Models(persistent units):

public class User {// any user in system: admin or driver

private String id; //GUID string

private String email;

private String password;

private String name;

private boolean admin; // if user is admin

private boolean deleted; //if marked as deleted

private String street;

private String address;

private String plz;

private String city;

private String telephoneNumber;

private TransportType transportType;

private TelephoneType telephoneType;

private String iban;

private String bic;

private ContractType contractType;

private String base64Image; //image binary array in base64 form

private int doneHours; // computable field , selected hours for current month

private int diffPrevHours; //computable field, (planned hours(45) - selected hours) for current month if user contract is minijob else =0

}

public class HourForecast { // defines manual forecasting for an hour

private int count; // how many drivers needed for the hour

}

public class ManualForecasting { // defines manual forecasting for all weeks

private HourForecast[][] days; // 7\*24 array for hour forecasting within week

}

public class HourStatistics { // parent class for AdminHourStatistics and DriverHourStatistics, defines current situation for the hour

protected int plannedHours; // (proper hourForecasting.count - count of all drivers who registered for the hour) if current user is driver , else proper hourForecasting.count

protected int index; // from 0 to 23, defining index of the hour within day

}

public class DriverHourStatistics extends HourStatistics {// defines current situation of the hour for driver

private boolean selected; //if the hour selected by the driver whom dayStatistics field belongs

private DriverDayStatistics dayStatistics; // parent object that holds all driverHourStatistics(current driverHourStatistics also)

}

public class DriverDayStatistics { // defines current situation of the day for driver

protected String userId; // id of driver whom belongs this statistics

protected LocalDate date; // date of the day

protected DriverHourStatistics[] hourStatisticsArray; //hourStatistics for the day

}

public class DriverCalendarWeek {//defines current situation of the week for driver

private String userId; // driver id

private LocalDate beginDate; //begin date of the week

private DriverDayStatistics[] dayStatisticsArray;// dayStatistics objects for the week

}

View Models(just for project logic, not persistent)

public class UserTableViewModel { //defines all parameters of filtering/sorting/paging/itemsPerPage

private String sortingField; //sorting field

private boolean reverse;// is sorting in reverse order

private String keyword;// keyword for filtering

private int beginIndex;// begin index after soring and filtering applied

private int maxNumber;// max number of items to fetch

}

public class MonthStatistics { //defines month statistics(on the top of month acardion) for selected month(tab) in the main page(calendar view) for the user

private int plannedHours; // for driver user type : 45 if minijob contract , else 0; for admin user type sum of forecasting data for days in month

private int doneHours; // for driver user type : selected hours for the month; for admin user type all selected hours by all users in the system

private LocalDate beginDate; // begin date of month

}

public class CalendarWeekLight { //defines a week

private LocalDate beginDate; // begin date

private LocalDate endDate; // end date

}

public class CalendarMonth { // defines a month

private LocalDate beginDate; // begin date of the month

private CalendarWeekLight[] calendarWeekLights; // calendarWeekLight objects of the month

}

public class CalendarViewModel { //defines all months in main page(calendar view) of the user

private CalendarMonth[] calendarMonths; // all calendarMonth objects of the calendar for the user

}

public class AdminHourStatistics extends HourStatistics { // defines current situation of the day for admin

private int doneHours; // count of all drivers who registered for the hour

private AdminDayStatistics adminDayStatistics; // parent object that holds all adminHourStatistics(current adminHourStatistics also)

}

public class AdminDayStatistics {// defines current situation of the day for admin

private LocalDate date;//date of the day

private AdminHourStatistics[] adminHourStatisticsArray;//hourStatistics for the day

}

public class AdminCalendarWeek {//defines current situation of the week for admin

private LocalDate beginDate; // begin date of the week

private AdminDayStatistics[] adminDayStatisticsArray; //dayStatistics objects for the week

}

public class DetailedDriverDayStatistics extends DriverDayStatistics { // defines driverDayStatistics + driverInfo string for driver whom the object belongs, used in admin's main page(calendar view)

private String driverInfo; //driver info(name of driver)

}

public class DetailedAdminDayStatistics { //defines day statistics for the opened day, used in admin main page(calendar view) after events: open month->open week->open day

private LocalDate date; // date of the day

private DetailedDriverDayStatistics[] detailedDriverDayStatisticsArray; //detailedDriverDayStatistics objects for the day

}

REST API we need:

* User getUserByEmail(String email)
* User getUserById(String id)
* User insertOrUpdateUser(User user) – depending if user exists
* User deleteUser(String id)
* User[] getSortedFilteredPagedUsersWithoutStatistics(UserTableViewModel userTableViewModel)
* int getAllUsersCount()
* ManualForecasting getManualForecasting()
* void setManualForecasting(ManualForecasting manualForecasting)
* DriverCalendarWeek getStatisticsForDriverWeek(DriverCalendarWeek driverCalendarWeekInDB)
* DriverCalendarWeek getDriverCalendarWeekFromDB(User user, LocalDate beginDate) //without statistics
* DriverCalendarWeek insertOrUpdateDriverCalendarWeek(DriverCalendarWeek driverCalendarWeek) – depending if driverCalendarWeek exists
* MonthStatistics getMonthStatisticsForDriverUser(User user, LocalDate monthBeginDate)
* MonthStatistics getMonthStatisticsForAdminUser(LocalDate monthBeginDate)
* AdminCalendarWeek getAdminCalendarWeek(LocalDate beginDate)
* boolean driverHasCalendarWeek(User user, LocalDate beginDateOfProperWeek)
* DetailedAdminDayStatistics getDetailedAdminDayStatistics(LocalDate date)
* DriverCalendarWeek getDriverCalendarWeekFromDB(String userId, LocalDate beginDateOfWeek)
* ArrayList<User> getActiveDriversForMonth(LocalDate beginDateOfMonth)
* ArrayList<User> getActiveDrivers() – for current date