Teletearbies:
Fernanda Cunha Bacelar
Amalie Vibe Larsen
Balázs Dombovári
Dániel Szabó

Module 4 - Roskilde Daycare

Introduction

In this project, we have used the following technologies: Java, Spring Boot, Thymeleaf, Angular and Hibernate.

Initially, we used JavaFX for building a GUI application. The development of the project with JavaFX seemed outdated, and we decided to apply our skills into challenging ourselves to build an application, using HTML, CSS, Bootstrap (web- development framework) and JavaScript, as front-end and Spring Boot, and all its dependencies, as a baseline for our back-end.

The decision of using these technologies was based on the fact Spring Boot is the most used framework for building enterprise system applications in Java. In Spring Boot, we learned about how to utilize annotations that can represent different layers of our application and get introduced to architecture components and its relationships.
Therefore, we were able to easily find documentation on how to solve problems while developing a web application using Spring.

Source code:

https://github.com/bacelarfc/Module4_Teleturbies

User instructions before first use:

#User instructions before first run:

#1: Run CREATE DATABASE IF NOT EXISTS daycare;

#2: Uncomment the "public void initialiseDatabase()" method in the "common/InitialData" class

#3: Run main method in Application

#4: Comment out the "public void initialiseDatabase()" method in "common/InitialData" class

#5: Have fun using the application!

CODE SNIPPETS

Teletearbies:
Fernanda Cunha Bacelar
Amalie Vibe Larsen
Balázs Dombovári
Dániel Szabó

Class: Children

It is an entity class.
@Entity is a domain object. It represents a table in a relational database, and each entity instance corresponds to a row in that table.

```
@Entity
@Table(name = "children")
public class Children {
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private Integer id;

        @Column(nullable = false, unique = false, length = 45, name =
"first_name")
        private String firstName;

        @Column(nullable = false, unique = false, length = 45, name =
"last_name")
        private String lastName;

        @Column(nullable = false, unique = false, length = 45, name =
"parent_first_name")
        private String parentFirstName;

        @Column(nullable = false, unique = false, length = 45, name =
"parent_last_name")
        private String parentLastName;

        @Column(length = 45, nullable = false, name = "address")
        private String address;
$
        @Column(length = 45, nullable = false, name = "phone_number")
        private String contactNumber;


        public Children(String firstName, String lastName, String
parentFirstName, String parentLastName, String address, String
contactNumber) {
                this.firstName = firstName;
                this.lastName = lastName;
                this.parentFirstName = parentFirstName;
                this.parentLastName = parentLastName;
                this.address = address;
                this.contactNumber = contactNumber;
        }

        public Children() {

        }
```

Class: ChildrenService

@Service annotates classes at the service layer. It is used to retrieve, delete, and update the application.

```
@Service
public class ChildrenService {

   @Autowired private ChildrenRepository childrenRepository;


    public void saveChildren(Children children) {
        childrenRepository.save(children);
    }

    public List<Children> getAllChildren() {
        return (List<Children>) childrenRepository.findAll();
    }

    public Children getChildren(Integer id) throws ChildrenNotFoundException
{
          Optional<Children> result = childrenRepository.findById(id);
          if (result.isPresent()) {
              return result.get();
          }
          throw new ChildrenNotFoundException("Could not find any Child with
id: " + id);
    }

    public void deleteChildren(Integer id) throws ChildrenNotFoundException {
        Long count = childrenRepository.countById(id);
        if (count == null || count == 0)
        {
            throw new ChildrenNotFoundException("Could not find child with id
"  + id);
        }

        childrenRepository.deleteById(id);
    }
}
```

Class: ChildrenController

@Controller determines what response to send back to a user when a user makes a browser request.
Usually @Controller is used in combination with @RequestMapping annotation handling requests.

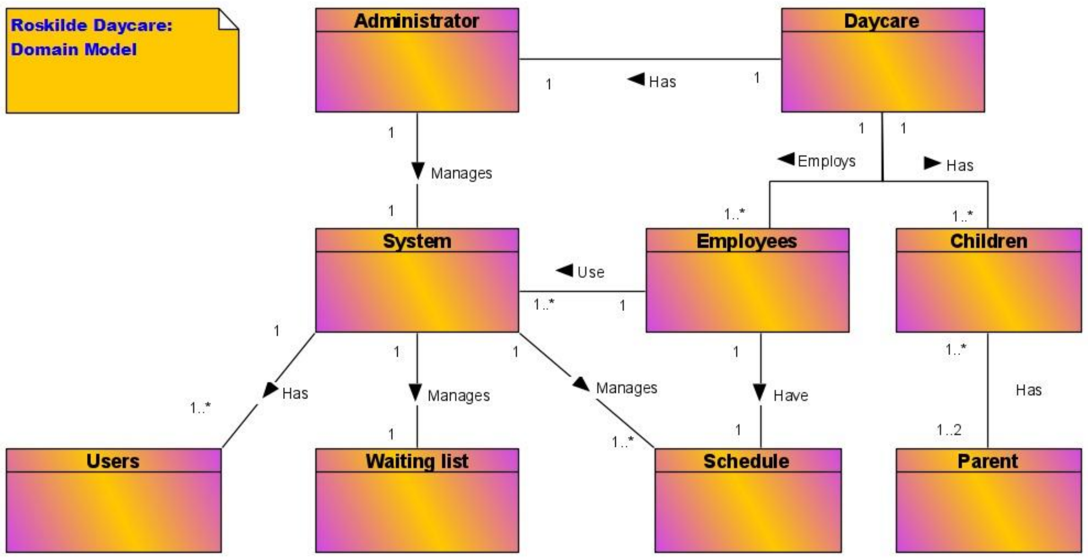
```
@Controller
public class ChildrenController {
    @Autowired
    private ChildrenService childrenService;


    @RequestMapping("/childrenList")
    public String displayChildrenList(Model model) {
        List<Children> listChildren = childrenService.getAllChildren();
        model.addAttribute("listChildren", listChildren);
        return "manage_children";
    }

    @RequestMapping("/children/add")
    public String addChildren(Model model) {
        model.addAttribute("children", new Children());
        return "children_form";
    }

    @PostMapping("/children/save")
    public String saveChildren(Children children, RedirectAttributes
redirectAttributes) {
        childrenService.saveChildren(children);
        redirectAttributes.addFlashAttribute("message", "Child info
saved!");
        return "redirect:/childrenList";
    }

    @RequestMapping("/admin")
    public String backToMainMenu() {
        return "admin_dashboard";
    }

    @RequestMapping("/children/edit/{id}")
    public String editChildren(@PathVariable("id") Integer id, Model model,
RedirectAttributes redirectAttributes) {
```

```
        try {
            Children children = childrenService.getChildren(id);
            model.addAttribute("children", children);
            return "children_form"; //Maybe not good

        } catch (ChildrenNotFoundException e) {
            redirectAttributes.addFlashAttribute("message", e.getMessage());
            model.addAttribute("pageTitle", "edit user (ID: " + id + ")");
            return "redirect:/childrenList"; //Maybe not good
        }
    }


        @RequestMapping("/children/delete/{id}")
        public String deleteChildren (@PathVariable("id") Integer id,
RedirectAttributes redirectAttributes){
            try {
                childrenService.deleteChildren(id);
                 redirectAttributes.addFlashAttribute("message", "Child was
deleted!");
            } catch (ChildrenNotFoundException e) {
                redirectAttributes.addFlashAttribute("message",
e.getMessage());

            }
            return "redirect:/childrenList";


        }
    }
```

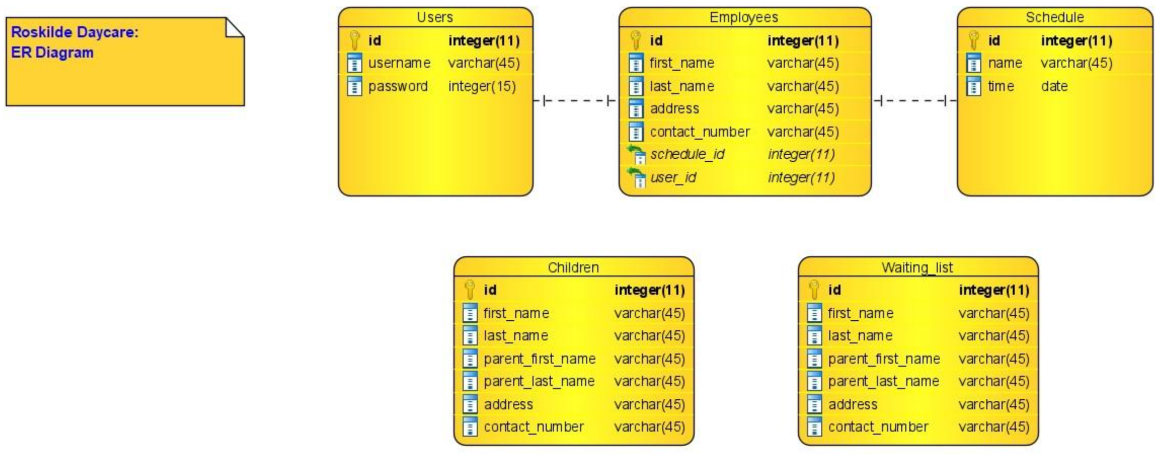
DIAGRAMS

Domain Model

Teletearbies:
Fernanda Cunha Bacelar
Amalie Vibe Larsen
Balázs Dombovári
Dániel Szabó

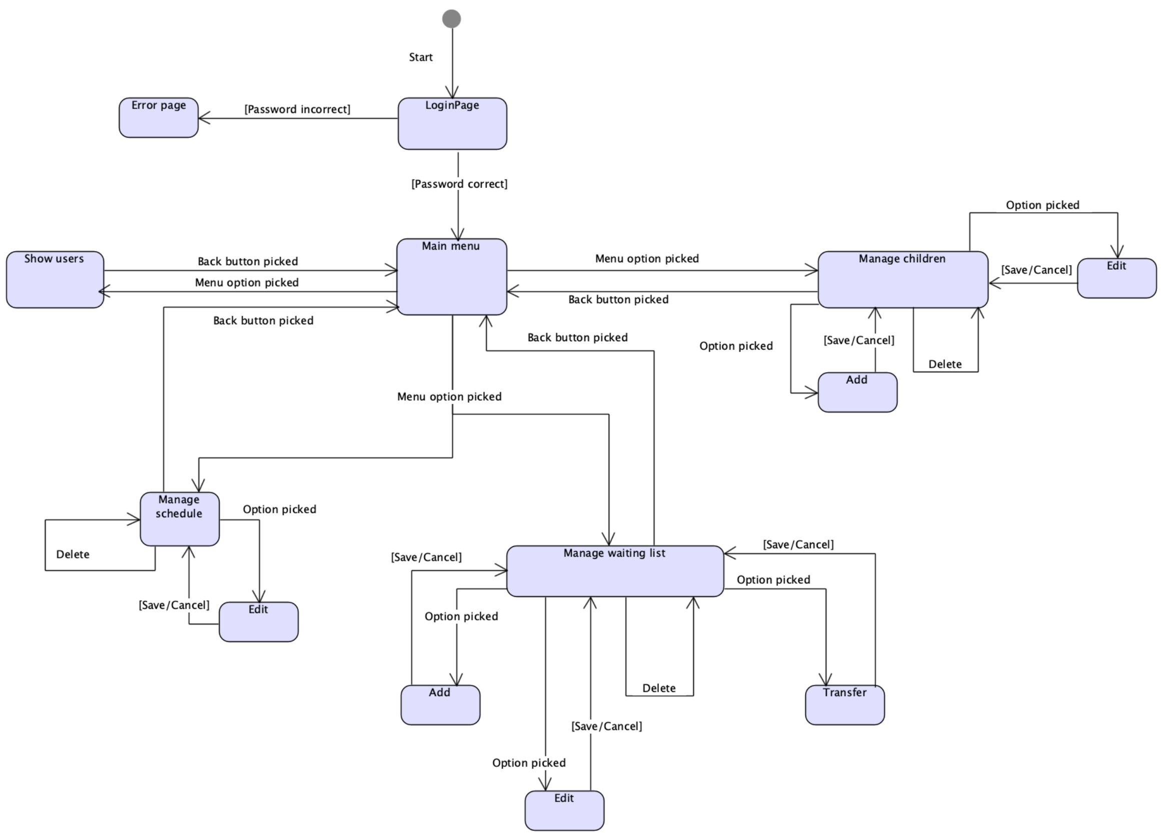


Entity Relationship Diagram

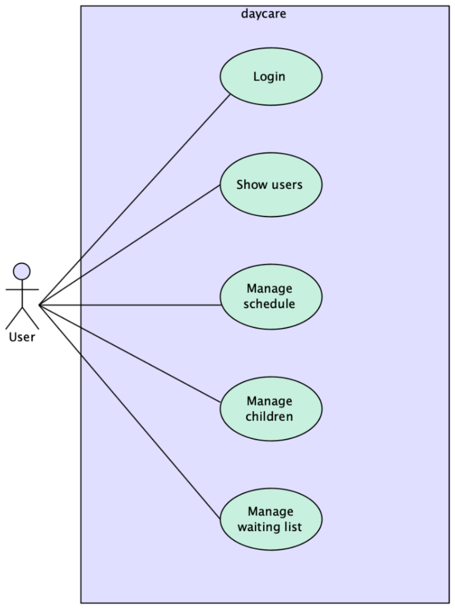


Teletearbies:
Fernanda Cunha Bacelar
Amalie Vibe Larsen
Balázs Dombovári
Dániel Szabó

State Machine Diagram



Use Case Diagram/Description



UC name: Login
Actor: user

Main Flow:
The user inputs accurate login info and is send to the main page.

Alternate flow:
User login is incorrect and user will be met with an error message.

UC name: Show users
Actor: user

Main Flow:
user is able to view users.

Alt. Flow:
User is met by error message.

UC name: Manage schedule
Actor: user

Main Flow:
user is able to edit and delete schedules.

Alt. Flow:
User is met by error message

UC name: Manage waiting list
Actor: user

Main Flow:
user is able to edit, add, transfer and delete children from the waiting list.

Alt. Flow:
User is met by error message

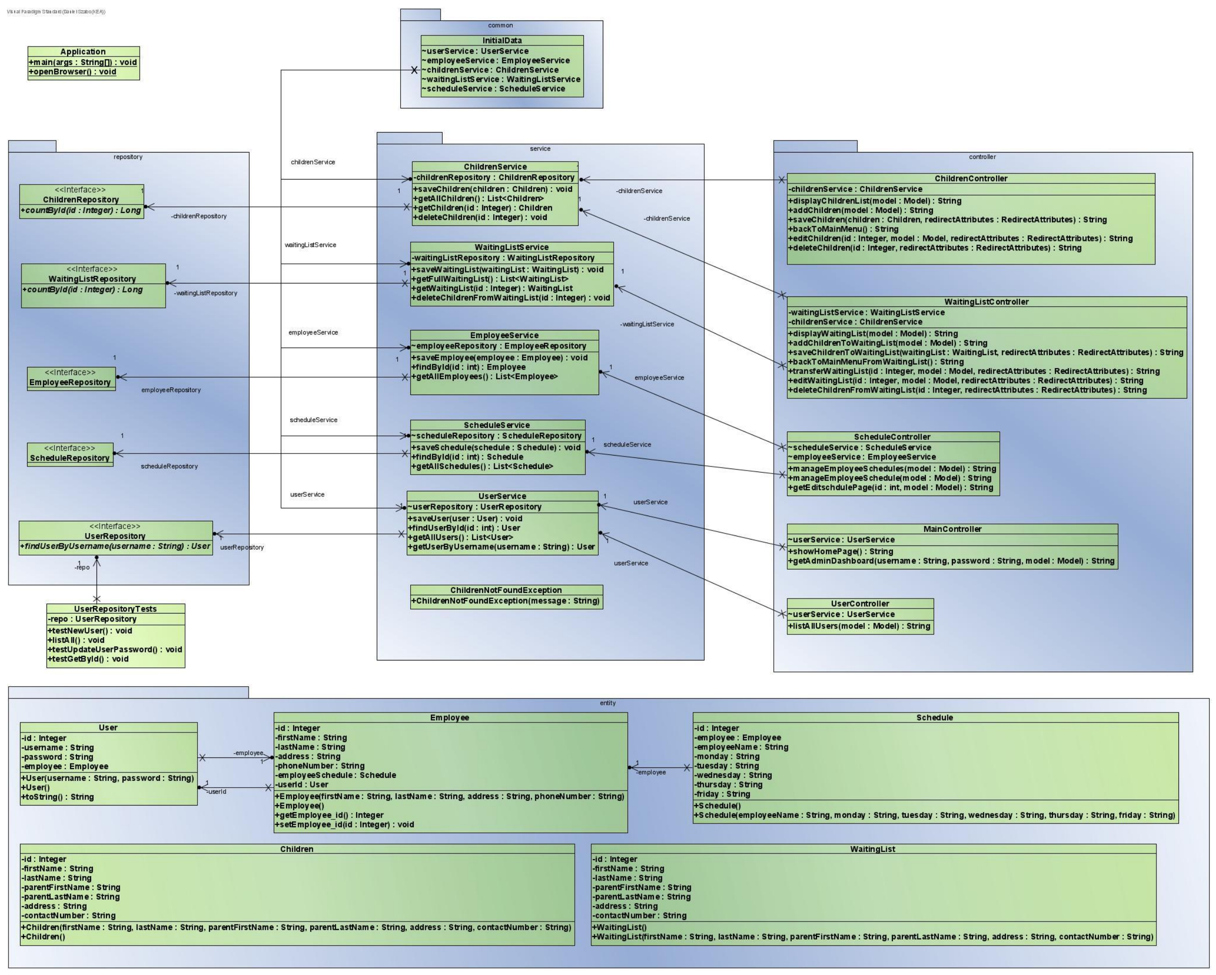
UC name: Manage children
Actor: user

Main Flow:
user is able to edit, add and delete children.

Alt. Flow:
User is met by error message

Teletearbies:
Fernanda Cunha Bacelar
Amalie Vibe Larsen
Balázs Dombovári
Dániel Szabó

Class Diagram


Visual Paradigm Standard (Daniel Szabo (KEA))
Application
+main(args : String[]) : void
+openBrowser() : void
common
InitialData
~userService : UserService
~employeeService : EmployeeService
~childrenService : ChildrenService
~waitingListService : WaitingListService
~scheduleService : ScheduleService
repository
<<Interface>>
ChildrenRepository
+countById(id : Integer) : Long
<<Interface>>
WaitingListRepository
+countById(id : Integer) : Long
<<Interface>>
EmployeeRepository
<<Interface>>
ScheduleRepository
<<Interface>>
UserRepository
+findUserByUsername(username : String) : User
-childrenRepository
-waitingListRepository
employeeRepository
scheduleRepository
userRepository
-repo
UserRepositoryTests
-repo : UserRepository
+testNewUser() : void
+listAll() : void
+testUpdateUserPassword() : void
+testGetById() : void
childrenService
waitingListService
employeeService
scheduleService
userService
service
ChildrenService
-childrenRepository : ChildrenRepository
+saveChildren(children : Children) : void
+getAllChildren() : List<Children>
+getChildren(id : Integer) : Children
+deleteChildren(id : Integer) : void
WaitingListService
-waitingListRepository : WaitingListRepository
+saveWaitingList(waitingList : WaitingList) : void
+getFullWaitingList() : List<WaitingList>
+getWaitingList(id : Integer) : WaitingList
+deleteChildrenFromWaitingList(id : Integer) : void
EmployeeService
~employeeRepository : EmployeeRepository
+saveEmployee(employee : Employee) : void
+findById(id : int) : Employee
+getAllEmployees() : List<Employee>
ScheduleService
~scheduleRepository : ScheduleRepository
+saveSchedule(schedule : Schedule) : void
+findById(id : int) : Schedule
+getAllSchedules() : List<Schedule>
UserService
~userRepository : UserRepository
+saveUser(user : User) : void
+findUserById(id : int) : User
+getAllUsers() : List<User>
+getUserByUsername(username : String) : User
ChildrenNotFoundException
+ChildrenNotFoundException(message : String)
-childrenService
-childrenService
-waitingListService
employeeService
scheduleService
userService
userService
controller
ChildrenController
-childrenService : ChildrenService
+displayChildrenList(model : Model) : String
+addChildren(model : Model) : String
+saveChildren(children : Children, redirectAttributes : RedirectAttributes) : String
+backToMainMenu() : String
+editChildren(id : Integer, model : Model, redirectAttributes : RedirectAttributes) : String
+deleteChildren(id : Integer, redirectAttributes : RedirectAttributes) : String
WaitingListController
-waitingListService : WaitingListService
-childrenService : ChildrenService
+displayWaitingList(model : Model) : String
+addChildrenToWaitingList(model : Model) : String
+saveChildrenToWaitingList(waitingList : WaitingList, redirectAttributes : RedirectAttributes) : String
+backToMainMenuFromWaitingList() : String
+transferWaitingList(id : Integer, model : Model, redirectAttributes : RedirectAttributes) : String
+editWaitingList(id : Integer, model : Model, redirectAttributes : RedirectAttributes) : String
+deleteChildrenFromWaitingList(id : Integer, redirectAttributes : RedirectAttributes) : String
ScheduleController
~scheduleService : ScheduleService
~employeeService : EmployeeService
+manageEmployeeSchedules(model : Model) : String
+manageEmployeeSchedule(model : Model) : String
+getEditschdulePage(id : int, model : Model) : String
MainController
~userService : UserService
+showHomePage() : String
+getAdminDashboard(username : String, password : String, model : Model) : String
UserController
~userService : UserService
+listAllUsers(model : Model) : String
entity
User
-id : Integer
-username : String
-password : String
-employee : Employee
+User(username : String, password : String)
+User()
+toString() : String
-employee
-userId
Employee
-id : Integer
-firstName : String
-lastName : String
-address : String
-phoneNumber : String
-employeeSchedule : Schedule
-userId : User
+Employee(firstName : String, lastName : String, address : String, phoneNumber : String)
+Employee()
+getEmployee_id() : Integer
+setEmployee_id(id : Integer) : void
-employee
Schedule
-id : Integer
-employee : Employee
-employeeName : String
-monday : String
-tuesday : String
-wednesday : String
-thursday : String
-friday : String
+Schedule()
+Schedule(employeeName : String, monday : String, tuesday : String, wednesday : String, thursday : String, friday : String)
Children
-id : Integer
-firstName : String
-lastName : String
-parentFirstName : String
-parentLastName : String
-address : String
-contactNumber : String
+Children(firstName : String, lastName : String, parentFirstName : String, parentLastName : String, address : String, contactNumber : String)
+Children()
WaitingList
-id : Integer
-firstName : String
-lastName : String
-parentFirstName : String
-parentLastName : String
-address : String
-contactNumber : String
+WaitingList()
+WaitingList(firstName : String, lastName : String, parentFirstName : String, parentLastName : String, address : String, contactNumber : String)