

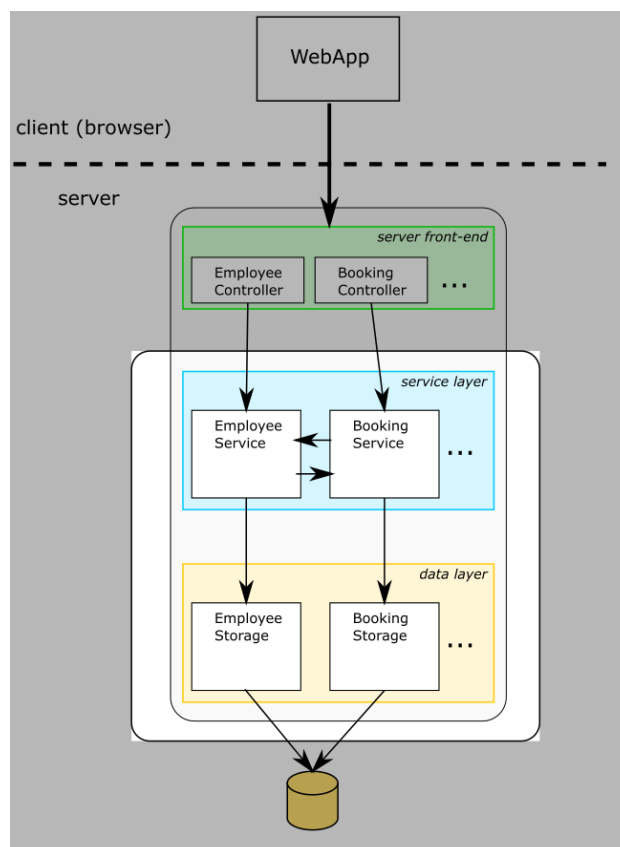
Assignment #3

Anders Kalhauge and Martin Vestergaard

Fall 2020

Booking system

You are going to work on a skeleton of a simple booking system. The idea is for customers to book appointments with an employee. The overview of the system is illustrated below. For now, we ignore the greyed-out parts.



Requirements

It must be possible to create customers, employees and bookings. A customer may have a phone number (this change requires a database migration script). When booking an appointment with a customer, an SMS must be sent¹.

Data layer

1. Create `BookingStorage`² and `BookingStorageImpl`³ with methods
 - `int createBooking(Booking booking)`⁴
 - `Collection<Booking> getBookingsForCustomer(int customerId)`
2. Create `EmployeeStorage` and `EmployeeStorageImpl` with methods
 - `int createEmployee(Employee employee)`⁵
 - `Collection<Employee> getEmployeeWithId(int employeeId)`

Service layer

1. Create `BookingService` and `BookingServiceImpl` with methods
 - `int createBooking(customerId, employeeId, date, start, end)`
 - `Collection<Booking> getBookingsForCustomer(customerId)`
 - `Collection<Booking> getBookingsForEmployee(employeeId)`
2. Create `EmployeeService` and `EmployeeServiceImpl` with methods
 - `int createEmployee(employee)`
 - `Employee getEmployeeById(employeeId)`

The repository

The project can be found at the following location: <https://github.com/mvdk-cph/test2020fall-BookingSystem>.

¹we only have the interface to the SMS service – imagine it's being developed elsewhere. But the behavior of calling the sms service must be verified in a unit test.

²interface

³implementation

⁴returns id of new booking

⁵returns id of new employee

The setup

Data is stored in a MySQL database with the following tables:

Listing 1: Customers table

```
create table Customers (  
  ID int not null auto_increment,  
  firstname varchar(255) not null,  
  lastname varchar(255),  
  birthdate date,  
  PRIMARY KEY (ID)  
)
```

Listing 2: Employees table

```
create table Employees (  
  ID int not null auto_increment,  
  firstname varchar(255) not null,  
  lastname varchar(255) not null,  
  birthdate date,  
  PRIMARY KEY (ID)  
)
```

Listing 3: Bookings table

```
create table Bookings (  
  ID int not null auto_increment,  
  customerId int not null,  
  employeeId int not null,  
  date Date not null,  
  start Time not null,  
  end Time not null,  
  primary key (ID),  
  foreign key (customerId)  
    references Customers(ID)  
    on delete cascade,  
  foreign key (employeeId)  
    references Employees(ID)  
    on delete cascade  
)
```

Running the MySQL server

If you don't want to pollute your workstation with a MySql server installation, you can easily run mysql in a docker container. That can be achieved by running the following command.

```
docker run -d --rm \  
  --name mysql-test-db \  
  -e MYSQL_ROOT_PASSWORD=testuser123 \  
  -p 3307:3306 \  
  mysql
```

You are welcome to come up with a better and easier way to run the SQL server. For example, you could look into testcontainers (<https://www.testcontainers.org>).

Hand-in

May be done in groups. It's a good opportunity to try out pair programming. Hand-in on the date given in peergrade. The hand-in should be code in a repository or zip-file, and a README.md with the written answers.