument
+ addDestinationDocumentr(destinationDocument): void
+ getDestinationDocumentr(destinationDocument): Dest
TransportController
+ addOrderToTransport(transportId,orderID):void
+ startTransport(transportId):void
+ placeDriver(transportId,carrierId):void
+placeTruck(transportId,licenceNumber):void
+ getTransport(transportID):Transport
TruckController
+ addTruck(licenceNumber, model,netWeight, maxCapacityWeight):void
+removeTruck(licenseNumber) :void
+getTruck(truckNumber):Truck
ConstraintController
+ addConstraint(date, shiftType) : void
+ getConstraint(date, shiftType) : Constraint
+ getConstraintEmployeeIDs(date, shiftType): Set<String>
+ getConstraintEmployees(date, shiftType): Set<Employees>
+ registerToConstraint(id, date, shiftType): void
+ unregisterToConstraint(id, date, shiftType): void
+ removeConstraint(date, shiftType): void
EmployeeController
+ registerEmployee(title, id, name, bankDetails, salary, employmentConditions, startingDate, certifications) : void
+ getEmployee(id) : Employee
+ editEmployee(id, property, value) : void
+ editCarrierLicenses(id, licenses) : void
+ validateID(id) : bool
+ checkUnusedID(id) : bool
ShiftController
+ createShift(date, shiftType, managerID, employeeCount[]): void
+ getShift(date, shiftType): Shift
+ editCount(employeeType, newCount) : void
+ editEmployees(employeeType, employees): void
+ getShiftsBetween(startDate, endDate) : Set<Shift>
+ getEmployeeWorkDetails(id): String
+ getAvailableEmployees(date, shiftType): Set<Employees>
<<Abstract>> Document
DestinationDocument
- serialNumber: int
- providedProducts: List<String>
TransportDocument
- serialNumber: int
- date: Date
- timeDeparture: Date
- truckNumber: int
- driverName: String
- sources: List<Source/Suplier>
- destinations: List<Destination>
- destDocumentSerialNumber: List<Integer>
- weightAtSource: Integer
Documentation
Constraint
- Date :Date
- Type : ShiftTypes
Manages
Available Employees
<<abstract>> Employee
- ID: String
- Name: String
- BankDetails: String
- Salary: Integer
- EmploymentConditions: String
- StartingDate: Date
- Certifications: Set<Certifications>
Assigned Employees
Managed by
<<abstract>> Shift
- Date: Date
- CarrierCount: int
- CashierCount: int
- StorekeeperCount: int
- SorterCount: int
- HR_ManagerCount: int
- Logistics_ManagerCount: int
- Transport_ManagerCount: int
EveningShift
MorningShift
TransportOrder
- source: Source
- destination: Destination
- productList: HashMap<String, Integer>
Contains
Transport
- date: Date
- startTime : Time
- DriverName: string
- sources: List<Source/Suplier>
- destinations: List<Destination>
Placed
Truck
- licenseNumber: int
- model: enum LicenseTypes
- netWeight: int
- maxCapacityWeight: int
Destination
Source
Address
- shippingArea: enum ShipingAreas
- exactAddress: String
<<Abstruct>> Site
- id: int
- phoneNumber : string
- contactPersonName: string
0..Logistics_ManagerCount
0..CarrierCount
0..CashierCount
0..StoreKeeperCount
0..SorterCount
0..HR_ManagerCount
0..Transport_ManagerCount
Logistics_Manager
Carrier
- Licenses: Set<LicenseTypes>
Cashier
StoreKeeper
Sorter
HR_Manager
Transport_Manager
holds
SiteMapper
OrderMapper
DocumentMapper
ProductOrderLink
DestinationMapper
SourceMapper
TransportOrderMapper
ProductDataMapper
DestinationDocumentDataMapper
TransportDocumentDataMapper
TruckDataMapper
TransportDataMapper
ConstraintMapper
+ get(LocalDate: date, ShiftTypes: constrain shiftTupe):Constraint
+ add(Constraint constraint,String eID):void
+ remove(Constraint constraint,String eID);void update(Constraint, propertyName)
EmployeeDataMapper
+ get(String: id):Emplyee
+ save(Employee: e):void update(String :propertyName, Employee e)
SiftMapper
+ get(LocalDate: date, ShiftTyps: Shift
+ save(Shift: shift):void
+ update(String :propertyName, Shift shift)
DAOs of
save the set of
EmployeeShiftMapperLink
CarrierLicenseLink
EmployeeCertificationLink
Logistics_ManagerDataMapper
CarrierDataMapper
CashierDataMapper
StoreKeeperDataMapper
SorterDataMapper
HR_ManagerDataMapper
EveningShiftMapper
MorningShiftMapper

Service Layer

the srervice objects are not in the object diadgam because they always create and disappear so they are not in the memory

Faced- facrd

TruckService- truckS
controller-truckC

TransportService-TransportS
controller- transportC

DocumentService- DocumentS
controller- DocumentC

ShiftService-shiftS

ConstraintService- constraintS

EmployeeService-employeeS

OrderService- orderS
controller- orderC

use

Business Layer

MorningShift1 - MorningShift
- Date: 27-05-2022
- ShiftManagerID: 3
- CarrierCount: 1
- CashierCount: 1
- StorekeeperCount: 1
- SorterCount: 0
- HR_ManagerCount: 1
- Logistics_ManagerCount: 0

EveningShift1 - EveningShift
- Date: 25-04-2022
- ShiftManagerID: 3
- CarrierCount: 1
- CashierCount: 1
- StorekeeperCount: 1
- SorterCount: 1
- HR_ManagerCount: 1
- Logistics_ManagerCount: 1

assigned workers

Ofek - Sorter
- ID: 11
- Name: Ofek Nov
- BankDetails: A654
- Salary: 20
- EmploymentConditions:
*ID: 11
Job Title: Sorter
Starting Date: 30-08-2020
Salary per shift: 20"
- StartingDate: 30-08-2020
-Certifications: []

Roi - Cashier
- ID: 3
- Name: Roi Tif
- BankDetails: A123
- Salary: 30000
- EmploymentConditions:
*ID: 3
Job Title: Cashier
Starting Date: 16-08-2020
Salary per shift: 30000"
- StartingDate: 16-08-2020
- Certifications: [ShiftManager]

Ein - Carrier
- ID: 13
- Name: Ein
- BankDetails:A 252
- Salary: 30000
- EmploymentConditions:
*ID: 13
Job Title: Carrier
Starting Date: 16-08-2021
Salary per shift: 30000"
- StartingDate: 16-08-2021
- Certifications : []
- Licenses: [B,C]

Rubin - HR_Manager
- ID: 53
- Name: Rubin
- BankDetails:A 252
- Salary: 23000
- EmploymentConditions:
*ID: 53
Job Title: HR Manager
Starting Date: 16-08-2021
Salary per shift: 23000"
- StartingDate: 16-08-2021
- Certifications : []

Constraint1 - Constraint
- Date :27-05-202
- Type : Morning

Truck-t
licenseNumber-123456
model-B
netWeight-3
maxCapacityWeight-4
currWeight-3

drive on

carry the products

TrackController-truckC

TransportController-transportC

DocumentController-DocumentC

ConstraintController-constraintC

EmployeeController-employeeC

SiteController-siteC

OrderController-orderC

ShiftController- shiftC

suplier to

Address- destAddress
shippingArea- south
exactAddress- Ashkelon Barnea1

is taking place in

Destinationd-Ashkelon1
id-2
address-destAddress
contactName- Israel
phoneNumber-0527654321

Address- sourceAddress
shippingArea- south
exactAddress- TLV Hashlom 54

Source-TLV1
id-1
address-sourceAddress
contactName- Avi
phoneNumber-0501234567

TransportOrder-order
id - 1
src- TLV1
dst-Ashkelon1
productList-<product,100>

order the product

Produto- product
id - 1
name- milk
weight-1

Data Access Layer

EmployeeDataMapper

TransportDataMapper
transport_IM= {1, transport

TruckDataMapper
truck_IM = {123456, t}

DocumentMapper
Manage
DestinationDocumentDataMapper
destinationDoc_IM- [<1,destdoc>]
TransportDocumentDataMapper
transportDoc_IM- [<1,tDoce>]

SiteMapper
Manage
DestinationMapper
destinations_IM- [<2,Ashkelon1>
SourceMapper
sources_IM- [<1,TLV1>]

OrderMapper
Manage
TransportOrderMapper
to_IM- [<1,order>]
ProductDataMapper
product_IM- [<1,product>]

ProductOrderLink

ConstraintMapper

SiftMapper
+ get(LocalDate: date, ShiftTypes: Shift
+ save(Shift: shift):void
+ update(String :propertyName, Shift shift)

EmployeeShiftMapperLink

CarrierLicenseLink

EmployeeCertificationLink

CarrierDataMapper
Carrier_IM = {13, Ein}

CashierDataMapper
Cashier_IM = {3, Roi}

SorterDataMapper
Sorter_IM = {11, Ofek}

EveningShiftMapper
EveningShift_IM = {15/04/2022Evening, EveningShift1 }

HR_ManagerDataMapper
HR_Manager_IM = {53, Rubin}

MorningShiftMapper
Morning_IM = {25/04/2022Morning, MorningShift1 }

employeeTypeLink

Logistics_ManagerDataMapper
Logistics_Manager_IM = {24, Oran}

StoreKeeperDataMapper
StoreKeeper_IM = {87, Shir}

use

is held

Manage

Service Layer

the srervice objects are not in the object diadgam because they always create and disappear so they are not in the memory

Faced- facrd

use

TruckService- truckS
controller-truckC

TransportService-TransportS
controller- transportC

DocumentService- DocumentS
controller- DocumentC

ShiftService-shiftS

ConstraintService- constraintS

EmployeeService-employeeS

OrderService- orderS
controller- orderC

Business Layer

MorningShift1 - MorningShift
- Date: 27-05-2022
- ShiftManagerID: 3
- CarrierCount: 1
- CashierCount: 1
- StorekeeperCount: 1
- SorterCount: 0
- HR_ManagerCount: 1
- Logistics_ManagerCount: 0

EveningShift1 - EveningShift
- Date: 25-04-2022
- ShiftManagerID: 3
- CarrierCount: 1
- CashierCount: 1
- StorekeeperCount: 1
- SorterCount: 1
- HR_ManagerCount: 1
- Logistics_ManagerCount: 1

assigned workers

TrackController-truckC

TransportController-transportC

DocumentController-DocumentC

ConstraintController-constraintC

EmployeeController-employeeC

SiteController-siteC

OrderController-orderC

ShiftController- shiftC

Eldad- TransportManeger
- ID: 1
- Name: Eldad vasker
- BankDetails: A654
- Salary: 20
- EmploymentConditions:
"ID: 1
Job Title: Transport Manager
Starting Date: 30-08-2020
Salary per shift: 20"
- StartingDate: 30-08-2020
-Certifications: []

Ofek - Sorter
- ID: 11
- Name: Ofek Nov
- BankDetails: A654
- Salary: 20
- EmploymentConditions:
"ID: 11
Job Title: Sorter
Starting Date: 30-08-2020
Salary per shift: 20"
- StartingDate: 30-08-2020
-Certifications: []

Roi - Cashier
- ID: 3
- Name: Roi Tif
- BankDetails: A123
- Salary: 30000
- EmploymentConditions:
"ID: 3
Job Title: Cashier
Starting Date: 16-08-2020
Salary per shift: 30000"
- StartingDate: 16-08-2020
- Certifications: [ShiftManager]

Ein - Carrier
- ID: 13
- Name: Ein
- BankDetails:A 252
- Salary: 30000
- EmploymentConditions:
"ID: 13
Job Title: Carrier
Starting Date: 16-08-2021
Salary per shift: 30000"
- StartingDate: 16-08-2021
- Certifications : []
- Licenses: [B,C]

execute the transport

drive on

Truck-t
licenseNumber-123456
model-B
netWeight-3
maxCapacityWeight-4
currWeight-3

carry the products

Transport- transport
startTime-09:00
endTime-15:00
driverName-firstDriver
truckWeight-3
transportOrders- [order]

Documents the transport

TransportDocument-tDoce
date-09:00
truckNumber-123456
driverName-Yossi
sources-1
destsSN-[2]

DestinationDocument-destdoc
destDocumentSN-1
destID-2
providedProducts-[milk]

suplier to

Address- destAddress
shippingArea- south
exactAddress- Ashkelon Barnea1

Destinationd-Ashkelon1
id-2
address-destAddress
contactName- Israel
phoneNumber-0527654321

is taking place in

Documents the transport to the destination

Address- sourceAddress
shippingArea- south
exactAddress- TLV Hashlom 54

Source-TLV1
id-1
address-sourceAddress
contactName- Avi
phoneNumber-0501234567

TransportOrder-order
id - 1
src- TLV1
dst-Ashkelon1
productList-<product,100>

order the product

Producte- product
id - 1
name- milk
weight-1

start from

Rubin - HR_Manager
- ID: 53
- Name: Rubin
- BankDetails:A 252
- Salary: 23000
- EmploymentConditions:
"ID: 53
Job Title: HR Manager
Starting Date: 16-08-2021
Salary per shift: 23000"
- StartingDate: 16-08-2021
- Certifications : []

Shir - Strorekeeper
- ID: 87
- Name: Shir
- BankDetails:A 741
- Salary: 21200
- EmploymentConditions:
"ID: 87
Job Title: Storekeeper
Starting Date: 17-09-2021
Salary per shift: 21200"
- StartingDate: 17-09-2021
- Certifications : [ShiftManager]

Oran - Logistics_Manager
- ID: 24
- Name: Oran
- BankDetails:A 12
- Salary: 21000
- EmploymentConditions:
"ID: 24
Job Title: Logistics Manager
Starting Date: 16-09-2021
Salary per shift: 30000"
- StartingDate: 16-09-2021
- Certifications : []

Constraint1 - Constraint
- Date :27-05-202
- Type : Morning

Data Access Layer

use

is held

EmployeeDataMapper

DocumentMapper

Manage

SiteMapper

OrderMapper

TransportDataMapper
transport_IM= {1, transport

TruckDataMapper
truck_IM = {123456, t}

DestinationDocumentDataMapper
destinationDoc_IM- [<1,destdoc>]

TransportDocumentDataMapper
transportDoc_IM- [<1,tDoce>]

DestinationMapper
destinations_IM- [<2,Ashkelon1>

SourceMapper
sources_IM- [<1,TLV1>]

TransportOrderMapper
to_IM- [<1,order>]

ProductDataMapper
product_IM- [<1,product>]

ProductOrderLink

ConstraintMapper

SiftMapper
+ get(LocalDate: date, ShiftTypes: Shift
+ save(Shift: shift):void
+ update(String :propertyName, Shift shift)

EmployeeShiftMapperLink

CarrierLicenseLink

EmployeeCertificationLink

CarrierDataMapper
Carrier_IM = {13, Ein}

CashierDataMapper
Cashier_IM = {3, Roi}

SorterDataMapper
Sorter_IM = {11, Ofek}

EveningShiftMapper
EveningShift_IM = {15/04/2022Evening, EveningShift1 }

HR_ManagerDataMapper
HR_Manager_IM = {53, Rubin}

MorningShiftMapper
Morning_IM = {25/04/2022Morning, MorningShift1 }

employeeTypeLink

Logistics_ManagerDataMapper
Logistics_Manager_IM = {24, Oran}

StoreKeeperDataMapper
StoreKeeper_IM = {87, Shir}

מסמך שינויים בין העבודות:

מודול הובלות

- הוספנו Actor נוסף למערכת שהוא הcarrier (היה המחלקה Driver במודל הקודם השתנה בעקבות המיזוג) שתפקידו הוא לעדכן על מיקום הובלה שאליה הוא משויך. מעדכן בהגעה למקורות ובהגעה ליעדים._ההוספה הייתה בשביל לדייק יותר מתי הובלה מגיעה לאתר ושיהיה מישהו שאחראי על הובלה ספציפית.
- הוספנו מחלקה שמתארת מוצר שאחד השדות שלו הוא משקל. בהוספת הזמנה להובלה מסוימת המערכת תבדוק את המשקל שההובלה צפויה להגיע אליו ותוסיף את ההזמנה רק כאשר המשקל הצפוי של ההובלה לא חורג מהמשקל המקסימלי של המשאית._על מנת שנוכל להימנע ממצב שבו נגיע לredesign ונאסור הובלה שחורגת מהמשקל עוד לפני שהיא יוצאת.
- הוספנו את המחלקות SiteController, OrderController על מנת שנוכל לנהל יותר בקלות את ההזמנות והאתרים במערכת.
- בנוסף בעקבות המיזוג נוספו לנו פונקציונליות חדשה למערכת לכן נאלצנו לעשות שינויים בשיבוצי ההובלות כך שיהיו מחסנאי ונהג זמינים בזמן ההובלה כלומר שיהיו עובדים זמינים במשמרת.

מודול עובדים:

שינויים בCLASS DIAGRAM מודל עובדים

1.יצירת קשר בין הקונטרולרים בbusiness עקב צורך בשימוש פונקציות אחת של השנייה והורדת לוגיקה לא רוצייה בpresentation

2.מחיקת כל DTO ושינוי מוחלט של הDAL עקב למידת dataMapper אשר יותר מתאים לעבודה שלנו זאת מאחר ואין שינויים שותפים ולא פוסקים, ואנו לא רוצים שאובייקט שDTO יהיה בעל גישה לDB

3.יצירת מחלקות אבסטרטיות DataMapper, DAO, Link אשר לכל אחת מהן יש תפקיד מובהק אשר מקל על כתיבת הקוד ומימושן עבור כל מחלקה אשר דורשת טבלה נפרדת בDB

4.הוספת מחלקה חדשה tranportManager, נוסף עקב חיבור מודל הובלות עם מודל עובדים, וצורך באובייקט זה בתור בעל תפקיד ואישיות נדרשת במשמרת

5.קשר בין transportController ל shiftController עקב חיבור מודל עובדים והובלות, הגרם ללוגיקה משותפת והצורך של transport לקבל מידע על המשמרות