Presentation Layer

View

Main
- + static main(String[] args): void

— calls →

MenuManager
- - finished: boolean
- + void run()
- + void terminate()

1 runs 1..*

<<Interface>> Menu
- + Menu run()
- + void printCommands()

△ implements

connection to Transport Module UI

LoginMenu ←redirects to→ **EmployeeMenu** — redirects to → **HRManagerMenu**

1   1   1

ViewModel

1   1   1

LoginMenuVM
- + login(username: String, password: String): String
- + isLoggedIn(): boolean
- + getUserAuthorizations(): List<String>

EmployeeMenuVM
- + registerShift(): String
- + logout(): String
- + requestShift(branchId: String, shiftDate, ...): String
- + getNextWeekShifts(branchId: String): String
- + getWeekShifts(branchId: String, weekStart): String
- + applyCancelCard(branchId: String, ...): String
- + reportShiftActivity(branchId: String, shiftDate, ...): String
- + getMyShifts(): String

HRManagerMenuVM
- + recruitEmployee(fullName: String, branchId: String, ...): String
- + createUser(username: String, password: String): String
- + logout(): String
- + createNextWeekShifts(branchId: String): String
- + createWeekShifts(branchId: String, weekStart): String
- + getNextWeekShiftRequests(branchId: String): String
- + getWeekShiftRequests(branchId: String, weekStart): String
- + approveShift(branchId: String, shiftDate, shiftType): String

1   1

Model

1

<<Singleton>> BackendController
- - String loggedUsername
- + isLoggedIn(): boolean
- - logout(): String
- + recruitEmployee(): String
- + createUser(username: String, password: String): String
- + requestShift(branchId: String, shiftTime: String, shiftDate: LocalDate): String
- + getUserAuthorizations(): List<String>
- + createWeekShifts(branchId: String, weekStart: LocalDate): String
- + createNextWeekShifts(branchId: String): String
- + setShiftNeededAmount(branchId: String, shiftDatE: LocalDate, shiftType: String, role: String, amount: int): String
- + setShiftEmployees(branchId: String, shiftDate: LocalDate, shiftType: String, role: String, List<String> employeeIds): String
- + getWeekShifts(branchId: String, weekStart: LocalDate): List<SShift[]>
- + getNextWeekShifts(branchId: String): List<SShift[]>
- + getMyShifts(): List<SShift[]>
- + getEmployee(): SEmployee
- + certifyEmployee(employeeId: String, role: String): String
- + uncertifyEmployee(employeeId: String, role: String): String
- + certifyDriver(driverId: String, driverLicense: String): String
- + approveShift(branchId: String, shiftDate: LocalDate, shiftType: String): String
- + addEmployeeToBranch(employeeId: String, branchId: String): String
- + deleteShift(branchId: String, shiftDate: LocalDate, shiftType: String): String
- + createBranch(branchId: String): String
- + updateBranchWorkingHours(branchId: String, morningStart: LocalTime, morningEnd: LocalTime, eveningStart: LocalTime, eveningEnd: LocalTime): String
- + updateEmployeeSalary(employeeId: String, hourlySalaryRate: double, salaryBonus: double): String
- + updateEmployeeBankDetails(employeeId: String, bankDetails: String): String
- + updateEmployeeEmploymentConditions(employeeId: String, employmentConditiosn: String): String
- + updateEmployeeDetails(employeeId: String, details: String): String
- + authorizeUser(username: String, autorization: String): String
- + cancelShiftRequest(branchId: String, shiftType: Stirng, shiftDate: LocalDate, role: String): String
- + reportShiftActivity(branchId: String, shiftDate: LocalDate, shiftType: String, activity:String): String
- + applyCancelCard(branchid: String, shiftDate: LocalDate, shiftType: String, productId: String): String

uses   uses   uses

Service Layer

ServiceFactory

connection to Transport Module

uses

Response<T>

- returnValue: T

- errorMessage: String

+ createErrorResponse(String errorMessage): Response

+ errorOccurred(): boolean

returns

creates

creates

uses

returns

EmployeesService

+ loadData(): void

+ resetData(): void

+ recruitEmployee(String actorUsername, String branchId, String employeeId, String bankDetails,
        LocalDate employmentDate, String employmentConditions, String details): json

+ addEmployeeToBranch(String actorUsername, String employeeId, String branchId): json

+ createWeekShifts(String actorUsername, String branchId, LocalDate weekStart): json

+ setShiftNeededAmount(String actorUsername, String branchId,
        LocalDate shiftDate, SShiftType type, String role, int amount) : json

+ requestShift(String actorUsername, String branchId, LocalDate shiftDate, SShiftType shiftType): json

+ cancelShiftRequest(String actorUsername, String branchId, LocalDate shiftDate, SShiftType shiftType
                String role): json

+ getWeekShifts(LocalDate shiftDate, String branchId, LocalDate weekStart)
        : json

+ setShiftEmployees(String actorUsername, String branchId, LocalDate shiftDate,
        SShiftType shiftType, String role, List<String> employeeIds): json

+ createBranch(String actorUsername, String branchId): json

+ getEmployeeShifts(String actorUsername): json

+ getEmployee(String actorUsername): json

+ certifyEmployee(String actorUsername, String employeeId, String role): json

+ uncertifyEmployee(String actorUsername, String employeeId, String role): json

+ certifyDriver(actorUsername: String, employeeId: String, driverLicense: String: json

+ approveShift(String actorUsername, String branchId, LocalDate shiftDate, SShiftType shiftType): json

+ applyCancelCard(String actorUsername, String branchId, LocalDate shiftDate, SShiftType shiftType,
            String productId): json

+ reportShiftActivity(String actorUsername, String branchId, LocalDate shiftDate, SShiftType shiftType,
            String activity): json

+ deleteShift(String actorUsername, String branchId, LocalDate shiftDate, SShiftType shiftType): json

+ updateBranchWorkingHours(String actorUsername,. String branchId, LocalTime morningStart,
            LocalTime morningEnd, LocalTime eveningStart, LocalTime eveningEnd): json

+ updateEmployeeSalary(String actorUsername, String employeeId, double hourlySalaryRate, double salaryBonus): json

+ getAvailableDrivers(atDateTime: String): String

+ checkStoreKeeperAvailability(dateToCheck: Strring, branchAddress: String): String

---

UserService

+ loadData(): void

+ resetData(): void

+ login(String username, String password): json

+ logout(String username): json

+ createUser(String actorUsername, String username, String password): json

+ createManagerUser(String actorUsername, String username, String password): json

+ isAuthorized(String username, String authorization) : json

+ isAuthorized(String username, Authorization auth) : json

+ getUserAuthorizations(String username) : json

+ authorizeUser(String actorUsername, String username, String authorization): json

---

returns as json

returns as json

uses

uses

SShift

- shiftDate: LocalDate

- shiftType: ShiftType

- shiftRequests: Map<String,List<SEmployee>>

- shiftWorkers: Map<String,List<SEmployee>>

- isApproved: boolean

has▶

**<<Enum>>
SShiftType**

Morning

Evening

SEmployee

0..*

- fullName: String

- id: String

- employmentDate: String

- employmentConditions: String

- details: String

- expectedSalary: double

- roles: Set<String>

Business Layer

<<Singleton>> EmployeeController

+ resetData(): void

+ recruitEmployee(Branch branch, String employeeId, String bankDetails, LocalDate employmentDate, String employmentConditions, String details): boolean

+ certifyEmployee(String employeeId, Role role): boolean

+ getBranch(String branchId): Branch

+ createBranch(String branchId): Branch

+ addEmployeeToBranch(String branchId, String employeeId): void

+ updateBranchWorkingHours(String brancHid, LocalTime morningStart, LocalTime morningEnd, LocalTime eveningStart, LocalTime eveningEnd): void

+ getAvailableDrivers(dateTime: LocalDateTime): List<Employee>

+ checkStorekeepeerAvailability(date: LocalDate, branchAddress: String): void

<<Singleton>> UserController

+ resetData(): void

+ login(String username, String password): boolean

+ logout(String username, String password): void

+ createUser(String username, String password): boolean

+ createManagerUser(String username, String password) : boolean

+ isAuthorized(String username, Authorization auth): boolean

+ getUserAuthorizations(String username): Set<Authorization>

+ authorizeUser(String username, Authorization auth): void

(uses)

Branch

(using BranchesDAO)

- LocalTime morningStart

- LocalTime morningEnd

- LocalTime eveningStart

- LocalTime eveningEnd

Employee

(using EmployeeDAO) 0..*

- fullName: String

- ID: String

- bankDetails: String

- hourlySalaryRate: double

- monthlyHours: double

- salaryBonus: double

- employmentDate: LocalDate

- employmentConditions: String

- details: String

<<Enum>> Authorization

uses

HRManager

Cashier

Storekeeper

ShiftManager

TransportManager

User

(using UserDAO) 0..*

- username: String

- password: String

- loggedIn: boolean

+ isAuthorized(Authorization auth): boolean

+ login(String password): boolean

+ logout(): void

+ authorize(Authorization auth): void

holds 0..*

<<Enum>> Role

1..*

Cashier

Storekeeper

ShiftManager

GeneralWorker

SecurityGuard

Steward

Cleaner

Driver

has

<<Enum>> ShiftType

Morning

Evening

characterized by

ShiftController

uses

+ resetData(): void

+ getShift(String branchId, LocalDate shiftDate, ShiftType shiftType): Shift

+ shiftExists(String branchId, LocalDate shiftDate, ShiftType shiftType: boolean

+ createShift(String branchId, LocalDate shiftDate, ShiftType shiftType): Shift

+ deleteShift(String branchId, LocalDate shiftDate, ShiftType shiftType): void

+ createWeekShifts(String branchId, LocalDate weekStart): void

+ getWeekShifts(String branchId, LocalDate weekStart): List<Shift[]>

+ setShiftNeededAmount(String branchId, LocalDate shiftDate, ShiftType type, Role role, int amount): void

+ requestShift(Employee employee, String branchId, LocalDate shiftDate, ShiftType shiftType, Role role): void

+ cancelShiftRequest(Employee employee, String branchid, LocalDate shiftDate, ShiftType shiftType, Role role): void

+ setShiftEmployees(String branchId, LocalDate shiftDate, ShiftType shiftType, String role, List<Employee> employees): void

+ getEmployeeShifts(Employee employee): List<Shift[]>

+ approveShift(String branchId, LocalDate shiftDate, ShiftType shiftType): void

+ applyCancelCard(String branchId, LocalDate shiftDate, ShiftType shiftType, String employeeId, String productId): void

+ reportShiftActivity(String branchId, LocalDate shiftDate, ShiftType shiftType, String employeeId, String activity): void

createBranch(String branchId): void

Shift

(using ShiftsDAO) 0..*

- branchId: String

- shiftDate: LocalDate

- shiftType: ShiftType

- isApproved: boolean

- shiftEmployeesAmounts: Map<Role, Integer>

- shiftRequests: Map<Role,List<Employee>>

- shiftWorkers: Map<Role,List<Employee>>

- cancelCardApplies: List<String>

- shiftActivities: List<String>

+ addShiftWorker(role: Role, employee: Employee) void

+ addShiftRequest(role: Role, employee: Employee) void

+ approve(): void

+ reportActivity(reportingEmployee: String, activity: String): void

using
⋮
∨

```
┌─────────────────────────────────┐
│        Data Access Layer         │
└─────────────────────────────────┘
```

Utils

DateUtils
+ DATE_PATTERN: String
+ DateFormat: DateTimeFormatter
+ getWeekDays(LocalDate d): LocalDate[]
+ getWeekNumber(LocalDate d): int
+ parse(String dateString): LocalDate
+ validDate(String dateInput): boolean

JsonUtils
+ <T> serialize(T object): String
+ <T> deserialize(String json, Type typeOfT): T

Design Updates

1) Added "ServiceFactory" class – We used the factory pattern, to make sure that the services and controllers are initialized correctly.

2) Updated some of the classes to not be used as a Singleton, to match with the factory pattern written above.

3) Updated the services return values to be a json String of the Response<T> object, instead of a Response<T> return type.

4) Added the needed integration with the Transports Module, through the service and business calls, and the unified Data Access Layer, shown on its dedicated class diagram.

5) Fixed the aggregation relationship between some of the classes, it is shown as a "using" association relationship.