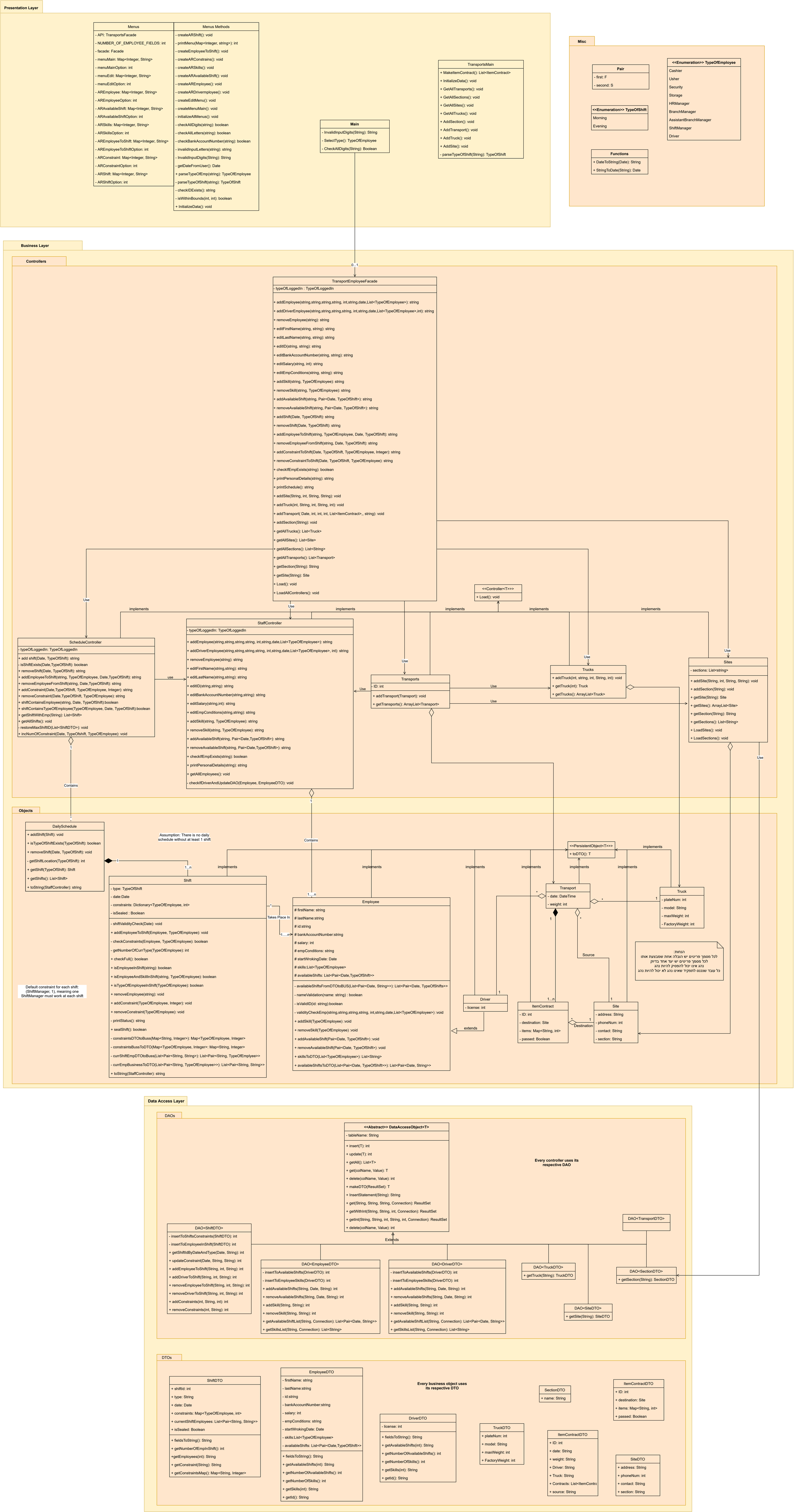
Presentation Layer
Menus
- API: TransportsFacade
- NUMBER_OF_EMPLOYEE_FIELDS: int
- facade: Facade
- menuMain: Map<Integer, String>
- menuMainOption: int
- menuEdit: Map<Integer, String>
- menuEditOption: int
- AREmployee: Map<Integer, String>
- AREmployeeOption: int
- ARAvailableShift: Map<Integer, String>
- ARAvailableShiftOption: int
- ARSkills: Map<Integer, String>
- ARSkillsOption: int
- AREmployeeToShift: Map<Integer, String>
- AREmployeeToShiftOption: int
- ARConstraint: Map<Integer, String>
- ARConstraintOption: int
- ARShift: Map<Integer, String>
- ARShiftOption: int
Menus Methods
- createARShift(): void
- printMenu(Map<Integer, string>): int
- createEmployeeToShift(): void
- createARConstrains(): void
- createARSkills(): void
- createARAvailableShift(): void
- createAREmployee(): void
- createARDrivermployee(): void
- createEditMenu(): void
- createMenuMain(): void
- initializeAllMenus(): void
- checkAllDigits(string): boolean
- checkAllLetters(string): boolean
- checkBankAccountNumber(string): boolean
- invalidInputLetters(string): string
- InvalidInputDigits(String): String
- getDateFromUser(): Date
+ parseTypeOfEmp(string): TypeOfEmployee
- parseTypeOfShift(string): TypeOfShift
- checkIDExists(): string
- isWithinBounds(int, int): boolean
+ InitializeData(): void
TransportsMain
+ MakeItemContract(): List<ItemContract>
+ InitializeData(): void
+ GetAllTransports(): void
+ GetAllSections(): void
+ GetAllSites(): void
+ GetAllTrucks(): void
+ AddSection(): void
+ AddTransport(): void
+ AddTruck(): void
+ AddSite(): void
- parseTypeOfShift(String): TypeOfShift
Main
- InvalidInputDigits(String): String
- SelectType(): TypeOfEmployee
- CheckAllDigits(String): Boolean
0..1
Misc
Pair
- first: F
- second: S
<<Enumeration>> TypeOfShift
Morning
Evening
Functions
+ DateToString(Date): String
+ StringToDate(String): Date
<<Enumeration>> TypeOfEmployee
Cashier
Usher
Security
Storage
HRtManager
BranchManager
AssistantBranchManager
ShiftManager
Driver
Business Layer
Controllers
TransportEmployeeFacade
- typeOfLoggedIn : TypeOfLoggedIn
+ addEmployee(string,string,string,string, int,string,date,List<TypeOfEmployee>): string
+ addDriverEmployee(string,string,string,string, int,string,date,List<TypeOfEmployee>,int): string
+ removeEmployee(string): string
+ editFirstName(string, string): string
+ editLastName(string, string): string
+ editID(string, string): string
+ editBankAccountNumber(string, string): string
+ editSalary(string, int): string
+ editEmpConditions(string, string): string
+ addSkill(string, TypeOfEmployee): string
+ removeSkill(string, TypeOfEmployee): string
+ addAvailableShift(string, Pair<Date, TypeOfShift>): string
+ removeAvailableShift(string, Pair<Date, TypeOfShift>): string
+ addShift(Date, TypeOfShift): string
+ removeShift(Date, TypeOfShift): string
+ addEmployeeToShift(string, TypeOfEmployee, Date, TypeOfShift): string
+ removeEmployeeFromShift(string, Date, TypeOfShift): string
+ addConstraintToShift(Date, TypeOfShift, TypeOfEmployee, Integer): string
+ removeConstraintToShift(Date, TypeOfShift, TypeOfEmployee): string
+ checkIfEmpExists(string): boolean
+ printPersonalDetails(string): string
+ printSchedule(): string
+ addSite(String, int, String, String): void
+ addTruck(int, String, int, String, int): void
+ addTransport(Date, int, int, int, List<ItemContract>,, string): void
+ addSection(String): void
+ getAllTrucks(): List<Truck>
+ getAllSites(): List<Site>
+ getAllSections(): List<String>
+ getAllTransports(): List<Transport>
+ getSection(String): String
+ getSite(String): Site
+ Load(): void
+ LoadAllControllers(): void
<<Controller<T>>>
+ Load(): void
Use
implements
StaffController
- typeOfLoggedIn: TypeOfLoggedIn
+ addEmployee(string,string,string,string, int,string,date,List<TypeOfEmployee>): string
+ addDriverEmployee(string,string,string,string, int,string,date,List<TypeOfEmployee>, int): string
+ removeEmployee(string): string
+ editFirstName(string,string): string
+ editLastName(string,string): string
+ editID(string,string): string
+ editBankAccountNumber(string,string): string
+ editSalary(string,int): string
+ editEmpConditions(string,string): string
+ addSkill(string, TypeOfEmployee): string
+ removeSkill(string, TypeOfEmployee): string
+ addAvailableShift(string, Pair<Date,TypeOfShift>): string
+ removeAvailableShift(string, Pair<Date,TypeOfShift>): string
+ checkIfEmpExists(string): boolean
+ printPersonalDetails(string): string
+ getAllEmployees(): void
- checkIfDriverAndUpdateDAO(Employee, EmployeeDTO): void
ScheduleController
- typeOfLoggedIn: TypeOfLoggedIn
+ add shift(Date, TypeOfShift): string
- isShiftExists(Date,TypeOfShift): boolean
+ removeShift(Date, TypeOfShift): string
+ addEmployeeToShift(string, TypeOfEmployee, Date,TypeOfShift): string
+ removeEmployeeFromShift(string, Date,TypeOfShift): string
+ addConstraint(Date,TypeOfShift, TypeOfEmployee, Integer): string
+ removeConstraint(Date,TypeOfShift, TypeOfEmployee): string
+ shiftContainsEmployee(string, Date, TypeOfShift):boolean
+ shiftContainsTypeOfEmployee(TypeOfEmployee, Date, TypeOfShift):boolean
+ getShiftWithEmp(String): List<Shift>
+ getAllShifts():: void
- restoreMaxShiftID(List<ShiftIDTO>): void
+ incNumOfConstraint(Date, TypeOfshift, TypeOfEmployee): void
use
Contains
Transports
- ID: int
+ addTransport(Transport): void
+ getTransports(): ArrayList<Transport>
Trucks
+ addTruck(int, string, int, String, int): void
+ getTruck(int): Truck
+ getTrucks(): ArrayList<Truck>
Sites
- sections: List<string>
+ addSite(String, int, String, String): void
+ addSection(String): void
+ getSite(String): Site
+ getSites(): ArrayList<Site>
+ getSection(String): String
+ getSections(): List<String>
+ LoadSites(): void
+ LoadSections(): void
Objects
DailySchedule
+ addShift(Shift): void
+ isTypeOfShiftExists(TypeOfShift): boolean
+ removeShift(Date, TypeOfShift): void
- getShiftLocation(TypeOfShift): int
+ getShift(TypeOfShift): Shift
+ getShifts(): List<Shift>
+ toString(StaffController): string
Assumption: There is no daily schedule without at least 1 shift
Shift
- type: TypeOfShift
- date:Date
- constraints: Dictionary<TypeOfEmployee, int>
- isSealed : Boolean
- shiftValidityCheck(Date): void
+ addEmployeeToShift(Employee, TypeOfEmployee): void
- checkConstraints(Employee, TypeOfEmployee): boolean
- getNumberOfCurrType(TypeOfEmployee): int
+ checkFull(): boolean
+ isEmployeeInShift(string): boolean
+ isEmployeeAndSkillInShift(string, TypeOfEmployee): boolean
+ isTypeOfEmployeeInShift(TypeOfEmployee): boolean
+ removeEmployee(string): void
+ addConstraint(TypeOfEmployee, Integer): void
+ removeConstraint(TypeOfEmployee): void
- printStatus(): string
+ sealShift(): boolean
- constraintsDTOtoBuss(Map<String, Integer>: Map<TypeOfEmployee, Integer>
- constraintsBussToDTO(Map<TypeOfEmployee, Integer>: Map<String, Integer>
- currShiftEmpDTOtoBuss(List<Pair<String, String>>: List<Pair<String, TypeOfEmployee>>
- currEmpBusinessToDTO(List<Pair<String, TypeOfEmployee>>): List<Pair<String, String>>
+ toString(StaffController): string
Takes Place In
Default constraint for each shift: (ShiftManager, 1), meaning one ShiftManager must work at each shift
Contains
Employee
firstName: string
lastName:string
id:string
bankAccountNumber:string
salary: int
empConditions: string
startWrokingDate: Date
skills:List<TypeOfEmployee>
availableShifts: List<Pair<Date,TypeOfShift>>
- availableShiftsFromDTOtoBUS(List<Pair<Date, String>>): List<Pair<Date, TypeOfShifts>>
- nameValidation(name: string) : boolean
- isValidID(id: string):boolean
- validityCheckEmp(string,string,string,string, int,string,date,List<TypeOfEmployee>) :void
+ addSkill(TypeOfEmployee) :void
+ removeSkill(TypeOfEmployee) :void
+ addAvailableShift(Pair<Date, TypeOfShift>) :void
+ removeAvailableShift(Pair<Date, TypeOfShift>) :void
+ skillsToDTO(List<TypeOfEmployee>): List<String>
+ availableShiftsToDTO(List<Pair<Date, TypeOfShift>>): List<Pair<Date, String>>
<<PersistentObject<T>>>
+ toDTO(): T
Transport
- date: DateTime
- weight: int
Truck
- plateNum: int
- model: String
- maxWeight: int
- FactoryWeight: int
Source
Driver
- license: int
extends
ItemContract
- ID: int
- destination: Site
- items: Map<String, int>
- passed: Boolean
Destination
Site
- address: String
- phoneNum: int
- contact: String
- section: String
הנחות:
לכל מסוף פרטים יש הובלה אחת שמבצעת אותו
לכל מסוף פרטים יש יעד אחד בדיוק
נהג אינו יכול להוסיף לו/ירות נהג
כל עובד שנכנס למסדיר שאינו נהג לא יכול לרשות נהג
Data Access Layer
DAOs
<<Abstract>> DataAccessObject<T>
- tableName: String
+ insert(T): int
+ update(T): int
+ getAll(): List<T>
+ get(colName, Value): T
+ delete(colName, Value): int
+ makeDTO(ResultSet): T
+ InsertStatement(String): String
+ get(String, String, String, Connection): ResultSet
+ getWithInt(String, String, int, Connection): ResultSet
+ getInt(String, String, int, String, int, Connection): ResultSet
+ delete(colName, Value): int
Every controller uses its respective DAO
Extends
DAO<ShiftDTO>
- insertToShiftsConstraints(ShiftDTO): int
- insertToEmployeeInShift(ShiftDTO): int
+ getShiftIdByDateAndType(Date, String): int
+ updateConstraint(Date, String, String): int
+ addEmployeeToShift(String, int, String): int
+ addDriverToShift(String, int, String): int
+ removeEmployeeToShift(String, int, String): int
+ removeDriverToShift(String, int, String): int
+ addConstraints(int, String, int): int
+ removeConstraints(int, String): int
DAO<EmployeeDTO>
- insertToAvailableShifts(DriverDTO): int
- insertToEmployeeSkills(DriverDTO): int
+ addAvailableShifts(String, Date, String): int
+ removeAvailableShifts(String, Date, String): int
+ addSkill(String, String): int
+ removeSkill(String, String): int
+ getAvailableShiftList(String, Connection): List<Pair<Date, String>>
+ getSkillsList(String, Connection): List<String>
DAO<DriverDTO>
- insertToAvailableShifts(DriverDTO): int
- insertToEmployeeSkills(DriverDTO): int
+ addAvailableShifts(String, Date, String): int
+ removeAvailableShifts(String, Date, String): int
+ addSkill(String, String): int
+ removeSkill(String, String): int
+ getAvailableShiftList(String, Connection): List<Pair<Date, String>>
+ getSkillsList(String, Connection): List<String>
DAO<TruckDTO>
+ getTruck(String): TruckDTO
DAO<TransportDTO>
DAO<SectionDTO>
+ getSection(String): SectionDTO
DAO<SiteDTO>
+ getSite(String): SiteDTO
DTOs
ShiftDTO
+ shiftId: int
+ type: String
+ date: Date
+ constraints: Map<TypeOfEmployee, int>
+ currentShiftEmployees: List<Pair<String, String>>
+ isSealed: Boolean
+ fieldsToString(): String
+ getNumberOfEmpInShift(): int
+getEmployees(int): String
+ getConstraints(String): String
+ getConstraintsMap(): Map<String, Integer>
EmployeeDTO
- firstName: string
- lastName:string
- id:string
- bankAccountNumber:string
- salary: int
- empConditions: string
- startWrokingDate: Date
- skills:List<TypeOfEmployee>
- availableShifts: List<Pair<Date,TypeOfShift>>
+ fieldsToString(): String
+ getAvailableShifts(int): String
+ getNumberOfAvailableShifts(): int
+ getNumberOfSkills(): int
+ getSkills(int): String
+ getId(): String
Every business object uses its respective DTO
DriverDTO
- license: int
+ getAvailableShifts(int): String
+ getNumberOfAvailableShifts(): int
+ getNumberOfSkills(): int
+ getSkills(int): String
+ getId(): String
TruckDTO
+ plateNum: int
+ model: String
+ maxWeight: int
+ FactoryWeight: int
SectionDTO
+ name: String
ItemContractDTO
+ ID: int
+ date: String
+ weight: String
+ Driver: String
+ Truck: String
+ Contracts: List<ItemContr
+ source: String
ItemContractDTO
+ ID: int
+ destination: Site
+ items: Map<String, int>
+ passed: Boolean
SiteDTO
+ address: String
+ phoneNum: int
+ contact: String
+ section: String