MINISTERUL EDUCAŢIEI ŞI CERCETĂRII ŞTIINŢIFICE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE SI TEHNOLOGIA INFORMATIEI

SOFTWARE DEVELOPERS MANAGEMENT

Autor: Viman Andrei-Liviu

Alti membrii: Hitu Octavian, Munteanu Cezar, Timis Iulia

Data: 03.01.2022

Grupa: 30236

Diagrama cazurilor de utilizare:

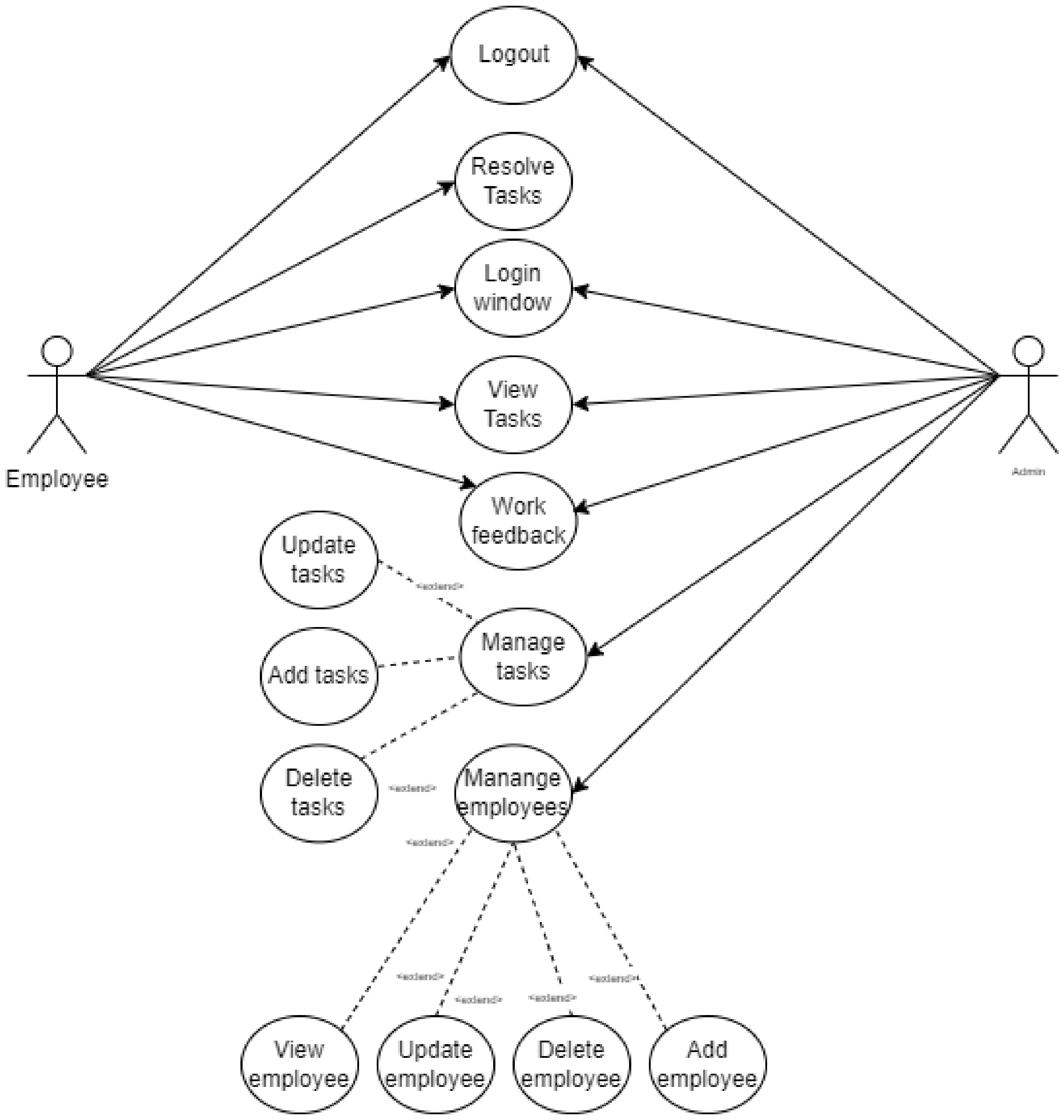


Diagrama de clase:

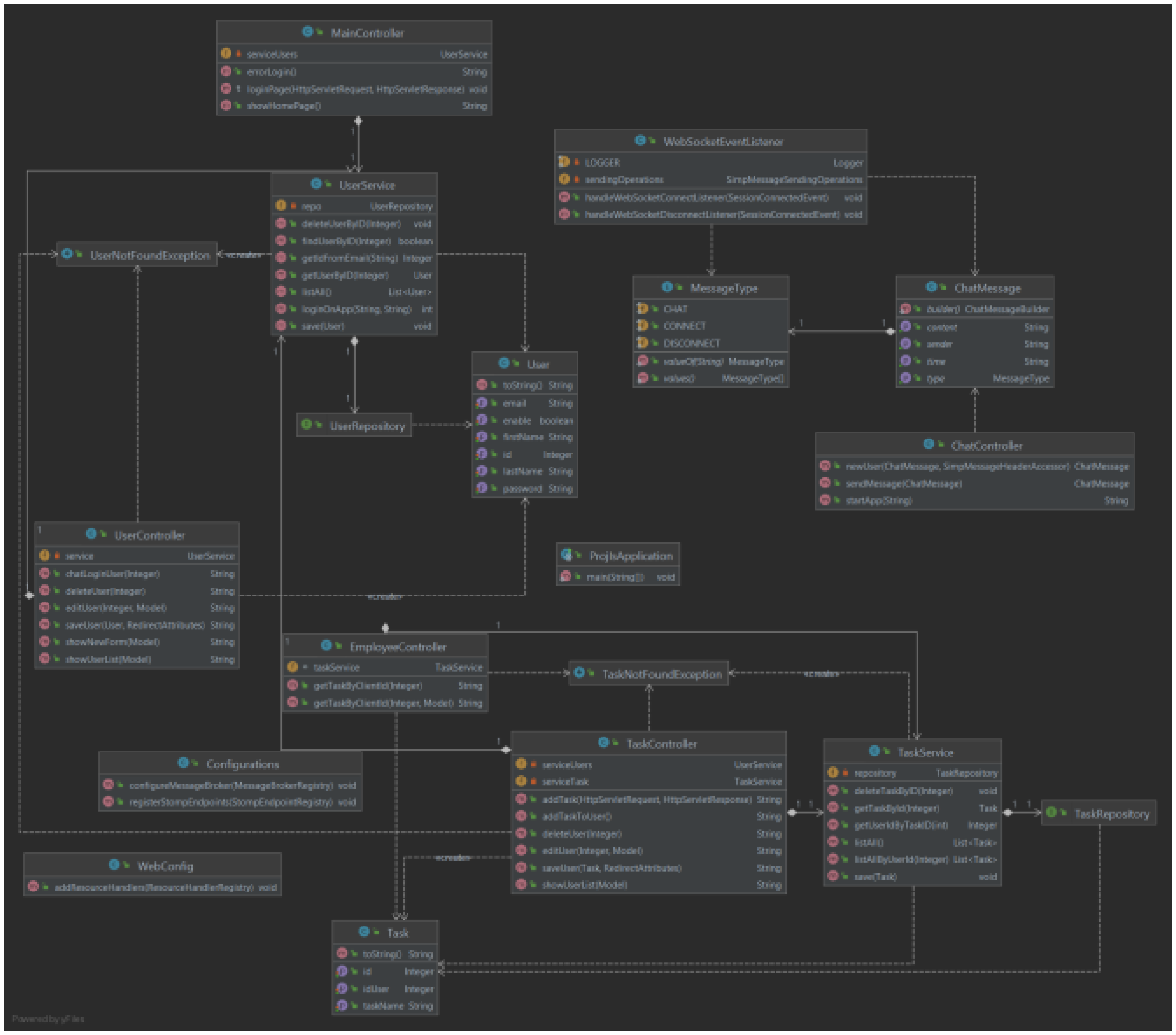
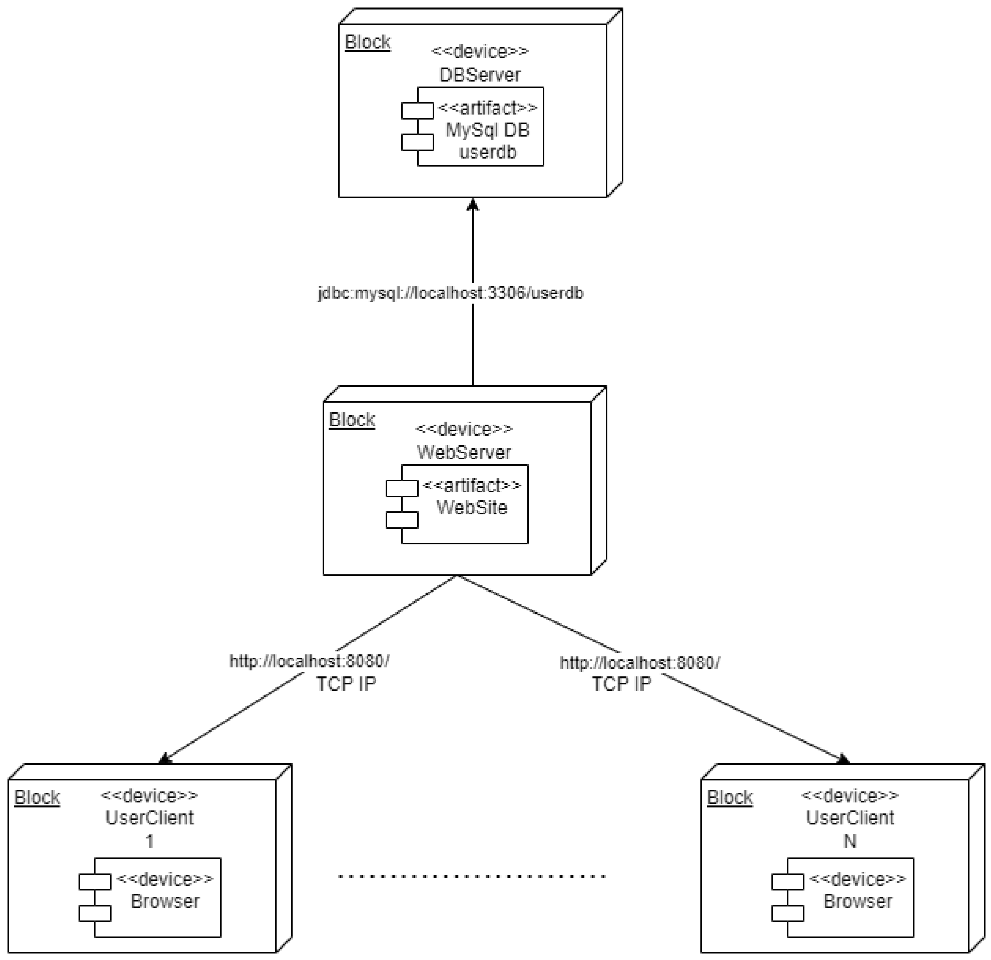

MainController
serviceUsers UserService
errorLogin() String
loginPage(HttpServletRequest, HttpServletResponse) void
showHomePage() String
WebSocketEventListener
LOGGER Logger
sendingOperations SimpMessageSendingOperations
handleWebSocketConnectListener(SessionConnectedEvent) void
handleWebSocketDisconnectListener(SessionConnectedEvent) void
UserService
repo UserRepository
deleteUserByID(Integer) void
findUserByID(Integer) boolean
getIdfromEmail(String) Integer
getUserByID(Integer) User
listAll() List<User>
loginOnApp(String, String) int
save(User) void
UserNotFoundException
«create»
MessageType
CHAT
CONNECT
DISCONNECT
valueOf(String) MessageType
values() MessageType[]
ChatMessage
builder() ChatMessageBuilder
content String
sender String
time String
type MessageType
User
toString() String
email String
enable boolean
firstName String
id Integer
lastName String
password String
UserRepository
ChatController
newUser(ChatMessage, SimpMessageHeaderAccessor) ChatMessage
sendMessage(ChatMessage) ChatMessage
startApp(String) String
UserController
service UserService
chatLoginUser(Integer) String
deleteUser(Integer) String
editUser(Integer, Model) String
saveUser(User, RedirectAttributes) String
showNewForm(Model) String
showUserList(Model) String
ProjIsApplication
main(String[]) void
«create»
EmployeeController
taskService TaskService
getTaskByClientId(Integer) String
getTaskByClientId(Integer, Model) String
TaskNotFoundException
«create»
TaskController
serviceUsers UserService
serviceTask TaskService
addTask(HttpServletRequest, HttpServletResponse) String
addTaskToUser() String
deleteUser(Integer) String
editUser(Integer, Model) String
saveUser(Task, RedirectAttributes) String
showUserList(Model) String
TaskService
repository TaskRepository
deleteTaskByID(Integer) void
getTaskById(Integer) Task
getUserIdByTaskID(int) Integer
listAll() List<Task>
listAllByUserId(Integer) List<Task>
save(Task) void
TaskRepository
Configurations
configureMessageBroker(MessageBrokerRegistry) void
registerStompEndpoints(StompEndpointRegistry) void
WebConfig
addResourceHandlers(ResourceHandlerRegistry) void
«create»
Task
toString() String
id Integer
idUser Integer
taskName String
Powered by yFiles

Diagrama de deployment:



Limbaje si tehnologii utilizate:

Java + Spring, HTML, CSS, JavaScript

Spring Data JPA, MySql, SQL

Thymeleaf and Bootstrap

Spring Boot CRUD(Create, Read, Update and Delete)

Lombok and WebSocket

Rezumat:

In acest proiect se doreste implementarea unei asa-zise retele sociale a unei firme de IT, in care angajatii pot discuta liber atat cu alti colegi cat si cu sefii lor, asfel ei putand da feedback care sa ii ajute pe colegi in terminarea task-urilor pe care un administrator le poate da fiecarui dintre angajati. Angajatii nu au constrangeri de timp pentru rezolvarea taskurilor dar sunt nevoiti sa le rezolve pentru ca ei sa isi primeasca salariul. Angajatul, odata cu terminarea taskului poate intra in interfata si sa anunte acest lucru prin apasarea butonului end task.

Pentru a putea intra in acest portal viirtual atat angajatii cat si administratorii sunt nevoiti sa se logheze, acest lucru se realizeaza cu ajutorul unei baze de date pe care o poate edita doar administratorul, dezavantajul este ca daca un angajat isi uita parola este obligat sa ia legatura cu angajatorul pentru schimbarea parolei. Administratorii pe langa ca pot vedea task-uri angajatilor pot si da task-uri, acestia mai au cateva functionalitati in plus, cum ar fi: sa modifice datele despre angajati si despre task-uri(Add, Delete, Update).

Cod Java:

Clasa UserService:

```
package com.example.projis.service;
```

```
import com.example.projis.exceptions.UserNotFoundException;
import com.example.projis.model.User;
import com.example.projis.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class UserService {
    @Autowired private UserRepository repo;

    public List<User> listAll(){
        return (List<User>) repo.findAll();
    }

    public void save(User user) {
        repo.save(user);
    }

    public int loginOnApp(String username, String password) {
        Iterable<User> users = repo.findAll();
        int id = -1;
        boolean isAdmin = false;
        for (User user: users ) {
            if (user.getEmail().equals(username) && user.getPassword().equals(password)) {
                id = user.getId();
                isAdmin = user.isEnable();
            }
        }
        if (id == -1){
            return -1;
        }else{
            if(isAdmin){
                return 1;

            }else{
                return 0;
            }
        }
    }

    public User getUserByID(Integer id) throws UserNotFoundException {
        Optional<User> result = repo.findById(id);
        if(result.isPresent()){
            return result.get();
        }
        throw new UserNotFoundException("User with id: "+id+" is not found!");
    }

    public void deleteUserByID(Integer id) throws UserNotFoundException {
        if(repo.existsById(id)) {
            repo.deleteById(id);
        }else{
            throw new UserNotFoundException("User with id: "+id+" is not found!");
        }
    }
    public boolean findUserByID(Integer id) throws  UserNotFoundException{
        Optional<User> byId = repo.findById(id);
        if( byId.isPresent()) {
            return true;
        }
        else{
            throw new UserNotFoundException("User with id: "+id+" is not found!");
        }

    }

    public Integer getIdFromEmail(String email){
        Iterable<User> users = repo.findAll();
        Integer id = 0;
        for (User user: users ) {
            if (user.getEmail().equals(email)) {
                id = user.getId();
            }
        }
        return id;
    }
}
```

Clasa UserRepository:

```
package com.example.projis.repository;


import com.example.projis.model.User;
import org.springframework.data.repository.CrudRepository;

public interface UserRepository extends CrudRepository<User,Integer> {

}
```

Clasa UserController:

```
package com.example.projis.controller;

import com.example.projis.exceptions.UserNotFoundException;
import com.example.projis.model.User;
import com.example.projis.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;


import java.util.List;

@Controller
public class UserController {
    @Autowired private UserService service;

    @GetMapping("/users")
    public String showUserList(Model model){
```

```
    List <User> listUsers = service.listAll();
    model.addAttribute("listUsers",listUsers);
   return  "users";
}
@GetMapping("/users/newUser")
public String showNewForm(Model model){
    model.addAttribute("user",new User());
    model.addAttribute("pageTitle","Add New User");
    return "user_form";
}
@PostMapping("/users/save")
public String saveUser(User user, RedirectAttributes redirectAttributes){
    service.save(user);
    redirectAttributes.addFlashAttribute("message","The user has been served successfuly !");
    return "redirect:/users";
}

@GetMapping("/users/edit/{id}")
public String editUser(@PathVariable("id") Integer id,Model model){
    try {
        User user = service.getUserByID(id);
        model.addAttribute("user",user);
        model.addAttribute("pageTitle","Edit User with ID: "+ id );
        return "user_form";
    } catch (UserNotFoundException e) {
        return "error_edit";
    }
}
```

```
    @GetMapping("/users/delete/{id}")
    public String deleteUser(@PathVariable("id") Integer id){
        try {
            service.deleteUserByID(id);
            return "redirect:/users";
        } catch (UserNotFoundException e) {
            return "error_edit";
        }
    }

    @GetMapping("/intermediar/{id}")
    public String chatLoginUser(@PathVariable Integer id){
        try {
            User user = service.getUserByID(id);
            String nume = user.getLastName();
            String prenume = user.getFirstName();
            String chatUsername = nume + "_"+ prenume;
            return "redirect:/chat/"+chatUsername;
        } catch (UserNotFoundException e) {
            return "error_edit";
        }
    }
}
```

Clasa User:

```
package com.example.projis.model;

import javax.persistence.*;
```

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

    @Column(nullable = false,unique = true,length = 45)
    private String email;
    @Column(nullable = false,unique = true,length = 15)
    private String password;

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

    @Column(nullable = false, name = "enable")
    private boolean enable;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getEmail() {
```

```
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
    }

    public boolean isEnable() {
        return enable;
    }

    public void setEnable(boolean enable) {
        this.enable = enable;
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", email='" + email + '\'' +
                ", password='" + password + '\'' +
                ", firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", enable=" + enable +
                '}';
    }
}
```

Cod HTML:

User.html:

```
<!DOCTYPE html>
<html lang="en" xmlns:th = "https://www.thymeleaaf.org">
<head>
```

```
  <meta charset="UTF-8">
  <title>Manage Users</title>
  <link rel="stylesheet" th:href="@{/webjars/bootstrap/4.3.1/css/bootstrap.min.css}"
type="text/css"/>

  <style>
    body{
      margin: 0;
      padding: 0;
      background: url(http://localhost:63342/projIS/templates/bsn.jpeg);
      background-size: cover;
      background-position: center;
      font-family: sans-serif;
      text-align: center;
    }

    .adminbox{
      background-color:rgb(245, 245, 245) ;
      border-radius: 1%;
      width: 1500px;
      height: 700px;
      top: 50%;
      left: 50%;
      position: absolute;
      transform: translate(-50%,-50%);
      box-sizing: border-box;
    }

    .button{
      background-color: rgba(245, 245, 245, 0.822);
      border: 2px solid #000;
      border-radius: 20px;
      color: #000;
      height: 40px;
      width: 120px;
    }
    thead{
      border: 2px solid black;
    }
    tr, td{
      text-align: left;
      border-bottom: 1px solid black;
    }
    tr:hover{
      background-color: green;
    }


  </style>
</head>
<body>
<dine class="adminbox">
  <div class = "container-fluid text-center" >
    <h1>Manage Users</h1>
    <a href="users/newUser">
      <button class="button">Add New User</button>
    </a>
    <a href="/users/addtask">
      <button class="button">Add New Task</button>
    </a>
    <a href="/users/viewtask">
      <button class="button">View Tasks</button>
    </a>
    <a href="/">
      <button class="button">LogOut</button>
    </a>
    <a href="/chat/Admin">
      <button class="button">Chat</button>
    </a>
    <br>
    <br>
    <div>
      <table class="table text-center">
        <thead class="thead-dark">
        <tr>
          <th>ID</th>
          <th>E-mail</th>
          <th>First Name</th>
          <th>Last Name</th>
          <th>Administrator</th>
          <th></th>
        </tr>
        </thead>
        <tbody class="">
        <th:block th:each = "user: ${listUsers}"  >
          <tr>
            <td>[[${user.id}]]</td>
```

```
            <td>[[${user.email}]]</td>
            <td>[[${user.firstName}]]</td>
            <td>[[${user.lastName}]]</td>
            <td>[[${user.enable}]]</td>
            <td>
              <a class="h5 mr-3" th:href = "@{'/users/edit/' + ${user.id}}">Edit</a>
              <a class="h5 mr-3" th:href = "@{'/users/delete/' + ${user.id}}">Delete</a>
            </td>
          </tr>
        </th:block>
        </tbody>
      </table>
    </div>
  </div>
</dine>
</body>
</html>
```

User_form.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th = "https://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Add New User</title>
    <link rel = "stylesheet" type="text/css" th:href = "@{/webjars/bootstrap/4.3.1/css/bootstrap.min.css}" />
    <style>
```

```
body{
    margin: 0;
    padding: 0;
    background: url(http://localhost:63342/projIS/templates/bsn.jpeg);
    background-size: cover;
    background-position: center;
    font-family: sans-serif;
    text-align: center;
}

.userformbox{
    background-color: rgb(245, 245, 245);
    border-radius: 10%;
    width: 700px;
    height: 500px;
    top: 50%;
    left: 50%;
    position: absolute;
    transform: translate(-50%,-50%);
    box-sizing: border-box;
}
.userformbox button[type="submit"]
{
    background-color: rgba(32, 97, 250, 0.733);
    border: 2px solid #000;
    border-radius: 20px;
    color: #000;
    height: 40px;
    width: 90px;
```

```
        }

        .userformbox button[type="button"]
        {
            background-color: rgba(245, 245, 245, 0.822);
            border: 2px solid #000;
            border-radius: 20px;
            color: #000;
            height: 40px;
            width: 90px;

        }

        .userformbox input[type="email"],input[type="text"],input[type="password"]{
            border: none;
            border-bottom: 2px solid #000;
            background: transparent;
            outline: none;
            height: 50px;

        }

    </style>
</head>
<body>

<dinv class="userformbox">
    <div class="text-center"><h2>[[${pageTitle}]]</h2></div>
```

```
  <div th:if = "${message}" class="alert alert-success text-center">
    [[${message}]]
  </div>
  <br>
  <br>
  <form th:action = "@{/users/save}" method="post" th:object = "${user}" style="max-width: 500px; margin: 0 auto;">
    <input type="hidden" th:field = "*{id}">
    <div class="form-group row">
      <label class="col-sm-4 col-form-label">E-mail:</label>
      <div class="col-sm-8">
        <input type="email" th:field = "*{email}" class="form-control" required minlength="8" maxlength="45"/>
      </div>
    </div>
    <div class="form-group row">
      <label class="col-sm-4 col-form-label">First Name:</label>
      <div class="col-sm-8">
        <input type="text" th:field = "*{firstName}" class="form-control" required minlength="2" maxlength="45" />
      </div>
    </div>
    <div class="form-group row">
      <label class="col-sm-4 col-form-label">Last Name:</label>
      <div class="col-sm-8">
        <input type="text" th:field = "*{lastName}" class="form-control" required minlength="2" maxlength="45"/>
      </div>
    </div>
    <div class="form-group row">
```

```
            <label class="col-sm-4 col-form-label">Password:</label>
            <div class="col-sm-8">
                <input type="password" th:field = "*{password}" class="form-control" required
minlength="5" maxlength="15"/>
            </div>
        </div>
        <div class="form-group row">
            <label class="col-sm-4 col-form-label">Administrator</label>
            <div class="col-sm-8">
                <input type="checkbox" th:field = "*{enable}"/>
            </div>
        </div>
        <div class="text-center">
            <button type="submit" class="btn btn-primary m-2">Save</button>
            <button type="button" class="btn btn-secondary m-2"
onclick="cancelForm()">Cancel</button>
        </div>

    </form>

</dinv>
<script type="text/javascript">
    function cancelForm(){
        window.location = "[[@{/users}]]";
    }
</script>
</body>
</html>
```

Error_edit.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th = "https://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>ERROR</title>
  <link rel = "stylesheet" type="text/css" th:href = "@{/webjars/bootstrap/4.3.1/css/bootstrap.min.css}" />
  <style>

    body{
      margin: 0;
      padding: 0;
      background: url(http://localhost:63342/projIS/templates/bsn.jpeg);
      background-size: cover;
      background-position: center;
      font-family: sans-serif;
      text-align: center;
    }

    .errorbox{
      background-color: rgb(245, 245, 245);
      border-radius: 10%;
      width: 500px;
      height: 200px;
      top: 50%;
      left: 50%;
      position: absolute;
      transform: translate(-50%,-50%);
      box-sizing: border-box;
    }
    .close {
      height: 100px;
      width: 100px;
      background-color: red;
      border-radius: 5px;
      position: relative;
      left: -40%;

    }

    .close:after {
      position: absolute;
      top: 0;
      bottom: 0;
      left: 0;
      right: 0;
      content: "\274c"; /* use the hex value here... */
      font-size: 50px;
      color: #FFF;
      line-height: 100px;
      text-align: center;

    }
```

```
    .errorbox input[type="button"]
    {
      background-color: rgba(245, 245, 245, 0.822);
      border: 2px solid #000;
      border-radius: 20px;
      color: #000;
      height: 40px;
      width: 90px;

    }
  </style>

</head>
<body>
<dine class="errorbox ">
  <div class="text-center"><h2>USER NOT FOUND</h2></div>
  <div class="close"/>
  <br>
  <br>
  <br>
  <br>

  <form>
    <div class="text-center">
      <button type="button" class="btn btn-secondary m-2"
onclick="cancelForm()">Back</button>
    </div>
  </form>
</dine>
```

```
<script type="text/javascript">
  function cancelForm(){
    window.location = "[[@{/users}]]";
  }
</script>

</body>
</html>
```

Bibliografie:

https://ktor.io/docs/creating-web-socket-chat.html

https://www.tutorialspoint.com/html/html_basic_tags.htm

https://www.w3schools.com/html/

https://spring.io/projects/spring-data-jpa#overview

https://www.javatpoint.com/spring-boot-crud-operations

https://frontbackend.com/thymeleaf/how-to-add-bootstrap-css-and-js-to-thymeleaf

https://www.thymeleaf.org/doc/articles/layouts.html

https://www.javatpoint.com/spring-mvc-tutorial

https://www.visual-paradigm.com/tutorials/how-to-draw-deployment-diagram-in-uml/

https://creately.com/diagram-type/template/gqm1ek4w1/deployment-diagram