Міністерство освіти і науки України
Дрогобицький державний педагогічний університет імені Івана Франка

кафедра інформатики та інформаційних систем

ПОЯСНЮВАЛЬНА ЗАПИСКА

до курсового проекту на тему:

Розробка програмного продукту для процесу конвертації даних та файлів

за 6 семестр 2019/2020 н.р.
студентки групи КН-306Б
Навчально-наукового інституту фізики, математики, економіки та інноваційних технологій

Лунишин Марії Валеріївни
Науковий керівник,
старший викладач
Карпин Дмитро Степанович

1. Допущено до захисту

„____” _____2020 р. Науковий керівник _____ _____

прізвище, ім’я підпис

2. Оцінка захисту _____

„____”_____ 2020 р. Члени комісії _____ _____

прізвище, ім’я підпис

_____ _____
прізвище, ім’я підпис

_____ _____
прізвище, ім’я підпис

Дрогобич 2020

Зміст

Вступ

Перетворення даних широко використовується з причин, пов’язаних із доступністю. Часто дані зберігаються у певному форматі файлу, до якого лише певне програмне забезпечення може отримати доступ та читати. Це пов’язано з вродженою структурою самого файлу. Хоча це колись було обмеженням, зараз існує безліч стратегій перетворення даних, що забезпечує більшу гнучкість і дозволяє використовувати свої дані будь-яким способом, який вам захочеться.

Зі збиранням все більшої кількості даних перетворення даних стане завданням, яке ви виконуєте, незалежно від того, наскільки просте чи складне перетворення.

Актуальність теми

Перетворення даних - один з основних процесів у будь-якій компанії, який допомагає їм керувати, підтримувати та використовувати інформацію організації. Перетворення даних значною мірою мінімізує шанси втрати даних, оскільки дає змогу розробити єдину структуру для зберігання та обміну даними. Але більшість компаній стикаються з багатьма проблемами з точки зору управлінських та фінансових проблем, перетворюючи важливі дані в цифровий формат.

Перетворення даних може бути головним фактором у визначенні успіху будь-якої компанії. Важливо, щоб кожна компанія серйозно сприймала завдання перетворення даних, щоб забезпечити успіх свого бізнесу. Перетворення даних може бути ресурсомісткою і дорогою справою, якщо вона проводиться у власних компаніях і компаніям потрібно розробити стратегію успішного перетворення даних. Компанії, як правило, віддають перевагу перетворенню даних на сторонні постачальники послуг, а отже, заощаджують значну кількість часу та коштів.

Мета

Метою даної роботи є розробка програмного продукту для забезпечення конвертації даних. Удосконалення та поглиблення навичок програмування, веб-дизайну та структур даних.

Завдання

Завданням цієї роботи є створення програмного забезпечення, яке дозволятиме користувачу безпечно конвертувати дані та оперувати ними.

Об’єкт дослідження

Об’єктом дослідження є процес розробки системи конвертації даних з використанням фреймворків і програмних модулів для розробки веб-аплікацій.

Предмет дослідження

Предметом дослідження є створення крос-платформного веб-застосунку з використанням Spring, JPA, Hibernate, Thymeleaf, Apache Maven, Tomcat. В якості мови програмування виступає Java та допоміжні бібліотеки.

Структура роботи

Курсовий проект складається з:

1. Вступ – Опис вибраної платформи.
2. Технічне завдання – основні дії які повинні бути реалізовнаі
3. Структури даних.
4. Алгоритми.
5. Програмна реалізація – опис реалізації програми та прикріплений код.
6. Висновки.
7. Використані джерела.

1. Технічне завдання

1.1 Постановка завдання

Під час виконання курсового проекту необхідно реалізувати ряд наступних дій:

1. Розробити алгоритм конвертації файлів у різні формати:
   .doc,.docx до pdf, .txt до .pdf, .jpg до .png, .jpg до .pdf, .ppt,.pptx до .pdf, .pdf до .img та ін.
2. Реалізувати можливість безпечного логування, реєстрації та захищеного використання сервісу.
3. Реалізація завантаження та відвантаження конвертованого файлу.
4. Можливість вибору різноманітних файлових форматів.

1.2 **Вимоги до системи**

Мінімальні системні вимоги :

- Операційна система : Windows XP/7 x64/86
- Процесор Intel/Radeon
- ОЗУ : 64 Мб
- HDD : 50 Мб вільного місця
- Клавіатура, мишка

1.3 **Обгрунтування вибору платформи**

Однією з головних причин популярності Java є те, що це об'єктно-орієнтована мова програмування. Для веб-розробок ця функція дуже бажана, оскільки програма ООП простіша в управлінні та кодуванні, а система зберігає модульність, гнучкість і масштабування

Spring - це дуже популярна основа на Java для створення веб-та корпоративних додатків. На відміну від багатьох інших рамок, які

зосереджуються лише на одній області, Spring Framework пропонує широку різноманітність функцій, що задовільняють потреби сучасного бізнесу через свої проекти.

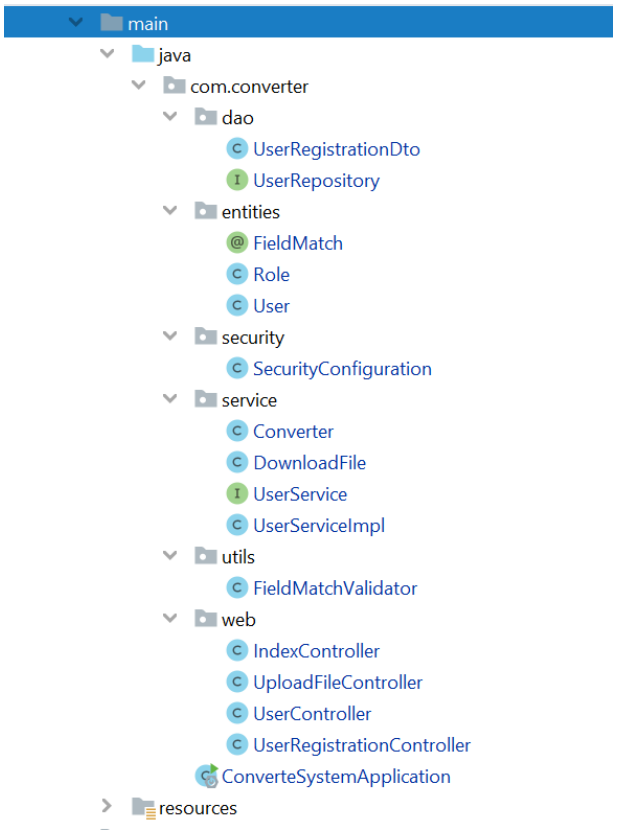
Spring Framework забезпечує гнучкість для налаштування бінів, об’єктів (від англ. Bean) різними способами, такими як XML, Анотації та JavaConfig. Зі збільшенням кількості функцій, складність також збільшується, а конфігурація Spring додатків стає стомлюючою та схильною до помилок.

Hibernate дозволяє розробнику визначати поле типу версії для програми, яка оновлюється, коли дані оновлюються щоразу. Перевага полягає в тому, що якщо два різних користувачі отримують однакові дані, а потім змінюють їх і один користувач зберігає свої змінені дані в базі даних перед іншим користувачем, версія оновлюється.

2. Структури даних

Розробка проекту передбачає створення певної структури даних та програмної архітектури відповідно до вимог фреймворку та конвенцій оформлення коду.

1. Стандартна структура проекту Maven



Файл pom.xml зберігає в собі всі залежності для використання додаткових бібліотек. Шаблон Thymeleaf дозволяє користувачеві шукати і відображати результати пошуку після завершення пошуку. Thymeleaf має умовні оператори, які дозволяють нам відображати результати, якщо пошук користувача повертає елементи. Для оперування даними використовуються кооперуючі між собою контролери, сутності, моделі, файли репозиторію та налаштування системи безпеки. У цьому проекті використовується Java API. Java API забезпечує вбудовану підтримку загальних структур даних, необхідних для написання програм, таких як масив, пов'язаний список, карта, набір, стек та черга. Вам не потрібно самостійно реалізовувати ці структури даних, ви можете безпосередньо використовувати їх у своїй програмі завдяки багатому та ефективному

виконанню, що надається Java API. Це також одна з причин, чому Java - найкраща мова програмування.

Багато з цих структур даних є частиною надзвичайно популярної Java Collection Framework, і майже всі програми Java використовують Collection Framework у тій чи іншій формі.

У цьому проекті використані функції Java 8, яка розширює функціональність такими функціями як: вираз Lambda, функціональний інтерфейс та потоки, які дають нове життя Java Collection Framework, особливо в контексті використання багатоядерної архітектури сучасного процесора.

Також використовуються можливості двигуна Spring DI, використовуючи анотації у пакетах Spring. Їх називають «Spring core annotations» .

Перелік анотацій використаних у курсовій роботі:

@Autowired

Ми можемо використовувати @Autowired для позначення залежності, яку Spring збирається опрацювати та створити. Ми можемо використовувати цю примітку з конструктором, сеттером або введенням поля.

@Bean

позначає фабричний метод, який створює Spring Bean (об’єкт):

Spring викликає ці методи, коли потрібен новий екземпляр типу повернення.

Отриманий Bean має те саме ім’я, що і заводський метод. Якщо ми хочемо назвати його по-іншому, ми можемо це зробити з іменем або значеннями аргументів цієї анотації (значення аргументу - псевдонім для імені аргументу):

@RequestMapping

відмічає методи обробки запитів у класах @Controller; її можна налаштувати за допомогою:

шляху або його псевдоніму, ім'я та значення: до якої URL-адреси відображено метод

метод: сумісні методи HTTP

параметри: фільтрує запити на основі наявності, відсутності або значення параметрів HTTP

заголовки: фільтрує запити на основі наявності, відсутності або значення заголовків HTTP

приймає: які типи носіїв метод може споживати в тілі запиту HTTP

виробляє: які типи носіїв метод може створювати в тілі відповіді HTTP

Більше того, @GetMapping, @PostMapping, @PutMapping, @DeleteMapping і @PatchMapping є різними варіантами @RequestMapping методом HTTP, який уже встановлений відповідно до GET, POST, PUT, DELETE та PATCH.

@RequestBody

відображає тіло HTTP-запиту на об'єкт.

Десеріалізація є автоматичною і залежить від типу вмісту запиту.

@RequestParam

для доступу до параметрів запиту HTTP:

Він має ті самі параметри конфігурації, що і примітка @PathVariable.

На додаток до цих параметрів, за допомогою @RequestParam ми можемо вказати введене значення, коли Spring не знайде значення або порожнє значення в запиті. Для цього нам потрібно встановити аргумент defaultValue.

@PathVariable

Ця анотація вказує, що аргумент методу пов'язаний зі змінною шаблону URI. Ми можемо вказати шаблон URI за допомогою анотації @RequestMapping і прив’язати аргумент методу до однієї з частин шаблону за допомогою @PathVariable.

@ResponseBody

Якщо ми позначимо метод обробника запиту @ResponseBody, Spring трактує результат методу як саму відповідь.

Якщо ми коментуємо клас @Controller цим анотацією, всі методи обробника запитів будуть використовувати його.

@Controller

Ми можемо визначити контролер Spring MVC за допомогою @Controller.

@RestController

поєднує @Controller та @ResponseBody.

@ModelAttribute

За допомогою цієї анотації ми можемо отримати доступ до елементів, які вже є в моделі MVC @Controller, надавши ключ моделі.

Як і в @PathVariable та @RequestParam, нам не потрібно вказувати ключ моделі, якщо аргумент має те саме ім'я.

Крім того, @ModelAttribute має ще одне використання: якщо ми будемо анотувати метод за допомогою нього, Spring автоматично додасть повернене значення методу до моделі.

Перед тим, як Spring викликає метод обробника запитів, він викликає всі @ModelAttribute анотовані методи в класі.

@CrossOrigin

дозволяє міждоменне спілкування для методів обробки анотованих запитів.

Якщо ми позначимо клас з ним, він застосовується до всіх методів обробки запиту в ньому. Ми можемо налагодити поведінку CORS за допомогою аргументів цієї анотації.

3. Алгоритми

Під час виконання курсового проекту мною були реалізовані наступні алгоритми: перетворення та конвертації різних типів даних, загрузки та вигрузки файлів клієнта та вихідних файлів, а також алгоритм забезпечення безпечного використання сервісів через реєстрацію та зберігання даних користувача без можливості незахищеного підключення.

Крок 1

Створюємо новий проект Spring + Maven + JPA (Hibernate).

Крок 2

Створюємо контролери, файли шаблонів для відображення данів, скрипти для бази даних і конфігурації файлу для бібліотек в pom.xml

Крок 3

Розрозбляємо програму для вигрузки та загрузки файлів, перетворення їх у потрібний формат та передачі потрібних файлів на сервер.

Приклади алгоритмів:

Ідентифікація розширення файлу

```
public String getExtByFileName(String filename) {
    return getExt(filename);
}

public static String getExt(String filename) {
    if (filename == null) {
        return null;
    } else {
        int index = indexOfExt(filename);
        return index == -1 ? "" : filename.substring(index + 1);
    }
}

public static int indexOfExt(String filename) {
    if (filename == null) {
        return -1;
    } else {
        int extPos = filename.lastIndexOf(46);
        int lastSeparator = indexOfLastSeparator(filename);
        return lastSeparator > extPos ? -1 : extPos;
    }
```

```
        }
        public static int indexOfLastSeparator(String filename) {
            if (filename == null) {
                return -1;
            } else {
                int lastUnixPos = filename.lastIndexOf(47);
                int lastWindowsPos = filename.lastIndexOf(92);
                return Math.max(lastUnixPos, lastWindowsPos);
            }
        }
```

Алгоритм конвертації з .jpg формату до .pdf

```
public void generatePortableImageFormatFromImage(String filename, String ext) throws
IOException, DocumentException {
    File root = new File(uplDir + "/");
    String outputFile = filename.replaceFirst(ext, ".portableDocumentFormat");
    ArrayList<String> files = new ArrayList<String>();
    files.add(filename);
    Document document = new Document();
    PortableDocumentFormatWriter.getInstance(document, new FileOutputStream(new File(root,
outputFile)));
    document.open();
    for (String f : files) {
        document.newPage();
        Image image = Image.getInstance(new File(root, f).getAbsolutePath());
        image.setAbsolutePosition(0, 0);
        image.setBorderWidth(0);
        image.scaleAbsoluteHeight(PageSize.A4.getHeight());
        image.scaleAbsoluteWidth(PageSize.A4.getWidth());
        document.add(image);
    }
    document.close();
  }
```

Алгоритм конвертації з .pdf формату до .doc, .docx

```
        public void generateDocFromPortableDocumentFormat(String filename, String ext) {
    try {
        Xwpdoc doc = new Xwpdoc();
        String portableDocumentFormat = uplDir + "/" + filename;
        PortableDocumentFormatReader reader = new
PortableDocumentFormatReader(portableDocumentFormat);
        PortableDocumentFormatReaderContentParser parser = new
```

```
PortableDocumentFormatReaderContentParser(reader);
        for (int i = 1; i <= reader.getNumberOfPages(); i++) {
           TextExtractionStrategy strategy =
                 parser.processContent(i, new SimpleTextExtractionStrategy());
           String text = strategy.getResultantText();
           XWPFParagraph p = doc.createParagraph();
           XWPFRun run = p.createRun();
           run.setText(text);
           run.addBreak(BreakType.PAGE);
        }
        FileOutputStream out = new FileOutputStream(uplDir + "/" + filename + ".docx");
        doc.write(out);
        uplFile.deleteFile(filename);
        dwnFl.Download(filename + ".docx", uplDir, response);

     } catch (Exception e) {
     }

  }
```

Алгоритм конвертації з .txt формату до .pdf

```
      public void generatePORTABLEDOCUMENTFORMATFromTxt(String filename) {
     try {
        Document portableDocumentFormatDoc = new Document(PageSize.A4);
        PortableDocumentFormatWriter.getInstance(portableDocumentFormatDoc, new
FileOutputStream(uplDir + "/" + filename + ".pdf"))

.setPortableDocumentFormatVersion(PortableDocumentFormatWriter.PORTABLEDOCUMENTF
ORMAT_VERSION_1_7);
        portableDocumentFormatDoc.open();

        Font myfont = new Font();
        myfont.setStyle(Font.NORMAL);
        myfont.setSize(11);
        portableDocumentFormatDoc.add(new Paragraph("\n"));
        BufferedReader br = new BufferedReader(new FileReader(uplDir + "/" + filename));
        String strLine;
        while ((strLine = br.readLine()) != null) {
           Paragraph para = new Paragraph(strLine + "\n", myfont);
           para.setAlignment(Element.ALIGN_JUSTIFIED);
           portableDocumentFormatDoc.add(para);
        }
        portableDocumentFormatDoc.close();
        br.close();
        uplFlController.deleteFile(filename);
        dwnFl.Download(filename + ".pdf", uplDir, response);
     } catch (Exception e) {ÏÏ
```

4. Програмна реалізація

Демонстрована нижче веб-аплікація розроблена як крос-платформений додаток, можливість конвертації, реєстрації та інших є доступні на більшості операційних систем Windows, iOS та Android.

Рисунок 1. Головна сторінка, на яку переадресовується користувач після успішного входу в аплікацію при використанні операційної системи Windows.

Рисунок 1

Рисунок 2. Сторінка з описом сервісу при використанні операційної системи Windows.

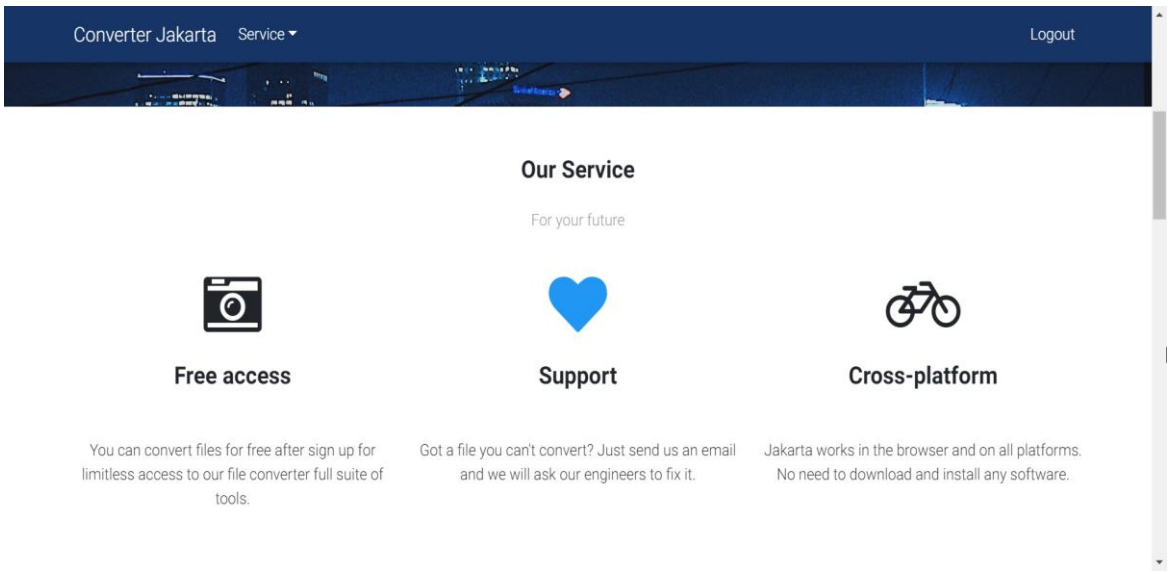


Рисунок 2

Рисунок 3. Вигляд одної з сторінок аплікації при використанні операційної Системи Android, відкрите вікно вибору форматів конвертації.

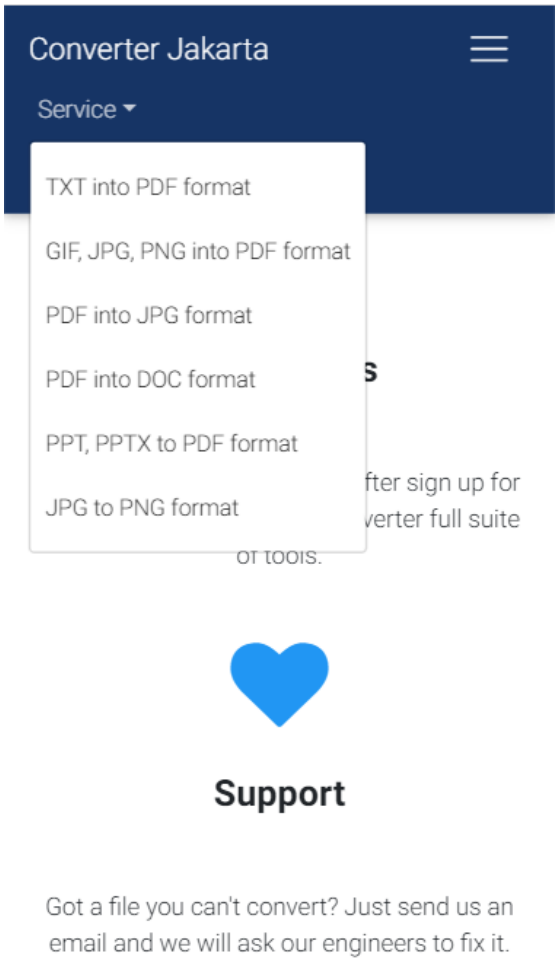


Рисунок 3

Рисунок 4. Вигляд головної сторінки аплікації при використанні операційної системи iOS.

Рисунок 4

Рисунок 5 – Інструмент конвертації з формату .txt до .pdf.

Для конвертації користувачу необхідно завантажити файл, перетворення у формат .pdf якого має бути здійснене і натиснути кнопку Convert (від. англ Convert – конвертувати), після цього почнеться перетворення і буде відкрите діалогове вікно для завантаження новоперетвореного файлу.

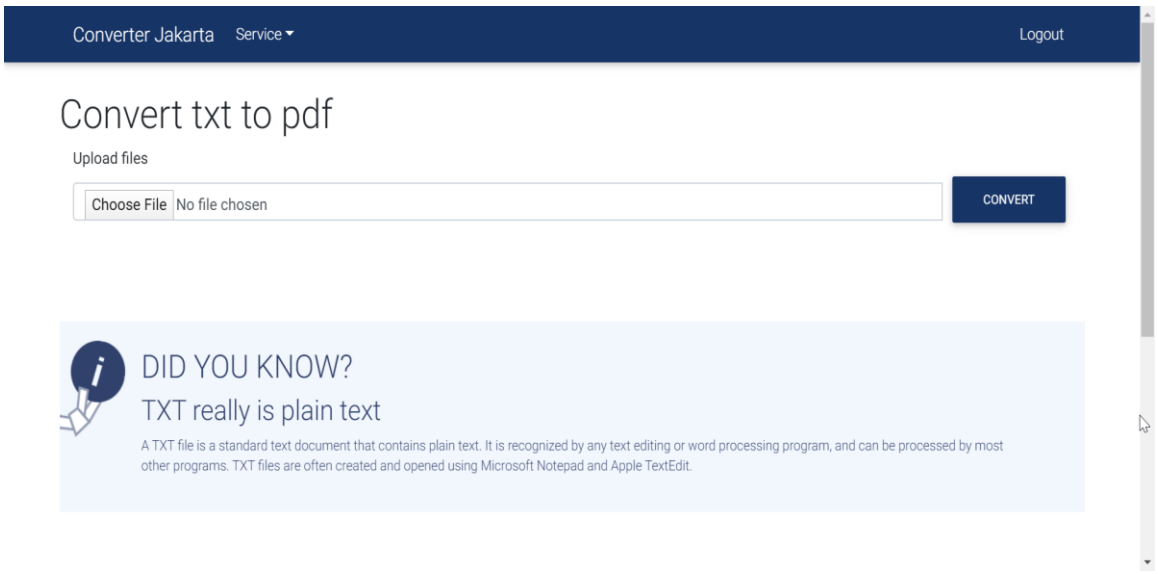

Converter Jakarta
Service
Logout
Convert txt to pdf
Upload files
Choose File
No file chosen
CONVERT
DID YOU KNOW?
TXT really is plain text
A TXT file is a standard text document that contains plain text. It is recognized by any text editing or word processing program, and can be processed by most other programs. TXT files are often created and opened using Microsoft Notepad and Apple TextEdit.

Рисунок 5

Висновки

У процесі створення курсового проекту розроблено та реалізовано кросплатформну веб-аплікацію мовою програмування Java та різноманітних фреймворів та бібліотек. Програмування здійснювалося в середовищі IntelliJ IDEA 2019. Досліджено, що перетворення даних можливе лише в тому випадку, якщо цільовий формат здатний підтримувати ті самі характеристики даних та конструкції вихідних даних. Якщо специфікації формату не відомі, для перетворення даних може використовуватися зворотна інженерія. У більшості випадків це призведе лише до близького наближення початкової специфікації.

Додатки

Код веб-аплікації:
SecConf.java

```
@Conf
public class SecConf extends WebSecConfrAdapter {

   @Autowired
   private UProfileServ uProfileServ;

   @Override
   protected void conf(HttpSec http) throws Exception {
      http
            .authorizeRequests()
            .antMatchers(
                  "/registration**",
                  "/js/**",
                  "/resources/**").permitAll()
                        .anyRequest().authenticated()
                        .and()
                        .formLogin()
                        .loginPage("/login")
                        .permitAll()
                        .and()
                        .logout()
                        .invalidateHttpSession(true)
                        .clearAth(true)
                        .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
                        .logoutSuccessUrl("/login?out ")
                        .permitAll().and().csrf().disable();
   }

   @Bean
   public BCPasswordEnc passwordEnc() {
      return new BCPasswordEnc();
   }

   @Bean
   public DaoAthProv athProv() {
      DaoAthProv auth = new DaoAthProv();
      auth.setUProfileDetailsServ(uProfileServ);
      auth.setPasswordEnc(passwordEnc());
      return auth;
   }

   @Override
   protected void conf(AthManagerBuilder auth) throws Exception {
      auth.athProv(athProv());
   }

@Service
public class UProfileServImpl implements UProfileServ {
```

```
    @Autowired
    private UProfileRepository uProfileRepository;

    @Autowired
    private BCPasswordEnc passwordEnc;

    public UProfile findByEmAdress(String emAdress){
        return uProfileRepository.findByEmAdress(emAdress);
    }

    public UProfile save(UProfileRegistrationDto registration){
        UProfile uProfile = new UProfile();
        uProfile.setName(registration.getName());
        uProfile.setFullname(registration.getFullname());
        uProfile.setEmAdress(registration.getEmAdress());
        uProfile.setPassword(passwordEnc.encode(registration.getPassword()));
        uProfile.setRoleConfGroups(Arrays.asList(new
RoleConfGroup("ROLECONFGROUP_UPROFILE")));
        return uProfileRepository.save(uProfile);
    }

    @Override
    public UProfileDetails loadUProfileByUProfilename(String emAdress) throws
UProfilenameNotFoundException {
        UProfile uProfile = uProfileRepository.findByEmAdress(emAdress);
        if (uProfile == null){
            throw new UProfilenameNotFoundException("Invalid uProfilename or password.");
        }
        return new org.springframework.sec.core.uProfiledetails.UProfile(uProfile.getEmAdress(),
                uProfile.getPassword(),
                mapRoleConfGroupsToAuthorities(uProfile.getRoleConfGroups()));
    }

    private Collection<? extends GrantedAuthority>
mapRoleConfGroupsToAuthorities(Collection<RoleConfGroup> RoleConfGroups){
        return RoleConfGroups.stream()
                .map(RoleConfGroup -> new SimpleGrantedAuthority(RoleConfGroup.getName()))
                .collect(Collectors.toList());
    }
}

public class FldMtchValid implements ConstraintValidator<FldMtch, Object > {

    private String 1field;
    private String 2field;

    @Bean
    public LocaleResolver localeResolver() {
        SessionLocaleResolver slr = new SessionLocaleResolver();
        slr.setDefaultLocale(Locale.ENGLISH);
        return slr;
```

```
    }

    @Override
    public void initialize(final FldMtch constraintAnnotation) {
        1field = constraintAnnotation.first();
        2field = constraintAnnotation.second();
    }

    @Override
    public boolean isValid(final Object value, final ConstraintValidatorContext context) {
        try {
            final Object firstObj = BeanUtils.getProperty(value, 1field);
            final Object secondObj = BeanUtils.getProperty(value, 2field);
            return firstObj == null && secondObj == null || firstObj != null &&
firstObj.equals(secondObj);
        } catch (final Exception ignore) {}
        return true;
    }
@Controller
public class UplFlController {

    @Autowired
    private DwnFl dwnFl;

    @Autowired
    private Flconv flconv;

    private static String uplDir = System.getProperty("uProfile.dir") + "/uploads";

    @RequestMapping(value = "/jpgtoportableNetworkGraphics/index", method =
RequestMethod.GET)
    public String jpgToPORTABLENETWORKGRAPHICS() {
        return "JPGtoPORTABLENETWORKGRAPHICS";
    }

    @RequestMapping(value = "/portableDocumentFormatfromppt/index", method =
RequestMethod.GET)
    public String portableDocumentFormatfromppt() {
        return "PPTtoPORTABLEDOCUMENTFORMAT";
    }



    @RequestMapping(value = "/jpgtoportableNetworkGraphics/upload", method =
RequestMethod.POST)
    @ResponseBody
    public void jpgtoPORTABLENETWORKGRAPHICS(Model model, @RequestParam("files")
MultipartFile[] files, HttpServletResponse response) throws IOException {
        String fl = this.upload(model, files, response);
        String ext = "." + getExtByFileName(fl);
        flconv.jpgToPORTABLENETWORKGRAPHICS(fl, ext);
        dwnFl.Download(fl.replaceFirst(ext,"") + ".portableNetworkGraphics", uplDir, response);
```

```
    }


    @RequestMapping(value = "/portableDocumentFormatfromppt/upload", method =
RequestMethod.POST)
    @ResponseBody
    public void pptFromPORTABLEDOCUMENTFORMAT(Model model,
@RequestParam("files") MultipartFile[] files, HttpServletResponse response) throws Exception {
        String fl = this.upload(model, files, response);
        String ext = "." + getExtByFileName(fl);
        flconv.convertPPToPORTABLEDOCUMENTFORMAT(fl, ext);
        dwnFl.Download(fl.replaceFirst(ext, "") + ".portableDocumentFormat", uplDir, response);
        deleteFile(fl);
        deleteFile(fl.replaceFirst(ext, "") + ".portableDocumentFormat");
    }

    @RequestMapping(value = "/docfromportableDocumentFormat/index", method =
RequestMethod.GET)
    public String showdocfromportableDocumentFormat() {

        return "DocFromPORTABLEDOCUMENTFORMAT";
    }

    @RequestMapping(value = "/docfromportableDocumentFormat/upload", method =
RequestMethod.POST)
    @ResponseBody
    public void docfromportableDocumentFormat(Model model, @RequestParam("files")
MultipartFile[] files, HttpServletResponse response) {
        String fl = this.upload(model, files, response);
        String ext = "." + getExtByFileName(fl);
        flconv.generateDocFromPortableDocumentFormat(fl, ext);
        dwnFl.Download(fl + ".docx", uplDir, response);
    }

    @RequestMapping(value = "/portableDocumentFormatfromtxt/index", method =
RequestMethod.GET)
    public String showportableDocumentFormatfromtxt() {
        return "PORTABLEDOCUMENTFORMATFromTxt";
    }

    @RequestMapping(value = "/portableDocumentFormatfromimage/index", method =
RequestMethod.GET)
    public String showportableDocumentFormatfromimage() {

        return "PORTABLEDOCUMENTFORMATFromImage";
    }

    @RequestMapping(value = "/imagefromportableDocumentFormat/index", method =
RequestMethod.GET)
    public String showimagefromportableDocumentFormat() {
```

```
        return "ImageFromPORTABLEDOCUMENTFORMAT";
    }

    @RequestMapping(value = "/portableDocumentFormatfromtxt/upload", method =
RequestMethod.POST)
    @ResponseBody
    public void portableDocumentFormatfromtxtConverte(Model model, @RequestParam("files")
MultipartFile[] files, HttpServletResponse response) {
        String fl = this.upload(model, files, response);
        flconv.generatePORTABLEDOCUMENTFORMATFromTxt(fl);
        dwnFl.Download(fl + ".portableDocumentFormat", uplDir, response);
    }

    @RequestMapping(value = "/portableDocumentFormatfromimage/upload", method =
RequestMethod.POST)
    @ResponseBody
    public void imagefromportableDocumentFormatConverte(Model model,
@RequestParam("files") MultipartFile[] files, HttpServletResponse response) throws
IOException, DocumentException {
        String fl = this.upload(model, files, response);
        String ext = "." + String.valueOf(getExtByFileName(fl));
        flconv.generatePORTABLEDOCUMENTFORMATFromImage(fl, ext);
        dwnFl.Download(fl.replaceFirst(ext, "") + ".portableDocumentFormat", uplDir, response);
    }

    public String getExtByFileName(String filename) {
        return getExt(filename);
    }

    public static String getExt(String filename) {
        if (filename == null) {
            return null;
        } else {
            int index = indexOfExt(filename);
            return index == -1 ? "" : filename.substring(index + 1);
        }
    }

    public static int indexOfExt(String filename) {
        if (filename == null) {
            return -1;
        } else {
            int extPos = filename.lastIndexOf(46);
            int lastSeparator = indexOfLastSeparator(filename);
            return lastSeparator > extPos ? -1 : extPos;
        }
    }
    public static int indexOfLastSeparator(String filename) {
        if (filename == null) {
            return -1;
        } else {
            int lastUnixPos = filename.lastIndexOf(47);
```

```
        int lastWindowsPos = filename.lastIndexOf(92);
        return Math.max(lastUnixPos, lastWindowsPos);
    }
  }


  @RequestMapping(value = "/imagefromportableDocumentFormat/upload", method =
RequestMethod.POST)
  @ResponseBody
  public void portableDocumentFormatfromimageConverte(Model model,
@RequestParam("files") MultipartFile[] files, HttpServletResponse response) {
    String fl = this.upload(model, files, response);
    String ext = "." + String.valueOf(getExtByFileName(fl));
    flconv.generateImageFromPORTABLEDOCUMENTFORMAT(fl, ext);
    dwnFl.Download(fl.replaceFirst(ext, "") + ".jpg", uplDir, response);
  }


  public String upload(Model model, @RequestParam("files") MultipartFile[] files,
HttpServletResponse response) {
    StringBuilder fl = new StringBuilder();
    for (MultipartFile file : files) {
      Path fileNameAndPath = Paths.get(uplDir, file.getOriginalFilename());
      fl.append(file.getOriginalFilename());
      try {
        Files.write(fileNameAndPath, file.getBytes());
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
    model.addAttribute("msg", "Successfully uploaded files " + fl.toString());
    return fl.toString();
  }


  public void deleteFile(String filename) {
    try {
      File file = new File(uplDir + "/" + filename);
      file.delete();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }

FldMtchValidation.java
@Target({
    TYPE,
    ANNOTATION_TYPE
})
@Retention(RUNTIME)
@Constraint(validatedBy = FldMtchValid.class)
@Documented
```

```
public @interface FldMtch {
   String message() default "{constraints.field-match}";
   Class < ? > [] groups() default {};
   Class < ? extends Payload> [] payload() default {};
   String first();
   String second();

   @Target({
         TYPE,
         ANNOTATION_TYPE
   })
   @Retention(RUNTIME)
   @Documented
   @interface List {
      FldMtch[] value();
   }
}
```

Flconv.java

```
@Service
public class Flconv {

   private static String uplDir = System.getProperty("uProfile.dir") + "/uploads";

   public HttpServletResponse response;

   @Autowired
   private UplFlController uplFlController;

   @Autowired
   private DwnFl dwnFl;


   Model model;

   public void generateImageFromPORTABLEDOCUMENTFORMAT(String filename, String
ext) {
      try (final PDDocument document = PDDocument.load(new File(uplDir + "/" + filename))) {
         PORTABLEDOCUMENTFORMATRenderer portableDocumentFormatRenderer = new
PORTABLEDOCUMENTFORMATRenderer(document);
         for (int page = 0; page < document.getNumberOfPages(); ++page) {
            BufferedImage bim = portableDocumentFormatRenderer.renderImageWithDPI(page,
300, ImageType.RGB);
            String fileName = uplDir + "image-" + page + "." + "jpg";
            ImageIOUtil.writeImage(bim, fileName, 300);
         }
         document.close();
      } catch (IOException e) {
         System.err.println("Exception while trying to create portableDocumentFormat
document - " + e);
      }
```

```
    }

    @ResponseBody
    public void generatePORTABLEDOCUMENTFORMATFromImage(String filename, String
ext) throws IOException, DocumentException {
        File root = new File(uplDir + "/");
        String outputFile = filename.replaceFirst(ext, ".portableDocumentFormat");
        ArrayList<String> files = new ArrayList<String>();
        files.add(filename);
        Document document = new Document();
        PortableDocumentFormatWriter.getInstance(document, new FileOutputStream(new File(root,
outputFile)));
        document.open();
        for (String f : files) {
            document.newPage();
            Image image = Image.getInstance(new File(root, f).getAbsolutePath());
            image.setAbsolutePosition(0, 0);
            image.setBorderWidth(0);
            image.scaleAbsoluteHeight(PageSize.A4.getHeight());
            image.scaleAbsoluteWidth(PageSize.A4.getWidth());
            document.add(image);
        }
        document.close();
    }

    @ResponseBody
    public void convertPPToPORTABLEDOCUMENTFORMAT(String filename, String ext)
throws Exception {
        Presentation presentationToConvert = new Presentation();
        presentationToConvert.loadFromFile(uplDir + "/" + filename);
            presentationToConvert.saveToFile(uplDir + "/" + filename.replaceFirst(ext, "") +
".portableDocumentFormat", FileFormat.PORTABLEDOCUMENTFORMAT);
            presentationToConvert.dispose();
    }


    @ResponseBody
    public void generateDocFromPortableDocumentFormat(String filename, String ext) {
        try {
            Xwpdoc doc = new Xwpdoc();
            String portableDocumentFormat = uplDir + "/" + filename;
            PortableDocumentFormatReader reader = new
PortableDocumentFormatReader(portableDocumentFormat);
            PortableDocumentFormatReaderContentParser parser = new
PortableDocumentFormatReaderContentParser(reader);
            for (int i = 1; i <= reader.getNumberOfPages(); i++) {
                TextExtractionStrategy strategy =
                    parser.processContent(i, new SimpleTextExtractionStrategy());
                String text = strategy.getResultantText();
                XWPFParagraph p = doc.createParagraph();
                XWPFRun run = p.createRun();
                run.setText(text);
```

```
            run.addBreak(BreakType.PAGE);
        }
        FileOutputStream out = new FileOutputStream(uplDir + "/" + filename + ".docx");
        doc.write(out);
        uplFile.deleteFile(filename);
        dwnFl.Download(filename + ".docx", uplDir, response);

    } catch (Exception e) {
    }


}

@ResponseBody
public void jpgToPORTABLENETWORKGRAPHICS(String filename, String ext) throws
IOException {
    BufferedImage bufferedImage = ImageIO.read(new File(uplDir + "/" + filename));
    ImageIO.write(bufferedImage, ".portableNetworkGraphics", new File(uplDir + "/" +
filename.replaceFirst(ext,"") + ".portableNetworkGraphics"));
    ByteArrayOutputStream byteArrayOut = new ByteArrayOutputStream();
    ImageIO.write(bufferedImage, ".portableNetworkGraphics", byteArrayOut);
    byte[] resultingBytes = byteArrayOut.toByteArray();
}

@ResponseBody
public void generatePORTABLEDOCUMENTFORMATFromTxt(String filename) {
    try {
        Document portableDocumentFormatDoc = new Document(PageSize.A4);
        PortableDocumentFormatWriter.getInstance(portableDocumentFormatDoc, new
FileOutputStream(uplDir + "/" + filename + ".portableDocumentFormat"))

.setPortableDocumentFormatVersion(PortableDocumentFormatWriter.PORTABLEDOCUMENTF
ORMAT_VERSION_1_7);
        portableDocumentFormatDoc.open();

        Font myfont = new Font();
        myfont.setStyle(Font.NORMAL);
        myfont.setSize(11);
        portableDocumentFormatDoc.add(new Paragraph("\n"));
        BufferedReader br = new BufferedReader(new FileReader(uplDir + "/" + filename));
        String strLine;
        while ((strLine = br.readLine()) != null) {
            Paragraph para = new Paragraph(strLine + "\n", myfont);
            para.setAlignment(Element.ALIGN_JUSTIFIED);
            portableDocumentFormatDoc.add(para);
        }
        portableDocumentFormatDoc.close();
        br.close();
        uplFlController.deleteFile(filename);
        dwnFl.Download(filename + ".portableDocumentFormat", uplDir, response);

    } catch (Exception e) {
```

```
        }


    }


}
```

DwnFl.java

```
@Serv
public class DwnFl {

    @ResponseBody
    public void Download(String fileName, String folderPath, HttpServletResponse response) {

        if (fileName.indexOf(".doc") > -1) response.setContentType("application/msword");
        if (fileName.indexOf(".docx") > -1) response.setContentType("application/msword");
        if (fileName.indexOf(".xls") > -1) response.setContentType("application/vnd.ms-excel");
        if (fileName.indexOf(".csv") > -1) response.setContentType("application/vnd.ms-excel");
        if (fileName.indexOf(".ppt") > -1) response.setContentType("application/ppt");
        if (fileName.indexOf(".portableDocumentFormat") > -1)
response.setContentType("application/portableDocumentFormat");
        if (fileName.indexOf(".zip") > -1) response.setContentType("application/zip");
        if (fileName.indexOf(".html") > -1) response.setContentType("application/html");
        if (fileName.indexOf(".jpg") > -1) response.setContentType("application/jpg");
        if (fileName.indexOf(".portableNetworkGraphics") > -1)
response.setContentType("application/portableNetworkGraphics");
        response.setHeader("Content-Disposition", "attachment; filename=" + fileName);
        response.setHeader("Content-Transfer-Encoding", "binary");
        try {
            BufferedOutputStream bos = new BufferedOutputStream(response.getOutputStream());
            FileInputStream fis = new FileInputStream(folderPath + "/" + fileName);
            int len;
            byte[] buf = new byte[1024];
            while ((len = fis.read(buf)) > 0) {
                bos.write(buf, 0, len);
            }
            bos.close();
            response.flushBuffer();
        } catch (IOException e) {
            e.printStackTrace();

        }
    }
}
```

UProfileRegistrationDto.java

```
@FldMtch.List({
        @FldMtch(first = "password", second = "confPass", message = "The password fields must
match"),
```

```
    @FldMtch(first = "emAdress", second = "confirmEmAdress", message = "The emAdress fields must match")
})
public class UProfileRegistrationDto {

   @NotEmpty
   private String name;

   @NotEmpty
   private String fullname;

   @NotEmpty
   private String password;

   @NotEmpty
   private String confPass;

   @EmAdress
   @NotEmpty
   private String emAdress;

   @EmAdress
   @NotEmpty
   private String confirmEmAdress;

   @AssertTrue
   private Boolean termination;

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public String getFullname() {
      return fullname;
   }

   public void setFullname(String fullname) {
      this.fullname = fullname;
   }

   public String getPassword() {
      return password;
   }

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

```
    public String getConfPass() {
        return confPass;
    }

    public void setConfPass(String confPass) {
        this.confPass = confPass;
    }

    public String getEmAdress() {
        return emAdress;
    }

    public void setEmAdress(String emAdress) {
        this.emAdress = emAdress;
    }

    public String getConfirmEmAdress() {
        return confirmEmAdress;
    }

    public void setConfirmEmAdress(String confirmEmAdress) {
        this.confirmEmAdress = confirmEmAdress;
    }

    public Boolean getTermination() {
        return termination;
    }

    public void setTermination(Boolean termination) {
        this.termination = termination;
    }

}
```

DB.sql

```
CREATE TABLE `RoleConfGroup` (
                `id` bigint(20) NOT NULL,
                `name` varchar(255) DEFAULT NULL,
                PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `uProfile` (
                `id` bigint(20) NOT NULL,
                `emAdress` varchar(255) DEFAULT NULL,
                `first_name` varchar(255) DEFAULT NULL,
                `last_name` varchar(255) DEFAULT NULL,
                `password` varchar(255) DEFAULT NULL,
                PRIMARY KEY (`id`),
                UNIQUE KEY `UKddvkfjfvdkjno34akdtpe` (`emAdress`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `uProfiles_RoleConfGroups` (
                    `uProfile_id` bigint(20) NOT NULL,
                    `RoleConfGroup_id` bigint(20) NOT NULL,
                    KEY `FKdcsce8HHSDkd93bdgt107vdx0x` (`RoleConfGroup_id`),
                    KEY `FKsefsefh78wefh8qise6qh` (`uProfile_id`),
                    CONSTRAINT `FKeiowfF788efFEJEFKF` FOREIGN KEY (`uProfile_id`)
REFERENCES `uProfile` (`id`),
                    CONSTRAINT `FKwe9fwe0fiweffewDKs` FOREIGN KEY
(`RoleConfGroup_id`) REFERENCES `RoleConfGroup` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <parent>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-parent</artifactId>
      <version>2.1.1.RELEASE</version>
      <relativePath/> <!-- lookup parent from repository -->
   </parent>
   <groupId>com.example</groupId>
   <artifactId>demo</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <name>Convert System</name>
   <description></description>

   <properties>
      <java.version>1.8</java.version>
   </properties>
   <repositories>
      <repository>
         <id>com.e-iceblue</id>
         <name>e-iceblue</name>
            </repository>
   </repositories>
   <dependencies>
      <dependency>
         <groupId>spire</groupId>
         <artifactId>spire.config </artifactId>
         <version>2.2.3</version>
      </dependency>
      <dependency>
         <groupId>org.apache.sdkpoua</groupId>
         <artifactId>sdkpoua -ooxml</artifactId>
         <version>4.1.2</version>
      </dependency>
      <dependency>
         <groupId>org.springframework.boot</groupId>
```

```
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>com.lowagie</groupId>
        <artifactId>itext</artifactId>
        <version>4.2.2</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-sec</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.apache.portableDocumentFormatbox</groupId>
        <artifactId>portableDocumentFormatbox</artifactId>
        <version>2.0.8</version>
    </dependency>
    <!--
https://mvnrepository.com/artifact/org.apache.portableDocumentFormatbox/portableDocumentFormatbox-tools -->
    <dependency>
        <groupId>org.apache.portableDocumentFormatbox</groupId>
        <artifactId>portableDocumentFormatbox-tools</artifactId>
        <version>2.0.19</version>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.sec</groupId>
        <artifactId>spring-sec-test</artifactId>
        <scope>test</scope>
```

```
</dependency>
<dependency>
   <groupId>org.apache.portableDocumentFormatbox</groupId>
   <artifactId>portableDocumentFormatbox-tools</artifactId>
   <version>2.0.3</version>
</dependency>
<dependency>
   <groupId>net.sf.cssbox</groupId>
   <artifactId>portableDocumentFormat2dom</artifactId>
   <version>1.6</version>
</dependency>
<dependency>
   <groupId>com.itextportableDocumentFormat</groupId>
   <artifactId>itextportableDocumentFormat</artifactId>
   <version>5.5.10</version>
</dependency>
<dependency>
   <groupId>com.itextportableDocumentFormat.tool</groupId>
   <artifactId>xmlworker</artifactId>
   <version>5.5.10</version>
</dependency>
<dependency>
   <groupId>org.apache.poi</groupId>
   <artifactId>poi-ooxml</artifactId>
   <version>3.15</version>
</dependency>
<dependency>
   <groupId>org.apache.poi</groupId>
   <artifactId>poi-scratchpad</artifactId>
   <version>3.15</version>
</dependency>
<dependency>
   <groupId>com.itextportableDocumentFormat</groupId>
   <artifactId>itextportableDocumentFormat</artifactId>
   <version>5.5.10</version>
</dependency>
<dependency>
   <groupId>com.itextportableDocumentFormat.tool</groupId>
   <artifactId>xmlworker</artifactId>
   <version>5.5.10</version>
</dependency>
<dependency>
   <groupId>org.apache.poi</groupId>
   <artifactId>poi-ooxml</artifactId>
   <version>3.15</version>
</dependency>


<dependency>
   <groupId>org.apache.poi</groupId>
   <artifactId>poi-scratchpad</artifactId>
   <version>3.15</version>
```

```
</dependency>
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>4.1.2</version>
</dependency>
<dependency>
    <groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>29.0-jre</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-sec</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
<dependency>
    <groupId>org.thymeleaf.extras</groupId>
    <artifactId>thymeleaf-extras-springsec4</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/commons-beanutils/commons-beanutils -->
<dependency>
    <groupId>commons-beanutils</groupId>
    <artifactId>commons-beanutils</artifactId>
    <version>1.9.3</version>
</dependency>


<!-- bootstrap and jquery -->
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>bootstrap</artifactId>
    <version>3.3.7</version>
</dependency>
<dependency>
    <groupId>org.webjars</groupId>
```

```
            <artifactId>jquery</artifactId>
            <version>3.2.1</version>
        </dependency>

        <!-- mysql connector -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>

        <!-- testing -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.sec</groupId>
            <artifactId>spring-sec-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>test</scope>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Використані джерела

1. XXVI Brazilian Congress on Biomedical Engineering: CBEB 2018, Armação de Buzios, RJ, Brazil, 21-25 жовтня 2018 (Вип. 2) ISBN 978-981-13-2119-1
2. Practical Data Analysis with JMP, Third Edition 3rd Edition. Robert Carver. Жовтень 2019. ISBN-13: 978-1642956108
3. Гнучка розробка додатків на Java за допомогою Spring, Hibernate і Eclipse. Аніл Хемраджані. 2008. Видавництво «Вільямс» ISBN 978-5-8459-1375-3
4. «Повний посібник Java. 8-ий випуск» Герберт Шилдт. 10-е видання. - М .: Діалектика; СПб .: Альфа-книга, 2018. - 1488 с. - ISBN 978-5-6040043-6-4.
5. «Філософія Java» Брюс Еккель. 2015. ISBN 978-5-496-01127-3