УДК 004.4

Одарущенко О. Б., к.т.н., доцент,
Томко В. Ю., магістрант
Полтавський національний технічний
університет імені Юрія Кондратюка

РОЗРОБКА СЕРВІСУ УПРАВЛІННЯ ДОКУМЕНТООБІГОМ ПІДПРИЄМСТВА У СФЕРІ ПОСЛУГ

Анотація. *У статті розглянуті методи розробки сервісу управління документообігом підприємства. Мета роботи полягає в розробці системи управління документообігом підприємства із обмеженим обсягом функцій, що дозволить ефективно та зручно проводити щоденні операції користувачем. Методи розробки включають вивчення матеріалу, використання об'єктно-орієнтованого програмування, робота з базами даних MySQL.*

Ключові слова: *управління документообігом, ООП, СУБД MySQL.*

Вступ

Основою роботи підприємства в сфері послуг є взаємодія з клієнтами. Крім основного завдання, а саме надання низки послуг клієнтам, потрібно вирішувати ряд задач, що відносяться до діяльності самого підприємства.

До них можна віднести: облік грошових коштів; формування клієнтської бази; статистичний облік клієнтів; формування статистики по прибуткам і витратам; вирішення задач з планування, бюджетування та фінансового обліку; нарахування заробітної плати працівникам; управління документообігом; формування первинних документів (рахунків, накладних, актів); формування бланків податкових декларацій. Наведений список є далеко не повним переліком завдань з обліку діяльності підприємства.

Для спрощення роботи з управління підприємством існує ряд засобів автоматизації. До них відносяться як стандартні офісні програми, так і спеціалізовані облікові системи. Стандартні офісні додатки можуть застосовуватися на підприємствах з малим переліком послуг і невеликою кількістю клієнтів. У свою чергу, спеціалізовані облікові системи підходять для будь-яких типів підприємств, але досить складні у попередньому налаштуванні та подальшому використанні у повсякденній праці.

На даний момент доступний великий перелік спеціалізованих систем обліку, які призначені для певних типів підприємств. Найбільш поширеними на території України є програмні продукти 1С:Підприємство [1], облікова система «Таксер»[2] та «Діловод»[2]. Дані системи дозволяють вести облік майна та керувати господарчими процесами, вирішувати завдання з обліку грошових коштів, основних та оборотних засобів, нарахування заробітної плати, та завдання з керування первинними документами та інші. Перелічені функціональні можливості роблять ці системи універсальними, що в свою чергу значно ускладнює процес роботи з ними.

Таким чином, *актуальним завданням* є розробка програми, яка дозволить вести облік підприємства, і при цьому містить набір лише тих функціональних можливостей, які щоденно необхідні користувачу. Дане усічення загальної множини функцій дозволить знизити рівень складності вихідної системи з точки зору первинного налагодження та подальшого застосування системи.

Виходячи з актуальності поставленої задачі *метою роботи* є розробка системи управління документообігом підприємства із обмеженим обсягом функцій, що дозволить ефективно та зручно проводити щоденні операції користувачем.

Для досягнення поставленої мети необхідно вирішити низку *окремих задач*, а саме: проаналізувати існуючі системи документообігу підприємства; провести дослідження та визначити обсяг мінімально необхідного та достатнього функціоналу; сформулювати технічне завдання на програмний продукт, що розробляється; розробити логіку функціонування програмного

продукту; розробити інтерфейс програмного продукту; спроектувати та розробити базу даних для зберігання необхідної інформації, щодо програмного продукту; спроектувати та розробити програмний продукт; провести верифікацію та тестування інтерфейсу, бази даних та безпосередньо програмного продукту.

Розроблений в *результаті* веб-додаток можна віднести до спеціалізованих систем обліку. Він має дозволяти у напівавтоматичному режимі здійснювати контроль за визначеними пунктами діяльності підприємства.

Для розробки серверної частини веб-додатку буде використано мову програмування Java. Даний вибір обумовлений портативністю, простотою опанування технологій розробки, оскільки платформа Java надає вичерпний набір документації. У зв'язку з вибором мови програмування, в якості інтегрованої середовища розробки була обрана Eclipce IDE. Критерієм вибору служить простота даного середовища розробки, за наявності необхідних інструментів для розробки.

Необхідний та достатній перелік функцій розроблюваного сервісу

Додаток, що розроблюється, повинен надавати засоби для створення та підтримки клієнтської бази, а також забезпечити можливість зберігання документів, які прикріплюються до клієнта, а саме договір, акти и рахунки.

В частині обліку додаток повинен забезпечити процес фіксування прибутків та витрат підприємства. При цьому користувач повинен мати можливість самостійно формувати перелік статей прибутків та витрат.

При цьому додаток має формувати деякі статистичні відомості про діяльність підприємства, а саме статистику за готівковим і безготівковим розрахунком, статистику за кількістю наданих послуг. А також потрібно забезпечити формування звіту про рух грошових коштів.

Звіт про рух грошових коштів надає відомості про надходження і вибуття грошових коштів протягом звітного періоду в результаті діяльності

підприємства. У підприємницькій діяльності такий звіт використовується як елемент планування та бюджетування. Також цей звіт може бути зручним інструментом фінансового контролю діяльності підприємства.

Звіт про рух грошових коштів дозволяє отримати стислу та чітку інформацію щодо фінансового стану підприємства, що сприяє прийняттю правильних оперативних управлінських рішень. А це в свою чергу зменшує часовий проміжок за який виконується петля управління підприємством [3].

На сьогоднішній день такий звіт для підприємства можна сформувати лише за допомогою розширення для платформи «1С:Підприємство», «ИНТАЛЄВ». Однак це рішення може виявитись надто важким для застосування в рамках діяльності підприємств малого бізнесу із вузько направленою діяльністю.

Архітектура розроблюваного сервісу

Архітектура визначає базову організацію розроблюваної системи. Визначити архітектуру означає визначити компоненти та їх зв’язки. Поведінка також визначається архітектурою, оскільки зв'язок між компонентами визначається бажаною поведінкою системи.

Архітектура програмного забезпечення включає в себе визначення структурованого рішення, що відповідає всім технічним і робочим вимогам. При цьому архітектура відображає тільки елементи, які оцінюються як значущі для системи. Набір елементів не є статичним, а може змінюватися в ході уточнення вимог та створення програми. Однак відносна стабільність є ознакою доброї архітектури.

Оскільки в розробці використовується Spring Framework, а саме Spring MVC, то додаток будується відповідно до логіки шаблону проектування MVC.

У загальному вигляді процес обробки запитів в Spring MVC наведено на Рис. 1. На етапі визначення архітектури та шаблону проектування, відповідно

до використовуваних технологій, було розроблено структура проекту зображену на Рис. 2.

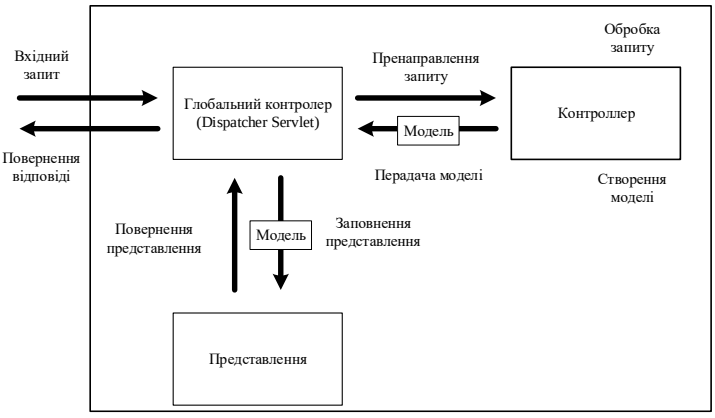

Рис. 1. Процес обробки запитів (сервер Apache Tomcat 8)

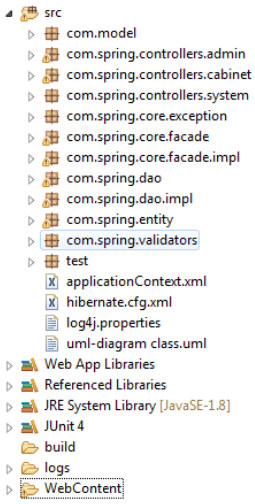

Рис. 2. Структура проекту на рівні пакетів

В даному випадку модель, представлена на Рис. 1 показує яким чином відбувається обробка запиту, що надходить від користувача.

Усі вхідні запити додатку передаються на обробку до глобального контролера (Dispatcher Servlet), який у свою чергу приймає рішення щодо перенаправлення запиту до потрібного контролера. Контролер виходячи з інформації, що міститься в запиті, формує певну модель, можливо виконує операції над моделлю. Далі контролер формує відповідь, передаючи модель глобальному контролеру. Він визначає представлення та заповнює його даними, що містяться в моделі. Після чого користувачу повертаються, візуалізуються дані, що були запитані.

На етапі визначення архітектури та шаблону проектування, відповідно до використовуваних технологій, розроблена наступна структура проекту (Рис. 2)

Пакет «com.spring» містить наступні пакети:

- «controllers», містить класи контролерів, що відповідають за обробку запитів користувача. Контролери визначено для кожного розділу додатку, а також для розділів частини адміністратора та для обробки дій, що пов’язані з авторизацією користувача.

- «core», включає в себе пакети «exception» та «facade». Пакет «exception» містить класи, що визначають можливі виняткові ситуації при роботі із базою даних. Ці класи є нащадками класу «Exception». У пакеті «facade» розташовані інтерфейси та породжені від них класи-сервіси Spring MVC фреймворку. Вони є додатковим рівнем для розмежування рівня доступу до даних та контролерами, що обробляють дії користувача.
- «dao», містить інтерфейси класи рівня доступу до даних.
- «entity», містить класи-сутності таблиць бази даних, для ORM Hibernate. Особливість цих класів полягає в тому, що вони повність відтворюють структуру таблиць бази даних, тим самим створюється об’єктна модель бази даних. Ці сутності використовуються класами пакету «dao». При цьому робота із класами виконується подібно до роботі із таблицями бази даних. А маніпуляція даними виконується за допомогою мови HQL. Мова HQL є об'єктно орієнтованою мовою запитів та за синтаксисом подібна до мови SQL. Різниця полягає в тому, що HQL працює не з таблицями бази даних, а із класами-сутностями. А вже Hibernate транслює HQL-запити у SQL-запити, які виконують необхідні дії із таблицями бази даних.
- «validators», містить класи, що відповідають за перевірку коректності обробки форм додатку. Ці класи є імплементацією інтерфейсу «Validator». Методи цих класів виконують перевірку об’єктив які повертаються формами, що заповнюються користувачем. Перевірка виконується шляхом визначення відповідності значень полів визначеним обмеженням.
- «model» включає в себе класи, що відповідають за специфічні операції додатку, а саме формування статистики.

Інтерфейс користувача

Для розробки інтерфейсу користувача використовується технологія JSP. Технологія JSP дозволяє створювати статичні і динамічні компоненти на сторінках додатка. JSP схожа з PHP, ASP, React's JSX, але використовується в

Java додатках. Для розгортання і запуску JSP необхідний веб-сервер сумісний з контейнером сервлетів. Статичні компоненти створюються з використанням HTML, XML, WML або SVG. У свою чергу динамічні компоненти створюються за допомогою JSP-елементів. Крім того, можливе використання бібліотеки JSP-тегів і EL. Expression Language (EL) використовується для включення Java коду в статичний вміст JSP-сторінок.

В якості мови розмітки статичної частини JSP-сторінок використовувався HTML. HTML є доступною, простою у вивченні і застосуванні мова розмітки. Вона інтерпретується браузером, а отриманий в результаті інтерпретації форматований текст відображається користувачеві. На даний момент останньою актуальною версією є HTML 5.

Для надання потрібного стилю сторінці використовуються каскадні таблиці стилів (CSS). Для спрощення і прискорення процесу розробки інтерфейсу користувача використовується CSS фреймворк Bootstrap. Даний фреймворк забезпечує два важливих аспекти: кросбраузерність і адаптивність. Також верстка сторінок з використанням Bootstrap дозволяє створити однаковий код, який в майбутньому полегшить процес внесення правок.

Останньою стабільною версією є Bootstrap 4.0.0.

Далі на Рис. 3 – 4 наведено приклади зовнішній вигляд розробленого інтерфейсу додатку, а саме інтерфейс головної сторінки та сторінки для роботи з клієнтами.

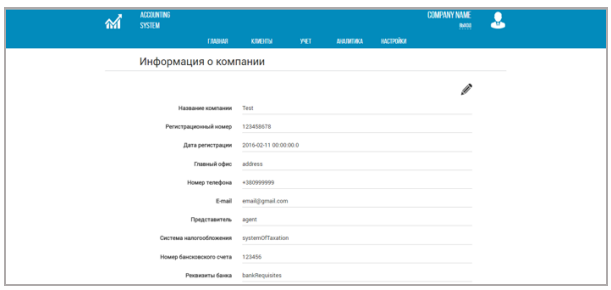

Рис. 3. Головна сторінка

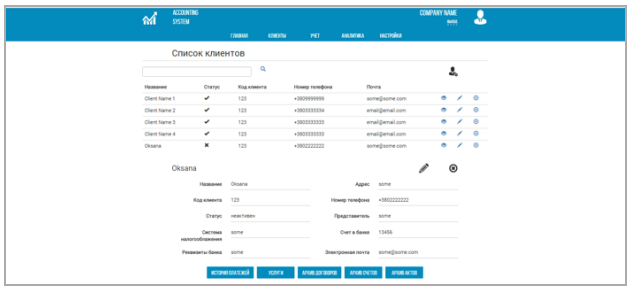

Рис. 4. Розділ роботи з клієнтами

1. База даних. Тестування рівня доступу до даних

Розробка бази даних полягала у створенні фізичної моделі з використанням конкретної СУБД. Специфіка СУБД може включати обмеження на імена об'єктів бази даних, обмеження підтримуваних типів даних та інші. Так само кожна СУБД визначає ряд специфічних рішень, пов'язаних зі способом зберігання даних (методи управління дискової пам'яті, поділ БД на файли, методи доступу до даних і ін.) і визначенням об'єктів БД (індексів, тригерів, збережених процедур і ін.)

В даному проекті використовується реляційна СУБД MySQL. MySQL підходить для малих і середніх рішень [4].

На основі логічної моделі бази даних були розроблені таблиці, що зберігають дані про: користувачів додатку; підприємство; клієнтів підприємства.

Процес тестування можна визначити як проведення випробувань програмного продукту, що мають на меті: продемонструвати відповідність програми специфікації; виявити ситуації у яких поведінка програми не відповідає специфікації та є небажаною; в залежності від мети тестування розрізняють наступні типи методів тестування; за об’єктом тестування, функціональне тестування, тестування продуктивності, юзабіліті тестування, тествання; за доступом до системи, тестування чорного ящика, тестування чорного ящика; за рівнем автоматизації, ручне автоматизоване та напів-автоматизоване; за рівнем ізольованості компонентів, модульне, інтеграційне та системне тестування; за періодом запуску тестів, альфа- та бета-тестування.

В якості метода тестування додатку обрано модульне тестування (юніт тестування). Даний вид тестування можна визначити як процес, що дозволяє перевірити правильність роботи окремих модулів вихідного коду програм.

Ідея полягає в тому, щоб визначати тест для кожної нетривіальної функції чи функції, що викликає складнощі при перевірці результатів її роботи. Такий метод дозволяє швидко перевірити правильність роботи функції та визначити, чи не привела чергова зміна до регресії, раніше протестованих частин програми. Формально модульне тестування можна визначити як процес перевірки

працездатності окремих частин програми за умови, що вони ізольовані від інших складових системи. Для більшості мов програмування високого рівня існують інструменти та фреймфорки, що дозволяють організувати процес юніт тестування. Зокрема для мови програмування Java існують наступні інструменти проведення модульного тестування: JUnit, TestNG, JavaTESK, Spock.

В якості інструменту для організації юніт тестування обрано JUnit 4 фреймфорк. Цей інструмент обрано тому, що він є найбільш використовуваним для тестування додатків.

У JUnit тест э звичайним Java об'єктом класу що визначає методи для тестування. Єдиною відмінністю є те, що методи цього класу повинні бути відмічені анотацією @Test. За необхідністю для методів також можна використовувати наступні анотації: @BeforeClass, метод відмічений даною анотацією буде виконуватись до виклику першого тестового методу; @AfterClass, означає що метод буде виконаний після визову останнього методу тестування; @Before, метод відмічений такою анотацією буде викликатися перед виконанням кожного методу тестування; @After, означає що метод буде викликано після виконання кожного методу тестування.

Для демонстрації процесу тестування обрано частину рівня доступу до даних, що відповідає за керування переліком послуг – ServiceDAO.

Розроблений клас із набором тестових випадків наведено на Лістингу 1.

Для запуску класу, що описує юніт тести використовується плагін для середовища розробки Eclipse JUnit. Для запуску класу тестування використовується контекстне меню та команда Debug As Junit test.

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = { "classpath:/applicationContext.xml" })
@Transactional
@TransactionConfiguration(defaultRollback = true)
public class ServiceDAOTest {
   @Autowired
   private ServiceDAO serviceDAO;
   @Test
   public void getServiceByIdTest() throws DatastoreException {
      Services newService = new Services();
      newService.setCost(54556);
```

```
        newService.setId_service(2);
        newService.setServiceName("Пакет ВИП");
        Services service = serviceDAO.getServiceById(2);
        assertEquals(newService, service);
    }
    @Test
    public void getServiceCreateSTest() throws CreateException {
        Services newService = new Services();
        newService.setCost(54556);
        newService.setServiceName("Пакет ВИП");
        try {
            Services service = serviceDAO.create(newService);
            assertEquals(newService, service);
        } catch (Exception e) {
            throw new CreateException(e);
        }
    }
    @Test
            public void getServiceUpdateTest() throws DatastoreException, UpdateException{
        try {
            Services newService = serviceDAO.getServiceById(2);
            newService.setCost(54556);
            Services service = serviceDAO.update(newService);
            assertEquals(newService, service);
        } catch (DatastoreException dex) {
            throw new DatastoreException(dex);
        } catch (UpdateException uex) {
            throw new DatastoreException(uex);
        }
    }
    @Test
    public void getServiceDeleteTest() throws DatastoreException {
        try {
            Services newService = serviceDAO.getServiceById(2);
            serviceDAO.delete(newService);
            Services service = serviceDAO.getServiceById(2);
            assertEquals(service, null);
        } catch (DatastoreException dex) {
            throw new DatastoreException(dex);
        } catch (RemoveException rex) {
            throw new DatastoreException(rex);
        }
    }
    @Test
    public void getServiceCountTest() throws DatastoreException {
        long count = (long) serviceDAO.getCount();
        assertEquals(count, 13);
    }
}
```

Лістинг 1 - Клас із набором тестових випадків для рівня доступу до даних, що відповідає за керування переліком послуг – ServiceDAO

Висновок

Відповідно до визначеної мети було розроблено веб-сервіс для керування документообігом, ведення обліку та підтримки клієнтської бази підприємства у сфері послуг, що дозволяє вести облік підприємства й при цьому містить набір лише тих функціональних можливостей, які щоденно необхідні користувачу.

Було проведено аналіз предметної області, порівняння існуючих рішень та визначення вимог до сервісу. Розглянуто і обрано найбільш прийнятні технології для розробки проекту.

В якості мови програмування було обрано Java. В якості фреймворку для побудови додатку використано Spring MVC. Розробка бази даних проводилась із використанням MySQL. Також для роботи з базою даних використано ORM Hibernate. Інтерфейс користувача побудований на основі технологій JSP, HTML, CSS (Bootstrap).

Особлива увага була приділена розробці структури веб-сервісу, інтерфейсу користувача та бази даних. Розроблена структура проекту дозволяє просто вносити зміни та додавати нові функції, вносячи мінімум змін до вже розроблених частин додатку.

Розроблений сервіс задовольняє всім вимогам, поставленим на етапі постановки задачі.

Даний проект може бути вдосконалений та доповнений новими функціональними можливостями. Оскільки функціональні можливості, існуючих на даний момент системи обліку підприємств, є надлишковими, та занадто важкі у використанні, то створений додаток є актуальним. Завдяки цьому є великий потенціал для розвитку. В якості елементів удосконалення можна відзначити розширення можливостей пошуку та фільтрації даних про клієнтів, послуги, прибутки та витрати, а також розширення функціональних можливостей для редагування документів.

Список літератури

1. Офіційний ресурс «1С» в Україні. Платформа 1С:Підприємство 8 [Електронний ресурс] – Режим доступу до ресурсу: http://1c.ua/ua/v8/

2. Taxer.ua – Податковий довідник і здача звітів до пенсійного фонду та податкової інспекції онлайн [Електронний ресурс] – Режим доступу до ресурсу: https://taxer.ua.

3. К. Дж. Дейт Введение в системы баз данных . Introduction to Database Systems. – 8-е изд. – М.: Вильямс, 2005. – 1328 с.

4. Ю.И. Ребрин «Управление качеством». Учебное пособие. Таганрог: Изд-во ТРТУ, 2004.

Authors:
Olena Odarushenko, Vladislav Tomko

DEVELOPMENT OF SERVICE MANAGEMENT WORKFLOW COMPANY IN THE SERVICE SECTOR

Abstract. In the article reviewed the methods development of service management workflow company. The purpose of the work is to develop a document management system for a company with a limited scope of functions, which will allow the user to conduct daily operations efficiently and conveniently. Development methods include studying the material, using object-oriented programming, working with databases MySQL.

Keywords: workflow control, OOP, SCDB MySQL.

Авторы:
Е. Б. Одарущенко, В. Ю. Томко

РАЗРАБОТКА СЕРВИСА УПРАВЛЕНИЯ ДОКУМЕНТООБОРОТОМ ОРГАНИЗАЦИИ В СФЕРЕ УСЛУГ

Аннотация. В статье рассмотрены методы разработки сервиса управления документооборотом предприятия. Цель работы состоит в разработке системы управления документооборотом предприятия с ограниченным набором функций, которые позволят эффективно и удобно проводить ежедневные операции пользователем. Методы разработки включают изучение материала, использование объектно-ориентированного программирования, работу с базами данных MySQL.

Ключевые слова: управление документооборотом, ООП, СУБД MySQL.