FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Fleet Management System

Design

Documentation of a project for the purpose of the course BIE-SI1.

Authors: Berker Katipoglu
Oleksii Tarbeiev
Atsamaz Akopyan
Sanan Pashayev
Abu Zaid Muntaser

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Contents

1. Architecture 3
   1.1 Presentation Layer 4
      1.1.1 Web User Interface 4
   1.2 Business Layer 5
      1.2.1 Controller 5
   1.3 Data Layer 5
2. Design Class Model 6
   2.1 view.employee 8
   2.2 view.fleet 8
   2.3 view.reservation 9
   2.4 view 10
   2.5 controller 11
      2.5.1 ReservationControllerInterface 13
   2.6 data 13
      2.6.1 entity 13
         2.6.1.1 Car 14
         2.6.1.2 Department 15
         2.6.1.3 Employee 15
         2.6.1.4 Reservation 15
         2.6.1.5 Role 15
         2.6.1.6 Trip 16
      2.6.2 dao 16
         2.6.2.1 ICarDao 18
         2.6.2.2 IDepartmentDao 18
         2.6.2.3 IEmployeeDao 18
         2.6.2.4 IReservationDao 18
         2.6.2.5 IRoleDao 19
         2.6.2.6 ITripDao 20
3. Class Diagram 21
   3.1 Car 21
      3.1.1 CarStateMachine_Availability 22
      3.1.2 CarStateMachine_Condition 22
   3.2 Department 23
   3.3 Employee 23
   3.4 Maintenance 24
   3.5 Reservation 24
      3.5.1 ReservationStateMachine 24
   3.6 Role 25
   3.7 Trip 25
      3.7.1 TripStateMachine 26

1. Architecture

This chapter describes the architecture of the web application of the Fleet Management System.

The web application will be implemented as Java using following technologies:
• Java 1.8
• Spring Boot Framework
• Vaadin Framework

The architecture is divided into three independent layers:
• Presentation layer - layer responsible for presentation of application data.
• Business layer- layer responsible for all business logic of the application.
• Data layer - layer responsible for data persistence.

The layers are isolated using interfaces:
• IBusiness - interface between the presentation and business layers. It consists of one main IBusiness to handle any request from presentation layer.
• IDao - interface between the business and data layer. It consists of several IDao interfaces defining the persistence operations for individual entities in the system.

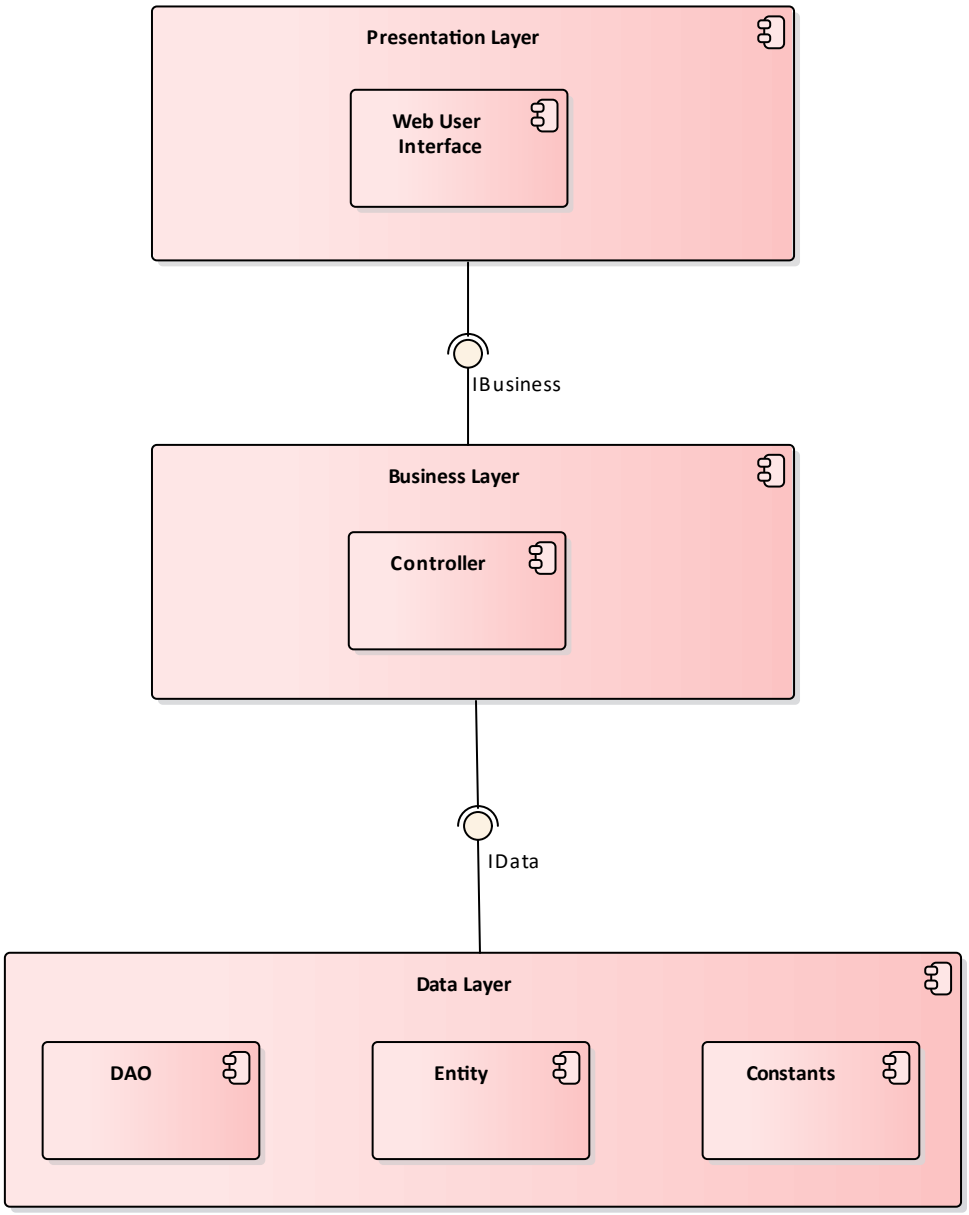


Figure 1 - Architecture

1.1 Presentation Layer

The presentation layer is responsible for sharing the information with a user or other systems. It consists of a web user interface and connected to business layer through business interface.

1.1.1 Web User Interface

Web User Interface is implemented in Java using Swing Boot and Vaadin frameworks.

1.2 Business Layer

The business layer contains implementation of the business logic and It consists of controllers implementing the system behavior. It is connected to data later through data interface.

1.2.1 Controller

Controller is implemented in Java using Spring Boot framework.

1.3 Data Layer

The data layer is responsible for data persistence. It consists of DAO classes implementing the persistence operations and entities representing the persistent data.

The implementation of the data layer is based on the Spring Boot Framework and its native support for database.

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

2. Design Class Model

This package describes the classes and packages of the implementation of the web application based on the Vaadin and Spring Boot Frameworks.

In the whole model, only important classes and their methods are shown. The primitive getters and setters are not shown.

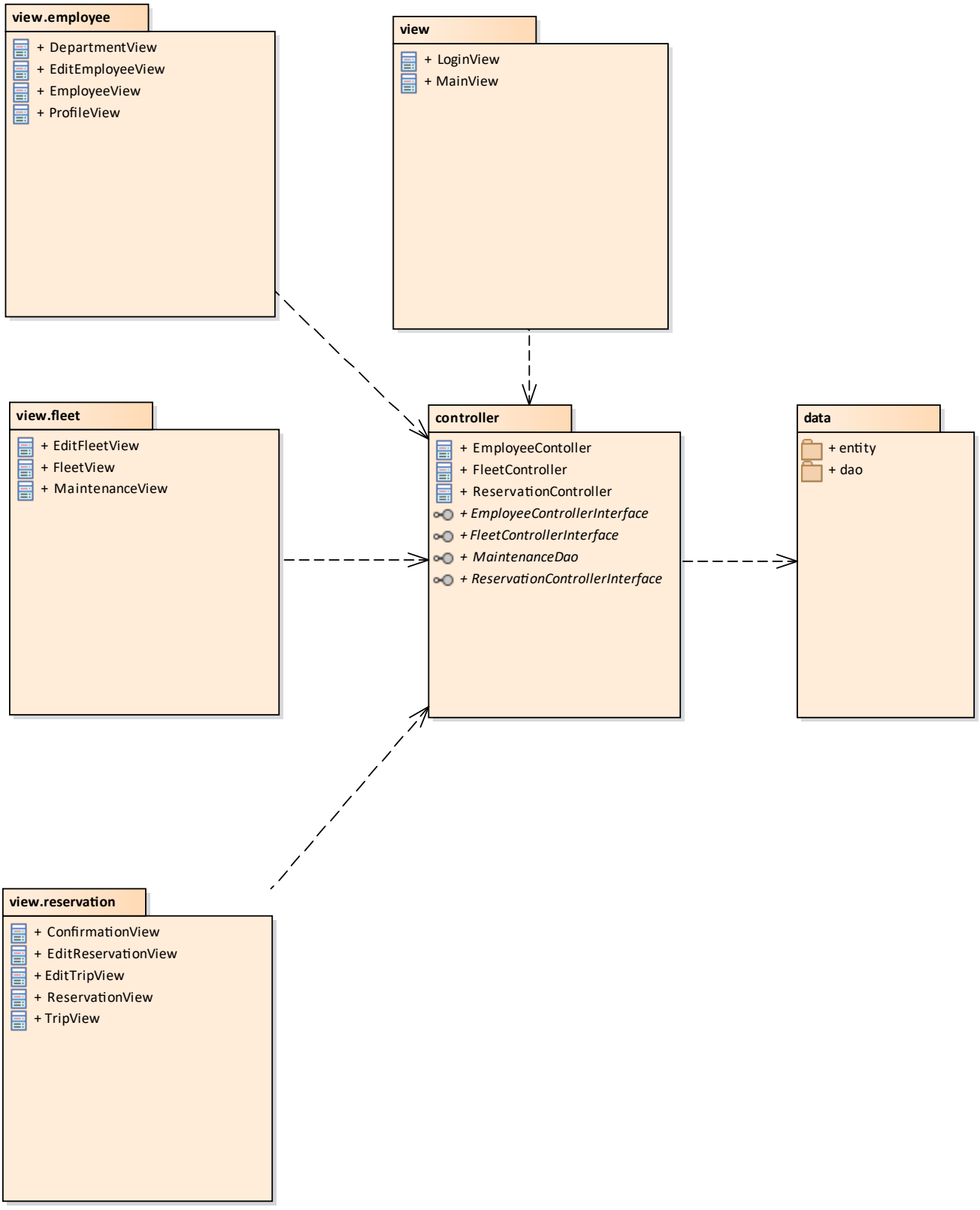

view.employee
+ DepartmentView
+ EditEmployeeView
+ EmployeeView
+ ProfileView
view
+ LoginView
+ MainView
view.fleet
+ EditFleetView
+ FleetView
+ MaintenanceView
controller
+ EmployeeContoller
+ FleetController
+ ReservationController
+ EmployeeControllerInterface
+ FleetControllerInterface
+ MaintenanceDao
+ ReservationControllerInterface
data
+ entity
+ dao
view.reservation
+ ConfirmationView
+ EditReservationView
+ EditTripView
+ ReservationView
+ TripView

Figure 2 - Design Class Model

2.1 view.employee

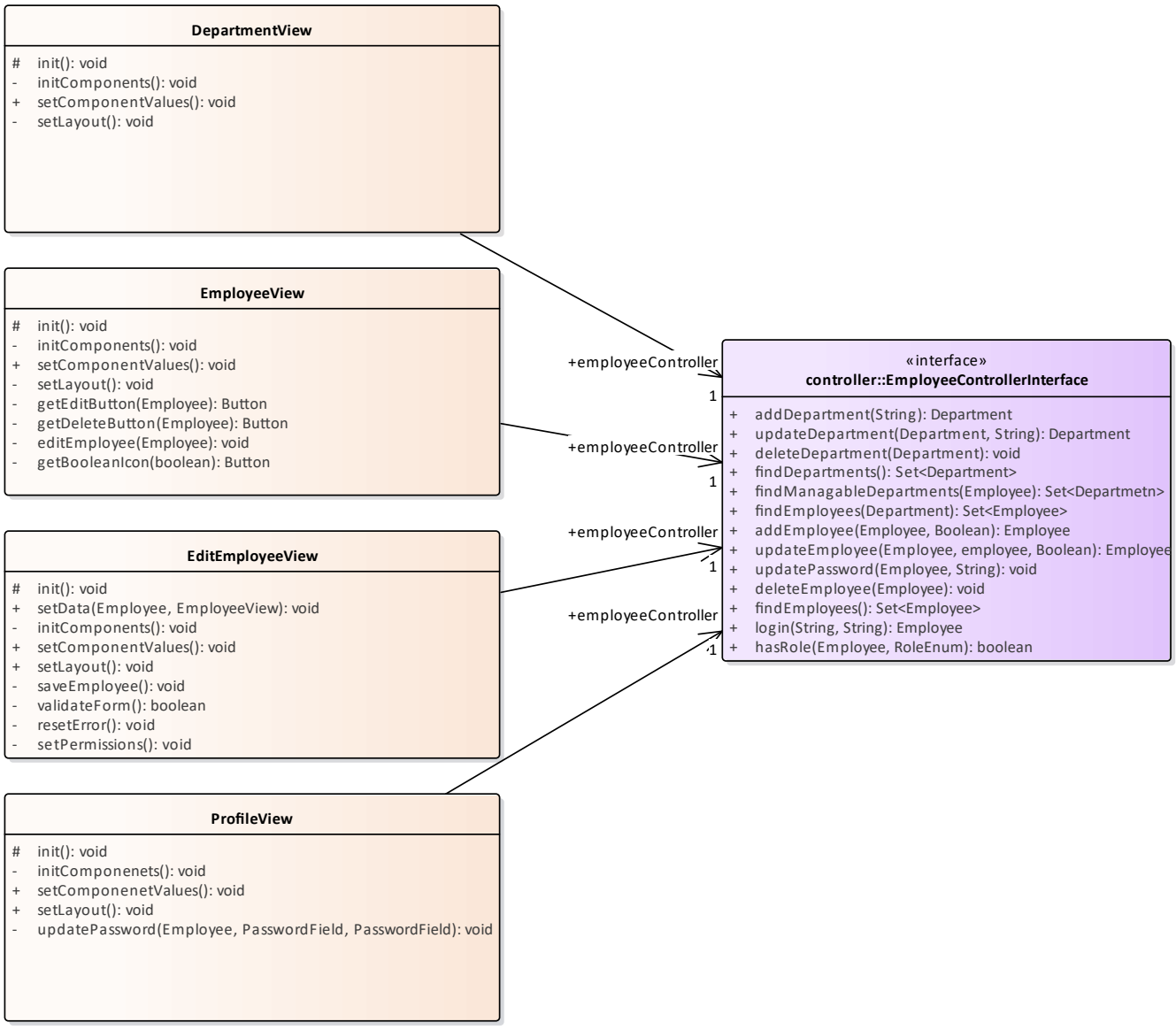


Figure 3 - view.employee

2.2 view.fleet

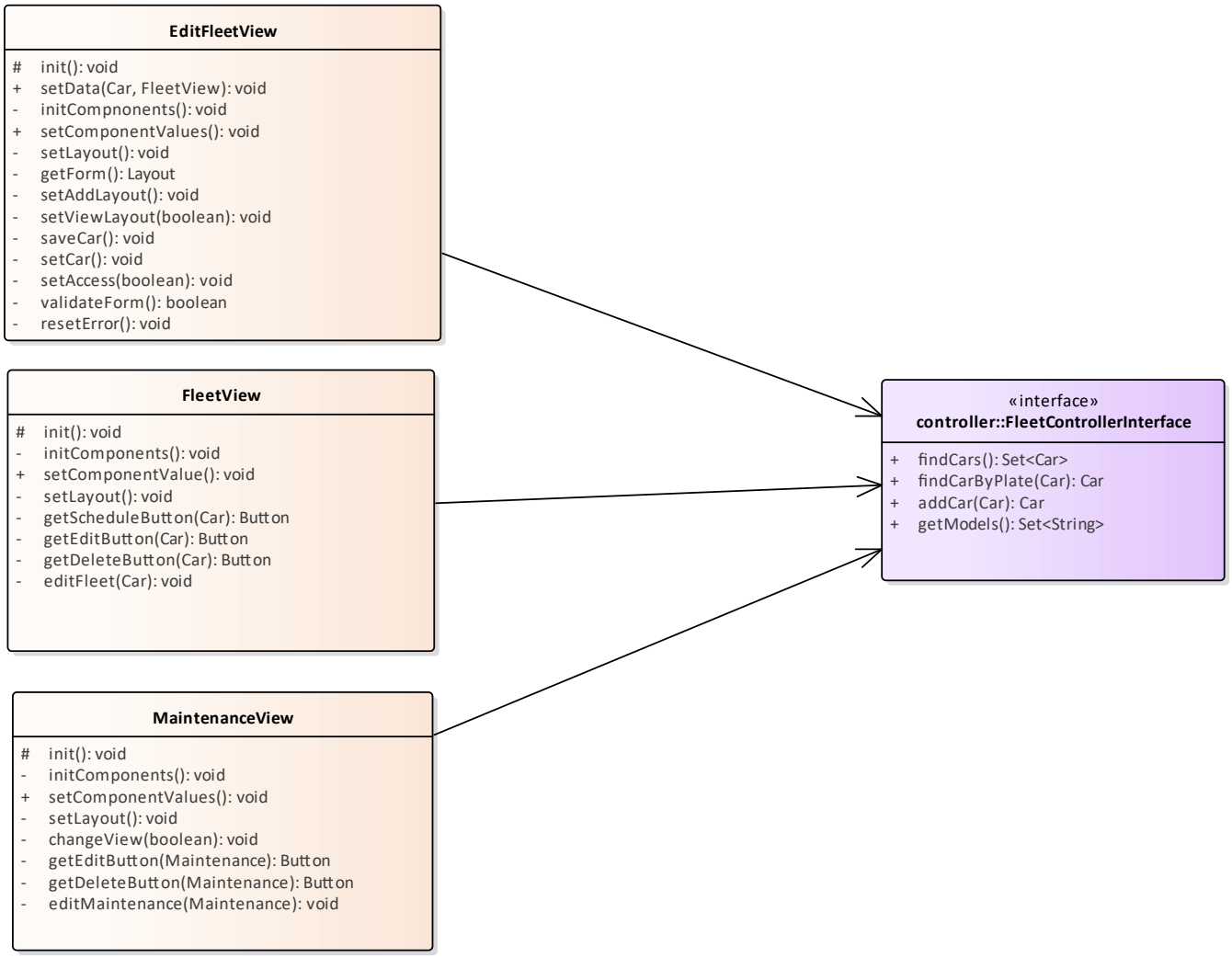


Figure 4 - view.fleet

2.3 view.reservation

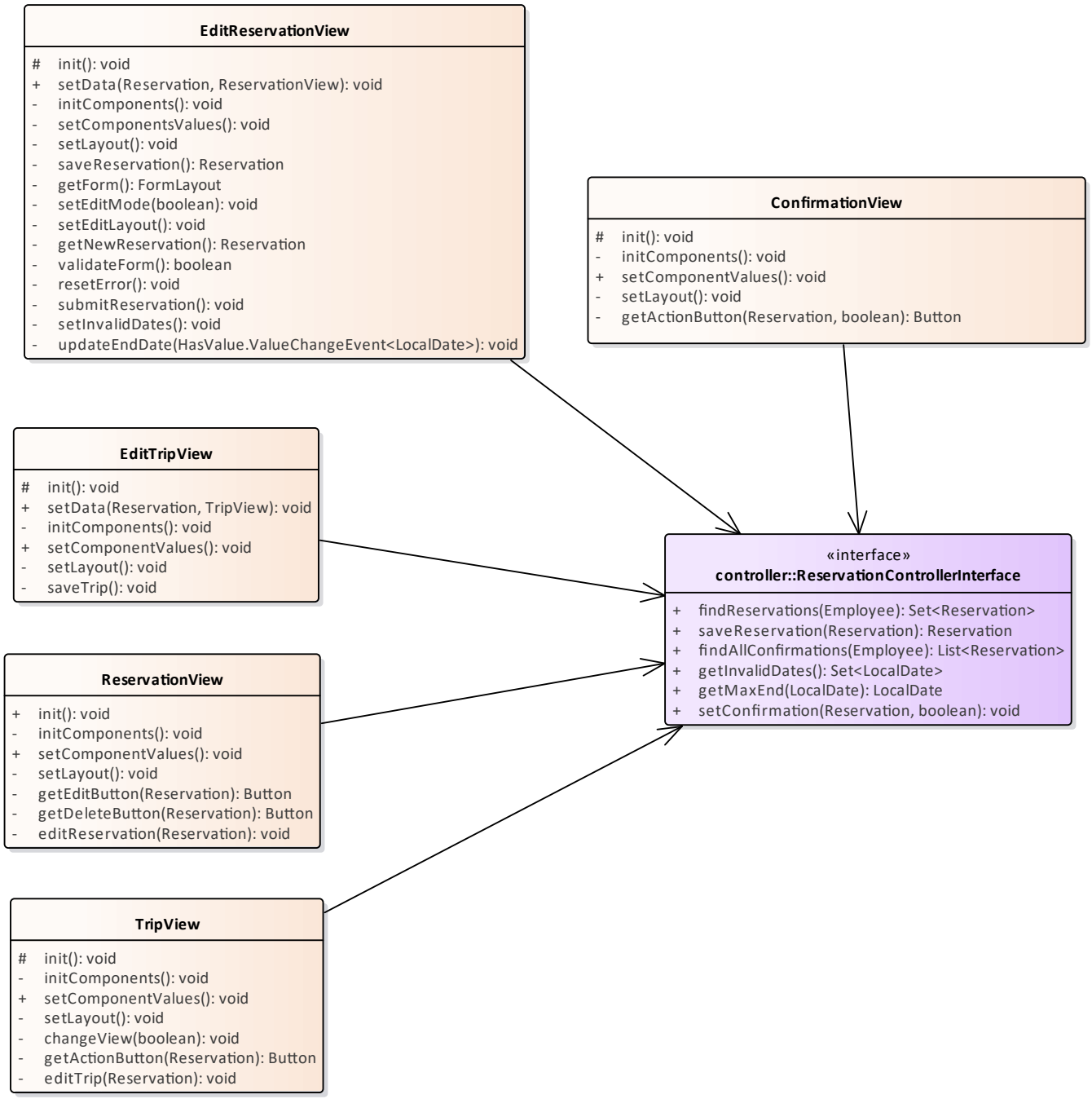


Figure 5 - view.reservation

2.4 view

The web package contains classes implementing the web user interface.

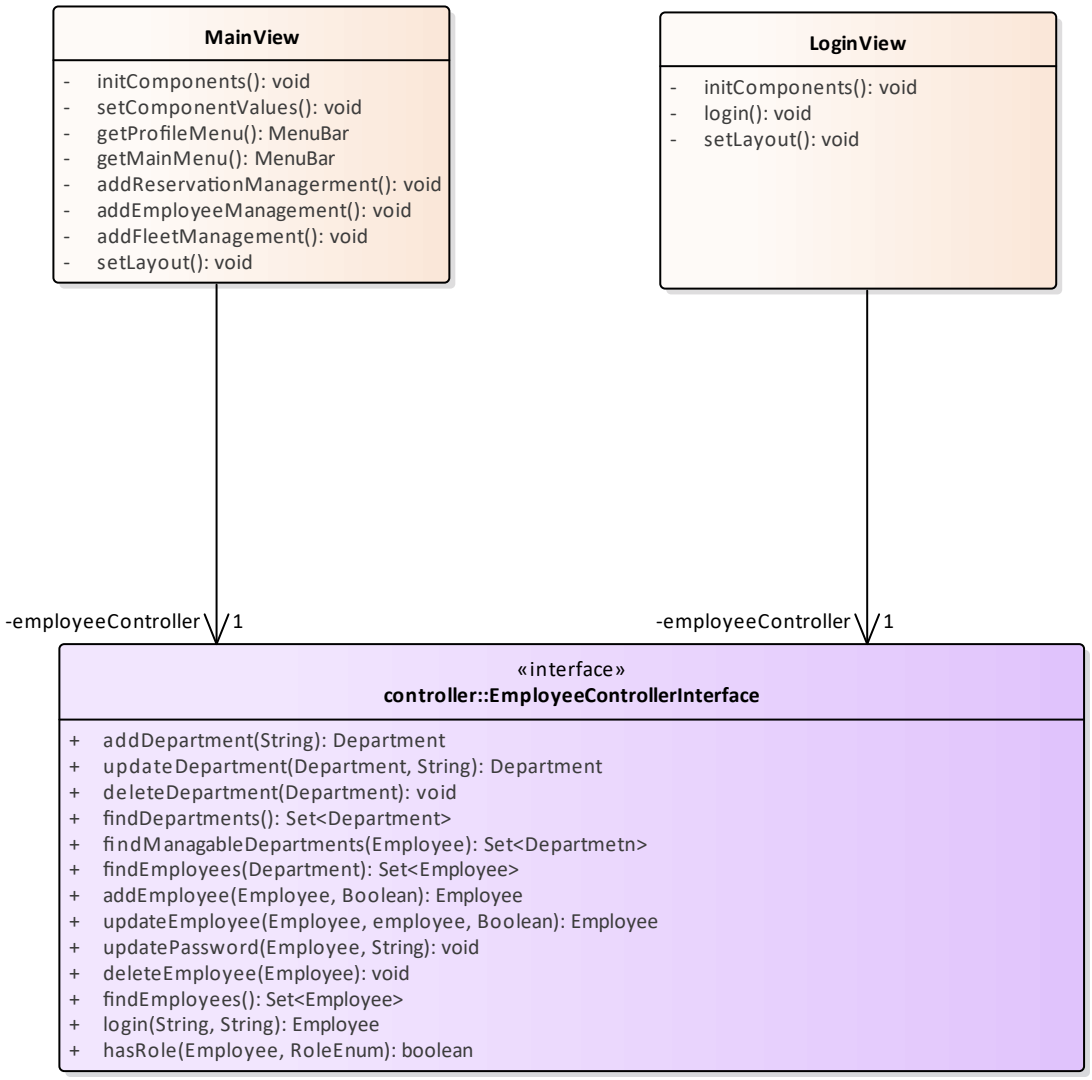


Figure 6 - view

2.5 controller

The model package contains the classes and packages of the business layer.

It contains the controller classes and interfaces, implementing the business logic of the system.

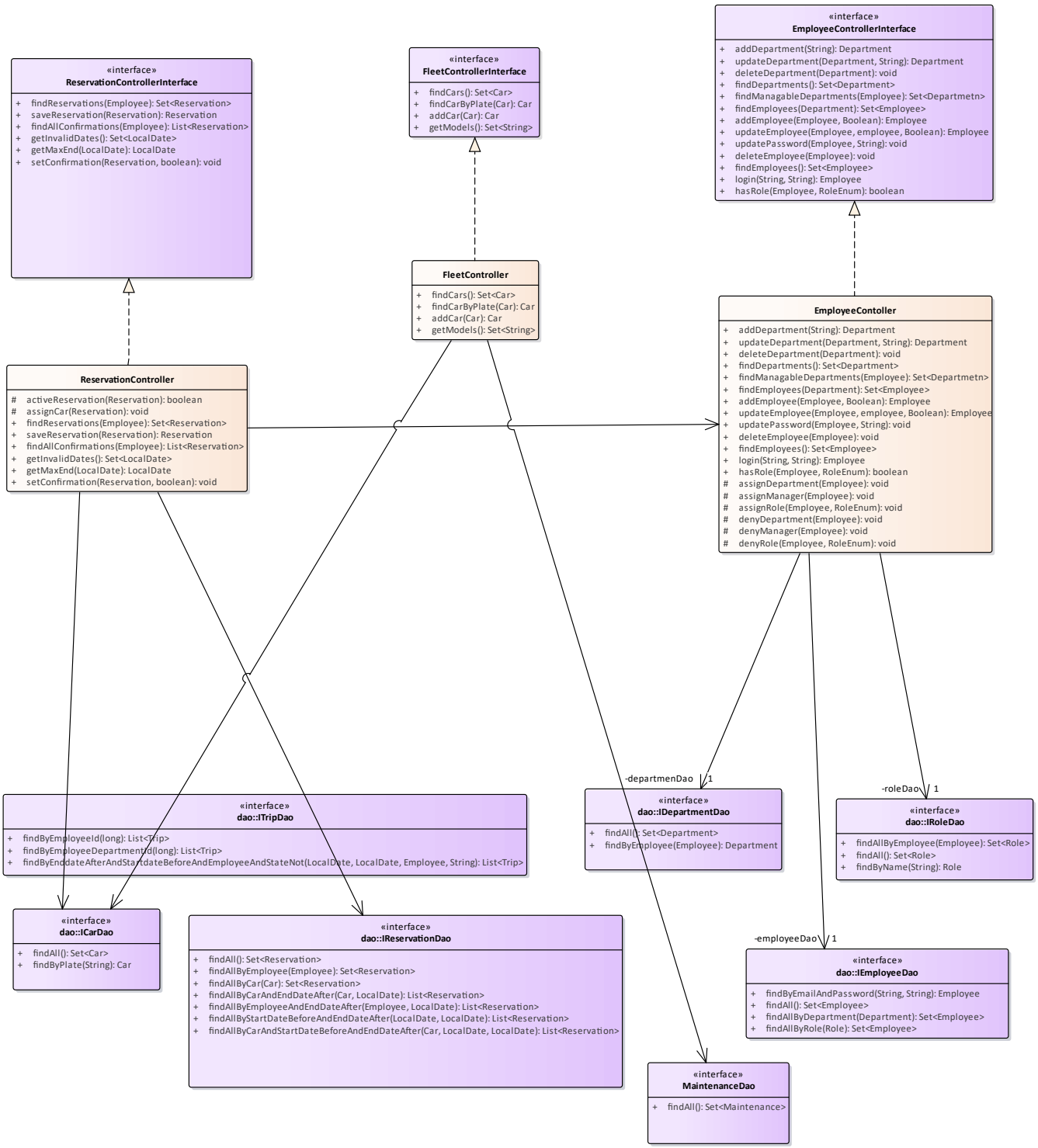


Figure 7 - ReservationController

2.5.1 ReservationControllerInterface

Interface operations related to reservation management use cases.

Method name	Return type	Description
findReservations	Set<Reservation>	Parameters: **user: Employee** -
saveReservation	Reservation	Parameters: **reservation: Reservation** -
findAllConfirmations	List<Reservation>	Parameters: **user: Employee** -
getInvalidDates	Set<LocalDate>	
getMaxEnd	LocalDate	Parameters: **value: LocalDate** -
setConfirmation	void	Parameters: **res: Reservation** - Parameters: **b: boolean** -

2.6 data

This package contains the classes and interfaces of the data layer.

Figure 8 - data

2.6.1 entity

This package contains the classes of the entities, defining the data objects synchronized with a persistent storage.

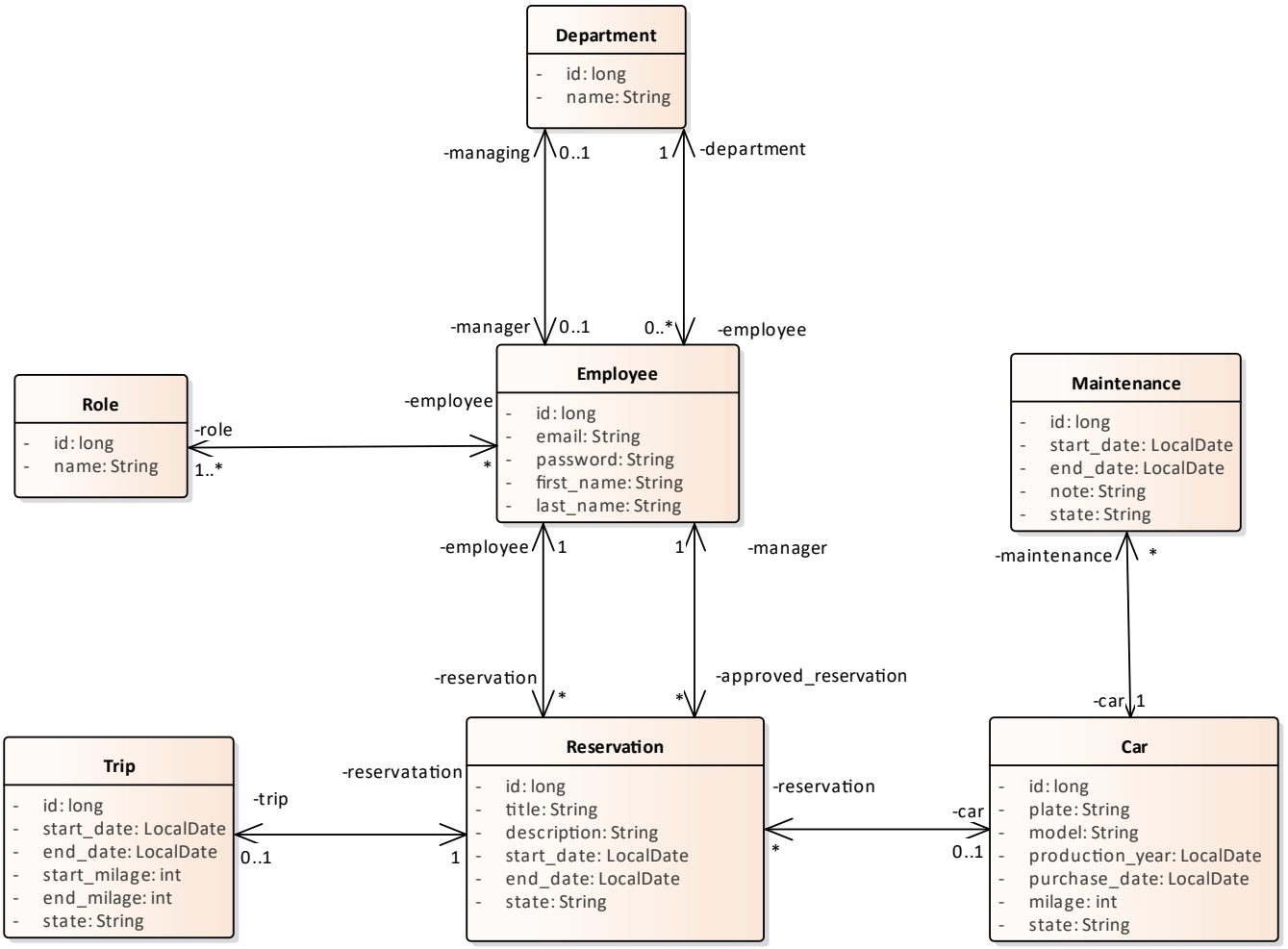


Figure 9 - entity

2.6.1.1 Car

The Car entity serves as a data holder for the data about car stored in a persistent storage.

Car represents information about a car which may be reserved for a business trip.

A car may be used in none or many reservations.

Attribute name	Data type	Description
id	long	Internal ID of car entity.
plate	String	Unique plate number of the car.
model	String	Model of the car.
production_year	LocalDate	
purchase_date	LocalDate	
milage	int	
state	String	State of the car (eg. OK, damaged).

2.6.1.2 Department

The Department entity serves as a data holder for the data about departments stored in a persistent storage.

Department represents information about a departments of the company.

A department must always have an employee managing it. It may also have one or many employees working.

Attribute name	Data type	Description
id	long	Internal ID of the department entity.
name	String	Name of the department.

2.6.1.3 Employee

The Employee entity serves as a data holder for the data about employees stored in a persistent storage.

Employee represents information about an employee working in the company.

An employee must have at least one role, also a department that s/he working for. It may also have a department that s/he is managing.

Attribute name	Data type	Description
id	long	Internal ID of the employee entity.
email	String	Unique email of the user.
password	String	Password of the user used to log in the system.
first_name	String	
last_name	String	

2.6.1.4 Reservation

The Trip entity serves as a data holder for the data about trips stored in a persistent storage.

Trip represents information about a business trips which are requested by employees.

A trip must always have an employee that requested it and also an employee to approve it, it may also have zero or one reservation requests.

Attribute name	Data type	Description
id	long	Internal ID of the trip entity.
title	String	Customer to be visited during the trip.
description	String	Optional short description about the trip.
start_date	LocalDate	Planned start date of the trip.
end_date	LocalDate	Planned end date of the trip.
state	String	State of the trip (eg. new, accepted, rejected).

2.6.1.5 Role

The Role entity serves as a data holder for the data about roles stored in a persistent storage.

Reservation represents information about a reservation which may be done for a business trip.

A reservation must always have a trip related to it, it may also have zero or one car according to availability.

Attribute name	Data type	Description
id	long	Internal ID of the role entity.
name	String	Name of the role.

2.6.1.6 Trip

The Reservation entity serves as a data holder for the data about reservations stored in a persistent storage.

Reservation represents information about a reservation which may be done for a business trip.

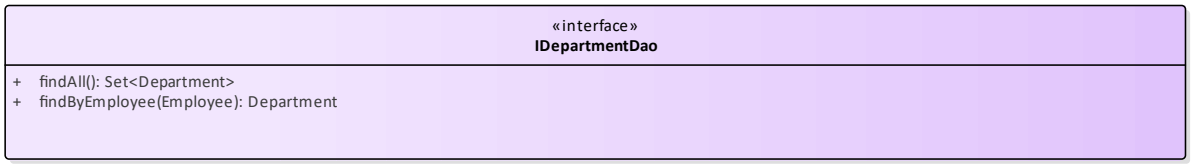
A reservation must always have a trip related to it, it may also have zero or one car according to availability.

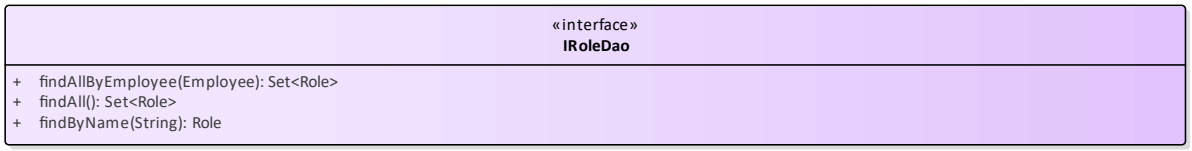
Attribute name	Data type	Description
id	long	Internal ID of reservation.
start_date	LocalDate	Realization start date of the reservation.
end_date	LocalDate	Realization end date of the reservation.
start_milage	int	
end_milage	int	
state	String	State of the reservation (eg. new, started, ended, expired).

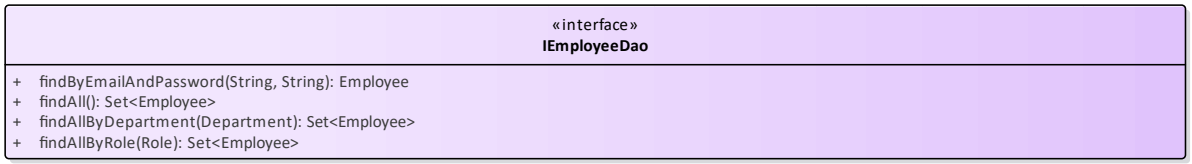
2.6.2 dao

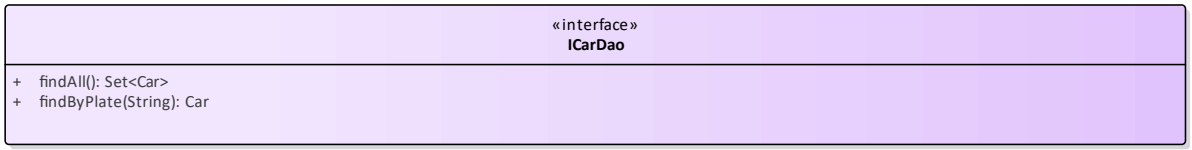
This package defines the interfaces for DAOs - Data access objects - responsible for retrieving data from the persistent storage and persisting the changes. It maps the database tables to entities of the system.

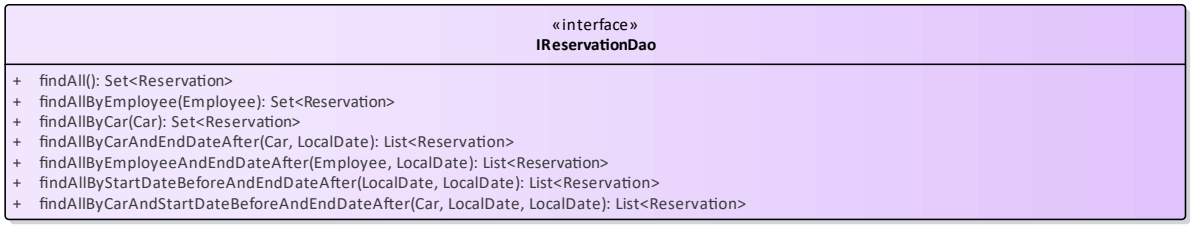
Spring Boot's generic interfaces allow usage of the interface without implementation and also does not require to specify basic database operations such as getById() or findAll().


«interface»
IDepartmentDao
+ findAll(): Set<Department>
+ findByEmployee(Employee): Department


«interface»
IRoleDao
+ findAllByEmployee(Employee): Set<Role>
+ findAll(): Set<Role>
+ findByName(String): Role


«interface»
IEmployeeDao
+ findByEmailAndPassword(String, String): Employee
+ findAll(): Set<Employee>
+ findAllByDepartment(Department): Set<Employee>
+ findAllByRole(Role): Set<Employee>


«interface»
ICarDao
+ findAll(): Set<Car>
+ findByPlate(String): Car


«interface»
IReservationDao
+ findAll(): Set<Reservation>
+ findAllByEmployee(Employee): Set<Reservation>
+ findAllByCar(Car): Set<Reservation>
+ findAllByCarAndEndDateAfter(Car, LocalDate): List<Reservation>
+ findAllByEmployeeAndEndDateAfter(Employee, LocalDate): List<Reservation>
+ findAllByStartDateBeforeAndEndDateAfter(LocalDate, LocalDate): List<Reservation>
+ findAllByCarAndStartDateBeforeAndEndDateAfter(Car, LocalDate, LocalDate): List<Reservation>

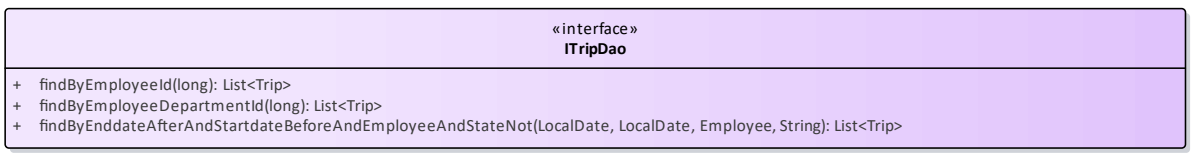

«interface»
ITripDao
+ findByEmployeeId(long): List<Trip>
+ findByEmployeeDepartmentId(long): List<Trip>
+ findByEnddateAfterAndStartdateBeforeAndEmployeeAndStateNot(LocalDate, LocalDate, Employee, String): List<Trip>

Figure 10 - dao

2.6.2.1 ICarDao

Interface defining the operations available for persistence of cars.

Method name	Return type	Description
findAll	Set<Car>	
findByPlate	Car	Parameters: **plate: String** -

2.6.2.2 IDepartmentDao

Interface defining the operations available for persistence of departments.

Method name	Return type	Description
findAll	Set<Department>	
findByEmployee	Department	Parameters: **employee: Employee** -

2.6.2.3 IEmployeeDao

Interface defining the operations available for persistence of employee.

Method name	Return type	Description
findByEmailAndPassword	Employee	Parameters: **email: String** - Parameters: **password: String** -
findAll	Set<Employee>	
findAllByDepartment	Set<Employee>	Parameters: **department: Department** -
findAllByRole	Set<Employee>	Parameters: **role: Role** -

2.6.2.4 IReservationDao

Interface defining the operations available for persistence of reservations.

Method name	Return type	Description
findAll	Set<Reservation>	
findAllByEmployee	Set<Reservation>	Parameters: **user: Employee** -
findAllByCar	Set<Reservation>	Parameters: **c: Car** -
findAllByCarAndEndDateAfter	List<Reservation>	Parameters: **c: Car** - Parameters: **value: LocalDate** -
findAllByEmployeeAndEndDateAfter	List<Reservation>	Parameters: **e: Employee** - Parameters: **now: LocalDate** -
findAllByStartDateBeforeAndEndDateAfter	List<Reservation>	Parameters: **plusDays: LocalDate** - Parameters: **minusDays: LocalDate** -
findAllByCarAndStartDateBeforeAndEndDateAfter	List<Reservation>	Parameters: **c: Car** - Parameters: **plusDays: LocalDate** - Parameters: **minusDays: LocalDate** -

2.6.2.5 IRoleDao

Interface defining the operations available for persistence of roles.

Method name	Return type	Description
findAllByEmployee	Set<Role>	Parameters: **employee: Employee** -
findAll	Set<Role>	
findByName	Role	Parameters: **name: String** -

Method name	Return type	Description

2.6.2.6 ITripDao

Interface defining the operations available for persistence of trips.

Method name	Return type	Description
findByEmployeeId	List<Trip>	Parameters: **id: long** -
findByEmployeeDepartmentId	List<Trip>	Parameters: **id: long** -
findByEnddateAfterAndStartdateBeforeAndEmployeeAndStateNot	List<Trip>	Parameters: **start_date: LocalDate** - Parameters: **end_date: LocalDate** - Parameters: **employee: Employee** - Parameters: **state: String** -

3. Class Diagram

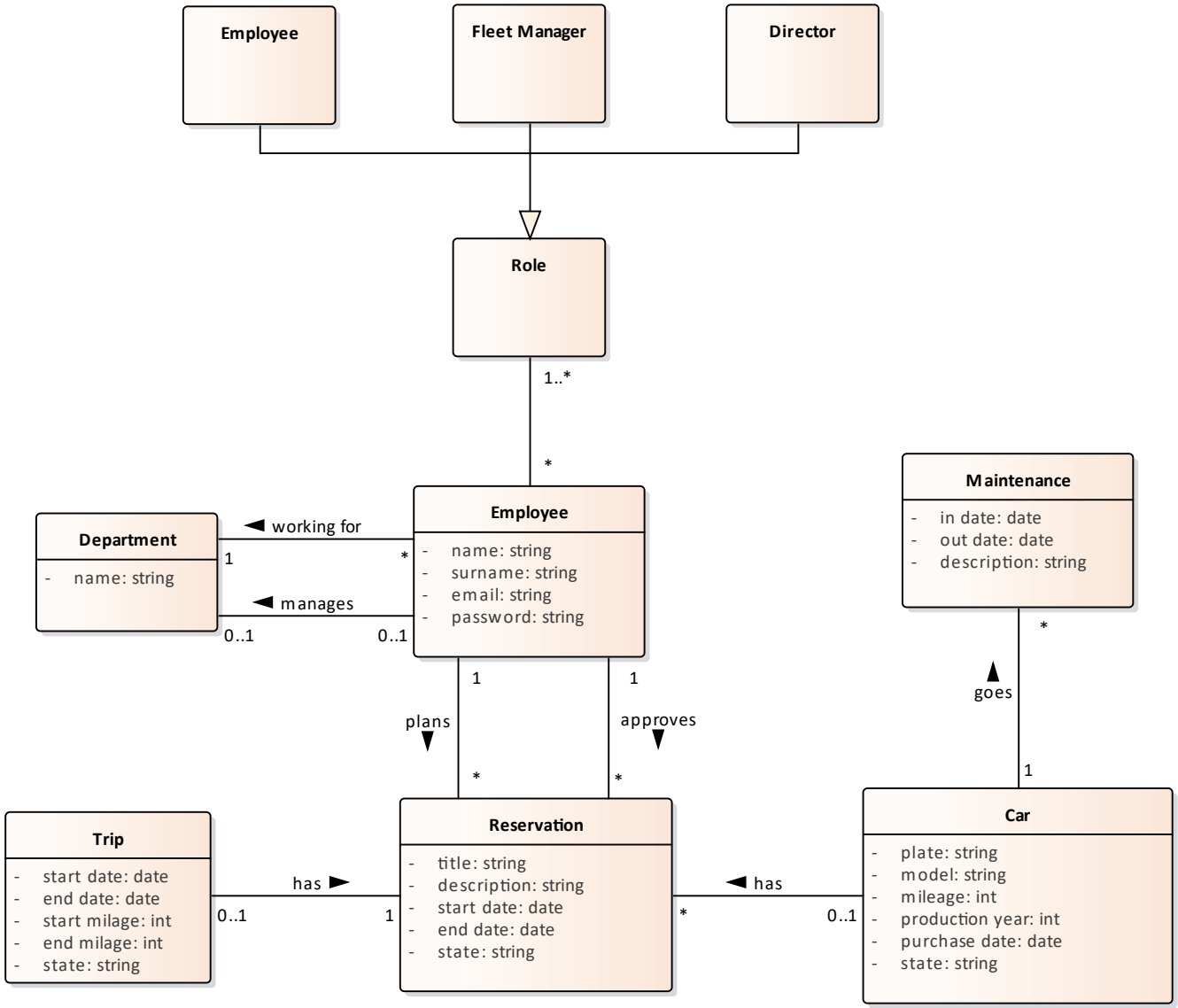


Figure 11 - ClassDiagram

3.1 Car

Car represents the cars in the company.
Each car may have zero or many maintenance that it was sent.
Also a car may be involved in zero or many reservations.

Attribute name	Data type	Description
plate	string	Plate number of the car stored as string.
model	string	Model of the car stored as string.
mileage	int	Current mileage of the car stored as integer.
production year	int	Production year of the car stored as integer.
purchase date	date	Next planned maintenance date for the car stored as date.
state	string	State of the car(eg. available, damaged, in maintenance etc.) stored as string.

3.1.1 CarStateMachine_Availability

The following diagram shows all possible availability state transitions for the car object.

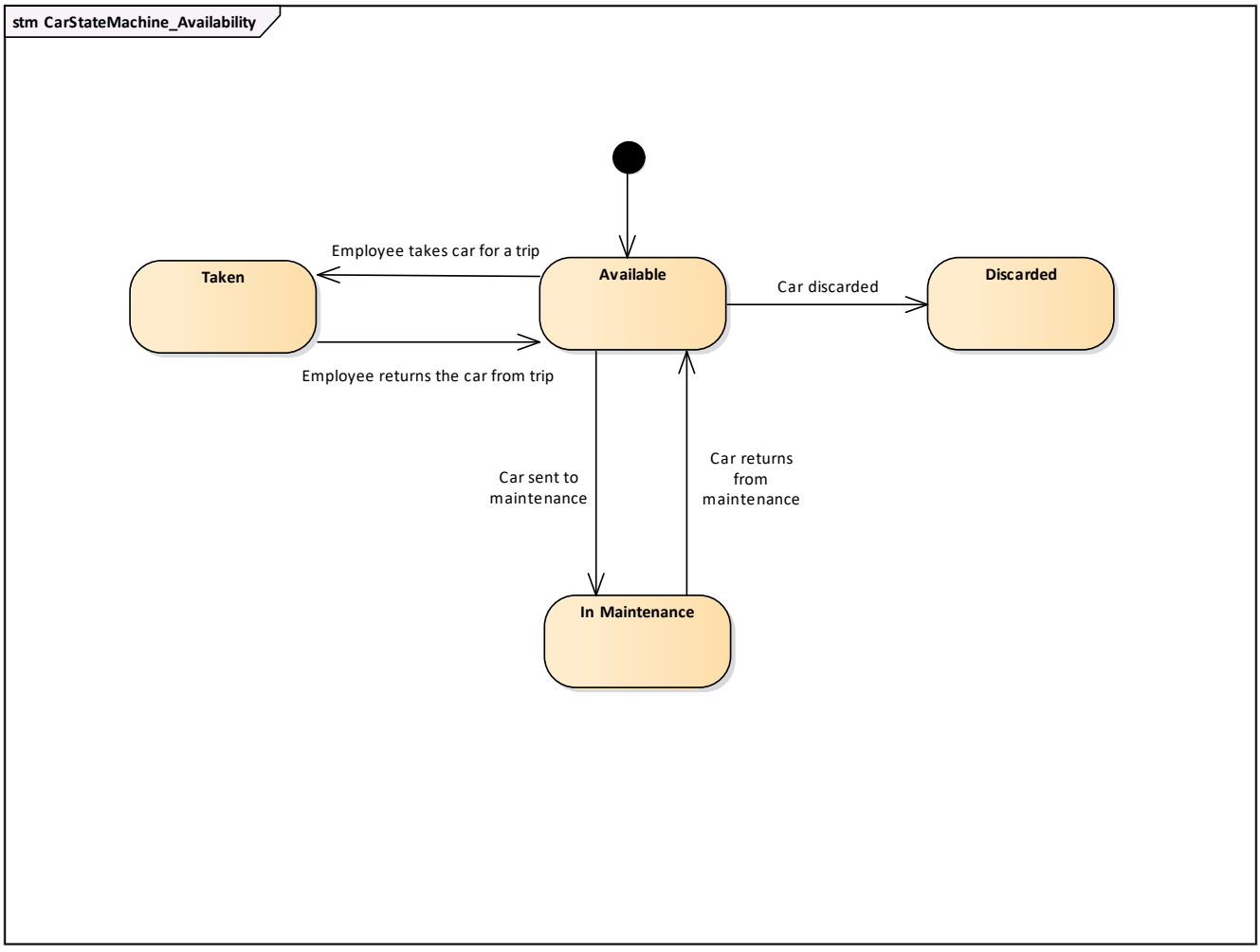


Figure 12 - CarStateMachine_Availability

3.1.2 CarStateMachine_Condition

The following diagram shows all possible condition state transitions for the car object.

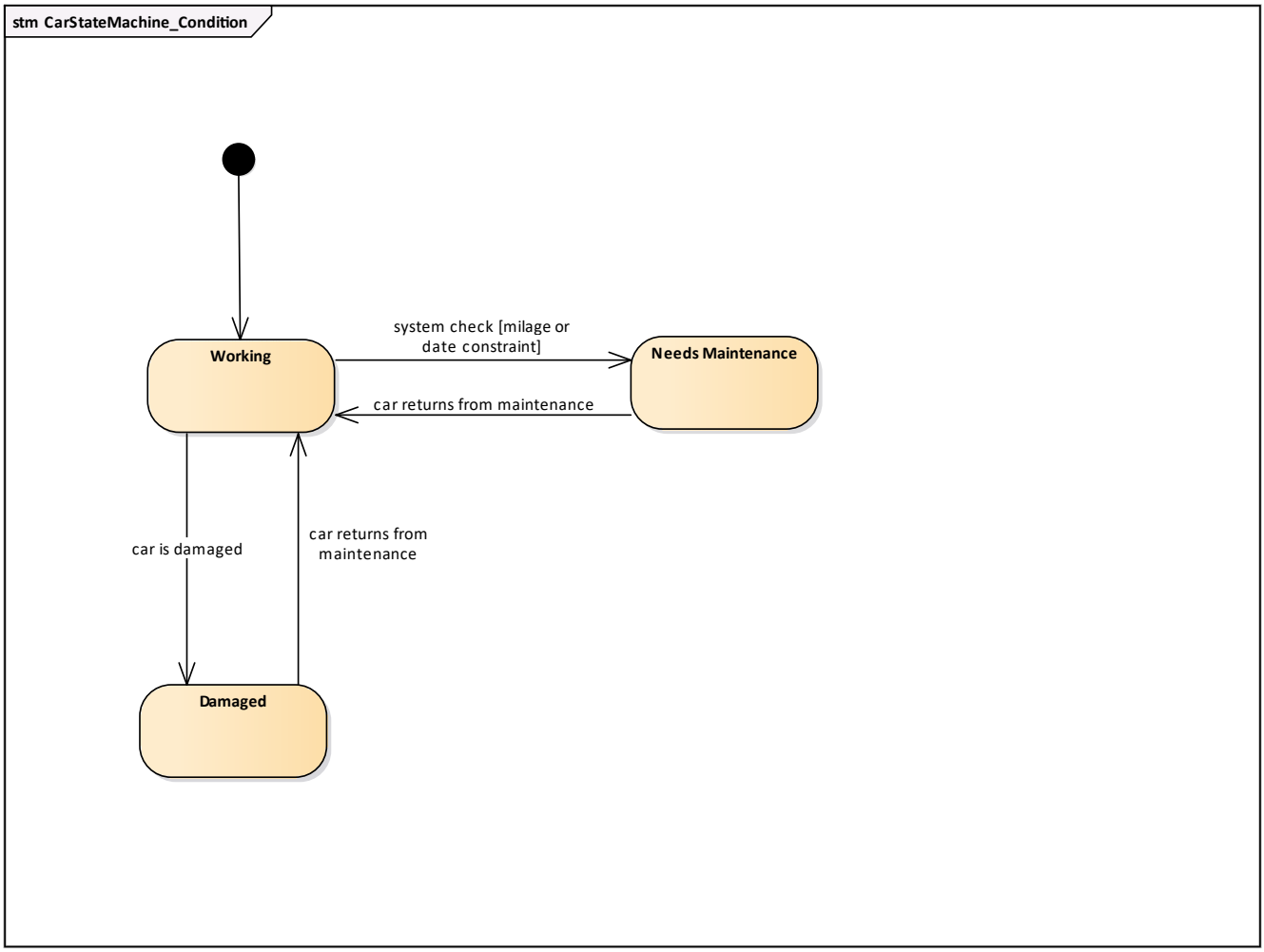


Figure 13 - CarStateMachine_Condition

3.2 Department

Department represents different departments in the company.
Each department is managed by exactly one employee.
Each department may have oneor more employees.

Attribute name	Data type	Description
name	string	Name of the department stored as a string.

3.3 Employee

Employee represents a person working in the company.
According to responsibilities each employee has one or many roles in the company.
Each employee has a department that he is working for and also some employees manages some departments.
Employee may have only one department that he is working for.
It is possible that employee does not manage any departments.
Also each employee may have zero or many business trips.

Attribute name	Data type	Description
name	string	Name of employee stored as string.
surname	string	Surname of employee stored as string.
email	string	Email of employee stored as string.
password	string	

3.4 Maintenance

Maintenance represents the maintenance history of each car in the company.
Maintenance must have exactly one car.

Attribute name	Data type	Description
in date	date	The date in which car was sent to maintenance stored as date.
out date	date	The date in which car was taken from maintenance stored as date.
description	string	General description about why car was sent to maintenance and what was done during maintenance stored as string.

3.5 Reservation

Reservation represents a car reservation made during the planning of a business trip.
It contains information about the trip and the car which was reserved for the given dates.
Each reservation must have trip object.
Each reservation may have a car, which represents a successful reservation.
Each reservation may lead to a loan, unless it was canceled.

Attribute name	Data type	Description
title	string	Mileage of the car when employee is taking the car stored as integer.
description	string	Mileage of the car when employee is returning the car stored as integer.
start date	date	Start date of realized reservation stored as date.
end date	date	End date of realized reservation stored as date.
state	string	State of the reservation(eg. accepted, rejected, canceled, realized) stored as string.

3.5.1 ReservationStateMachine

This diagram shows all possible state transitions for the reservation object.

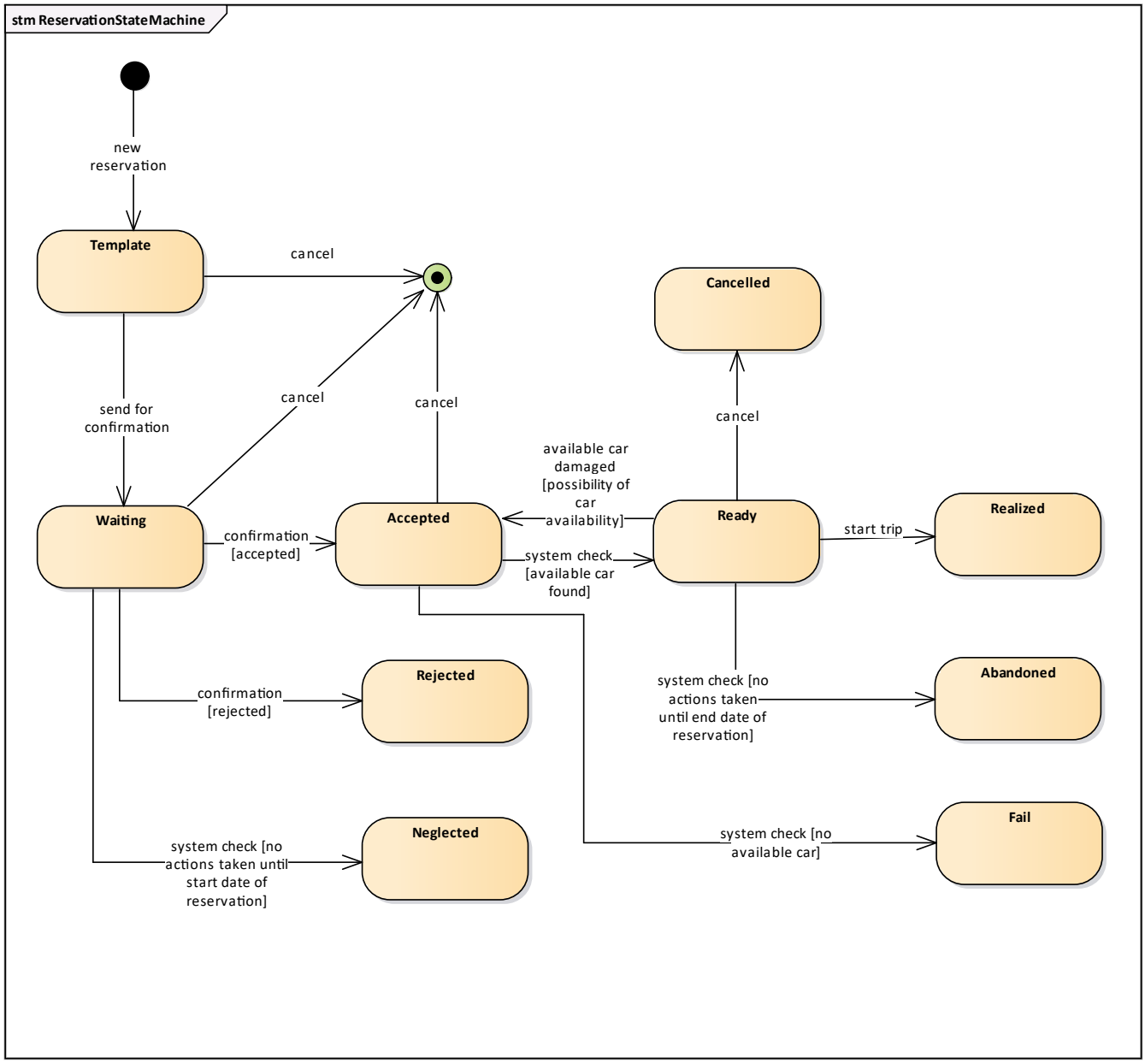


Figure 14 - ReservationStateMachine

3.6 Role

Role represents the role of an employee in the company.
Each role may have zero or many employees.

3.7 Trip

Trip represents a business trip that was planned by an employee.

Each trip needs to be approved by exactly one DM, who is managing the department who made the trip request.
Each trip also must have an employee who planned it.

Attribute name	Data type	Description
start date	date	Trip start date stored as date.
end date	date	Trip end date stored as date.
start milage	int	
end milage	int	
state	string	State of the trip(eg. accepted, rejected, cancelled) stored as string.

3.7.1 TripStateMachine

This diagram shows all possible state transitions for the trip class.

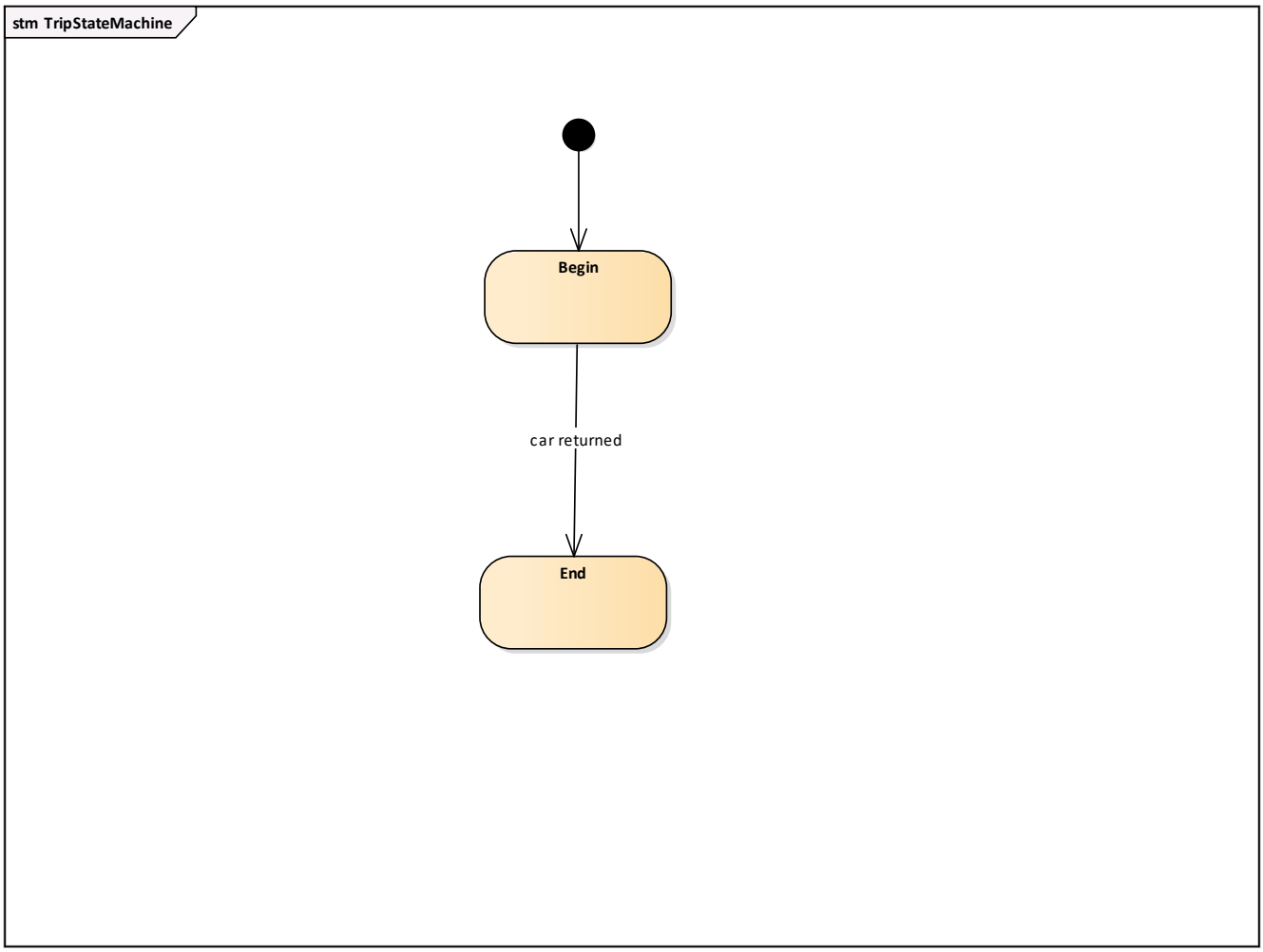


Figure 15 - TripStateMachine

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE