





Business Layer

<<Singleton>>
EmployeeController

- employees: Map<Branch, Map<String, Employee>> // branch to <employeeId to Employee>
- branches: Map<String, Branch> // branchId to Branch
- instance: EmployeesController

+ loadData(): void
+ 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

Employee

- fullName: String
- ID: String
- bankDetails: String
- hourlySalaryRate: double
- monthlyHours: double
- salaryBonus: double
- employmentDate: LocalDate
- employmentConditions: String
- details: String
- roles: Set<Role>

Branch

- LocalTime morningStart
- LocalTime morningEnd
- LocalTime eveningStart
- LocalTime eveningEnd

<<Singleton>>
ShiftController

- shifts: Map<String, Map<LocalDate,Map<ShiftType,Shift>>> // branchId to <Date to <ShiftType to Shift>>
- instance: ShiftController

+ 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

<<Singleton>>
UserController

- users: Map<String,User>
- instance: UserController

+ loadData(): void
+ 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

User

- username: String
- password: String
- authorizations: List<Authorization>
- loggedIn: boolean
+ isAuthorized(Authorization auth) : boolean
+ login(String password): boolean
+ logout(): void
+ authorize(Authorization auth): void

<<Enum>>
Authorization

HRManager
Cashier
Storekeeper
ShiftManager
LogisticsManager

<<Enum>>
Role

Cashier
Storekeeper
ShiftManager
GeneralWorker
SecurityGuard
Steward
Cleaner
Driver

<<Enum>>
ShiftType

Morning
Evening

Shift

- 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>

...

Utils

DateUtils

```
+ DATE_PATTERN: String
+ DateFormat: DateTimeFormatter

+ getWeekDays(LocalDate d): LocalDate[]
+ getWeekNumber(LocalDate d): int
+ parse(String dateString): LocalDate
+ validate(String dateInput): boolean
```

JsonUtils

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