

Проект Address book

Изготвен от Димана Стоянова, F99109 и Антонио Сотиров, F98702

Използвани технологии: Java, Spring Boot, MySQL, Bootstrap, Thymeleaf

1. Регистриране на потребители и вход в системата

Address Book

Login Register

Login

Username

Password

Login

Address Book

Login Register

Registration

Username

Email

Password

Confirm Password

First name

Методите Login и Register са реализирани в класа userService и се извикват в userController. При регистрацията данните се валидират и се запазват в локална база от данни, за да може да се използват за вход в системата.

```
11 usages  disto *
public abstract class Validator {

    2 usages  disto *
    public static String validateEmail(String email) {
        String emailRegex = "[A-Z0-9._%+-]+@[A-Z0-9.-]+\\.[A-Z]{2,6}$";
        Pattern pattern = Pattern.compile(emailRegex, Pattern.CASE_INSENSITIVE);
        String message = "";
        if (email == null || !pattern.matcher(email).matches()) {
            message = "Invalid email.";
        }
        return message;
    }

    2 usages  disto *
    public static String validatePassword(String password) {
        String message = "";
        if (password == null || password.isEmpty()
            || containsOnlySpaces(password)) {
            message = "Invalid password.";
        }
    }
}
```

```
no usages  disto
@GetMapping("/register")
public String register(Model model) {
    model.addAttribute("registerRequestUserDTO", new RegisterRequestUserDTO());
    return "register";
}

no usages  disto
@PostMapping("/register")
public String register(@Valid RegisterRequestUserDTO registerRequestUserDTO, BindingResult bindingResult) {
    userService.register(registerRequestUserDTO, bindingResult);

    if (bindingResult.hasErrors()) {
        return "register";
    }

    return "redirect:/login";
}

no usages  disto
@GetMapping("/login")
public String login(Model model) {
    model.addAttribute("loginUserDTO", new LoginUserDTO());
    return "login";
}
```

2. Наличие на текстов редактор за въвеждане и запазване на текст (коментар към конкретен запис)

Към всеки контакт може да се добави коментар или допълнително поле с текст по избор. Допълнителното поле е реализирано в клас `AdditionalField`. При добавяне на допълнително поле, информацията се запазва в базата от данни и можем да я достъпваме, когато преглеждаме дадения контакт.

```
3 pages  disto
@Service
public class AdditionalFieldService {
    7 usages
    AdditionalFieldRepository<AdditionalField> additionalFieldRepository;

    no usages  disto
    @Autowired
    public AdditionalFieldService(AdditionalFieldRepository<AdditionalField> additionalFieldRepository) {
        this.additionalFieldRepository = additionalFieldRepository;
    }

    1 usage  disto
    @Transactional
    public void addAdditionalField(AdditionalField additionalField){

        if (additionalField == null){}

        if(additionalField.getTitle() == null){}

        if(additionalField.getText() == null){}

        additionalFieldRepository.save(additionalField);
    }
}
```

Field name:

X

Field text:

Add Field

Submit

3. Възможност за предварително създаване, редактиране и изтриване на етикети (с различни цветове), с които може да се маркират записи, с цел подреждането им в йерархична структура.

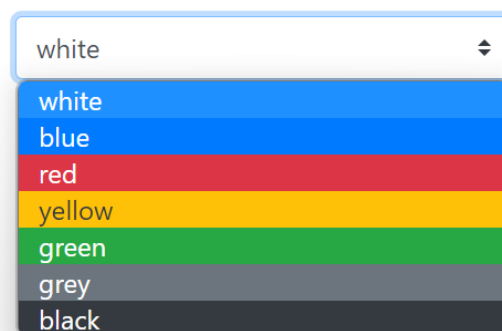
При създаване или редактиране на контакт можем да сложим етикет с различен цвят или да променим първоначалния етикет. Етикетите са реализирани като списък в клас Contact.

```
<select class="custom-select" name="label">
  <option value="WHITE" class="" th:selected="${contact.label == T(nbu.java.entity.Contact.Label).WHITE}">white</option>
  <option value="BLUE" class="bg-primary text-white" th:selected="${contact.label == T(nbu.java.entity.Contact.Label).BLUE}">blue</option>
  <option value="RED" class="bg-danger text-white" th:selected="${contact.label == T(nbu.java.entity.Contact.Label).RED}">red</option>
  <option value="YELLOW" class="bg-warning text-dark" th:selected="${contact.label == T(nbu.java.entity.Contact.Label).YELLOW}">yellow</option>
  <option value="GREEN" class="bg-success text-white" th:selected="${contact.label == T(nbu.java.entity.Contact.Label).GREEN}">green</option>
  <option value="GREY" class="bg-secondary text-white" th:selected="${contact.label == T(nbu.java.entity.Contact.Label).GREY}">grey</option>
  <option value="BLACK" class="bg-dark text-white" th:selected="${contact.label == T(nbu.java.entity.Contact.Label).BLACK}">black</option>
</select>
</div>
```

```
<h6 class="text-primary">Label</h6>
</label>
<select class="custom-select" name="label" id="label">
  <option value="WHITE" class="">white</option>
  <option value="BLUE" class="bg-primary text-white">blue</option>
  <option value="RED" class="bg-danger text-white">red</option>
  <option value="YELLOW" class="bg-warning text-dark">yellow</option>
  <option value="GREEN" class="bg-success text-white">green</option>
  <option value="GREY" class="bg-secondary text-white">grey</option>
  <option value="BLACK" class="bg-dark text-white">black</option>
</select>
</div>
</div>
```

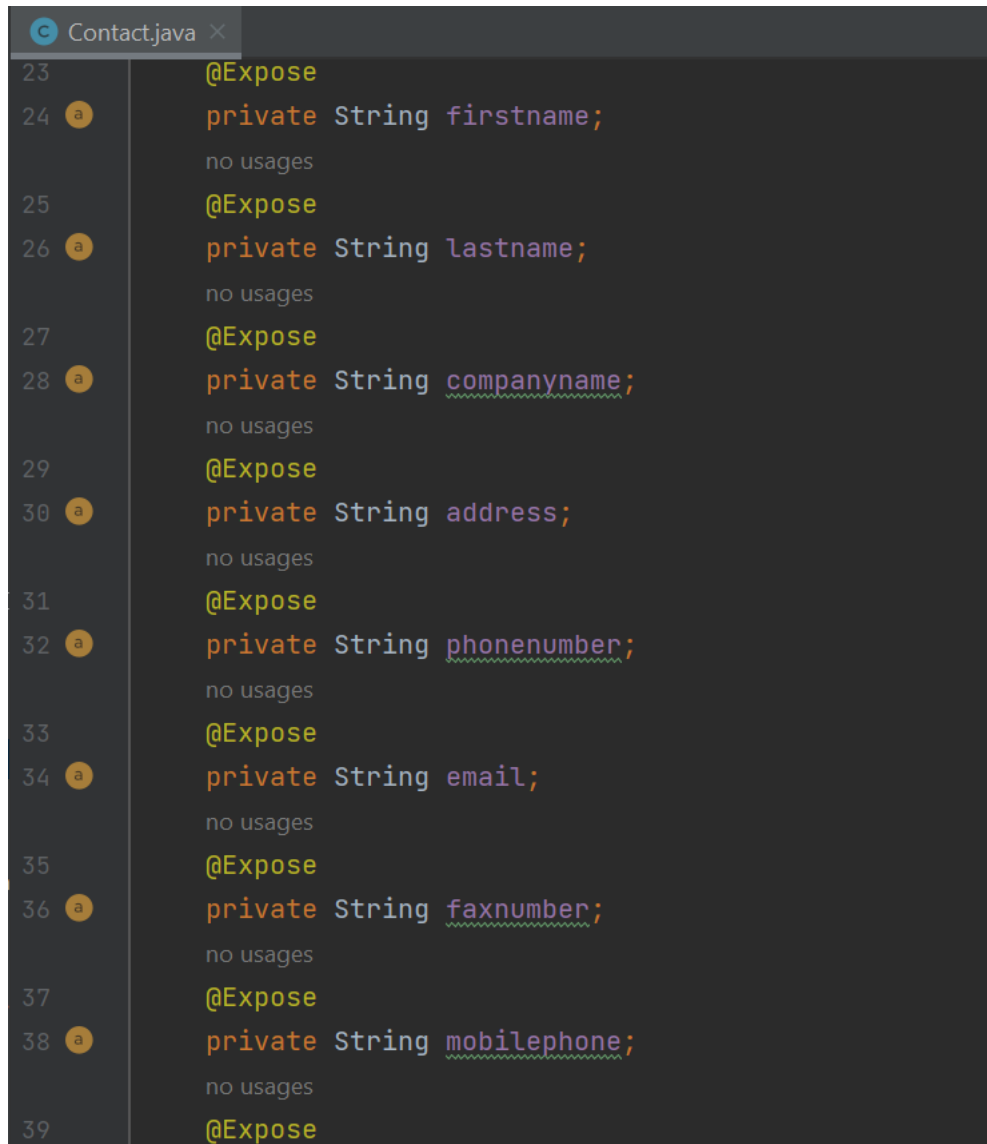
```
3 usages  disto
public enum Label {
    no usages
    WHITE, YELLOW, GREEN, RED, BLUE, GREY, BLACK
}
}
```

Label



4. Всеки запис да съдържа няколко полета (име, фамилия, име на фирма, адрес, телефонен номер, имейл адрес, номер на факс, номер на мобилен телефон, коментар).

Всяко от тези полета е променлива в клас Contact и се заявява при създаването на нов контакт.



```
23 @Expose
24 private String firstname;
    no usages
25 @Expose
26 private String lastname;
    no usages
27 @Expose
28 private String companyname;
    no usages
29 @Expose
30 private String address;
    no usages
31 @Expose
32 private String phonenumber;
    no usages
33 @Expose
34 private String email;
    no usages
35 @Expose
36 private String faxnumber;
    no usages
37 @Expose
38 private String mobilephone;
    no usages
39 @Expose
```

Информацията се валидира при създаване на нов контакт и се запазва в базата от данни.

5. Възможност за добавяне на персонализирани полета към вече съществуващите.

Отчасти се препокрива с точка 1 – additional field.

6. Въвеждане, показване, редактиране и изтриване на данни за:

6.1. Налични записи;

6.2. Профила на потребителя

```
ContactController.java x ContactService.java x
94
    1 usage  disto
95     @Transactional
96     @
97     public void updateContact(int id, ContactDTO contactDTO) {
98
99         Contact contact = contactRepository.findById(id);
100
101         if (contactDTO.getFirstname() != null) {
102             contact.setFirstname(contactDTO.getFirstname());
103         }
104
105         if (contactDTO.getLastname() != null) {
106             contact.setLastname(contactDTO.getLastname());
107         }
108
109         if (contactDTO.getCompanyname() != null) {
110             contact.setCompanyname(contactDTO.getCompanyname());
111         }
112
113         if (contactDTO.getAddress() != null) {
114             contact.setAddress(contactDTO.getAddress());
115         }
116
117         if (contactDTO.getPhonenumber() != null) {
118             contact.setPhonenumber(contactDTO.getPhonenumber());
119         }
120     }
```

```
no usages  disto
98     @GetMapping(value = "/contacts/edit")
99     public String showEditPage(@RequestParam String id, Model model, HttpSession httpSession) throws NotFoundException {
100
101         Contact contact = contactService.findById(Integer.parseInt(id));
102         if (httpSession.getAttribute(s: "LOGGED_USER_ID") == null || (Integer) httpSession.getAttribute(s: "LOGGED_USER_ID") != contact.getUser(
103             return "redirect:/contacts";
104         model.addAttribute(attributeName: "contact", contact);
105         model.addAttribute(attributeName: "additionalFields", additionalFieldService.findByContactId(Integer.parseInt(id)));
106         return "editContact";
107     }
108
109     no usages  disto
110     @PostMapping(value = "/contacts/edit")
111     public String editContact(@ModelAttribute ContactDTO contactDTO, @RequestParam String id,
112                             @RequestParam(value = "title", required = false) String[] title,
113                             @RequestParam(value = "text", required = false) String[] text,
114                             HttpSession httpSession) throws NotFoundException {
115
116         Contact contact = contactService.findById(Integer.parseInt(id));
117         if (httpSession.getAttribute(s: "LOGGED_USER_ID") == null || (Integer) httpSession.getAttribute(s: "LOGGED_USER_ID") != contact.getUser(
118             return "redirect:/contacts";
119
120         if (title != null && text != null) {
121             List<String> titles = Arrays.asList(title);
122             List<String> texts = Arrays.asList(text);
123             additionalFieldService.save(contact.getId(), titles, texts);
124         }
125         return "editContact";
126     }
```

Edit Page

First Name

Antonio

Last Name

Sotirov

Phone

0889335789

Mobile Phone

524163

Address

Sofia

Company

Crypto

Email

antonios@gmail.com

Fax

52634

Comment

Label

blue

Edit

```
65 no usages disto
66 @GetMapping("/{editMyProfile}")
67 public String edit(Model model) {
68     model.addAttribute("editRequestUserDTO", new EditRequestUserDTO());
69     return "editMyProfile";
70 }
71
72 no usages disto
73 @PostMapping("/{editMyProfile}")
74 public String edit(@Valid EditRequestUserDTO editRequestUserDTO, HttpSession session, BindingResult bindingResult) {
75     User loggedUser = sessionManager.getLoggedUser(session);
76     userService.editUser(editRequestUserDTO, loggedUser, bindingResult);
77     return "editMyProfile";
78 }
79
80 no usages disto
81 @PostMapping("/{deleteMyProfile}")
82 public String delete(HttpSession session) {
83     sessionManager.validateLogged(session);
84     int userId = sessionManager.getLoggedUser(session).getId();
85     String response = userService.deleteUser(userId);
86     sessionManager.logoutUser(session);
87     return "redirect:/login";
88 }
```

```
1 usage disto
15 @ public ResponseUserDTO editUser(EditRequestUserDTO userDTO, User loggedInUser, BindingResult bindingResult) {
16     String newUsername = userDTO.getUsername();
17     if (newUsername != null && (!newUsername.isEmpty()) && userRepository.findByUsername(newUsername) != null) {
18         ObjectError error = new ObjectError(objectName: "global", defaultMessage: "Username already exists");
19         bindingResult.addError(error);
20     } else {
21         loggedInUser.setUsername(newUsername);
22     }
23
24     String newEmail = userDTO.getEmail();
25     String validateEmailMessage = Validator.validateEmail(newEmail);
26     if (!validateEmailMessage.isEmpty()) {
27         ObjectError error = new ObjectError(objectName: "global", validateEmailMessage);
28         bindingResult.addError(error);
29     } else if (newEmail != null) {
30         if (userRepository.findByEmail(newEmail) != null) {
31             ObjectError error = new ObjectError(objectName: "global", defaultMessage: "Email already exists!");
32             bindingResult.addError(error);
33         } else {
34             loggedInUser.setEmail(newEmail);
35         }
36     }
}
```

Edit My Profile

Username

Email

Current Password

New Password

Confirm Password

Save

7. Справки за:

- 7.1. Всички запазени в системата записи;
- 7.2. Записите с най-често срещани етикети (обобщени данни);
- 7.3. Всички записи с еднакви имена и различни фамилии;
- 7.4. Всички записи с еднакви фамилии и различни имена;
- 7.5. Запис с определено име и фамилия.

```
no usages  disto
@PostMapping("/contacts/search")
public String search(@RequestParam(value = "radio", required = false) String choice,
                    @RequestParam(value = "firstname", required = false) String firstname,
                    @RequestParam(value = "lastname", required = false) String lastname,
                    Model model, HttpSession httpSession) {

    if (choice == null) return "search";

    if (choice.equals("allRecords")) {
        model.addAttribute("contacts", contactService.findById((Integer) httpSession.getAttribute("LOGGED_USER_ID")));
    } else if (choice.equals("searchByFirstAndLastname")) {
        model.addAttribute("contacts", contactService.findByFirstnameAndLastnameAndUserId(firstname, lastname, (Integer) httpSession.getAttribute("LOGGED_USER_ID")));
    } else if (choice.equals("mostCommonLabels")) {
        model.addAttribute("contacts", contactService.getContactsWithMostCommonLabels((Integer) httpSession.getAttribute("LOGGED_USER_ID")));
    } else if (choice.equals("sameFirstnames")) {
        model.addAttribute("contacts", contactService.findBySameFirstnameAndDistinctLastname((Integer) httpSession.getAttribute("LOGGED_USER_ID")));
    } else if (choice.equals("sameLastnames")) {
        model.addAttribute("contacts", contactService.findBySameLastnameAndDistinctFirstname((Integer) httpSession.getAttribute("LOGGED_USER_ID")));
    }

    return "search";
}
```

```
1 usage  disto
public List<Contact> getContactsWithMostCommonLabels(Integer userId) {
    Map<Contact.Label, List<Contact>> groupedContacts = findById(userId) List<Contact>
        .stream() Stream<Contact>
        .collect(Collectors.groupingBy(Contact::getLabel));

    int maxSize = groupedContacts
        .values() Collection<List<Contact>>
        .stream() Stream<List<Contact>>
        .max((contacts1, contacts2) -> contacts1.size() - contacts2.size()) Optional<List<Contact>>
        .orElse(new ArrayList<>()) List<Contact>
        .size();

    return groupedContacts
        .values() Collection<List<Contact>>
        .stream() Stream<List<Contact>>
        .filter(contacts -> contacts.size() == maxSize)
        .flatMap(List::stream) Stream<Contact>
        .collect(Collectors.toList());
}
```

Search

- ☐ All records
- ☐ Most common labels
- ☐ Same first names
- ☐ Same last names
- ☐ Search by first name and last name

Search

```
1 usage  disto
@Query(value = "Select * from (SELECT * FROM contact WHERE user_id = :id) as t Where firstname IN " +
"(SELECT firstname FROM (SELECT * FROM contact WHERE user_id = :id) as t GROUP BY firstname HAVING COUNT(distinct lastname) > 1)", na
List<Contact> findBySameFirstnameAndDistinctLastname(int id);

1 usage  disto
@Query(value = "Select * from (SELECT * FROM contact WHERE user_id = :id) as t Where lastname IN " +
"(SELECT lastname FROM (SELECT * FROM contact WHERE user_id = :id) as t GROUP BY lastname HAVING COUNT(distinct firstname) > 1)", na
List<Contact> findBySameLastnameAndDistinctFirstname(int id);
```

8. Възможност за импортиране в различни формати на всички запазени записи

Методите са реализирани в клас DownloadController.

Download Contacts

Choose data type:

- ☐ .json
- ☐ .excel
- ☐ .csv

Download

```

24 @GetMapping("/contacts/download")
25 public String getDownloadPage(HttpSession httpSession) {
26     if (httpSession.getAttribute("LOGGED_USER_ID") == null) return "redirect:/login";
27     return "downloadContacts";
28 }
29
30 @PostMapping("/contacts/download")
31 public ResponseEntity<byte[]> downloadContacts(@RequestParam(value = "radio", required = false) String choice,
32                                             HttpSession httpSession) {
33
34     HttpHeaders headers = new HttpHeaders();
35     headers.add("Location", "/contacts/download");
36     if (choice == null) {
37         return new ResponseEntity<byte[]>(headers, HttpStatus.FOUND);
38     }
39
40     User loggedUser = sessionManager.getLoggedUser(httpSession);
41     if (choice.equals("json")) {
42         return downloadService.exportToJson(loggedUser.getId());
43     } else if (choice.equals("csv")) {
44         return downloadService.exportToCsv(loggedUser.getId());
45     } else if (choice.equals("excel")) {
46         return downloadService.exportToExcel(loggedUser.getId());

```

```

<h2 class="title">Download Contacts</h2>

<form action="/contacts/download" method="post">

    <p>Choose data type:</p>
    <div class="form-check">
        <label class="form-check-label">
            <input type="radio" class="form-check-input" name="radio" value="json" id="json">
            <label for="json">.json</label>
        </label>
    </div>
    <div class="form-check">
        <label class="form-check-label">
            <input type="radio" class="form-check-input" name="radio" value="excel" id="excel">
            <label for="excel">.excel</label>
        </label>
    </div>
    <div class="form-check">
        <label class="form-check-label">
            <input type="radio" class="form-check-input" name="radio" value="csv" id="csv">
            <label for="csv">.csv</label>
        </label>
    </div>

```