

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ
РОССИЙСКОЙ ФЕДЕРАЦИИ»**

Кафедра информационных технологий

Пояснительная записка к курсовой работе

по дисциплине «Современные технологии программирования»

на тему:

«Информационная система учета обращений пользователей в техническую
поддержку»

Выполнил(а):

студент(ка) группы ДПИ22-1с институт
открытого образования Берникова В.С.

Научный руководитель:

Кандидат технических наук Пальчевский
Е.В.

2024 г

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ	2
ВВЕДЕНИЕ.....	3
ОПИСАНИЕ ПРОГРАММЫ	4
1.1 Алгоритмические решения.....	4
1.1.1 Безопасность	4
1.1.2 Клиент	5
1.1.3 Сервер.....	6
1.1.4 База данных.....	7
1.1.5 Зависимости программы	8
1.2 Описание интерфейса программы	9
1.2.1 Авторизация (Регистрация).....	9
1.2.2 Личный кабинет	13
1.2.3 Панель администратора.....	18
ЗАКЛЮЧЕНИЕ	20
СПИСОК ИСТОЧНИКОВ.....	21
ПРИЛОЖЕНИЯ.....	23

ВВЕДЕНИЕ

Безусловно, темпы развития технологий, задаваемые нынешним временем - не только быстры, но и растут с каждым днём. Никакая компания сегодня не может существовать без IT-решения будь то сайт, приложение или веб-приложение. И конечно, при взаимодействии пользователей с таким IT-решением, будут появляться технические проблемы, из-за которых пользователи будут обращаться за помощью. Хорошо организованная система технической поддержки пользователей – залог успешного взаимодействия организации с клиентами, потому что все любят, когда их проблемы решают быстро и четко.

Таким образом, было решено разработать информационно-справочную систему типа ServiceDesk для регулирования коммуникации клиентов организации с технической поддержкой. Система подобного типа предоставляет инструменты для эффективной организации и приоритезации заявок. Более того, включает в себя оперативную и управленческую отчетность. [11]

Следуя из всего выше сказанного, целью текущей курсовой работы является реализация системы учёта обращений пользователей в техническую поддержку.

Достижение цели планируется по средствам выполнения нескольких этапов:

- создание серверной части,
- создание клиентской части,
- создание базы данных.

ОПИСАНИЕ ПРОГРАММЫ

1.1 Алгоритмические решения

В текущем разделе будут представлены описания использованных в процессе разработке решений.

1.1.1 Безопасность

Безопасность – неотъемлемая часть любого IT-решения. Поэтому ей было уделено отдельное внимание в рамках работы над данной информационно-справочной системой.

Было решено использовать Spring Security, являющуюся надежной платформой, поднимающей безопасность веб-приложений. Особенно на базе Spring. Функции типа аутентификация, авторизация и защита от распространенных атак предоставляются. [5]

Три основные концепции, на базе которых работает Spring Security, можно представить в виде списка:

- аутентификация (установление личности).
- авторизация (контроль доступа).
- защита (от веб-уязвимостей). [5]

Spring Security добавляет цепочки фильтров в приложение. Эти фильтры перехватывают HTTP-запросы и применяют правила. Современный подход к определению правил безопасности в Spring Security заключается в использовании SecurityFilterChain компонента. [5]

1.1.2 Клиент

Клиентская часть приложения – это та часть приложения, которая видна клиенту, и с которой клиент, собственно, взаимодействует. Поэтому для создания клиентской части были использованы инструменты разработки, подходящие к выбранному веб-модулю Spring, а именно:

- HTML,
- CSS,
- JavaScript,
- Bootstrap,
- Thymeleaf.

HTML самый популярный среди языков гипертекстовой разметки, к которому чаще всего прибегают разработчики. Потому что он состоит из элементов или тегов, которые определяют различные части веб-страницы. [7]

Каскадные таблицы стилей CSS будут нужны для внешнего оформления и настройки стилей, составляющих экран HTML-объектов. [8]

И хотя основная часть текущей работы будет выполнена с помощью Java, при работе с HTML так или иначе затрагивается и JavaScript, скрипты которого можно удобно вставлять прямо в гипертекстовую разметку. [2]

Для большей оптимизации и облегчения работы над дизайнерской частью веб-приложения было решено использовать Bootstrap, который содержит все необходимые компоненты и шаблоны веб-интерфейса. [9]

1.1.3 Сервер

Серверную часть было решено разрабатывать, используя веб-модуль Spring Boot. Потому что, обладая большим функционалом, он позволяет создавать web-приложения, не требуя от программиста отдельных усилий для его настройки, что позволяет сконцентрироваться на самом главном. [1]

1.1.4 База данных

В последнее время база данных PostgreSQL обогнала MySQL по популярности и частоте выбора среди разработчиков. Обусловлено это явление расширяемостью, масштабируемостью и кроссплатформенностью. Таким образом, было решено использовать PostgreSQL для реализации части базы данных [14].

Всё необходимое для работы описание параметров, без которых взаимодействие приложения с системой управления базами данных PostgreSQL было бы невозможным, находится в файле конфигураций «application.properties».

В самом начале вышеописанного файла задано имя приложения через параметр `spring.application.name`. Ему было присвоено значение «demo1».

Далее, с помощью параметра «`spring.datasource.url`» задаётся подключение к PostgreSQL.

И наконец, с помощью параметров «`spring.datasource.name`» и «`spring.datasource.password`» задаются данные для доступа к базе данных.

1.1.5 Зависимости программы

Управление зависимостями – непростая задача, особенно если речь идёт о больших проектах. Однако Spring Boot решает эту проблему путём предоставления набора зависимостей, облегчая жизнь разработчикам. [13]

В текущем проекте зависимости управляются с использованием файла `build.gradle`. Зависимости, использованные в программе, можно представить в виде списка:

- `spring-boot-starter-thymeleaf` (для поддержки шаблонизаторов),
- `spring-boot-starter-security` (для обеспечения безопасности),
- `spring-boot-starter-web-services` (для работы с веб-службами),
- `spring-boot-starter-data-jpa` (для работы с данными),
- драйвер `postgresql` (для работы с данными),
- `lombok` (для упрощения кода).

1.2 Описание интерфейса программы

Текущий раздел предоставляет полное описание интерфейса информационно-справочной системы.

1.2.1 Авторизация (Регистрация)

При запуске программы пользователя встречает страница авторизации. На рисунке 1 представлена страница авторизации.

Рисунок 1 – Страница авторизации

Стандартная авторизация производится за счет ввода логина и пароля пользователя. Здесь стоит уделить отдельное внимание реализованной функции уведомления пользователя о неправильном вводе данных авторизации. Если пользователь ввел неверные данные, он увидит сообщение, представленное на рисунке 2.

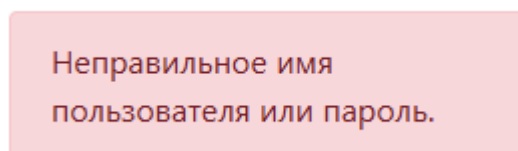


Рисунок 2 – Сообщение об ошибке

У каждого пользователя своя роль, «ROLE_ADNIN» или «ROLE_USER» и в зависимости от роли, пользователь будет перенаправлен на соответствующую страницу.

Если у пользователя нет аккаунта, то он может зарегистрироваться в системе, используя ссылку под кнопкой войти. После попадания на страницу регистрации пользователь увидит форму для заполнения. Форма для регистрации представлена на рисунке 3.

Регистрация

Имя пользователя:

Эл. почта:

Номер телефона:

Пароль

[Зарегистрироваться](#)

[Уже есть аккаунт? Войти](#)

© 2025 Все права защищены.

Рисунок 3 – Страница регистрации

После успешной регистрации пользователь может авторизоваться в системе и начать работу. Каждому новому пользователю при регистрации выдаётся роль «ROLE_USER», которую в дальнейшем при необходимости администратор может изменить.

Дополнением на страницы авторизации является раздел «об авторе», куда пользователь может перейти, нажав соответствующую ссылку.

Оказавшись на странице, представленной на рисунке 4, пользователь может ознакомиться с актуальной информацией об авторе проекта.

Об авторе

ФИО автора:

Берникова Валерия Сергеевна

Группа:

ДПИ22-1с

Контактные данные:

Email: 224911@edu.fa.ru

Телефон: +7 (968) 927-54-64

Использованные технологии

Java, SpringBoot, HTML, CSS, JavaScript.

Даты начала и завершения работ над проектом:

Начало работы: 01.01.2025

Завершение работы: 27.02.2025

[Назад в главное меню](#)

© 2025 Все права защищены.

Рисунок 4 – Об авторе

1.2.2 Личный кабинет

Личный кабинет – это первая страница, которую видит пользователь, авторизовавшись на информационно-справочной системе обращения пользователей в тех. поддержку.

В своём личном кабинете пользователь имеет возможность управлять своими заявками, просматривать их статус, создавать новые и редактировать существующие.

Конечно же функционал, а соответственно и интерфейс пользователей отличается в зависимости от роли. В верхнем правом углу находится информация о текущем пользователе, и кнопка «Выйти» которая предоставляет возможность покинуть текущий раздел информационной системы.

Верхняя часть страницы, представляет название раздела, оповещая пользователя о том, где он в данный момент находится.

Далее мы имеем несколько функциональных блоков.

Кнопка «Добавить заявку» перенаправляет пользователя на страницу создания заявки, представленную на рисунке 7.

Далее, как можно видеть, идёт поисковый блок. Он включает в себя поисковую строку, кнопку «Найти» и кнопку «Очистить».

При введении поискового запроса в поисковую строку и нажатии на кнопку «Найти» пользователь получает результаты из базы данных. Поддерживается полное и частичное совпадение.

На рисунке 5 представлена часть интерфейса личного кабинета пользователя.

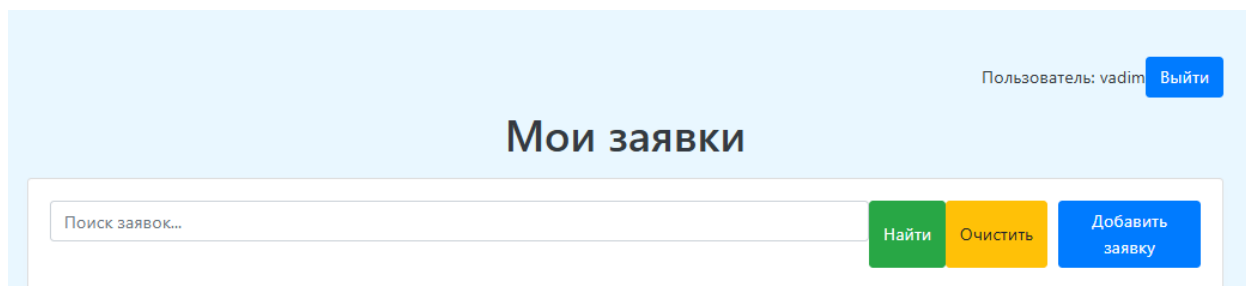


Рисунок 5 – Личный кабинет, верхняя часть страницы

В центральной/средней части, представленной на рисунке 6 (страница от лица пользователя с ролью «ROLE_USER»), располагается таблица, в которую происходит вывод всех заявок из базы данных.

Каждая заявка состоит из следующего набора информации, позволяющего быстро взаимодействовать с таблицей:

- номер заявки (уникальный),
- тема заявки,
- дата создания,
- дата изменения,
- статус (стадия работы с заявкой),
- описание заявки (сообщение прикрепленное при создании).

В отдельную колонку вынесена кнопка «Редактировать», существующая для каждой заявки. При нажатии на эту кнопку пользователь попадает на страницу редактирования, представленную на рисунке 8.

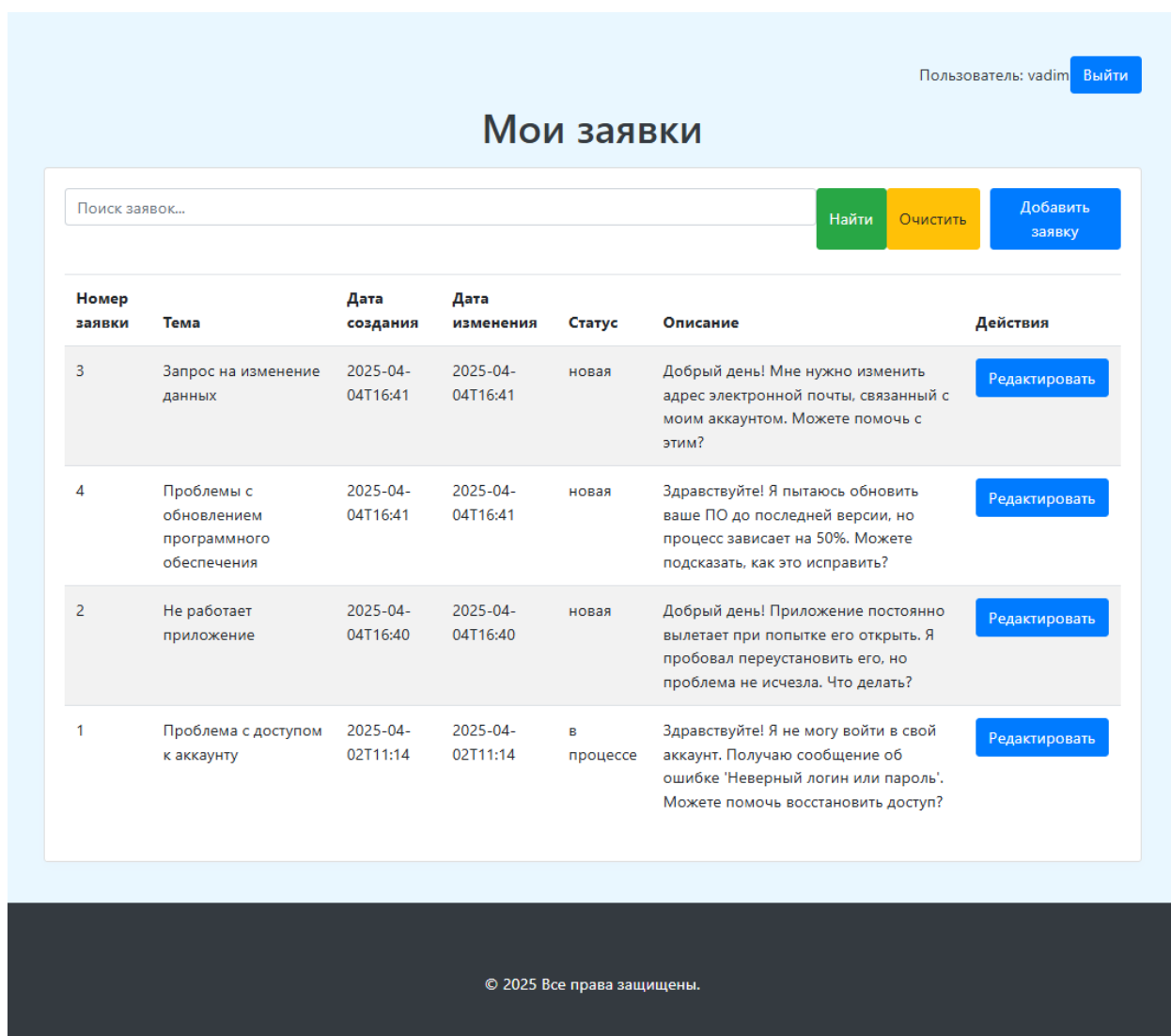




Рисунок 6 – Личный кабинет, полный скриншот

Упомянутая выше страница создания заявки, представленная на рисунке 7, позволяет пользователю создать новую заявку с обращением. Здесь можно указать тему обращения и подробно описать в чем именно заключается вопрос. Дата создания выставляется пользователем самостоятельно. Статус при создании выставляется как «Новая», что логично.

Кнопка «На главную» даёт возможность выйти из текущего раздела.

Создание заявки

Тема:	<input type="text"/>
Дата создания:	<input type="text" value="дд.мм.гггг --:--"/> 
Дата изменения:	<input type="text" value="дд.мм.гггг --:--"/> 
Статус:	<input type="text" value="Новая"/>
Описание:	<div><div></div><div></div></div>
<div>Сохранить</div>	


© 2025 Все права защищены.

Рисунок 7 – Создание заявки

Страница «Редактирование заявки», как видно на рисунке 8, даёт возможно изменить некоторые параметры заявки включая даты изменения статус, описание, что логично. Текущая страница создана для внесения изменений в существующую в базе данных заявку.

Кнопка «На главную» даёт возможность выйти из текущего раздела.

Редактирование заявки

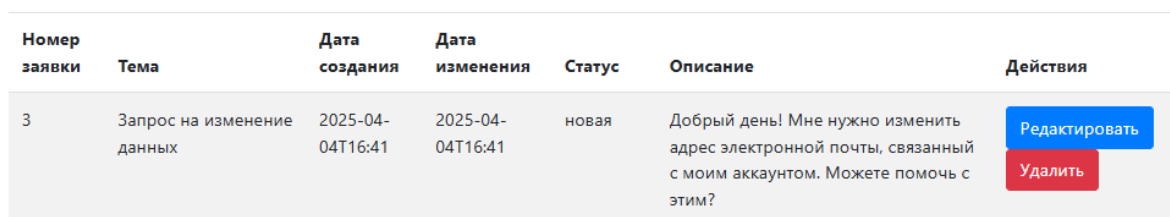
Номер заявки:	<input type="text" value="4"/>
Тема:	<input type="text" value="Проблемы с обновлением программного обеспечения"/>
Дата создания:	<input type="text" value="04.04.2025 16:41"/>
Дата изменения:	<input type="text" value="04.04.2025 16:41"/> 
Статус:	<input type="text" value="Отменена"/>
Описание:	<div>Здравствуйте! Я пытаюсь обновить ваше ПО до последней версии, но процесс зависает на 50%. Можете подсказать, как это исправить?</div>
<input type="button" value="Сохранить"/>	

© 2025 Все права защищены.

Рисунок 8 – Редактирование заявки

1.2.3 Панель администратора

Зайдя в систему от лица пользователя «ROLE_ADMIN», появляется дополнительная функция удаления существующих заявок, реализованная с помощью кнопки «Удалить» в колонке «Действия» основной таблицы в личном кабинете (рисунок 9).



Номер заявки	Тема	Дата создания	Дата изменения	Статус	Описание	Действия
3	Запрос на изменение данных	2025-04-04T16:41	2025-04-04T16:41	новая	Добрый день! Мне нужно изменить адрес электронной почты, связанный с моим аккаунтом. Можете помочь с этим?	Редактировать Удалить

Рисунок 9 – Личный кабинет пользователя «ADMIN»

Также у администратора есть отдельный раздел, где храниться информация о всех зарегистрированных пользователях системы.

Страница представляет собой таблицу, которая состоит из следующего набора информации:

- email (уникальный),
- номер телефона,
- активность,
- роли,
- заблокировать пользователя.

В колонке «Email», находится электронные почты всех зарегистрированных пользователей. В колонке «Номер телефона», номера пользователей, в колонке «Активность», имеет ли пользователь доступ к системе, в колонке «Роли», права доступа, ROLE_USER – для обычных пользователей и ROLE_ADMIN – для администраторов системы. Последняя колонке «Заблокировать пользователя», позволяет администратору заблокировать пользователя. Панель администратора представлена на рисунке 10.

[← На главную](#)

Техническая поддержка

Панель администратора

Email	Номер телефона	Активность	Роли	Заблокировать пользователя
v.bernikova@ma.ru	89999999999	Активен	ROLE_ADMIN	<input checked="" type="checkbox"/>
v.volkov@sd.com	84959999999	Активен	ROLE_USER	<input checked="" type="checkbox"/>

Рисунок 10 – Панель администратора

ЗАКЛЮЧЕНИЕ

В заключение стоит сказать, что в ходе выполнения текущей курсовой работы поставленная цель была достигнута.

Была разработана информационная система учета обращений пользователей в техническую поддержку типа ServiceDesk для организации коммуникации клиентов организации с технической поддержкой.

Были представлены подробные описания хода выполнения каждого этапа, а именно:

- описания этапа выбора инструментов для создания серверной части и создания серверной части;
- описание этапа выбора инструментов для создания клиентской части создания клиентской части;
- описания этапа выбора и создания базы данных.

Помимо этого, были закреплены практические навыки проектирования и разработки веб-приложения с нуля, используя SpringBoot, Java и HTML, CSS, JS и PostgreSQL.

В дальнейшем планируется продолжение работы над системой в рамках дипломного проекта.

СПИСОК ИСТОЧНИКОВ

1. Введение в Spring Boot: создание простого REST API на Java – [Электронный ресурс]. – URL: <https://habr.com/ru/articles/435144/> (дата обращения: 20.02.2025). – Текст: электронный.
2. Что такое JavaScript – [Электронный ресурс]. – URL: <https://digitalocean.ru/n/chto-takoe-javascript> (дата обращения: 20.02.2025). – Текст: электронный.
3. Thymeleaf – [Электронный ресурс]. – URL: <https://www.thymeleaf.org/> (дата обращения: 20.02.2025). – Текст: электронный.
4. Spring Boot – [Электронный ресурс]. – URL: <https://spring.io/projects/spring-boot> (дата обращения: 20.02.2025). – Текст: электронный.
5. Защита приложения Spring MVC с помощью Spring Security – [Электронный ресурс]. – URL: <https://www.geeksforgeeks.org/securing-a-spring-mvc-application-with-spring-security/> (дата обращения: 20.02.2025). – Текст: электронный.
6. Регистрация и авторизация с помощью Spring Security на примере простого приложения – [Электронный ресурс]. – URL: <https://habr.com/ru/articles/482552/> (дата обращения: 20.02.2025). – Текст: электронный.
7. Почему HTML называется Языком разметки? – [Электронный ресурс]. – URL: <https://www.geeksforgeeks.org/why-html-is-called-markup-language/> (дата обращения: 20.02.2025). – Текст: электронный.
8. Типы CSS (каскадная таблица стилей) – [Электронный ресурс]. – URL: <https://www.geeksforgeeks.org/types-of-css-cascading-style-sheet/> (дата обращения: 20.02.2025). – Текст: электронный.

9. Bootstrap – [Электронный ресурс]. – URL: <https://blog.skillfactory.ru/glossary/bootstrap/?ysclid=m7lq7h6wgy14424758> (дата обращения: 20.02.2025). – Текст: электронный.
10. Запускаем первое веб-приложение на Spring Boot – [Электронный ресурс]. – URL: <https://skillbox.ru/media/code/zapuskajem-pervoe-vebprilozhenie-na-spring-boot/> (дата обращения: 20.02.2025). – Текст: электронный.
11. ServiceDesk - система учета обращений – [Электронный ресурс]. – URL: <https://yandex.ru/q/article/servicedesk> (дата обращения: 20.02.2025). – Текст: электронный.
12. Драйвер PostgreSQL JDBC – [Электронный ресурс]. – URL: <https://www.geeksforgeeks.org/postgresql-jdbc-driver/> (дата обращения: 03.03.2025). – Текст: электронный.
13. Зависимости в Spring Boot – [Электронный ресурс]. – URL: <https://otus.ru/nest/post/1372/> (дата обращения: 03.03.2025). – Текст: электронный.
14. PostgreSQL (база данных) – [Электронный ресурс] – URL: <https://skillbox.ru/media/code/postgresql-vsye-cto-nuzhno-znat-dlya-bystrogo-starta/> (дата обращения 20.02.2025) – Текст: электронный

ПРИЛОЖЕНИЯ

Приложение А – логика контроллера заявок.

```
@Controller
@RequestMapping("/req")
public class ApplicationController {

    @Autowired
    private RequestService service;

    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String homePage(Model model, @RequestParam(name = "keyword", required = false) String keyword, Principal principal) {
        System.out.println("Keyword: " + keyword);
        List<Request> listRequest = service.listAll(keyword);
        model.addAttribute("listRequest", listRequest);
        model.addAttribute("keyword", keyword);
        model.addAttribute("user", service.getUserByPrincipal(principal));
        return "index";
    }

    @RequestMapping("/create")
    public String newForm(Model model) {
        var request = new Request();
        model.addAttribute("request", request);
        return "createRequest";
    }

    @RequestMapping(value = "/save", method = RequestMethod.POST)
    public String saveForm(@ModelAttribute("request") Request request) {
        service.save(request);
        return "redirect:/req/";
    }

    @RequestMapping("/edit/{id}")
    public ModelAndView editForm(@PathVariable(name = "id") Long id) {
        ModelAndView mav = new ModelAndView("editRequest");
        var request = service.get(id);
        mav.addObject("request", request);
        return mav;
    }

    @RequestMapping("/delete/{id}")
    public String deleteForm(@PathVariable(name = "id") Long id) {
        service.delete(id);
        return "redirect:/req/";
    }
}
```

Приложение Б – логика контроллера администратора.

```
@Controller
@RequiredArgsConstructor
@PreAuthorize("hasAuthority('ROLE_ADMIN')")
public class AdminController {
    private final UserService userService;

    @GetMapping("/admin")
    public String admin(Model model) {
        model.addAttribute("users", userService.list());
        return "admin";
    }

    @PostMapping("/admin/user/ban/{id}")
    public String userBan(@PathVariable("id") Long id) {
        userService.banUser(id);
        return "redirect:/admin";
    }
}
```


Приложение В – логика контроллера пользователя.

```
@Controller
@RequiredArgsConstructor
public class UserController {
    private final UserService userService;

    @GetMapping("/login")
    public String login() { return "login"; }

    @GetMapping("/registration")
    public String registration() { return "registration"; }

    @PostMapping("/registration")
    public String createUser(User user, Model model) {
        if (!userService.createUser(user)) {
            model.addAttribute("errorMessage", "Пользователь с email " + user.getEmail() + " уже существует!");
            return "registration";
        }

        return "redirect:/login";
    }

    @GetMapping("/hello")
    public String hello() {
        return "hello";
    }
}
```

Приложение Г – логика html страницы пользователей.

```
<div class="container">
  <!-- Блок авторизации -->
  <div class="btn-container">
    <div th:if="{user.email != null}" class="d-flex gap-2">
      <span class="text-dark align-self-center">Пользователь: [{user.name}]/</span><br>
      <form th:action="{@{/logout}}" method="post">
        <input type="hidden" name="_csrf" th:value="{_csrf.token}">
        <button type="submit" class="btn btn-custom">Выйти</button>
      </form>
      <a th:href="{@{/admin}}" th:if="{user.isAdmin()}" class="btn btn-custom">Админ-панель</a>
    </div>
    <a th:href="{@{/login}}" th:unless="{user.email != null}" class="btn btn-custom">Войти</a>
  </div>

  <!-- Заголовок и основная карточка -->
  <blockquote class="blockquote text-center"><h1>Мои заявки</h1></blockquote>
  <div class="card">
    <!-- Поиск и добавление -->
    <div class="btn-container mb-4">
      <form th:action="{@{/req/}}" method="get" class="d-flex gap-2 w-100">
        <input type="text" name="keyword" class="form-control" placeholder="Поиск заявок..." th:value="{keyword}">
        <input type="submit" class="btn btn-success" value="Найти">
        <input type="button" class="btn btn-warning" value="Очистить" onclick="clearSearch()">
      </form>
      <a th:href="{@{/req/create}}" class="btn btn-custom">Добавить заявку</a>
    </div>

    <!-- Таблица -->
    <table id="requestsTable" class="table table-striped table-hover">
      <thead>
        <tr>
          <th scope="col" onclick="sortTable(0)" title="Сортировать по номеру заявки">Номер заявки</th>
          <th scope="col" onclick="sortTable(1)" title="Сортировать по теме">Тема</th>
          <th scope="col" onclick="sortTable(2)" title="Сортировать по дате создания">Дата создания</th>
          <th scope="col" onclick="sortTable(3)" title="Сортировать по дате изменения">Дата изменения</th>
          <th scope="col" onclick="sortTable(4)" title="Сортировать по статусу">Статус</th>
          <th scope="col" onclick="sortTable(5)" title="Сортировать по описанию">Описание</th>
          <th scope="col">Действия</th>
        </tr>
      </thead>
    </table>
  </div>
</div>
```

Приложение Д – логика html страницы администратора.

```
<div class="container">
  <!-- Кнопка возврата -->
  <a href="/req/" class="btn btn-custom back-button">← На главную</a>

  <blockquote class="blockquote text-center">
    <h1>Панель администратора</h1>
  </blockquote>

  <!-- Таблица -->
  <table class="table table-striped table-hover">
    <thead>
      <tr>
        <th>Email</th>
        <th>Номер телефона</th>
        <th>Активность</th>
        <th>Роли</th>
        <th>Заблокировать пользователя</th>
      </tr>
    </thead>

    <tbody>
      <tr th:each="user : ${users}">
        <td th:text="${user.email}">Email отсутствует</td>
        <td th:text="${user.phoneNumber}">Номер телефона отсутствует</td>
        <td th:text="${user.active ? 'Активен' : 'Неактивен'}">Активность отсутствует</td>
        <td><span th:each="role : ${user.roles}" th:text="${role} + ' ' ">Роли отсутствует</span></td>
        <td>
          <form th:action="@{/admin/user/ban/{id}(id=${user.id})}" method="post">
            <input type="hidden" name="_csrf" th:value="${_csrf.token}">
            <button type="submit" class="btn btn-danger btn-sm">Да</button>
          </form>
        </td>
      </tr>
    </tbody>
  </table>

  ⚡</div>
```

Приложение Е – логика html страницы редактирования заявок.

```
<div class="container mt-5">
  <h1>Редактирование заявки</h1>
</blockquote>
<div class="row justify-content-center">
  <div class="col-md-8">
    <form action="#" th:action="@{/req/save}" th:object="${request}" method="post">
      <table class="table table-bordered text-center">
        <tr>
          <td>Номер заявки:</td>
          <td><input type="text" th:field="${id}" class="form-control" readonly="readonly" /></td>
        </tr>
        <tr>
          <td>Тема:</td>
          <td><input type="text" th:field="${title}" class="form-control" readonly="readonly" /></td>
        </tr>
        <tr>
          <td>Дата создания:</td>
          <td><input type="datetime-local" th:field="${createdDate}" class="form-control" readonly="readonly" /></td>
        </tr>
        <tr>
          <td>Дата изменения:</td>
          <td><input type="datetime-local" th:field="${modifiedDate}" class="form-control" /></td>
        </tr>
        <tr>
          <td>Статус:</td>
          <td>
            <select th:field="${status}" class="form-control">
              <option value="" disabled selected>Выберите статус</option>
              <option value="Новая">Новая</option>
              <option value="В процессе" th:if="${#authorization.expression('hasRole('ROLE_ADMIN'))}">В процессе</option>
              <option value="завершена" th:if="${#authorization.expression('hasRole('ROLE_ADMIN'))}">Завершена</option>
              <option value="отменена">Отменена</option>
            </select>
          </td>
        </tr>
        <tr>
          <td>Описание:</td>
          <td><textarea th:field="${description}" class="form-control" rows="4"></textarea></td>
        </tr>
      </table>
    </form>
  </div>
</div>
```

Приложение Ж – логика html страницы создания заявок.

```
<div class="container mt-5">
  <blockquote class="blockquote text-center">
    <h1>Создание заявки</h1>
  </blockquote>
  <div class="row justify-content-center">
    <div class="col-md-8">
      <form action="#" th:action="@{/req/save}" th:object="${request}" method="post">
        <table class="table table-bordered text-center">
          <tr>
            <td>Тема:</td>
            <td><input type="text" th:field="${title}" class="form-control" ></td>
          </tr>
          <tr>
            <td>Дата создания:</td>
            <td><input type="datetime-local" th:field="${createdDate}" class="form-control" ></td>
          </tr>
          <tr>
            <td>Дата изменения:</td>
            <td><input type="datetime-local" th:field="${modifiedDate}" class="form-control" ></td>
          </tr>
          <tr>
            <td>Статус:</td>
            <td>
              <select th:field="${status}" class="form-control">
                <option value="" disabled selected>Выберите статус</option>
                <option value="Новая">Новая</option>
              </select>
            </td>
          </tr>
          <tr>
            <td>Описание:</td>
            <td><textarea th:field="${description}" class="form-control" rows="4"></textarea></td>
          </tr>
          <tr>
            <td colspan="2">
              <button type="submit" class="btn btn-primary" data-toggle="button" aria-pressed="false" autocomplete="off">Сохранить</button>
            </td>
          </tr>
        </table>
      </form>
    </div>
  </div>
</div>
```

Приложение 3 – логика репозитория заявок.

```
public interface RequestRepo extends JpaRepository<Request, Long> { 2 usages
    @Query("SELECT p FROM Request p WHERE LOWER(CONCAT(p.id, ' ',
        "p.title, ' ', p.createdDate, ' ', p.modifiedDate, ' ',
        "p.status, ' ', p.description)) " +
        "LIKE LOWER(CONCAT('%', ?1, '%'))")
    List<Request> search(String keyword);
}
```

Приложение И – логика сущности заявок.

```
public class Request {  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
  
    private String title; 3 usages  
  
    @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss") 3 usages  
    @Temporal(TemporalType.TIMESTAMP)  
    private LocalDateTime createdDate;  
  
    @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss") 3 usages  
    @Temporal(TemporalType.TIMESTAMP)  
    private LocalDateTime modifiedDate;  
  
    private String status; // "NEW", "IN_PROGRESS", "COMPLETED" 3 usages  
    private String description; 3 usages  
  
    public Long getId() { return id; }  
    public void setId(Long id) { this.id = id; }  
    public String getTitle() { return title; }  
    public void setTitle(String title) { this.title = title; }  
    public LocalDateTime getCreatedDate() { return createdDate; }  
    public void setCreatedDate(LocalDateTime createdDate) { this.createdDate = createdDate; }  
    public LocalDateTime getModifiedDate() { return modifiedDate; }  
    public void setModifiedDate(LocalDateTime modifiedDate) { this.modifiedDate = modifiedDate; }  
    public String getStatus() { return status; }  
    public void setStatus(String status) { this.status = status; }  
    public String getDescription() { return description; }  
    public void setDescription(String description) { this.description = description; }
```

Приложение К – логика сервиса заявок.

```
@Service 4 usages
public class RequestService {
    @Autowired
    private RequestRepo repo;

    public List<Request> listAll(String keyword) { 2 usages
        if (keyword != null) {
            return repo.search(keyword);
        }
        return repo.findAll();
    }

    public void save(Request request) { 2 usages
        repo.save(request);
    }

    public Request get(Long id) { 2 usages
        return repo.findById(id).get();
    }

    public void delete(Long id) { 1 usage
        repo.deleteById(id);
    }
}
```


Приложение Л – логика сущности пользователей.

```
@Table(name = "users")
@Data
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Long id;

    @Column(name = "email", unique = true)
    private String email;

    @Column(name = "phone_number")
    private String phoneNumber;

    @Column(name = "name")
    private String name;

    @Column(name = "active")
    private boolean active;

    @Column(name = "password", length = 1000)
    private String password;

    @ElementCollection(targetClass = Role.class, fetch = FetchType.EAGER)
    @CollectionTable(name = "user_role", joinColumns = @JoinColumn(name = "user_id"))
    @Enumerated(EnumType.STRING)
    private Set<Role> roles = new HashSet<>();

    public boolean isAdmin() { return roles.contains(Role.ROLE_ADMIN); }
    @Override no usages
    public Collection<? extends GrantedAuthority> getAuthorities() { return roles; }
    @Override
    public String getUsername() { return email; }
    @Override no usages
    public boolean isAccountNonExpired() { return true; }
    @Override no usages
    public boolean isAccountNonLocked() { return true; }
    @Override no usages
    public boolean isCredentialsNonExpired() { return true; }
    @Override
    public boolean isEnabled() { return active; }
}
```

Приложение М – логика сервиса пользователей.

```
@Service 4 usages
@Slf4j
@RequiredArgsConstructor
public class UserService {
    private final UserRepository userRepository;
    private final PasswordEncoder passwordEncoder;

    public boolean createUser(User user) { 1 usage
        String email = user.getEmail();
        if (userRepository.findByEmail(email) != null) return false;
        user.setActive(true);
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        user.getRoles().add(Role.ROLE_USER);
        log.info("Saving new User with email: {}", email);
        userRepository.save(user);
        return true;
    }

    public List<User> list() {
        return userRepository.findAll();
    }

    public void banUser(Long id) { 1 usage
        User user = userRepository.findById(id).orElse( other: null);
        if (user != null) {
            boolean isActive = user.isActive();
            user.setActive(!isActive);
            userRepository.save(user);
        }
    }
}
```

Приложение Н – логика безопасности приложения.

```
@EnableWebSecurity
@Configuration
@RequiredArgsConstructor
public class WebSecurityConfig {
    private final CustomUserDetailsService userDetailsService;
    private final PasswordEncoder passwordEncoder;

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
            .authorizeHttpRequests(authorize -> authorize
                .requestMatchers(Ⓢ"/login", Ⓢ"/registration", Ⓢ"/aboutTheAuthor").permitAll()
                .anyRequest().authenticated()
            )
            .logout(logout -> logout
                .logoutUrl("/logout")
                .logoutSuccessUrl("/login")
                .permitAll()
            )
            .formLogin(form -> form
                .loginPage("/login")
                .loginProcessingUrl("/login")
                .defaultSuccessUrl("/req/")
                .failureUrl(authenticationFailureUrl: "/login?error")
            )
            .csrf(csrf -> csrf
                .csrfTokenRepository(CookieCsrfTokenRepository.withHttpOnlyFalse())
            );
        return http.build();
    }

    @Bean
    public AuthenticationManager authenticationManager(AuthenticationConfiguration authenticationConfiguration) throws Exception {
        return authenticationConfiguration.getAuthenticationManager();
    }

    @Bean
    public DaoAuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
        authProvider.setUserDetailsService(userDetailsService);
        authProvider.setPasswordEncoder(passwordEncoder);
        return authProvider;
    }
}
```

Приложение О – логика html страницы входа в приложение.

```
<div class="login-form">
  <h2>Вход в систему</h2>
  <!-- Отображение сообщения об ошибке -->
  <div th:if="{param.error}" class="alert alert-danger">
    Неправильное имя пользователя или пароль.
  </div>
  <form th:action="{/login}" method="post">
    <div class="form-group">
      <label for="email">Эл. почта:</label>
      <input type="text" class="form-control" id="email" name="username" placeholder="Введите почту" required>
    </div>
    <div class="form-group">
      <label for="password">Пароль</label>
      <input type="password" class="form-control" id="password" name="password" placeholder="Введите пароль" required>
    </div>
    <input type="hidden" name="_csrf" th:value="{_csrf.token}">
    <button type="submit" class="btn btn-primary btn-block">Войти</button>
  </form>
  <div class="text-center mt-3">
    <a th:href="{/registration}">Регистрация</a>
    <br>
    <a th:href="{/aboutTheAuthor}">Об авторе</a>
  </div>
</div>
```