АНОТАЦІЯ

Магістерська робота на тему «Розробка web-додатку підтримки науковця для електронної бібліотеки з використанням Hibernate API та шифрування AES-256» Сороки Ігоря Олеговича. Тернопільський національний технічний університет імені Івана Пулюя, Факультет комп'ютерно-інформаційних систем і програмної інженерії, Кафедра Програмної інженерії, група СПмз – 61 // Тернопіль, 2019.

С. – , рис. – , табл. – , слайдів – , додат. – 4, бібліогр. – .

Метою дипломної роботи є розробка електронної онлайн бібліотеки, яку би могли використовувати навчальні заклади.

Методи та програмні засоби, які були використані в розробці системи: мова програмування JAVA та додаткові фреймворки включно з ORM Hibernate, середовище розробки IntelliJ IDEA та сервер JEE-додатків Glassfish.

Результатом роботи є готовий, згідно до поставлених вимог, веб-сайт.

Ключові слова: АВТОМАТИЗАЦІЯ, ЕНТЕРПРАЙЗ, БІБЛІОТЕКА, JAVA, ПРОГРАМНА СИСТЕМА, АЛГОРИТМИ.

ABSTRACT

Master's thesis on "Development of a web-based application for scientific's support for the electronic library using the Hibernate API and AES-256 encryption" by Soroka Ihor Olehovich. Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, SPmz - 61 // Ternopil, 2019.

S. -, fig. -, Table. -, slides -, add. - 4, bibliography. -.

The aim of the thesis is to develop an online library for use by educational institutions.

Methods and software used in the development of the system: JAVA programming language and additional frameworks including ORM Hibernate, the IntelliJ IDEA development environment, and the Glassfish JEE application server.

The result of the work is a ready-made website according to the requirements.

Keywords: AUTOMATION, ENTERPRISE, LIBRARY, JAVA, SOFTWARE, ALGORITHMS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення.

ЖЦ – життєвий цикл.

ПК – персональний комп’ютер.

JAVA – мова программування від компанії Sun.

JDK (java development kit) – безкоштовно розповсюджуваний компанією ORACLE комплект розробника, на мові JAVA.

JRE (java runtime environment) – мінімальна реалізація віртуальної машини, необхідної для виконання програм на мові JAVA.

БД – база даних.

СУБД – система для управління базами даних. Сукупність програмного забезпечення для забезпечення керуванням, та організацією баз даних.

Стек (stack, англ) – набір технологій, або мов програмування для забезпечення розробки програмного забезпечення.

JEE (java enterprise edition) – платформа JAVA, яка надає можливості та API для розробки корпоративних WEB-додатків, сервісів тощо.

API (application program interface) – інтерфейс програмування; готовий код для виконання певних функцій.

Фреймворк (framework, анг.) – набір бібліотек, які містять готовий функціонал для виконання певних задач.

ВСТУП

Один із найважливіших видів ресурсів сучасного суспільства, наряду з матеріальним та енергетичним є інформація, яка може бути представлена у виглядів архівів, документів, музеїв тощо. Актуальність магістерської роботи пов'язана з тим, що одне із важливих завдань яке завжди стояло перед людьми, суспільством та людством в цілому – це передача досвіду наступним поколінням, шляхом збереження інформації в часі та просторі. І. Гуттенберг зробив великий крок в переході від папірусу і пергаменту до паперу, які в майбутньому стали передумовами для активного розвитку книжкової справи. Більшість книг, особливо наукових, в той період були друкованими.

Перші книги книги церковнослов'янською мовою було надруковані в Кракові Швайпольтом Фіолем, та у Кракові Франциском скориною. Поштовх для книгодрукарства в Україні дав Іван Федоров. Власне І. Федоров відновив занедбане книгодрукарство, втікши з Москви від переслідувань редакційного духовенства. За допомогою меценатів Федоровим було створено друкарні у Львові, де роком пізніше надрукував свій знаменитий «Апостол» (книга про опис життя святих).

Друкарство активно розвивалось по всій Україні. Більше 20 друкарень нараховувалось вже в першій половині вісімнадцятого століття. Найбільшою з яких була друкарня в Києво-Печерській лаврі. Створення друкарень спонсорувалось меценатами, наприклад Запорожським Військом. Активну діяльність по розвитку типографії виконували братства.

Прогрес в розвитку книжкової справи в свою чергу дав потужний поштовх для літератури. Дана сфера культури сповна відбивала перехідний характер епохи. Найяскравіше нові тенденції в культурі відображала література, яка перекладалась з інших мов. Наприклад були перекладені та

опубліковані різноманітні наукові трактати і довідники, які в свою чергу допомагали розвитку інших сфер тогочасного суспільства.

Після виникнення книгодрукарства, форма яка стала основною в фіксування інформації, стали друковані видання, книги. Останні в свою чергу були зібрані та ретельно зберігались в бібліотеках. Бібліотеки забезпечували швидкий доступ до інформації та її збереження. За допомогою багатьох поколінь спеціалістів, дослідницької та практичної роботи, використання та збереження рукописних та друкованих документів було освоєно достатньо добре. З часом об'єми інформації, які зберігаються в традиційний формі, збільшувались і збільшувались. Працювати з ними ставало дедалі складніше: зберігання, розповсюдження, пошук, аудит, облік і т.д і т.п. Розвиток обчислювальної техніки дозволив зберігати і розповсюджувати інформацію в електронній формі, що грає революційну роль в історії аналогічну винаходу книгодрукування.

Електронна форма дозволяє на сьогодні зберігати книги найбільш надійно і компактно, розповсюджувати її набагато оперативніше і ширше, і крім того, надає можливості маніпулювання нею, яких не могло би бути при інших формах. У зв'язку з цим за останні роки у всьому світі інтенсивно збільшується кількість електронних публікацій. Значна кількість різноманітних документів вже зараз існують в електронній формі.

В сучасній історії Україні є дві події, які власне дуже вплинули на процеси інформатизації є прийняття Указу Президента України, Леоніда Даниловича Кучми, № 663/97 від 17 червня 1997 року «Про невідкладні заходи впровадження системи здійснення державної інформаційної політики та удосконалення державного регулювання інформаційних відносин» та Закону України № 74/98-ВР від 04.02.1998 «Про національну програму інформатизації».

Дана програма була призвана для розв'язання проблемних питань, які виникли в процесі використання інформаційних ресурсів, та щоб зменшити розрив між зростаючими потребами в інформації та рівнем її використання;

досягнення Україною рівня інформатизації інших розвинутих країн світу. Включала вона в себе комплекс взаємодоповнюючих завдань, що були спрямовані на реалізацію державної політики в створенні державної мережі інформаційного забезпечення різного роду галузей виробничої сфери, та суспільства.

Однією із основних запорук розвитку будь-якого суспільства, та українського безпосередньо – є належний доступ кожному члену суспільства до інформації, які є йому необхідна, доступу до культурних надбань світу. Інвестиція в розвиток інформатизації суспільства – це також розвиток освіти, науки, національної свідомості, національної безпеки, зокрема бібліотек для університетів, тощо.

Останнім часом в суспільстві виникло підґрунтя для нового розуміння значення бібліотеки. У зв’язку з переходом до інформаційного суспільства, бібліотеки розглядаються як суб’єкти інформаційних процесів та інформаційних ресурсів. Це пов’язано з тим, що до традиційних ресурсів бібліотек додались аудіоматеріали, відеоматеріали та електронні ресурси.

Всесвітня мережа Інтернет поклала більшість функцій бібліотек на себе; стала джерелом інформації яке стало в пріоритеті в суспільстві. Невпинний процес модернізації суспільства спричинив більшу доступність онлайн бібліотек ніж бібліотек в їхньому класичному розумінні.

1 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

1.1 Аналіз вимог до програмної системи

1.1.1 Аналіз предметної області

У світі цифрових технологій розрізняють оцифровану групу документів, сучасну та електронну. Всі вони походять від різноманітних безпаперових технологій.

Першій групі (оцифрованим документам) належать документи, які з'являються в результаті конвертування друкованих документів (книг) в цифрову форму. Даний спосіб переведення документів з однієї форми в іншу називають оцифровкою. Оцифровка дає перетворення із традиційної форми друкованих матеріалів в цифрову, іншими словами – комп'ютерну. Вона дозволяє перевести ілюстраційні, графічні документи, фотографії в електронну форму – в файли.

Іншою (другою) групою електронних документів, якими користуються нині, є документи існуючі тільки в цифровому вигляді. Дані документи були утворені за допомогою інформаційних технологій, таких як Інтернет і мультимедіа. Сучасні електронні документи можуть бути представлені у вигляді вербальної (словесної) інформації, відеоматеріалів, звуку та анімації, графіки. Власне такі файли займають більше місця на жорстких дисках.

Третьою групою продуктів безпаперових документів нині, яка набрала значну популярність серед користувачів – є бази даних. Також прикладом безпаперових технологій може слугувати цифрове фотографування, за допомогою якого можна отримати різноманітні текстові, графічні і ілюстраційні документи. Значної популярності також набули повідомлення відправленні через електронну пошту. В комерційних цілях останні почали використовувати більше ніж 30 років тому.

Розглянемо на першій групі документів – оцифрованих документах (цифрових).

В даній роботі я спробую показати найвідоміші нині види цифрових документів в різних контекстах.

В контексті доступу до електронного змісту, цифрові документи мають:

- вільний доступ;
- обмежений доступ (платні, файли компанії, особисті);

По відношенню власника ресурсу до авторства:

- власного авторства;
- авторства третьої сторони.

По розміщенню:

- автономні сайти;
- розділи сайтів.

По способу передачі інформації:

- текст;
- відео;
- аудіо;
- графіка;
- мультимедіа.

З точки зору фізичного носія інформації цифрові документи зберігаються на:

- накопичувачах на гнучких магнітних дисках (FDD), вони ж флоппі-диски, вони ж дискети;
- накопичувачі на жорстких магнітних дисках (HDD), вони ж вінчестери;
- накопичувачі на оптичних компакт-дисках: CD-ROM (Compact Disk ROM), DVD-ROM.

На типології оригіналу цифрового документу зупинятися немає необхідності, так як це окрема і обширна тема.

Сьогодні навряд чи можна конкретизувати термін стосовно цифрових документів. Багато хто з професіоналів в своїх публікаціях визначення електронного документу дають як «документ, який містить інформацію у вигляді електронних даних». Якщо мається на увазі друкований документ, або будь-який твір чи робота яка має фізичну копію, то можна розглянути визначення «електронна копія».

Електронна копія – комп’ютерний аналог друкованого видання:

- текстового, який містить переважно текстову інформацію, представлену в формі, яка допускає посимвольну обробку;

- текстового, який містить переважно текстову інформацію, представлену в формі, яка допускає посимвольну обробку;
- образотворчого, який містить переважно електронні зразки об’єктів, які розглядаються як цілі графічні сутності, представлені в формі, яка допускає перегляд і друковане відтворення, але не допускає посимвольну обробку;
- мультимедійного, який містить взаємопов’язану інформацію різного роду, яка є забезпеченою відповідними програмними засобами.

Кожна електронна копія документу повинна мати свій бібліографічний опис, створений на базі бібліографічної записки оригіналу. В бібліографічний опис електронної копії документу додається: клас матеріалу – електронний ресурс, відомості про відповідальність – обличчя і організації, які приймають участь у створенні електронного ресурсу, область вигляду і об’єму ресурсу – перелік основних виглядів електронних даних – текстові, звукові, графічні, числові, шрифтові і т.п.

В межах планового оцифрування можливо виготовляти щонайменше 3 типи цифрових копії, які розрізняються за об’ємом інформації в залежності від призначення і особливостей використання:

1) Майстер-копія. Ця копія містить максимально можливу кількість інформації. Вона використовується для відновлення об’єкту у випадку його втрати, для інших поліграфічних цілей, для деяких видів досліджень і як основа для виготовлення інших типів цифрових копій. До файлів, яка містять архівні копії, не застосовуються алгоритми зжимання з утратами. Мається на увазі, що дані копії будуть розміщені на носіях з довгостроковим часом існування. В цілях охорони авторських прав доступ до копій даного типу обмежений а користувачам вони надаються тільки в особливих випадках.

2) Користувацька копія високої або середньої роздільної здатності. Ця копія виготовляється із майстер-копії для цінних і рідких об’єктів, або шляхом прямої оцифровки для тиражних видань. Вона призначена для представлення користувачам в базах даних і електронних бібліотеках. Допускається застосування алгоритмів стискування з утратою якості, але при

цьому вона повинна зберігати читабельність тексту і можливість розрізнення деталей графічних елементів. Об’єм файлу повинен відповідати пропускній здатності внутрішньої комп’ютерної мережі. Роздільна здатність цієї копії повинна бути достатньо низькою, щоб не дозволити поліграфічне копіювання об’єкту.

3) Службова копія низької роздільної здатності. Ця копія виготовляється із майстер-копії або користувацької копії. Вона призначена для публікування на web-сайті і для попереднього перегляду при пошуці по локальній ком’ютерній мережі, у тому числі для представлення користувачам в банах даних і електронних бібілотеках.

Можна застосовувати різні алгоритми стискування даних: алгоритм стискування з утратою роздільної здатності графічного матеріалу, впровадження в документи графічних даних (водяних знаків, метаданих тощо). Об’єм файлу не повинен перевищувати пропускну здатність інтернет-шлюзу.

Оцифровка книг – це процес конвертації друкованих видань в електронний (комп’ютерний) вигляд. Електронні копії книг можуть утворювати електронні бібліотека і розповсюджуватись в глобальній мережі Інтернет. Цифрова форма книг дозволяє достатньо легко їх розповсюджувати, відтворювати та читати за допомогою вашого електронного пристрою (планшет, смартфон, комп’ютер тощо). Як показують дослідження, найпопулярнішими форматами нині є pdf, djvu, jpg та tiff.

Повне оцифрування документу пояснюється бажанням зберегти і забезпечити відкритий доступ користувачів до найцінніших друкованих (або навіть рукописних) документів.

Якщо ми розглянемо на методи по формуванню фондів електронних видавництв методом оцифровки, то ми можемо виокремити декілька модельних форм, перевага яких залежить від типу електронної бібліотеки, системи її фондів і передбачуваного контингенту користувачів.

Кожна із моделей дозволяє організовувати електронний фонд як відкритого типу (доступний через Інтернет), так і призначений для роботи тільки в локальній мережі або на одному комп’ютері.

Модель 1. Постійна оцифровка бібліотечних фондів (організацій). Метою цього підходу є максимізація доступності бібліотечних фондів для необмеженої кількості користувачів та вирішення деяких проблем із їх зберіганням (зокрема, шляхом випуску найцінніших та рідкісних видавців в електронній формі, зменшення навантаження на оригінал). Зрозуміло, що бібліотека створює електронну копію свого інвентаря шляхом сканування. Перш за все важливо врахувати доцільність такої масштабної роботи та пов'язані з цим фінансові, часові та юридичні проблеми. У методі безперервного оцифрування ігноруються, по суті, такі фактори:

- нерівність (наукова, культурна, історична, інформативна) бібліотечних документів, що має бібліотечний фонд, що так чи інакше ставить проблему вибору або встановлення пріоритетів для оцифрування друкованих видавців;

- невизначеність щодо контингенту потенційних користувачів електронних фондів (якщо вони мають доступ до мережі);

- дублювання подібної інформації у різних видавцях (наскільки це відображено в бібліографічній інформації), що спричиняє електронне збирання документів генерувати більше інформаційного шуму.

Звичайно, найкращим рішенням для організації, яка вибирає цю модель, було б взаємодія з іншими власниками, оскільки оцифровувачі зазвичай повинні публікуватися у фонді без бібліотек, і очевидно, що цей процес потрібно впорядкувати по всій країні. Дана модель в найбільшій мірі підходить бібліотекам, фонд яких представляє собою цілісну колекцію з визначеним цільовим призначенням і чіткими принципами формування, яку має сенс зробити загальним надбанням. Тоді стає найбільш визначеним і контингент користувачів, що полегшує організацію матеріалу і створення необхідного набору пошукових коштів.

Модель 2. Перетворення активно використовуваної частини запасів документів в електронну форму. На перший погляд, це найкращий варіант для ЕБ, оскільки існує чіткий критерій вибору видавців для конвертації в цифровий формат і, звичайно, для надання громадськості самих сучасних документів. Крім того, можна доповнити електронну бібліотеку матеріалами, отриманими в результаті надання послуг електронної доставки документів.

При реалізації цієї моделі основну увагу необхідно звернути на наступні аспекти:

- дотримання авторських і майнових прав власників друкованих оригіналів;
- вироблення критеріїв активності використання фондів, окремих видавництв і їхніх частин (статей із журналів, глав із книг, монографій, звітів і т.п.);
- обліку коливань активності запиту для різних галузей знань і типів документів (відома закономірність, згідно якої активний запит на наукові видання протягом 2-3 років після публікації в деяких науках заміняється на повний пасив в майбутньому, оскільки оргінальні ідеї отримують адаптації і розвиток в більш пізніх працях). Не виключено, що ЕБ може з часом отримати більший масив невикористаних електронних копій, особливо в області природничих наук .

Модель в найбільшій ступені підходить бібліотекам вищих учбових закладів і науково-дослідницьких інститутів, оскільки в них є можливість переводити в цифровий формат примірники, які користуються найбільшим запитом, методичні розробки, труди своїх працівників, матеріали конференцій тощо.

Модель 3. Формування колекцій за видами документів та окремими темами. На мою думку, ця модель є одним з найбільш прийнятних варіантів для більшості ЕБ. Основна проблема полягає у визначенні тих електронних документів, які мають активну та наполегливу вимогу, яку можна очікувати з часом. З іншого боку, завдяки сучасним реаліям можна створити електронні

колекції за спеціальним замовленням або з огляду на підвищену актуальність певної теми. В світовій практиці є приклади, коли шкільні, спеціальні, публічні і університетські бібліотеки провінцій створюють масиви електронних документів по якомусь кругу проблем (історія, новини, спорт, культура, література, і т.д.).

Для деяких типів документів доцільно використовувати ресурси, які вже існують.

Одним із типів колекцій, для яких особливо важлива проблема широкої доступності в поєднанні з підвищеним збереженням оригіналу є колекція рідких, стародрукованих, особливо важливих видавництв, рукописів, документів з погасаючим текстом. Подібна модель визначена як базова для державних бібліотек.

Модель 4. Формування комплексної культурно-освітньої програми. Річ йде про створення цілісної мультимедійної бази даних конкретної тематики, включаючи, крім друкованих видавництв, архівні матеріали, зображення речових пам'ятників, звуковий ряд і т.д. Прикладами таких проектів являються «Пам'ять Америки», «Культура країн Середземномор'я», SCRAN («Шотландська мережа культурної спадщини») та інші.

Реалізації цієї моделі потребує:

- визначення базової одиниці інформації, тобто прийняття рішення про те, що буде лежати в основі колекції: текст твору чи конкретне видання;
- розробка програмного забезпечення, яке дозволятиме вести пошук по різного роду критеріям;
- наявність вихідного сегменту (колекції зображень, збірник текстів і т.д.), найбільше підготовленого для формування на його основі (шляхом доповнення іншими видами документів) кінцевого тематичного блоку;
- визначення статусу цієї бібліотеки (довідкова, публічна або науково-дослідницька), в залежності від того, буде вона розрахована на широкі круги користувачів чи тільки на спеціалістів.

Дана модель найбільш успішно може бути реалізована при наявності партнера між організаціями різних типів, які приймають участь в комплексних культурно-образотворчих програмах (регіональних, тематичних, образотворчих). Безумовно, реалізація даного модельного рішення буде великим вкладом в розвиток вітчизняної культури. При цьому варто прийняти до уваги наявність світової тенденції в розробці саме таких проектів і того, що деякі великі бібліотеки вже приступили до часткової реалізації подібних програм.

В ситуації швидкого поширення інформаційно-комунікаційних технологій виникають додаткові завдання пов'язані зі створенням електронних бібліотек, доступних для більшості населення. Електронні бібліотеки сьогодні навіть не напрямок, це – ідеологія. Електронні бібліотеки стають невід'ємною частиною діяльності практично в будь-якій області, з необхідністю мати, розвивати і використовувати електронну бібліотеку сьогодні стикаються практично всі. Як раніше всі прагнули отримати комп'ютери, так зараз, коли вже відбулося певне насичення комп'ютерами, кажуть в основному про дві речі – Інтернет і електронні бібліотеки. Причому все більше і більше про електронну бібліотеку в цьому сенсі найбільше не пощастило звичайним бібліотекам: з одного боку, саме бібліотеки є однією з головних рушійних сил розвитку електронної бібліотеки, а з іншого саме слово бібліотека в назві електронна плутає багатьох і часто нівелює це поняття , зводячи його до задачі оцифровки якихось фрагментів або всього фонду бібліотеки. Основна відмінність традиційної публікації (інформації "на твердому носії") від електронної полягає в тому, що користувач, звертаючись до сервісів електронної бібліотеки за необхідною інформацією в результаті отримує можливість безпосередньої роботи з самими документами. Головне в електронній бібліотеці – це система каталогів і можливість зручного і всебічного пошуку. І тоді будь-яка інформація буде знайдена і використана. Сьогоднішні можливості програмування відкривають воістину безмежні можливості для організації електронних бібліотек організацій. Електронні

бібліотеки та бази даних стають невід'ємною частиною діяльності практично в будь-якій області: науки, культури, освіти, інститутів влади. Практично будь-яка організація, має свою сторінку в Інтернеті, а часто і аматорський сайт.

Технології 21 століття дозволяють черпати різноманітну інформацію, не виходячи з дому. Електронна бібліотека дає можливість прочитати будь-яку художню книгу, знайти необхідну документацію, прочитати спецлітературу і т.д. Не можна не помітити, що подібна система має масу переваг.

Так, одна з переваг – це отримання необхідної інформації, незалежно від вашого місця знаходження. Читати необхідну літературу або знайти потрібний документ можна в робочому офісі, в інтернет-кафе або будинку (всюди, де є комп'ютер, підключений до інтернету).

При перекладі інформації в цифрову форму, її використання є більш повним і практично безмежним. Завдяки електронній бібліотеці, стає можливим доступ читачів до продукції, яка є в обмеженій кількості або в єдиному екземплярі, і не може бути придбана більшістю зі звичайних бібліотек (рідкісні книги, зарубіжні видання, фотоальбоми і т.д.). Використовувати електронну бібліотеку можуть люди будь-якого віку, професій і уподобань. Тут знайдеться і класична література, і твори сучасних авторів, і технічна документація, і навіть казки та оповідання для самих маленьких.

Полегшується реалізація нових форм бібліотечного та інформаційного обслуговування користувачів, в тому числі – обслуговування інвалідів по зору та інвалідів у зв'язку з хворобами опорно-рухового апарату.

Робота з цифровими електронними документами може вийти за рамки простого читання тексту або перегляду. Фрагменти вихідних даних можна використовувати в роботі, об'єднуючи, додаючи і редагуючи матеріали.

Електронні бібліотеки та бази даних стають невід'ємною частиною діяльності практично в будь-якій області: науки, культури, освіти, інститутів влади.

Незалежно від того, чи є ЕБ локальна або вона доступна з Інтернету (з різними умовами доступу), її створення має бути спрямоване на досягнення основної мети, яка бачиться в задоволенні інформаційних потреб. Використовувані технології і методики повинні відповідати специфіці позначених інформаційних потреб, раціональної організації масиву електронних документів, сформованого за обраними критеріями відбору.

Але на жаль не всі онлайн бібліотеки в мережі відповідають даним вимогам. І це нормально, так як розробники електронних бібліотек в мережі, частіше за все, як показала практика, робили акцент або на дуже спеціалізованій тематиці, або на певному типові літератури: наукова література, підручники, наукові видання, наукові періодичні видання, тощо. В теперішній ситуації, що виникла, більшість електронних бібліотек старається збільшувати своє «меню» контенту науковою літературою, та такою, що найбільше потребується в тій чи інший тематиці.

Оцінюючи існуючи в мережі електронні-бібліотеки я враховував декілька основних критеріїв:

– ліцензійно чистий контент (формування контенту в суворій відповідності до законодавства України: висновок з кожним правовласником ліцензійного договору, що підтверджує згоду власника авторських прав на розміщення в електронній бібліотеці належних йому наукових або навчальних матеріалів);

– актуальність контенту (поповнення електронними виданнями останніх років, обсяг, повнота і різноманіття колекцій, відповідність науковим і навчальним цілям вузу);

– зручний пошук, інтерфейс;

– відповідність фінансовим можливостям бібліотеки в умовах скорочення бюджетних коштів на комплектування фондів і потребам бібліотеки (колективний доступ і ін.).

В результаті аналізу безлічі інформаційних мережевих ресурсів мною були виділені три види електронних бібліотек, що представляють свої ресурси

в повнотекстовому вигляді з можливістю або безкоштовного користування, або підписки на них. Як вже говорилося вище, «універсальних» ЕБ, що надають і періодичні видання, і навчальну, наукову і методичну літературу якої вкрай мало. Я вибрав 3 таких універсальних бібліотеки і оцінив їхній зміст з точки зору відповідності потребам. Також в аналізі є характеристики 3 бібліотек періодичних видань. Електронних бібліотек, які формують свій контент з навчальної, наукової та методичної літератури, – 14, і це при тому, що переважна більшість бібліотек художньої літератури просто не включено в аналіз (як приклад таких бібліотек я охарактеризує три електронних бібліотеки), що не включено безліч бібліотек довідкової, аналітичної і оглядової спрямованості.

1.1.1.1 Національна бібліотека імені Вернадського

Адреса: http://www.nbuv.gov.ua/

Опис ресурсів: обсяг фондів - близько 15 млн. Одиниць зберігання. Книги, надані авторами; статті з періодичних видань, отримані від редакцій журналів; автореферати дисертацій, захищених в Україні в 1998 і наступних роках; публікації по бібліотечної справи т інформаційної діяльності; класиків української літератури (І. Котляревського, Т. Шевченка т ін.); електронної копії раритетів Бібліотеки. Щорічно до фондів надходять 160-180 тис. Документів (книг, журналів, газет і т.д.).

Умови доступу: Безкоштовні доступ і скачування більшості даних.

Оцінка: Зручний розширений пошук, хороший інтерфейс (рис. 1.1). Швидкий перегляд. Дуже багато методичної літератури.

Рисунок 1.1 – Головна сторінка сайту

1.1.1.2 Європейська електронна бібліотека «Europeana»

Адреса: http://www.europeana.eu/portal/

Опис ресурсів: містить оцифровані об'єкти культурної спадщини Європи: книги, картини, фотографії, аудіозаписи. Матеріали поширюються з ліцензією для некомерційного використання, або, що знаходяться в суспільному надбанні, на листопад 2008 містила більше 2-х млн об'єктів, до 2019 р загальна кількість планується довелося до 10-13 млн.

Умови доступу: безкоштовний.

Оцінка: найкраща європейська цифрова бібліотека (рис. 1.2), мета якої - забезпечити доступ до відсканованих сторінок книг, що відображають різні аспекти європейської культури. Зараз доступна інформація на французькому, німецькою та англійською мовами. Надалі передбачається представити джерела на всіх європейських мовах. Станом на 2008 рік, найбільший внесок у створення бібліотеки внесла Франція (50% - оцифрованих даних), 10% - Великобританія, 1,4% - Іспанія, 1% - Німеччина.

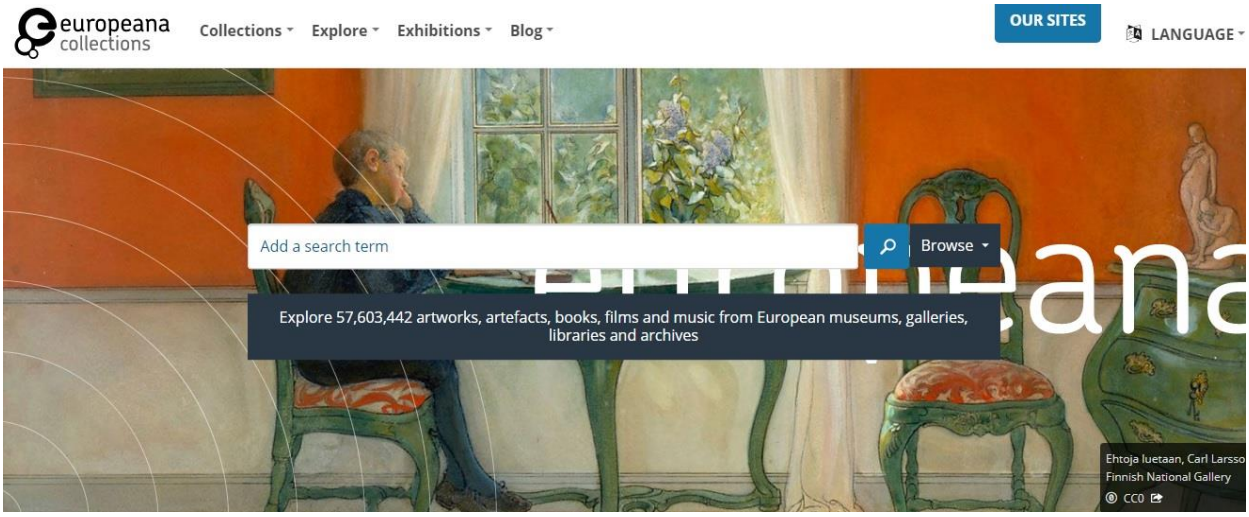

Рисунок 1.2 – Головна сторінка сайту

1.1.1.3 Бібліотека української літератури

Адреса: http://www.ukrlib.com.ua/

Опис ресурсів: найбільша в Інтернеті електронна бібліотека української літератури, яка крім українських книжок пропонує також літературну енциклопедію, біографії, твори, учнівські реферати, стислі перекази змісту творів з української та зарубіжної літератури (рис. 1.3).

Умови доступу: вільний доступ до скачування текстів літературних творів (фольклор, класична і сучасна література), творів і ін.

Література надана в більшій кількості, дворівневий пошук. У 2010-му році "Бібліотека української літератури" отримала Знак Якості!

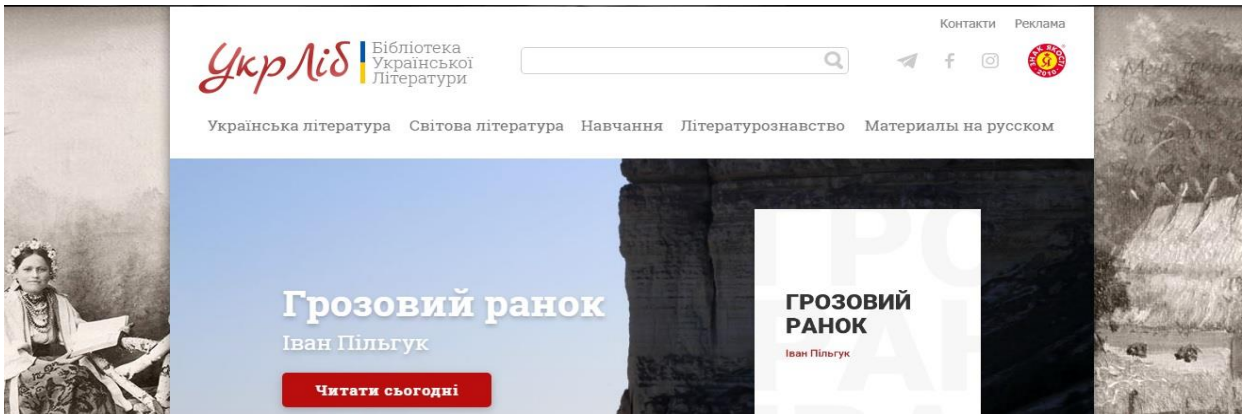

Рисунок 1.3 – Головна сторінка сайту

1.1.2 Постановка задачі

Виконавши аналіз предметної області, треба встановити основну мету та цілі для виконання завдання.

Отже ціллю є розробка WEB-сайту для електронної бібліотеки навчального закладу. Її контентом буде переважно науково-технічна література. В кінцевій роботі про книгу як об'єкт повинні бути відомі:

- *назва книги (документу)* – власна офіційна назва документу, за допомогою якої можна ідентифікувати оригінальний виріб;
- *ім'я автора або авторів* – сукупність праць, які створені в результаті досліджень, теоретичних узагальнень, зроблених в рамках наукового проекту, та які покликані для інформування фахівців про останні досягнення в тій чи іншій області, як правило досягаються за участю декількох авторів;
- *рік видання* – рік коли була видана певна книга;
- *видавництво* – назва видавництва, яке друкувало певну книгу;
- *кількість сторінок* – загальна кількість сторінок книги;
- *опис* – короткий опис про що саме книга;
- *ISBN* – універсальний ідентифікаційний номер книги, що присвоюється книзі або брошурі з метою їх ідентифікації. ISBN призначений для ідентифікації окремих книг або різних видань та є унікальним для кожного видання книги.

Web-сайт повинен мати наступні особливості:

- гнучкістю, зручністю адміністрування;
- для користувачів повинна бути доступність прочитати та роздрукувати будь-яку сторінку книги або книгу повністю;
- повинен бути реалізований розділ для зворотного зв'язку, в якій користувач зміг би задавати питання, які його цікавлять і отримувати на них відповіді в найкоротші терміни;
- web-сайт повинен мати можливість для часткової зміни контенту, і можливістю розмежувати доступ до певного жанру;

- на сайті повинен бути доступний пошук по назві певної книги;
- на сайті повинен бути доступний пошук по першій літері в назві книги;
- на сайті повинен бути присутня(ній) категорія/жанр книги, до якого вона належить;
- на сайті доступний відбір по певній категорії книги.

1.1.3 Пошук актантів та варіантів використання

Основними суб'єктами, що взаємодіють із системою – є адміністратор та користувач. Адміністратор і користувач – це абстрактні поняття конкретних кінцевих користувачів, що певним чином взаємодіють із системою і являють собою контекст, що існує поза нею. В даній системі дійові особи не являються окремими сутностями. І адміністратор і користувач є екземплярами класу User. Надання ролей для можливості використовувати певні можливості сайту, та розмежування для користування цими самими можливостями, відбувається в базі даних для кожного певного ідентифікатора користувача.

За замовчуванням додаток (сайт) містить кореневого користувача root, для якого доступна сторінка реєстрації користувачів (за адресою \signup.xhtml). На сторінці реєстрації користувачів, можна ввести ім'я користувача, та надати йому одну із доступних груп в системі: users або administrators. Якщо користувач не залогується в системі, то на доступній йому сторінці він зможе зареєструвати користувача по-замовчуванню з групою users, без можливості вибору.

Користувачі обох груп (ролей) можуть переглядати доступні книги, скачувати їх та роздруковувати прямо у вікні веб-браузера. На сайті доступний пошук по назві книги, або по автору книги. В рядку введення назви доступний випадаючий список, який дає змогу здійснити пошук по імені автора, або імені (назві) книги. Тобто коли користувач введе певну послідовність літер, то пошук введеної комбінації чи то пак введеного слова, буде здійснюватися або в імені книг, або в імені авторів.

Доступний список категорій (тобто математика, фізіка, хімія тощо) літератури, який є присутній на певний момент в базі даних. Вибірку книг можна здійснювати також по ньому.

Для адміністраторів системи, тобто для сутностей класу User, ідентифікатор якого був присвоєний до групи admins, доступне також видалення, редагування та введення нових категорій для літератури. Видалення, редагування існуючих книг, та додавання нових. Розмежування можливостей у користувача та адміністратора наведено у таблиці 1.1.

	Admins	Users
Редагування	+	-
Видалення	+	-
Перегляд	+	+
Завантаження	+	+
Друк	+	+
Реєстрація нових користувачів	+	+
Реєстрація нових користувачів з вибором групи	+	-
Обмеження по перегляду користувачів певних категорій книг	+	-

Таблиця 1.1 – Права групи admins та групи users

1.1.4 Опис ключових варіантів використання

Більшість сучасних веб-браузерів мають вбудовану підтримку формату PDF. Тобто файли з даним форматом можна прочитати безпосередньо у вікні браузера. Дана особливість економить час та сили на розробці власного програмного забезпечення, яке би дозволяло працювати з файлами даного формату. Формат документів PDF представляє собою універсальний файловий формат, який дозволяє зберегти шрифти, зображення і сам макет початкового документу незалежно від того, на якій із множини платформ і в якій із множини додатків даний документ було створено. Формат Adobe PDF

вважається визнаним загальносвітовим стандартом в області тиражування і обміну надійно захищених електронних документів і бланків. Файли Adobe PDF мають невеликий розмір, і вони є самодостатніми. Враховуючи все вищесказане, в якості контенту книги був обраний PDF-формат. Також вибір даного формату впав через його розповсюдженість, компактність та зручність у використанні.

Розроблений мною додаток повинен надавати можливість редагування сутностей класу Book, тобто книг. В режимі редагування адміністратор може змінити опис книги, її назву, кількість сторінок, жанр до якого вона належить, видавництво, автора, ISBN, рік видавництва, її обкладинку та контент. Якщо адміністратор під час редагування певної книги, захоче змінити її контент і обере файл з іншим форматом, то система сповістить його про невірний формат у спливаючому вікні. Обкладинка обмежена форматами png та jpeg. Все вищеописане справедливе і для додавання нових книг до бази даних електронної бібліотеки.

Резюмуючи все вище сказане, варіанти використання можна зобразити схематично на рисунку 1.4.

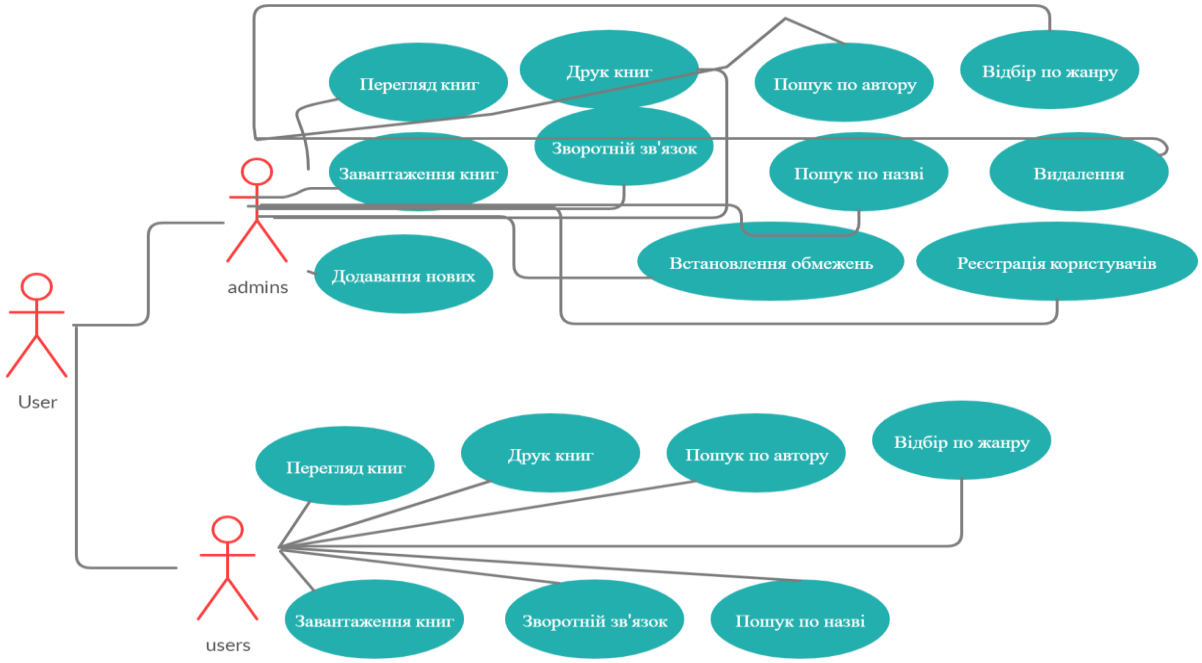

Рисунок 1.4 – Варіанти використання системи

1.2 Проектування програмної системи

1.2.1 Вибір процесу розробки

Процес розробки програмного забезпечення – це сукупність дій, які виконують розробники програмного забезпечення під час всього життєвого циклу розробки. Існує декілька моделей такого процесу, кожна з яких описує свій підхід, у вигляді завдань та/або діяльності, які мають місце в ході процесу.

1.2.1.1 Водоспадна модель

Водоспадна модель життєвого циклу програмного забезпечення має на увазі те, що різні етапи розробки програмного забезпечення , включно аналіз вимог, проектування, кодування та тестування окремих модулів програми, тестування білдів і інтегрування мають виконуватись послідовно. Кожен етап має чітку розмежованість з іншими, та передає напрацьовані документи в якості вхідних даних для наступних етапів. Таким чином кожен вид діяльності виконується на якійсь одній фазі життєвого циклу ПЗ. Рух назад є неможливий.

Етапи у відповідності до каскадної моделі:

- аналіз – на даному етапі вивчається і визначається задача, яку повинна виконувати програма. Результатом діяльності даного етапу є сукупність вимог до ПЗ;
- проектування – список вимог на попередньому етапі перетворюються в набір рішень. На даному етапі формується документ, який є підставою для конкретних рішень (вибір технології розробки, архітектури тощо) по розробці програмного забезпечення. Основний результат роботи на даному етапі – це текст природною мовою, модель ПЗ, алгоритми, таблиці, математичні формули тощо;
- реалізація – Після того, коли етапи аналізу і проектування будуть завершені, наступає етап реалізації, на якому створюються та тестуються

програмні модулі, які були визначені на проектуванні. Головними результатами цього етапу являються модулі вихідного коду і автономні тести модулів. Після реалізації переходять до тестування системи, а потім до її здачі в експлуатацію.

Недоліки водоспадного підходу:

- накопиченні помилок, які були допущені на різних стадіях проекту;
- невиправдане збільшення термінів виконання розробки;
- невиправдане збільшення бюджету;
- всі ключові рішення приймаються тоді, коли у аналітиків і розробників немає повного розуміння системи;
- метод водоспаду не дає можливості швидкої адаптації до змін, особливо на пізніх етапах життєвого циклу ПЗ.

Ця модель виходить з того, що всі помилки будуть зосереджені на реалізації, а тому їх усунення відбувається рівномірно під час тестування компонентів і системи. Таким чином, водоспадна модель для великих проектів мало реалістична і може бути ефективно використана тільки для створення невеликих систем.

1.2.1.2 Ітеративна модель

На жаль не всі моделі життєвого циклу ПЗ є послідовними. В світі розробки існують також ітеративні (або інкрементні як їх ще називають) моделі, в яких використовується підхід який кардинально відрізняється. В цій моделі, одній довгій послідовності дій, приходить на заміну декілька послідовностей, які були розбиті на окремі міні-цикли. Характерно, що кожен з них складається з тих самих базових стадій моделі життєвого циклу. Ці міні-цикли і називаються ітераціями. В кожній із ітерацій розробляється окремний компонент системи, після чого цей компонент додається до вже раніше розробленого функціоналу.

Ітеративна модель не надає повного об’єму вимог для початку робіт над продуктом. Розробка програми може початись з вимог до частини

функціоналу, які можуть в свою чергу доповнюватись і змінюватись. Процес повторяється доти, доки не забезпечить створення нової версії продукту.

Якщо дещо спростити ітеративну модель, то її можна представити у вигляді чотирьох стадій, які повторяються в кожній із ітерацій:

- визначення і аналіз вимог;
- дизайн і проектування згідно вимогам. Причому дизайн може або бути розроблений конкретно для даного функціоналу, або розроблюватись під вже існуючий;
- розробка і тестування – кодування, інтеграція і тестування нового компоненту;
- фаза рев'ю коду – оцінка, перегляд поточних вимог і пропозицій як їх можна доповнити.

Після виконання кожної ітерації приймається рішення чи будуть використані її результати для доповнення існуючого функціоналу в якості вхідної точки для початку наступної ітерації. В підсумку досягається точка, в якій всі вимоги були втілені в продукт – відбувається реліз.

Основні стадії процесу розробки в ітеративній моделі фактично повторюють модель водоспаду. В кожній ітерації створюється програмне забезпечення, яке вимагає тестування на всіх рівнях.

Плюси:

- швидкий реліз вже працюючого додатку;
- гнучкість – готовність до змін вимог на будь-якому етапі розробки;
- кожна ітерація – маленький етап, для якого тестування і аналіз ризиків забезпечити простіше, ніж для всього життєвого циклу продукту.

Мінуси:

- кожна фаза є самостійною, окремі ітерації не накладаються;
- можуть виникнути проблеми з реалізацією загальної архітектури системи, оскільки не всі потреби відомі до початку проектування.

1.2.1.3 Спіральна модель

В світі розробки існує також так звана «спіральна» модель розробки програмного забезпечення. Вона включає в себе елементи як водоспадної, так і ітеративної моделі. Її суть складається в тому, щоб представити весь процес створення програмного забезпечення на умовну площину, розбиту на 4 сектори, кожен з яких являється окремим етапом його розробки: визначення цілей, оцінка ризиків, розробка і тестування, планування нової ітерації.

В спіральній моделі життєвий шлях розроблюваного продукту зображений у вигляді спіралі, які почавшись на етапі планування, розкручується з проходженням кожного наступного кроку. Таким чином на виході з чергового витка ми повинні отримати готовий протестований прототип, який доповнює існуючий білд. Прототип, який задовольняє вимоги – готовий до релізу.

Головна особливість спіральної моделі – концентрація на всіх ризиках. Для їхньої оцінки зазвичай навіть виділяється окрема стадія.

Мінуси:

- абсолютно нереалістичні бюджет та терміни;
- дефіцит розробників;
- часті зміни вимог;
- надмірна оптимізація;
- низька продуктивність системи;
- рівень кваліфікації із різних відділів не відповідає один одному;
- не підходить для невеликих проектів.

Плюси:

- покращений аналіз ризиків;
- хороша документація процесу розробки;
- гнучкість – можливість внесення змін і додавання нового функціоналу навіть на відносно пізніх етапах;
- швидке створення прототипів.

Враховуючи все вищеописане, я вибрав каскадну модель розробки програмного забезпечення. По-перше вона влучно попадає в методичні вказівки до пояснювальної записки, по-друге добре підходить для одного розробника.

1.2.2 Вибір архітектури програмного забезпечення

Значення програмного забезпечення слід розуміти під назвою «програмне забезпечення». Слово «забезпечення» означає продукт. Ідея програмного забезпечення полягає в тому, щоб змінити поведінку комп'ютера. Ви також можете змінити поведінку свого комп'ютера за допомогою деяких апаратних засобів, але це набагато складніше. Щоб досягти цього, програмне забезпечення повинно працювати безперебійно, тобто його потрібно легко змінювати. Оскільки зацікавлені сторони змінюють свою думку щодо певних характеристик, це повинно бути простим завданням, щоб пристосувати їх до потреб зацікавлених сторін. Складність у таких випадках повинна визначатися архітектурою, яка в свою чергу залежить від парадигми програмування.

Парадигма – це спосіб програмування, який не залежить від конкретної мови програмування. Парадигма визначає, які структури використовувати і коли їх використовувати.

ООП – парадигма програмування, основними концепціями якої являються поняття об’єктів і класів. В центрі ООП знаходиться поняття об’єкту. Дана парадигма дозволяє вести розробку з використанням модульності, яка в свою чергу надає розробнику можливість здійснювати розробку гнучко та використовувати переваги при побудові різноманітних складових частин програми (оскільки часи статичних html-сайтів минули, то ми можемо сміливо називати сучасні сайти програмами). ООП дозволяє розглядати програму як множину об’єктів, які певним чином взаємодіють між собою, та можуть об’єднуватись в модулі. Це означає, що розробивши клас, ми можемо використовувати його в різних варіантах компоновки системи, тобто формувати інші модулі або навіть цілі системи. Такий спосіб розробки

дозволяє зменшити об’єм лістингу коду програми, зробити їх прозорішими та прискорити написання різних реалізацій та тестів для них. Також це мінімізує витрати на супровід та впровадження програми.

Враховуючи свій попередній досвід розробки і для того щоби оминути проблеми з підтримкою коду, коли при зміні певної частини програми ламається інша, я поставив перед собою розробити «правильну» архітектуру програми, чи то пак стратегію розробки.

Розроблювана програма – це група об’єктів, які вказують один одному, що робити, за допомогою повідомлень. Об’єкти ідентичні у всьому крім внутрішнього стану під час роботи програми. Створення абстрактних типів даних є фундаментальне поняття у всьому об’єктно-орієнтованому програмуванні. Абстрактні типи даних діють майже так, як і вбудовані типи: ви можете створювати змінні типів (названі об’єктами або екземплярами класів в термінах ООП) і маніпулювати їми (що і називається надсиланням повідомлень або запитом; ви робите запит, і об’єкт вирішує що з ними робити). Члени кожного класу мають схожі властивості: кожен екземпляр класу Book має ім’я, автора та номер ISBN, наприклад. В той же час, кожен об’єкт відрізняється від іншого своїм внутрішнім станом: кожен має своє ім’я, свій номер та своїх авторів.

В цілому програма повинна складатися з декількох умовно незалежних в реалізації та функціональному наповненню підсистем. Тобто дані підсистеми можна охарактеризувати за допомогою діаграми компонентів системи. Одними з таких підсистем є система автентифікації користувача, система реєстрації, та безпосередньо основний функціонал веб-сайту. На рисунку 5 схематично зображено взаємодію компонентів підсистеми автентифікації:

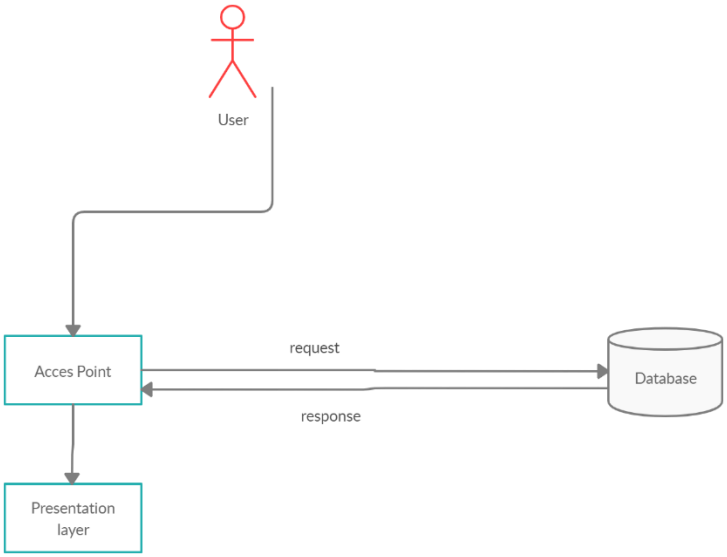

Рисунок 5 - взаємодія компонентів підсистеми автентифікації

Підсистему автентифікації користувачів можна представити у вигляді декількох класів, які взаємодіють між собою. У цій частині реалізовано валідацію введених даних користувачем на рівні presentation layer та на рівні бази даних. На рівні presentation layer проводиться аналіз валідності введених даних. Тобто якщо користувач ввів валідні дані, то далі на основі введених даних проводиться перевірка чи існує конкретний користувач в базі даних, та чи відповідає даному користувачу введений пароль.

Наступна діаграма зображує процес автентифікації користувача:

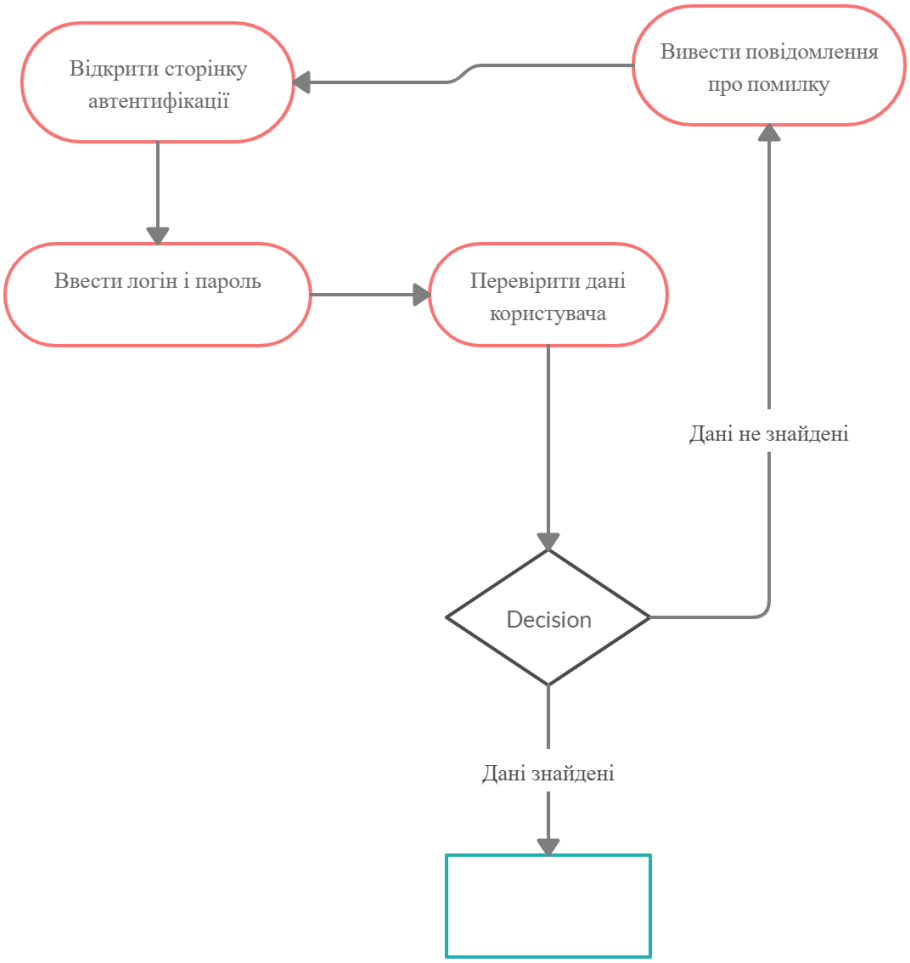

Рисунок 6 – процес автентифікації

Система реєстрації потребує від користувача ввести ім’я користувача та двічі пароль. Якщо ім’я користувача не задовольняє вимоги, або був вже кимось зайнятий, то користувач отримає відповідне повідомлення про помилку. Якщо пароль не задовольняє вимоги, або введені паролі відрізняються один від одного, то користувач також отримує відповідне до помилки повідомлення. Після успішної реєстрації в системі користувача буде переадресовано до сторінки автентифікації, після чого він зможе здійснити логін.

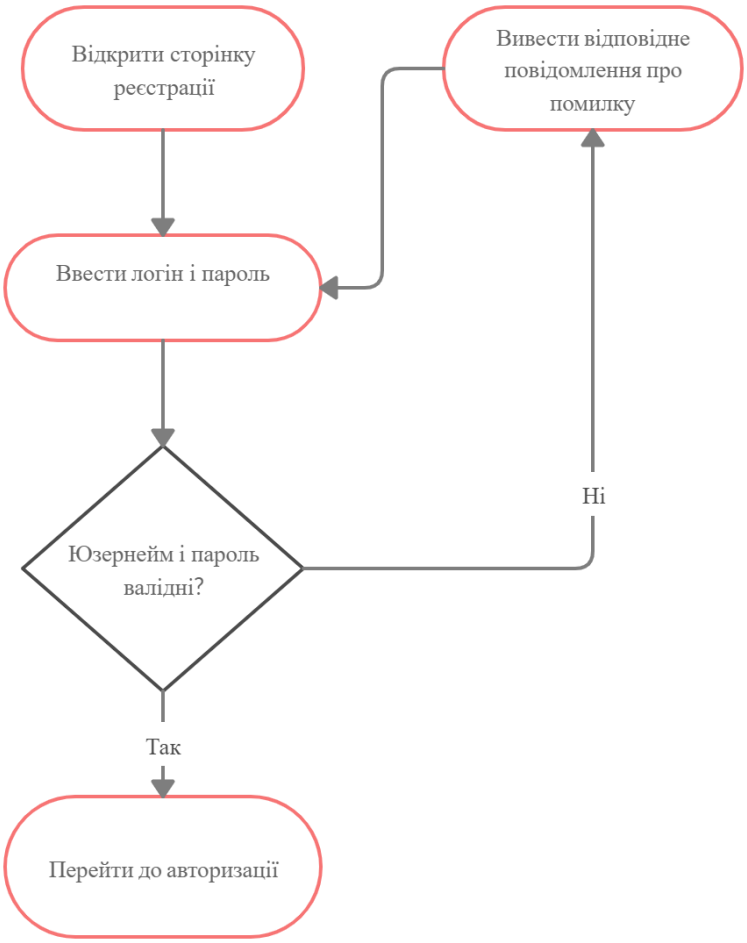

Рисунок 7 – система реєстрації

Практично ні одну ефективну програму не можна побудувати без використання програмних бібліотек. Якщо програміст буде розпоряджатись тільки самою мовою програмування, і не надати йому вже скомпільованих модулів, які були розроблені для вирішення найпоширеніших завдань, то йому доведеться вирішувати багато додаткових невеликих завдань.

У звичайному житті, якщо ви підете в бібліотеку або книжковий магазин, то ви там побачите велику кількість книжок, які відповідають на різні запитання: по історії, математиці, хімії, соціології, тощо. І не треба все придумувати – взяв книжку і отримав відповіді/рішення на багато питань. При чому всі книжки є розкладені строго по полицям. Так ось Java має віртуальну бібліотеку протестованого та скомпільованого коду, який вирішує безліч задач в повсякденній роботі програміста. А це дуже і дуже економить час

програміста, тому що не треба писати абсолютно весь код з нуля. В цій віртуальній бібліотеці Java інформація розбита по пакетам (packages) – це своєрідний аналог полиць в книжковому магазині. Кожен пакет має протестований код, які логічно розділені відповідно до своїх завдань. Наприклад: java.applet, java.lang. java.util, тощо. В пакеті лежать класи. Тобто, якщо до прикладу нам необхідно порахувати квадратний корінь, то нам не треба писати код, який покроково програмував би обрахування квадратного кореня. А такий код зайняв би на дуже і дуже багато рядків коду. Ми просто беремо із пакету java.lang клас Math, та використовуємо потрібний нам метод.

Така структура та існуюче готове рішення з програмних бібліотек значно економлять час виконання поставленого завдання. Розбиття та підтримка такого модульного способу конструювання ПЗ дозволяє гнучко та ефективно підійти до розробки програмних систем.

Як згадувалось у попередніх розділах, використаний об’єктно-орієнтований підхід до розробки ґрунтується на взаємодії об’єктів. Дані об’єкти в системі є екземплярами класів, які реалізуються з використанням парадигм ООП. У даній системі проектовано інтерфейси, класи що реалізують ці інтерфейси, класи що унаслідковуються від інших класів, методи для виконання операцій та обчислень. Такий підхід до розробки дозволяє уникнути суперечностей та помилок, надлишковості коду, зробити програмний код доступним для розуміння та читабельним, легким ля розробки для модифікації.

Так як об’єкти також мають складну структуру та виникають в результаті багатьох операцій з іншими об’єктами та класами, то вибудовується складне дерево викликів та використань об’єктів інших класів. Загалом така схема побудови системи дозволяє розгорнути модель від самого початку виклику найглобальнішого об’єкту до дрібніших, і таким чином дійти до того стану, коли буде отримано результат роботи програми за одним зі сценаріїв викликів. На виході програми як результат пророблених операцій користувач

отримує об’єкт системи з відповідним значенням аргументів, який вже може використовувати систему, в яку інтегрується дана модель.

1.2.3 Абстрактна модель системи

Один об’єкт або система може виступати в ролі моделі іншого об’єкту або системи, якщо між ними встановлено деякою мірою схожість. Моделлю системи (або будь-якого іншого об’єкту або явища) може бути формальний опис системи в якому виділені основні об’єкти, які складають систему, і відносини між цими об’єктами.

Побудова моделей – широко розповсюджений спосіб вивчення складних об’єктів та явищ. В моделі опущено багато деталей, які ускладнюють розуміння. Моделювання широко розповсюджено в науці, техніці та програмуванню.

Моделі допомагають перевірити працездатність розроблюваної системи на ранніх етапах її розробки; спілкуватись із замовником системи, уточнюючи його вимоги до системи; вносити (у випадку необхідності) зміни в проект системи (як на початку проектування, так і на інших фазах її життєвого циклу).

Шаблон проектування Model-View-Controller (далі MVC) ліг в основу архітектурного рішення першого середовища програмування з графічним інтерфейсом користувача. Вперше MVC описав ще в 1978 році Трюгве Рінскауг, який працював деякий час в лабораторії Xerox PARC. Шаблон проектування MVC передбачає розподіл даних програми, користувацького інтерфейсу та бізнес-логіки на три окремих компоненти: модель, представлення і контролер – таким чино, що модифікація кожного компоненту може відбуватися незалежно. Модель (model) забезпечує дані предметної області представленню (view) і реагує на команди контролера (controller), міняючи свій стан. Представлення відповідає за відображення предметних даних користувачу, реагуючи на зміни моделі. Контролер інтерпретує дії користувача, сповіщаючи модель про необхідність змін. Повний цикл MVC-тріади: модель, представлення і контролер перемикача – можна описати

наступним шляхом. При ініціалізації представлення користувачем воно звертається до моделі і встановлює текст мітки у відповідність з поточним станом перемикача. Користувач ініціює зміни перемикача, натискуючи на відповідний тригер. При цьому представлення надсилає відповідну команду контролеру: увімкнути або вимкнути. Контролер інтерпретує команду і змінює модель. Представлення реєструє зміни моделі: по цій події воно змінює текст мітки у відповідність до нового стану моделі перемикача.

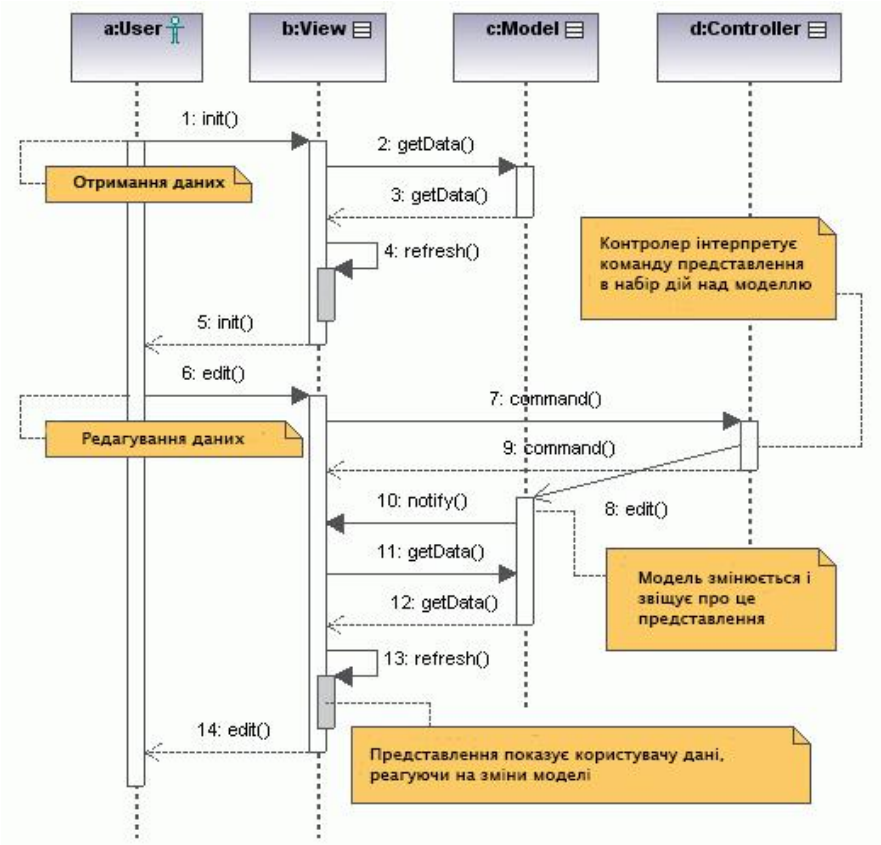

Рисунок 1.8 – Робота паттерну MVC.

На основі цих рішень було створено програмне забезпечення, яке реалізовує паттерн MVC.

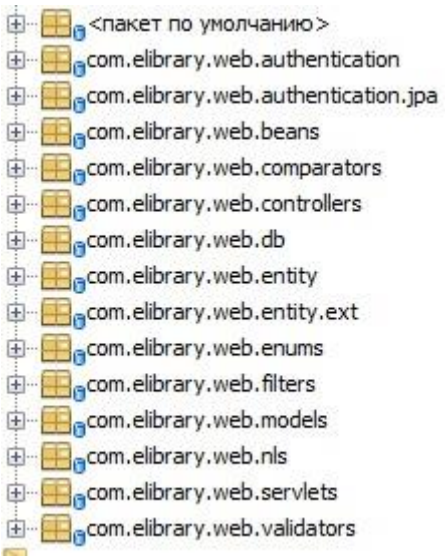

Рисунок 1.8 – Структура пакетів в проекті

Згідно паттерну MVC всі класи були розбиті згідно ролям по відповідним пакетам. В моєму випадку в ролі моделі виступають класи із пакету beans, вони ж сутності бази даних, в ролі контролерів виступають класи із пакету controllers, а в ролі представлення виступають самі веб-сторінки. Способом розробки можна вважати реверс-інжиніринг, що передбачає розробку та реалізацію всіх ключових елементів (класів та їх взаємодії), а уже на базі реалізованого виконується побудова UML-діаграми класів. Даний підхід був обраний у зв’язку зі складністю оцінити весь масштаб та структуру системи на початкових стадіях розробки.

UML (Unified Modeling Language – уніфікована мова моделювання) – мова графічного опису для об’єктного моделювання в області розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаною UML моделлю. Він був створений для того, щоб визначати, візуалізувати та проектувати програмні системи[23].

Система складається з класів та зв’язків між ними. Так як згадувалось вище, згідно паттерну проектування MVC система розбивається на пакети та класи, роль яких логічно покладена в назву пакета або класа. Розглянемо власне основні класи із пакетів: db та controllers. Чому основні? Тому що немає сенсу концентрувати увагу на всіх класах, так як робота тоді перетвориться буквально на документацію.

Пакет com.elibrary.web.controllers містить в собі класи, завдання яких забезпечувати зв’язок між користувачем та системою. Використовує модель та представлення для реалізації необхідної реакції на дії користувача, як правило, на рівні контролера відбувається фільтрація отриманих даних і авторизація (перевіряються права користувача на виконувані дії і отримання інформації. Власне клас BookListController відповідає за наповнення книгами в поточний момент сесії для конкретного користувача.

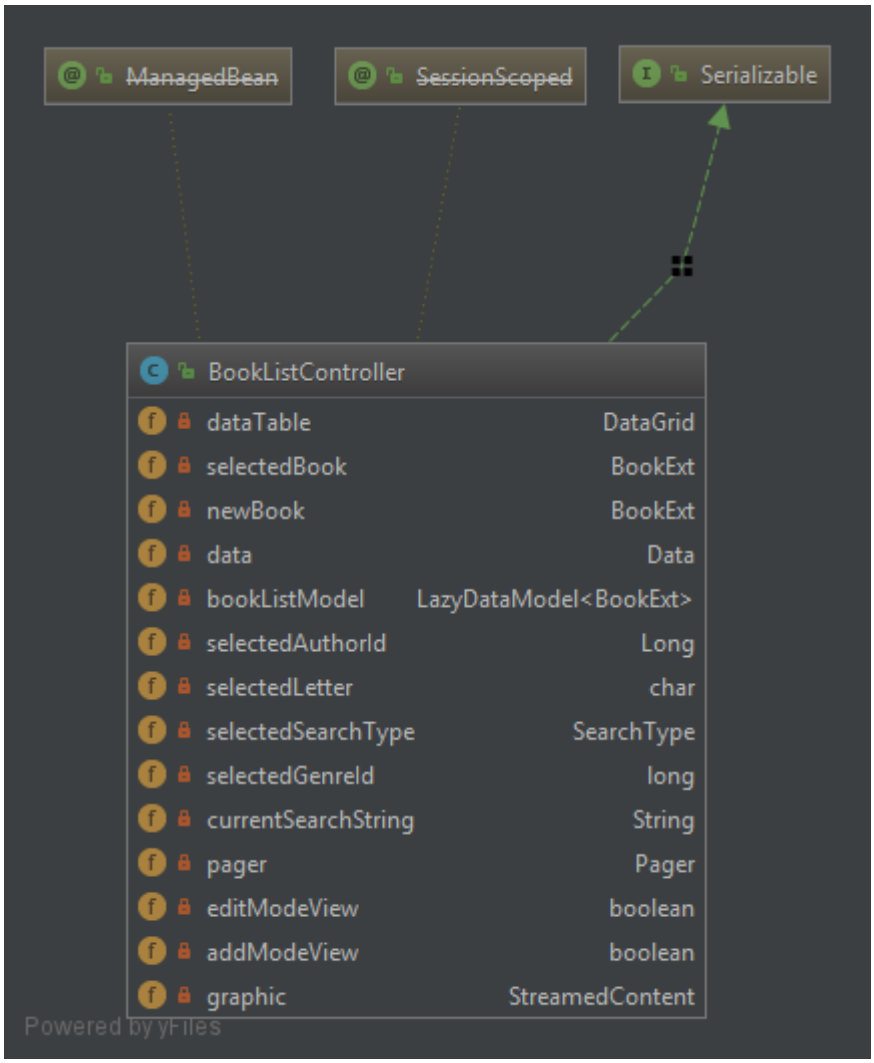

Рисунок 1.9 – Діаграма класу BookListController

Об’єкт класу Pager (рис. 10) містить в собі колекцію list типу ArrayList зі списком книг відібраних для користувача по заданих критеріях (назва, літера, категорія тощо).

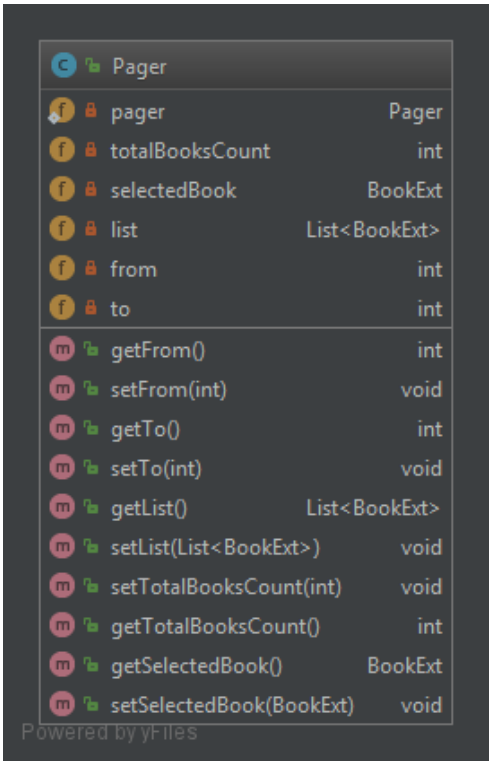

Рисунок 1.10 – діаграма класу Pager.

Також клас Pager використовується для навігації по списку з книжками на веб-сторінці. Клас GenreController працює з заповненням категорій книг (жанрів) (рис. 11).

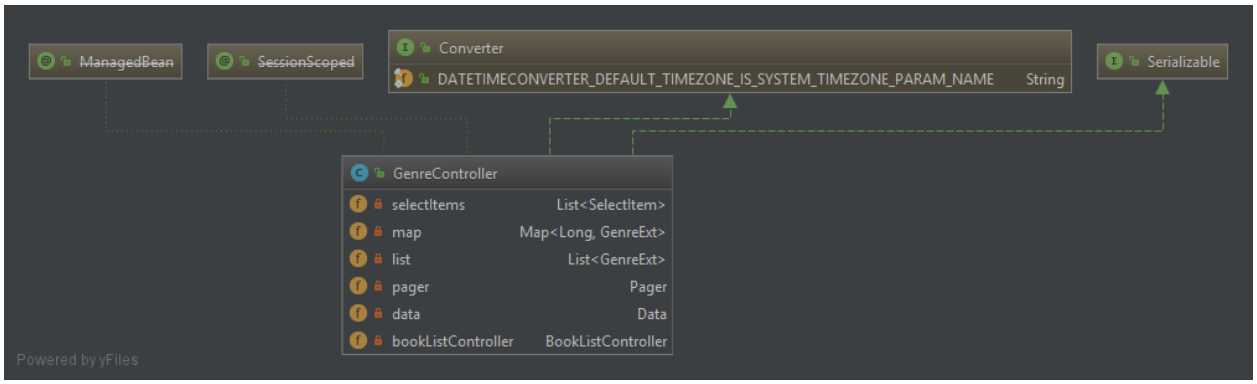

Рисунок 1.11 – діаграма класу GenreController

Клас (рис. 1.12) AuthorController працює з наповненням списку авторів в колекції, та відповідно на веб-сторінці. Надає можливість змінювати авторів в книжках.

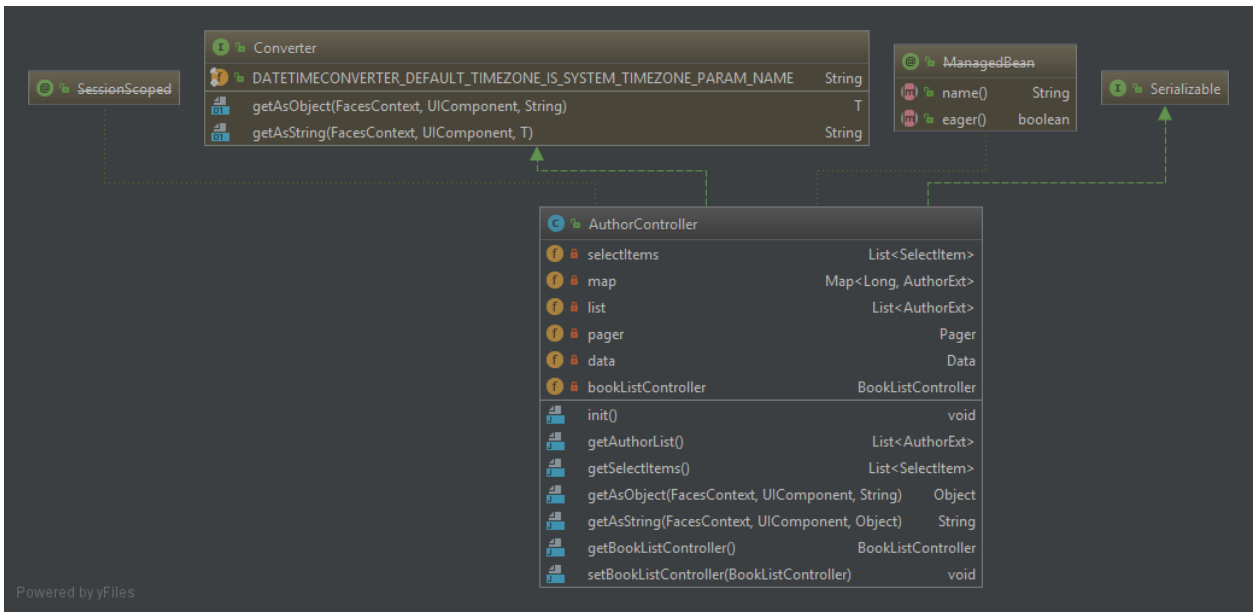

Рисунок 1.12 – діаграма класу AuthorController

Як ми можемо бачити з діаграм, кожен з класів-контролерів містить в собі посилання на змінну типу BookListController. Це зроблено для того, що дані класи мали доступ до вже ініційованої змінної класу Pager (рис. 13). Схематично це можна зобразити наступним чином:

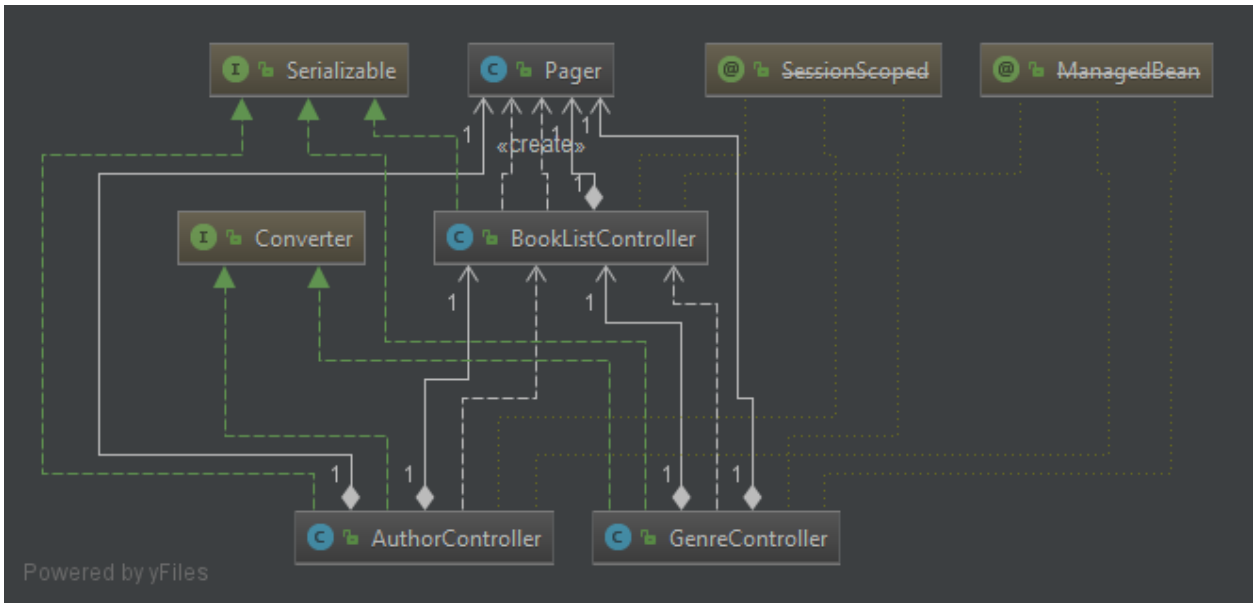

Рисунок 1.13 – діаграма залежностей між класами

Класи-контролери містять аннотацію ManagedBean. ManagedBean – це звичайний Java клас, котрий взаємодіє з JSF веб-сторінкою. ManagedBean містить в собі бізнес-логіку, геттери, сеттери. В більшості випадків вони

працюють як модель для UI компонентів. Але в даній роботі в ролі моделі виступають класи-сутності із пакету com.elibrary.web.entity (рис. 14)

Рисунок 1.14 – Структура пакету com.elibrary.web.entity

Hibernate дозволяє визначити звичайний Java клас, як частину моделі реляційної бази даних, точніше відповідність певного класу до таблиці в базі даних. Для початку, зупинимося на основних вимогах JPA-специфікації, які необхідно враховувати при створенні entity-класу:

- клас-сутність повинен бути анотований анотацією javax.persistence.Entity, або бути визначений як такий в XML-співставленні (використовується в даній роботі);
- клас-сутність повинен мати public або protected конструктор без аргументів. Entity клас також може визначати окремі додаткові конструктори з аргументами;
- клас-сутність повинен мати властивість – унікальний ідентифікатор, анотований анотацією javax.persistence.Id;
- клас-сутність повинен бути класом верхнього рівня;
- перерахування (enum) або інтерфейс (interface) не можуть бути визначені як сутність;

- клас сутності не повинен бути як final. Також ні один з методів або постійних екземплярів змінних класу сутності не можуть бути визначені як final;
- якщо екземпляр entity класу повинен використовуватись віддалено як окремий об’єкт, то він повинен реалізовувати інтерфейс Serializable;
- Entity класи можуть бути абстрактними;
- Entity класи можуть розширювати звичайні Java-класи (класи не сутності);
- Звичайні Java-класи можуть розширювати Entity класи;

Hibernate в свою чергу дозволяє спростити деякі вимоги JPA специфікації, та робить класи-сутності такими, які важко перенести на інші специфікації, тому не варто порушувати вимоги без гострої необхідності.

1.3 Розробка програмної системи

1.3.1 Вибір основних технологій для розробки

Перед розробкою будь-якого програмного забезпечення перед розробником постає питання у виборі тієї чи іншої технології для реалізації поставленої задачі, яка підходить найбільше за думкою розробника. Дуже важливо зробити правильний вибір, так як це напряму впливатиме на результативність, швидкість та подекуди і якість кінцевого програмного забезпечення. Також це впливатиме на трудомісткість та затрати.

Для того щоб реалізувати задачу в межах розробленою нами архітектури, нам потрібна мова яка реалізує парадигми ООП програмування. Одними із таких, та найпоширенішими на ринку є: Java, C++, C# та Python. Оскільки розроблюваний проект являється WEB-додатком, то одразу можемо зробити висновок, що C++ в цьому випадку нам не дуже підходить в силу складності та трудоємності розробки в сфері веб-програмування. Python міг би бути хорошим кандидатом, оскільки розробка на даній платформі є не дуже

складною та швидкою, але в силу своєї не гнучкості, вона нам не зовсім підходить. Лишається Java та C#.

Поява як Java, так і C# тісно пов'язано з переходом від низкорівневих мов програмування, таких як мова програмування C++ до мов більш високого рівня, які компілюються в байт-код. Байт-код можна запустити на віртуальній машині. З цим пов’язаний ряд переваг, в першу чергу, можливість написання коду, який буде зрозумілий людині, і буде працювати на будь-якій апаратній архітектурі, на якій встановлена віртуальна машина. Якщо відкинути синтаксичні причуди в сторону, то не дивно, що ці дві подібні між собою мови є такими популярними для розробників додатків. Ось декілька основних подібностей:

- Безпека типів. Помилка типу виникає тоді, коли тип даних одного об’єкту помилково назначається іншому об’єкту, створюючи побічні ефекти. І C# і Java, працюють на те, щоб гарантувати вияв таких типів незаконних приведень під час компілювання. Якщо приведення не може бути застосовано до нового типу, тоді під час виконання такі виключення будуть видалені.
- Garbage collector. На мовах більш низького рівня керування пам’яттю може бути дуже трудоємним процесом, тому що треба пам’ятати про те, що необхідно видалити нові об’єкти, щоб звільнити ресурси. На C# і Java є вбудований збирач сміття, який допомагає передбачити витік пам’яті шляхом видалення об’єктів, які більше не використовуються додатком.
- Можна наслідуватись тільки від одного класу. Обидві мови дозволяють наслідуватись від одного класу - це означає, що існує тільки один шлях з будь-якого базового класу в будь-який з його похідних класів. Це обмежує ненавмисні побічні ефекти, які можуть виникати при наявності декількох шляхів між декількома базовими класами і похідними класами. Diamond pattern - книжковий приклад цієї проблеми.
- Інтерфейси. Інтерфейс являє собою абстрактний клас, де всі методи абстрактні. Абстрактним методом є той метод, який оголошений, але не містить подробиць його реалізації. Код, що визначає будь-які методи або

властивості, певні інтерфейсом, має надаватися класом, який його реалізує. Це допомагає уникнути двозначності паттерна diamond, оскільки завжди ясно, який базовий клас реалізує даний похідний клас під час виконання. Результатом є чиста ієрархія лінійних класів одиночного наслідування в поєднанні з деякою універсальністю множинного спадкоємства. Фактично використання абстрактних класів є одним із способів множинного успадкування мов, які можуть подолати проблему паттерна diamond.

Сьогодні C# використовується в основному для реалізації CLI на .NET Framework, Mono і Portable.NET. Але java-комьюніті постійно створює нові бібліотеки і інструменти. Відповідно підтримка по тим чи іншим питанням краща, тому вибір був зроблений на користь мови програмування Java.

Головна особливість цієї мови є використання байт-коду. Після успішної компіляції всіх класів, вони транслюються в байт-код, який оброблюється віртуальною машиною JVM. Дана віртуальна машина входить в пакет для встановлення JRE, який і забезпечує роботу java-програм на різних операційних системах.

Основні переваги мови програмування Java:

- безпечність;
- ефективність;
- об’єктно-орієнтована спрямованість;
- стійкість до помилок;
- підтримка багатозадачності;
- незалежність від архітектури;
- доступ до інструментарію та матеріалів;
- ефективність розробки;
- garbage collector;
- велике комьюніті.

Одним із ключових принципів розробки мовою Java полягає у забезпеченні від несанкціонованого доступу. Програми на мові програмування

Java не можуть викликати глобальні методи і отримувати доступ до довільних системних ресурсів, що забезпечує рівень безпеки недоступний для інших мов.

Java має одну із найкращих підтримок. Як у вигляді документації, різноманітних інструкцій, так і користувацької. Як правило документація існує у повному обсязі, з детальним описом та навіть з прикладами. Також у глобальній мережі існує багато напрацювань, якими ви можете скористатись абсолютно безкоштовно.

1.3.1.1 ORM Hibernate

Як було згадано вище, за замовчуванням бібліотека мови програмування Java містить в собі додаткові пакети класів, які допомагають швидше вирішувати ті чи інші завдання, які були поставлені перед розробником. Але однієї стокової бібліотеки буває замало, і якщо користуватись виключно нею, то продуктивність розробки буде не такою швидкою, як при використанні фреймворків.

Фреймворк – це інструмент, який полегшує процес написання і запуску веб-додатків. Вам не потрібно самостійно писати купу коду і витрачати час на пошук потенційних прорахунків і помилок. Фреймворк – це програмне забезпечення, яке забезпечує роботу з певними технологіями на вищому рівні абстракції. Отже для роботою з базою даних я вибрав фреймворк під назвою Hibernate.

Hibernate – є проектом з відкритими вихідними кодами, який допомагає поєднати сутності в базі даних, з об’єктами класів. Постачається з абстрактним прошарком, та реалізує їх внутрішньо. Реалізація включає в себе задачі типу запису query-стрічок в базу даних, CRUD-операції або з’єднання з базою даних, тощо.

1.3.1.2 Java Server Faces

JavaServerFaces (JSF) є технологією, яка входить в специфікацію J2EE, допомагає будувати web-сторінки (view прошарок в моделі MVC) і пройшов

довгий шлях від першого релізу. Здебільшого це пов’язаного з великою екосистемою навколо цієї технології. Першим великим розширенням JSF був Facelets, який позбавив тягар розробки на JSP. З приходом версії 2.0 Facelets став стандартним для ентерпрайз специфікації, і разом з цим приніс багато нових функцій, наприклад таких як AJAX, на появу якого вплинула спільнота JAVA-розробників. JSF 2.0 послідував тому самому підходу.

Однією з основних нетехнічних вимог до корпоративних додатків була їхня довготривала підтримка і працездатність. Відповідно для цього їм потрібні технології, які можуть реалізувати дану вимогу, і мають підтримку на ринку та гарантії того, що давно розроблене програмне забезпечення зможе підтримуватись новими розробниками. Тут живе і процвітає Java, і спільнота Java (JCP) є двигуном розробки даної мови програмування.

Проте дещо складно підтримувати корпоративні сайти старими технологіями, в світі IT-технологій, де все швидко розвивається. Першим продуктом такої «еволюції» стали опенсоурс (OSS) проекти. Коли вийшла перша версія JSF, то корпоративний ринок сайтів з великою підозрою дивились на дану технологію з відкритим кодом. «Чи можемо ми довіряти їй? Чи надійна вона?» – задавались питанням розробники. З часом ринок корпоративних додатків прийняв опенсоурс, і JSF очолив даний тренд. Щоправда без підтримки JCP дана технологія прийшла би в непридатність.

Відкриття для вступу JCP спільноти стало ключом для розвитку еко-системи навколо JSF – стало революцією в розробці JSF-компонентів.

1.3.1.3 Primefaces

Primefaces являється бібліотекою компонентів, які дозволяють будувати функціональні, зручні та красиві web-сторінки. Також дана бібліотека є опенсоурс-проектом і власне є розширенням, чи то пак доповненням до JSF.

1.3.2 Вибір та огляд середовища для розробки

Розробка будь-якого програмного забезпечення виконується в тому чи іншому середовищі розробки. Середовище розробки – це програма або декілька програм, які ви використовуєте для створення програм. В цей процес входить, власне, написання коду, його налагодження, запуск інтеграція з системи управління версіями. Якщо раніше розробники часто користувались простим текстовим редактором (іноді навіть без підсвічування синтаксису), наприклад «Блокнот» і консоллю, то зараз частіше всього використовують інтегровані середовища для розробки або IDE.

Інтегроване середовище розробки або англійською Integrated Development Environment – IDE – це програма, яка містить в собі інструменти для розробки програмного забезпечення. Зазвичай середовище розробки включає в себе:

- текстовий редактор з підсвічуванням коду;
- компілятор або інтерпретатор;
- браузер класів, інспектор об’єктів і діаграму ієрархії класів;
- засоби автоматизації збирання;
- дебагер;
- засоби для інтеграції з системами керування версіями (Git);
- інструменти для спрощення конструювання графічного інтерфейсу користувача.

Для порівняння розглянемо IntelliJ IDEA, Eclipse Neon Java EE і NetBeans 8.1 Java EE.

Сучасна IDE повинна підтримувати Java 8, Scala, Groovy, а також інші мови віртуальної машини Java, яку вона використовує. Не була би зайвою і підтримка основних серверів веб-додатків і найбільш популярних веб-структур, у тому числі – Spring MVC, JSF Struts, GWT, Play, Wicket, Grails і Vaadin. IDE повинна бути сумісна з системами контролю версій, такими як Git та SVN. Додатково для середовища розробки важливо вміти працювати з базами даних і клієнтським рівнем вашого стеку.

1.3.2.1 IntelliJ IDEA

IntelliJ IDEA з точки зору можливостей і ціни поставляється у двох варіантах: безкоштовного Community Edition, і платного Ultimate Edition з розширеною функціональністю.

Community Edition призначена для JVM- і Android-розробки. Безкоштовна версія підтримує Java, Kotlin, Groovy і Scala; Android; Maven, Gradle і SBT; працює з системами контролю версій Git, SVN, Mercurial і CVS.

Ultimate Edition пристосована для веб і ентерпрайз-розробки. Ця версія IDE працює не тільки з Git, SVN, Mercurial і CVS, але також з Perforce, ClearCase і TFS; в ній ви можете писати на JS і TypeScript; є підтримка JavaEE, Spring, GWT, Vaadin, Play, Grails і ряд інших фреймворків. І, звичайно, не обійшлось без SQL і інструментів для роботи з базами даних.

Ідея якою керуються розробники даного середовища розробки, формуючи цінову політику, заключається в тому, шо її комерційна версія (Ultimate) займе своє місце на комп'ютерах професіоналів, за рахунок чого їхні продуктивність підвищиться. На даний момент ціна для бізнесу падає, для стартапів, фрілансерів вона суттєво нижча, а для студентів, вчителів і open-source розробників взагалі безкоштовна.

IntelliJ IDEA підкуповує своїм глибоким розумінням коду, розумною ергономікою, вбудованими функціями для розробки і підтримки багатьох мов (рис. 1.15).

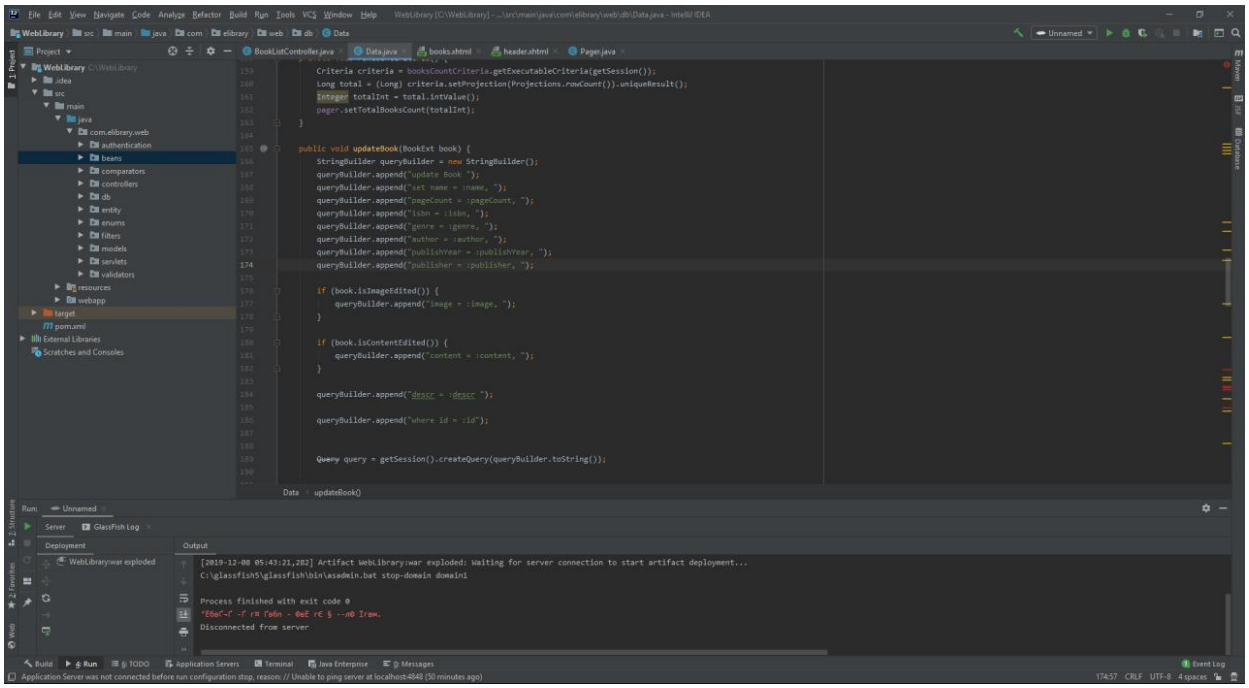

Рисунок 1.15 – інтерфейс IntellIJ IDEA

Підсвічування синтаксису і просте автодоповнення коду – звичайна справа для будь-яких сучасних Java-редакторів. IDEA пішла далі, пропонуючи «розумне автодоповнення». Цей термін означає, що середовище розробки показує список найбільш релевантних символів, які можна застосувати в даному контексті. Список символів залежить не тільки від контексту як такого, «загальноприйнятого», але від стилю програмування розробника, від того, наскільки часто він використовує ті чи інші оператори. «Завершення ланцюжка» і зовсім показує список застосовних символів, допустимих через методи або геттери в поточному контексті. Крім того, у випадку зі статичними членами або константами IDEA автоматично додає будь-які необхідні оператори імпорту (import). У всіх випадках автодоповнення, IDEA намагається вгадати тип символу під час виконання, уточнити свій вибір і навіть застосувати приведення типів якщо необхідно.

Код Java часто включає фрагменти з інших мов в вигляді рядків. IDEA може вводити код SQL, XPath, HTML, CSS або JavaScript в рядкові літерали Java. У цьому сенсі IDE може проводити рефакторинг коду на декількох мовах. Наприклад, якщо ви можете перейменувати клас в JPA-відображенні, IDEA оновить відповідний клас сутностей і виразів JPA.

Під час рефакторінга фрагмента коду, у розробника виникає одне (цілком природне) бажання: щоб всі дублікати цього коду також зарефакторились. IDEA Ultimate знаходить дублікати і схожі фрагменти і також застосовує до них рефакторинг.

IntelliJ IDEA аналізує код при завантаженні і безпосередньо при введенні. Вона вказує на передбачувані проблеми (як на нашому малюнку вище) і, за бажанням, пропонує список ймовірних швидких правок до виявлених проблем.

IntelliJ IDEA спроектована так, щоб не вибивати розробника зі стану потокової продуктивності, якщо він вже в нього потрапив. Вікно Project, зникає по простому кліку мишки, щоб програміст міг зосередитися на вікні редактора коду. На всі дії, які потрібні під час написання коду, тобто комбінації клавіш для швидкого доступу до них, в тому числі - визначення символів у спливаючих вікнах. Спочатку всі ці комбінації складно запам'ятати, але поступово до них звикаєш, і тільки ними і користуєшся. Проте, навіть якщо програміст не використовує комбінації клавіш, як показує практика, він звикає до роботи в IDEA легко і швидко.

Хочеться особливо відзначити відладчик IDEA: значення змінних відображаються безпосередньо у вікні редактора, поруч з відповідним вихідним кодом. При зміні стану змінної, колір підсвічування також змінюється.

IntelliJ IDEA забезпечує єдиний інтерфейс взаємодії з більшістю систем контролю версій, включаючи Git, SVN, Mercurial, CVS, Perforce і TFS. Ви можете управляти змінами безпосередньо в IDE, що дуже зручно. Коли я тестував IDEA, у мене виникало бажання, щоб остання зміна в вихідному коді показувалося в вікні редактора у вигляді анотації (як це відбувається, наприклад, в Visual Studio). Як виявилося, у IDEA є для цього спеціальний плагін.

Також IDEA оснащена інструментами для збірки, середовищем виконання тестів, інструментами покриття (coverage tools) і вбудованим

термінальним вікном. У IntelliJ немає власного профайлера, але за допомогою плагінів до неї можна під'єднати сторонні. Наприклад, YourKit, створений колишнім ведучим розробником JetBrains або VisualVM (це переупакованої версія профайлера NetBeans).

Налагодження в Java може бути болісним, коли відбуваються всякі загадкові речі з класами, початкових кодів яких у вас немає. У комплект IDEA входить декомпілятор для таких випадків.

Серверне програмування на Java передбачає часте взаємодія з базами даних, так що програмісти IDEA версії Ultimate оцінять зручність інструментів для роботи з SQL і БД. Але якщо комусь їх можливостей буде мало, можна придбати версію IDEA Ultimate з вбудованою SQL IDE (DataGrip). Правда, це буде трохи дорожче, ніж звичайна підписка IDEA Ultimate.

IntelliJ IDEA підтримує всі основні сервери додатків JVM, і дозволяє розгортати і проводити налагодження на цих серверах, що нівелює добре знайомі всім програмістам Java Enterprise труднощі. IDEA також підтримує Docker (за допомогою плагіна, який додає до середовища розробки спеціальне вікно інструментів Docker. До речі, плагінів у IDEA – велика кількість.

IDEA розширила підтримку коду Spring, Java EE, Grails, Play, Android, GWT, Vaadin, Thymeleaf, Android, React, AngularJS і інших фреймворків. Ви, мабуть, помітили, що не всі з них відносяться до Java. IDEA безпосередньо з коробки «розуміє» та іншими мовами - Groovy, Kotlin, Scala, JavaScript, TypeScript і SQL. Якщо ви не знайшли в цьому списку потрібного вам мови, зараз є 19 мовних плагінів IntelliJ, зокрема, для підтримки R, Elm і D.

1.3.2.2 Eclipse IDE

Років 10 тому на питання про краще IDE, Java-розробник відповідав впевнено: Eclipse. Довгі роки це середовище розробки впевнено тримала пальму першості серед Java IDE. Це середовище повністю безкоштовне, з відкритим вихідним кодом, написаним переважно на Java. Проте, її модульна архітектура дозволяє використовувати Eclipse і з іншими мовами. Проект

Eclipse, ініційований IBM, з'явився в 2001 році. Їм хотіли замінити сімейство середовищ розробки IBM Visual Age, заснованих на Smalltalk.

Ну а головною метою, про що навіть назва говорить, було затьмарити Microsoft Visual Studio (eclipse англійською означає затемнення).

Портативність Java допомагає Eclipse бути кросплатформовим середовищем: ця IDE працює на Linux, Mac OS X, Solaris і Windows.

Добре це чи погано, Java Standard Widget Toolkit (SWT), принаймні частково, відповідає за зовнішній вигляд Eclipse.

Своєю продуктивністю (або, як кажуть деякі доброзичливці, її відсутності) Eclipse зобов'язана JVM. Eclipse працює досить повільно, оскільки впирається корінням в досить старе «залізо» і старі версії JVM. Навіть сьогодні вона здається повільною, особливо якщо встановити на неї багато плагінів.

Частина витрат ресурсів Eclipse можна віднести на рахунок її вбудованого інкрементного компілятора, який запускається кожного разу при завантаженні файлу або оновленні коду. Корисна штука, саме вона ловить помилки при введенні тексту.

Незалежно від збірки, проект Eclipse підтримує модель контенту, яка містить інформацію про ієрархію типів, посилань і оголошеннях Java-елементів.

Поточна версія Eclipse носить ім'я Neon (4.6.0). Я інсталював Eclipse Java EE IDE для веб-розробників (це далеко не єдина опція, ви можете вибрати щось ще). Він містить мінімальну Eclipse SDK, а плагіни додаються на вимогу. До слова, робота з плагінами в цій IDE - не для людей зі слабкими нервами. Сторонні плагіни часто конфліктують між собою, хоча в їхній офіційній специфікації про це нічого не сказано.

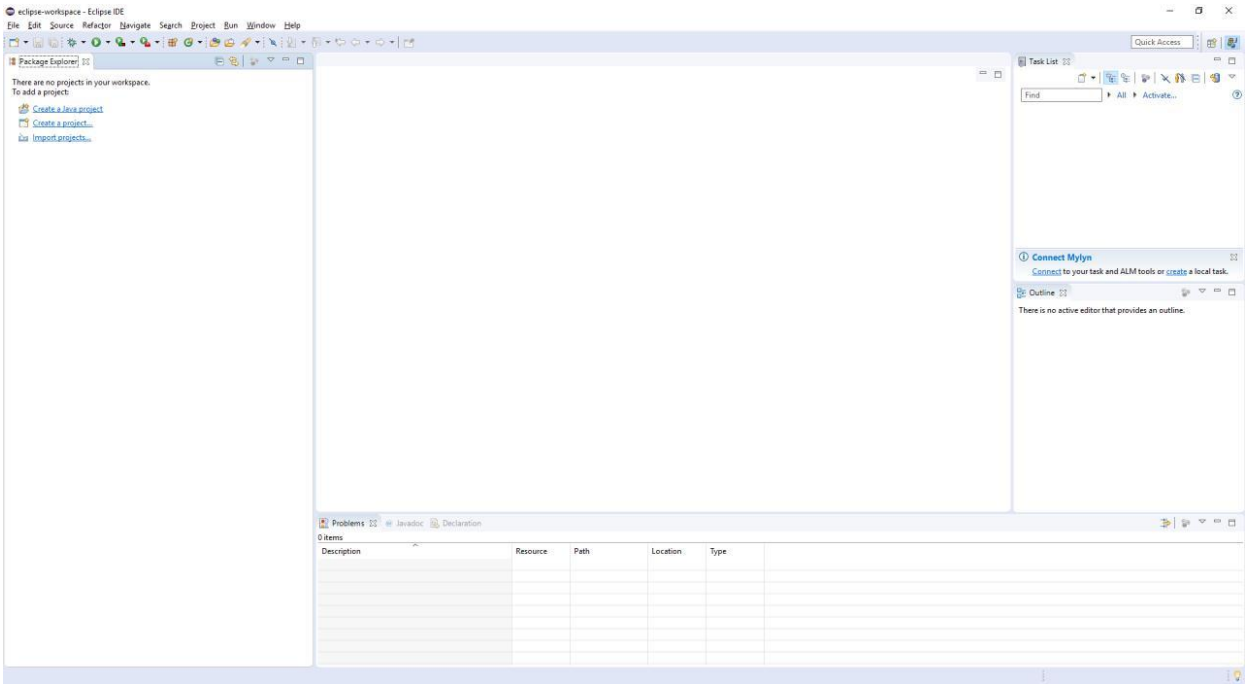

Рисунок 1.16 – Інтерфейс IDE Eclipse

Екосистема плагінів Eclipse – це одночасно сильна сторона цієї IDE і одна з головних її проблем. Саме через несумісність плагінів часом «падають» цілі збірки, і програмістам доводиться починати роботу спочатку.

В даний час для Eclipse розроблено понад 1700 плагінів, офіційних і неофіційних, які можуть працювати відмінно, а можуть погано. Модулі Eclipse, підтримують понад 100 мов програмування і майже 200 фреймворків для розробки додатків. Більшість серверів Java також підтримуються: якщо ви позначили нове з'єднання з сервером з Eclipse, ви потрапите в список папок виробників, де знайдете близько 30 серверів додатків. Тільки варіантів Apache Tomcat буде цілих дев'ять штук. Комерційні виробники, як правило, збирають свої рішення разом: наприклад, є тільки один пункт Red Hat JBoss Middleware, а вже всередині ви знайдете WildFly і інструменти EAP-сервера, а також JBoss AS.

Перший досвід роботи з Eclipse, може дещо збентежити, і навіть збити з пантелику. Спочатку необхідно налаштувати Eclipse і звикнути до її концептуальної архітектурі робочих просторів, ракурсів і видів. Все це

визначається плагінами, які ви встановили. Для серверної розробки на Java, ви, ймовірно, будете використовувати ракурси Java, Java EE і Java browsing, вид, що відображає структуру пакета (Package Explorer), ракурс налагодження, ракурс командної синхронізації веб-інструментів, ракурс розробки баз даних і ракурс налагодження бази даних. На практиці все набуває сенсу, коли ви відкриєте потрібні вам вікна.

Eclipse практично завжди пропонує кілька способів вирішення тієї чи іншої задачі. Наприклад, ви можете переглядати код за допомогою ракурсу перегляду Java (Java browsing perspective). Що вибрати - справа смаку і вибору. Спеціальний пошук Java дозволяє знайти оголошення, посилання і входження Java-пакетів, типів, методів, полів. Ви також можете використовувати швидкий доступ до пошуку і попередній перегляд.

Поширені патерни коду можна згенерувати з шаблонів коду. Рефакторинг Java в Eclipse, підтримує 23 операції, починаючи від загальноприйнятих операцій з перейменування і закінчуючи менш очевидними перетвореннями (як в книзі Мартіна Фаулера).

Eclipse, підтримує налагодження як локально, так і віддалено, за умови, що ви використовуєте JVM, яка підтримує віддалену налагодження. Налагодження досить стандартна: ви визначаєте контрольні точки, а потім переглядаєте змінні в закладці налагодження. Звичайно, можна покроково виконувати свій код і обчислювати вирази.

У Eclipse – широка база документації самого різного віку, цінності і корисності. На жаль, виявити невідповідність поточної версії картинки в інструкції, наприклад, із застарілим інтерфейсом і розташуванням кнопок – звичайна справа для цієї IDE. На жаль, проблема запізнілого поновлення документації дуже характерна для будь-яких проектів з вихідним кодом.

1.3.2.3 NetBeans

NetBeans з'явилася як студентський університетський проект в Празі в 1996 році. У 1997 році IDE стала комерційним продуктом, а в 1999 році її

викупила компанія Sun Microsystems і вже на наступний рік представила open source-реліз.

Актуальна версія 8.1 працює на комп'ютерах під керуванням ОС Windows, Mac OS X, Linux і Solaris. Ну а пакет portable можна запустити на будь-яких системах, для яких існує Java-машина. Собі я завантажив Java EE bundle, це один з шести можливих пакетів завантаження. Цей бандл підтримує JavaScript і HTML, GlassFish і Tomcat, але не підтримує PHP, C / C ++ / Fortran, Groovy і Grails: їх можна отримати в пакеті «Все включено» (або просто «All»). Проте, при бажанні, я в будь-який момент зможу завантажити підтримку вищеназваних мов, вибравши відповідний плагін (та й будь-який інший). Їх у NetBeans менше, ніж у Eclipse, проте вони зазвичай не конфліктують один з одним.

Цієї осені Oracle (їй NetBeans дісталася після поглинання Sun Microsystems) вирішила передати цю середу розробки під крило Apache Software Foundation разом з усіма правами, вихідними кодами, торговою маркою, доменом "netbeans.org" і низкою інших елементів інфраструктури. Тому, майбутнє проекту поки туманно, хоча раніше у системи були певні родинні привілеї. Так, саме NetBeans першою отримала підтримку Java 8 практично відразу після виходу оновленої платформи, і була названа «офіційної IDE для Java 8». Втім, через місяць після виходу ця перевага була втрачено: саме тоді інші IDE також отримали підтримку восьмої Java.

Проте, хочу зазначити, підтримка Java 8 в NetBeans дійсно хороша, і ця IDE відмінно підходить для вплетіння в «старий» код елементів восьмої версії. Її редактори, аналізатори коду і конвертери допоможуть програмісту провести покращення коду, використовуючи в ньому конструкції, характерні для Java 8 - лямбда-вирази, функціональні оператори і посилання на методи. Модулі JavaScript в NetBeans 8 відмінно справляються з підтримкою Node.js і новітніх інструментів JavaScript, таких як Gulp і Mocha, так само як і підтримку інтерпретатора JavaScript Nashorn.

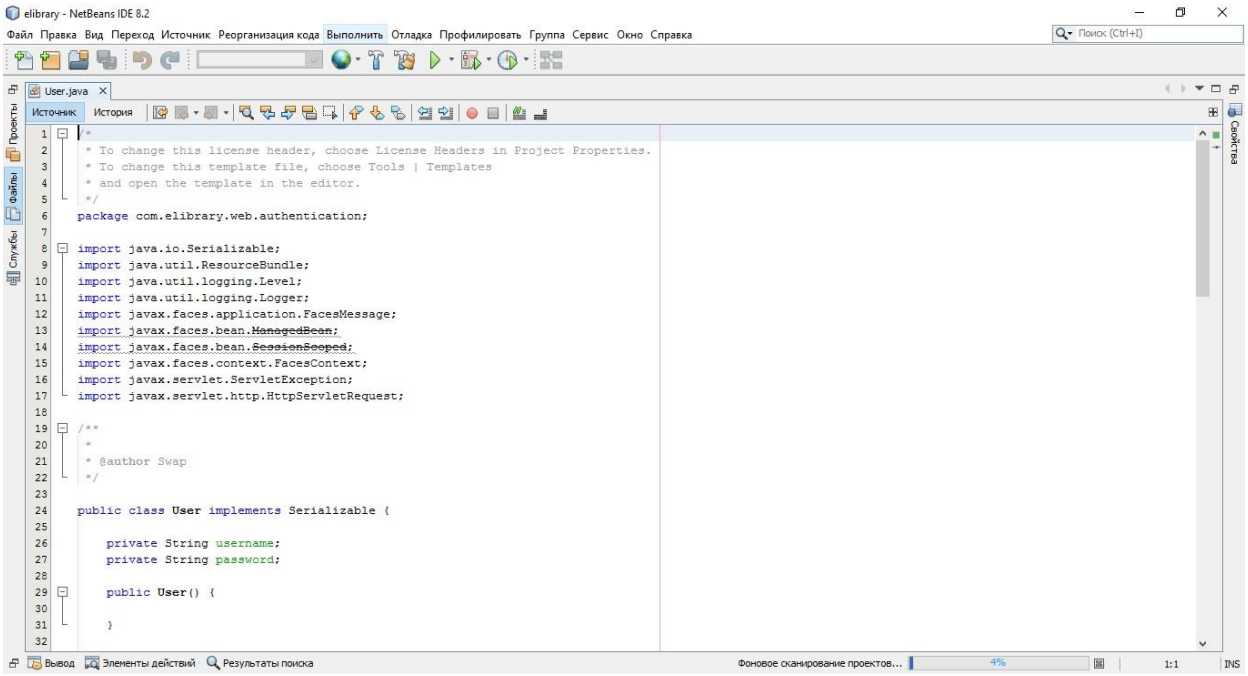

Рисунок 1.17 – Інтерфейс головного робочого вікна NetBeans

Редактор NetBeans підтримує мови, виявляє помилки в той час, коли ви друкуєте, і допомагає вам за допомогою спливаючих підказок і «розумним» автодоповненням коду. За суб'єктивним відчуттям IDE справляється з цим завданням швидше, ніж Eclipse, але дещо повільніше за IntelliJ IDEA. Крім того, NetBeans володіє повним спектром інструментів рефакторингу, які дозволяють програмісту реструктуризувати код, не ламаючи його, виконувати аналіз початкових кодів, а також пропонує широкий набір підказок для швидких виправлень або розширення коду. До складу NetBeans входить

інструмент проектування для графічного інтерфейсу користувача Swing, раніше відомий як "Project Matisse".

Розробники високо оцінюють засіб автоматизованого рефакторингу Inspect & Transform, що з'явилося в версії NetBeans 7.1. Воно дозволяє провести аналіз коду проекту і зробити пропоновані поліпшення. Хоча особисто я вважаю за краще спочатку перевірити весь власний код unit-тестами, і тільки потім запускати інструменти, які можуть внести радикальні зміни.

У NetBeans є відмінна вбудована підтримка Maven і Ant, а також плагіна для Gradle. Проекти Maven сприймаються системою як «рідні». Це означає, що їх можна просто відкривати, а не імпортувати. NetBeans також містить привабливе (і корисне) графічне відображення для залежностей Maven.

Відладник Java NetBeans непоганий, але з застереженнями. Окремий візуальний відладник дозволяє програмісту робити знімки екрану графічного інтерфейсу і вивчати інтерфейси додатків, виконаних за допомогою JavaFX і Swing. Профайлер NetBeans робить більш очевидним те, яким чином використовується процесор і пам'ять, і володіє відмінними інструментами для пошуку витоків пам'яті.

Отже протестувавши всі три IDE, моя я вирішив зупинитись на IntelliJ IDEA, так як продуктивність розробки в даному інтегрованому середовищі була найвища.

1.3.3 Вибір СУБД та її фізична модель

Веб-розробник росте разом з проектами, які він створює і розвиває. З ростом проектів збільшується складність програмної частини, неминуче зростає кількість оброблюваних нею даних, а так само складність схеми даних. Спілкування з іншими розробниками показує, що величезною популярністю серед користуються бази даних MySQL. В своєму арсеналі MySQL має зручний інструмент для управління базами даних MySQL Workbench. MySQL Workbench - інструмент для візуального проектування баз даних, що інтегрує

проектування, моделювання, створення та експлуатацію БД в єдине безшовне оточення для системи баз даних MySQL.

На даний момент MySQL представляє собою одну із самих надійних, швидких, якісних і відомих систем управління базами даних. Основною причиною цього є її безкоштовне розповсюдження разом з її вихідними кодами, інша причина – це те, що MySQL доволі швидка СУБД.

Головна особливість роботи СУБД MySQL заключається в використання мови структурованих запитів – SQL в ролі основного в управлінні базою даних, а саме: для створення або видалення таблиць в базі даних, здійснення вибірки із бази даних, для безпосереднього заповнення таблиць даними.

MySQL володіє високою стійкістю, високою швидкістю роботи, простотою в налаштуванні і використанні, вихідні коди сервера компілюються на багатьох платформах, тому СУБД MySQL є достойним рішенням для невеликих проектів.

1.3.3.1 Створення фізичної моделі бази даних

Для представлення схеми даних, сутностей і їх зв'язків в графічному вигляді в MySQL Workbench існує редактор EER-діаграм. Для створення діаграми у розділі «Model» тиснемо «Create EER Model from Database» (рис. 5)

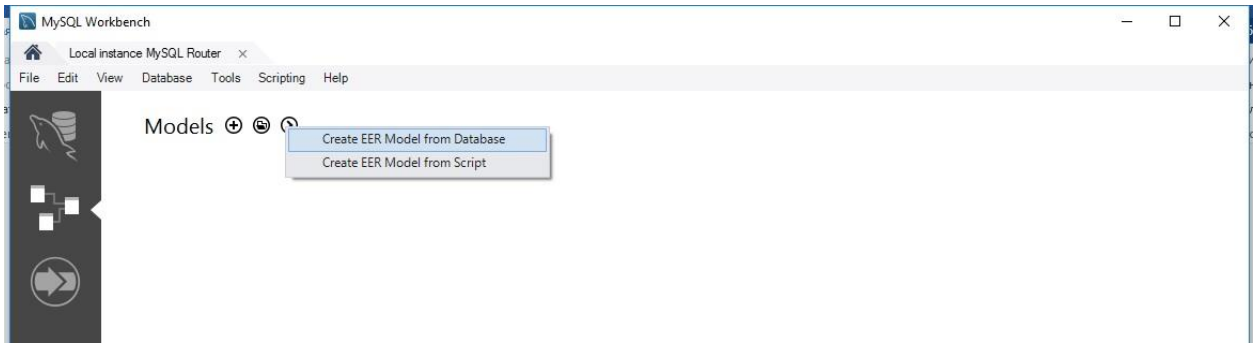

Рисунок 1.18 – Створення EER моделі

Далі вводимо дані для підключення до бази даних (рис. 19)

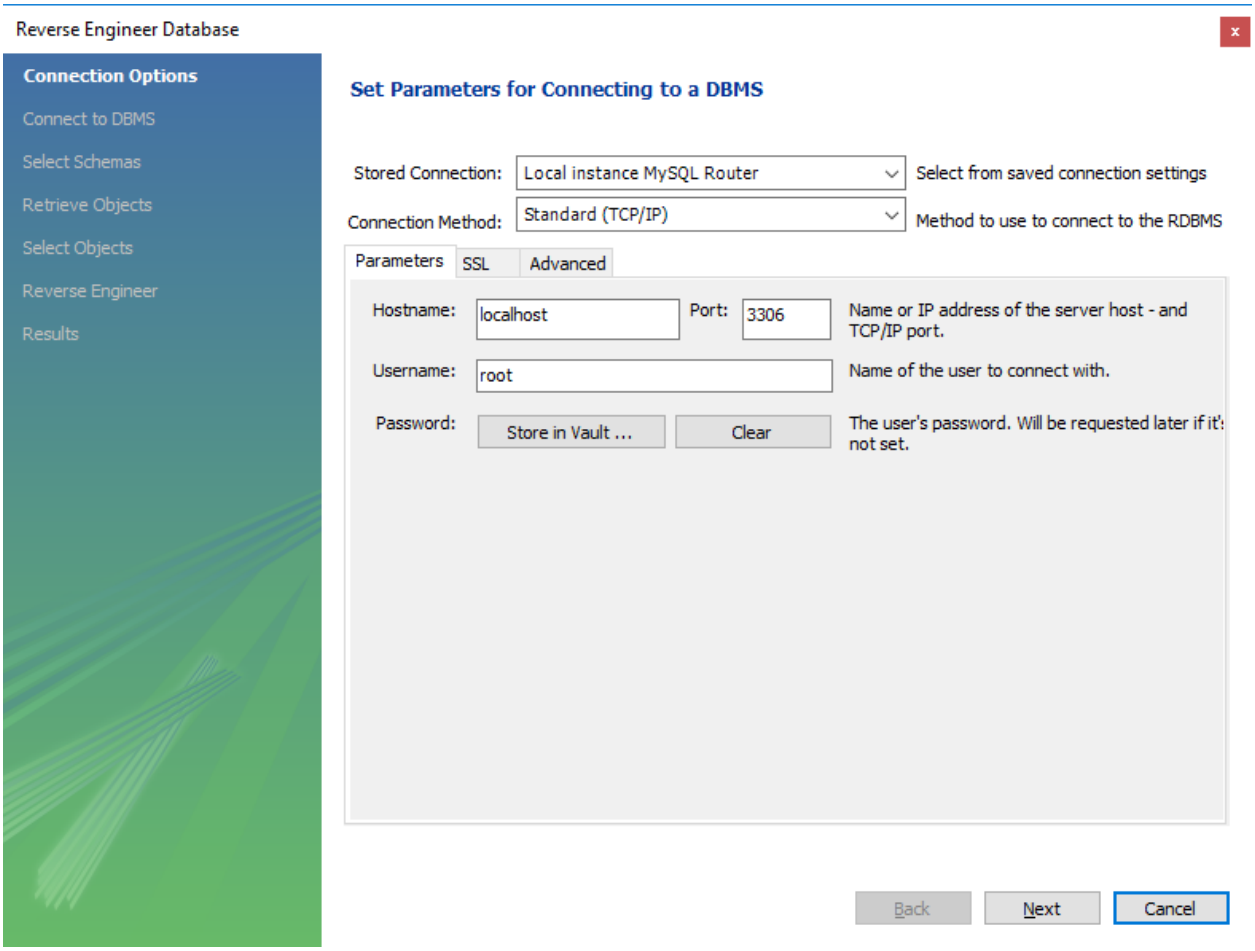

Рисунок 1.19 – Підключення до бази даних для створення EER моделі

Далі тиснемо «Next», обираємо потрібну нам базу даних, тиснемо ще раз «Next», «Next», «Execute», «Next» та «Finish». В результаті отримуємо фізичну модель нашої бази даних (рис. 1.20).

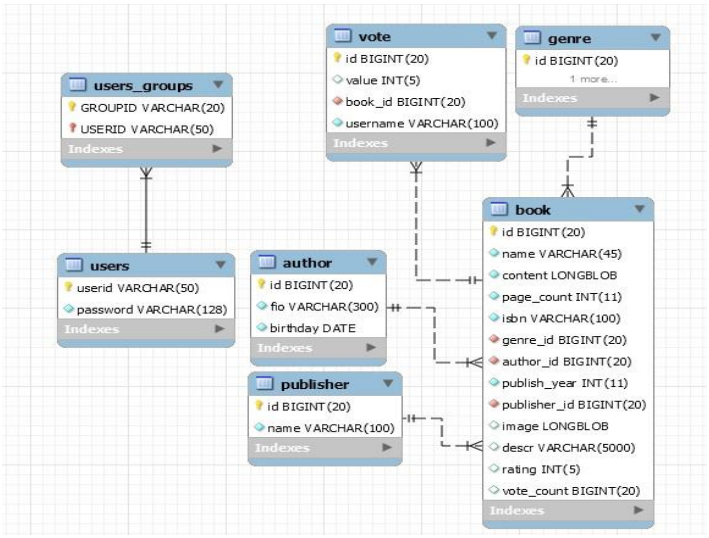

Рисунок 1.20 – Фізична модель бази даних

1.5 Реалізація основних класів та методів

1.5.1 Клас LoginViewer

Робота з сайтом починається з логування. Тому розглянемо LoginViewer. Клас LoginViewer знаходиться в пакеті com.elibrary.web.jpa.authentication, та працює для автентифікації користувачів. Власне метод login() (ліс. 1.1) виконує дану роботу.

```
2   public String login() {
3           FacesContext context = FacesContext.getCurrentInstance();
4           HttpServletRequest request = (HttpServletRequest)
    context.getExternalContext().getRequest();
5           String error =
    ResourceBundle.getBundle("com.elibrary.web.nls.messages",
    context.getViewRoot().getLocale()).getString("login_error");
6
7           try {
8               request.login(username, password);
9           } catch (ServletException e) {
10              e.printStackTrace();
11              context.addMessage("formLogin:messages", new
    FacesMessage(FacesMessage.SEVERITY_ERROR, error, null));
12              System.out.println("Login failed for user " + username + "with
    password "+ password);
13              return "login";
14          }
15
16          Principal principal = request.getUserPrincipal();
17          this.user = userEJB.findUserById(principal.getName());
18
19          log.info("Authentication done for user: " + principal.getName());
20
21          ExternalContext externalContext =
    FacesContext.getCurrentInstance().getExternalContext();
22          Map<String, Object> sessionMap = externalContext.getSessionMap();
```

```
23          sessionMap.put("User", user);
24
25          if (request.isUserInRole("users") ||
    request.isUserInRole("admins")) {
26              return "/pages/books.xthml?faces-redirect=true";
27          } else {
28              return "login";
29          }
30      }
31
```

Лістинг 1.1 – Імплементація методу login() класу LoginViewer

В ньому ми отримуємо об’єкт класу HttpServletRequest, пробуємо провести логін за допомогою методу login() цього класу з ім’ям користувача та паролем які зберігаються в полях String username, String password класу LoginViewer, які в свою чергу були завбачливо отримані з відповідних полей на веб-сторінці login.xhtml. Якщо виникає виключна ситуація (exception), то користувачу відобразиться помилка на сторінці, якщо ні – далі в роботу вступає JPA (java persistence API), і користувача буде переадресовано на сторінку з контентом.

1.5.2 Клас BookListController

Даний клас знаходиться в пакеті com.elibrary.web.controllers, та відповідає свій назві – виконує роботу по наповненню та керування контентом. Посилається на: змінну для додавання нової книги, змінну для редагування існуючих книг, декілька методів які роблять запити в базу даних. В контексті цього класу ми їх розглядати не будемо, так як вони є проміжними, і основна робота з базою даних виконується в класі Data. Також містить посилання на екземпляр класу LazyDataModel<T> (Primefaces), який виконує роботу по рівномірному наповненню сторінки контентом. Для того, щоб увімкнути «ледаче завантаження», потрібно реалізувати клас LazyDataModel таким чином, щоб він робив запит до бази даних під час pagination, sorting і filtering.

1.5.2 Клас Data

Data – це клас, який за допомогою гібернейту працює з базою даних. Він зв’язує додаток з базою даних.

Спроектований він таким чином, щоб завантажувати контент (файл pdf) тільки після безпосереднього запиту на файл (ліс. 1.2) (завантаження або читання).

```
    public Data(Pager pager) {

        this.pager = pager;

        prepareCriterias();

        sessionFactory = HibernateUtil.getSessionFactory();

        bookProjection = Projections.projectionList();
        bookProjection.add(Projections.property("id"), "id");
        bookProjection.add(Projections.property("name"), "name");
        bookProjection.add(Projections.property("image"), "image");
        bookProjection.add(Projections.property("genre"), "genre");
        bookProjection.add(Projections.property("pageCount"), "pageCount");
        bookProjection.add(Projections.property("isbn"), "isbn");
        bookProjection.add(Projections.property("publisher"), "publisher");
        bookProjection.add(Projections.property("author"), "author");
        bookProjection.add(Projections.property("publishYear"),
"publishYear");
        bookProjection.add(Projections.property("descr"), "descr");
        bookProjection.add(Projections.property("rating"), "rating");
        bookProjection.add(Projections.property("voteCount"), "voteCount");

        getAllBooks();
    }
```

Лістинг 1.2 – конструктор класу Data

Projection вказує на ті поля, які потрібно завантажити при ініціалізації даного класу.

Також можна робити різноманітні вибірки. Розглянемо на прикладі пошуку по назві книги.

```
public void getBooksByName(String bookName) {

        Criterion criterion = Restrictions.ilike("b.name", bookName,
MatchMode.ANYWHERE);

        prepareCriterias(criterion);
        populateList();
    }

    private void prepareCriterias(Criterion criterion) {
        bookListCriteria = DetachedCriteria.forClass(Book.class, "b");
        createAliases(bookListCriteria);
        bookListCriteria.add(criterion);

        booksCountCriteria = DetachedCriteria.forClass(Book.class, "b");
        createAliases(booksCountCriteria);
        booksCountCriteria.add(criterion);
    }
```

```
    private void createAliases(DetachedCriteria criteria) {
        criteria.createAlias("b.author", "author");
        criteria.createAlias("b.genre", "genre");
        criteria.createAlias("b.publisher", "publisher");
    }
    public void populateList() {
        runCountCriteria();
        runBookListCriteria();
    }
    private void runBookListCriteria() {
        Criteria criteria =
bookListCriteria.getExecutableCriteria(getSession());

criteria.addOrder(Order.asc("b.name")).setProjection(bookProjection).setResul
tTransformer(Transformers.aliasToBean(BookExt.class));


criteria.setFirstResult(pager.getFrom()).setMaxResults(pager.getTo());

        List<BookExt> list = criteria.list();
        pager.setList(list);
    }
```

Лістинг 1.3. – Ланцюжок викликів для запиту книг по імені

Як ми бачимо з коду, гібернейт дозволяє писати не типові SQL запити в базу, а використовувати Criterias. Фактично на найнижчому рівні доступу до бази даних йдуть звичайні SQL запити, але у вигляді Criterias їх набагато зручніше та простіше конструювати. Даний код може здатися дещо збитковим, якщо розглядати його тільки в контексті одного пошуку по книзі, та методи prepareCriterias(), createAliases(), populateList() та runBookListCriteria() виконуються для всіх методів, які запитують інформацію з бази даних.

1.6 Розгортання додатку

1.6.1 Налаштування веб-серверу Glassfish

Для того, щоб можна було працювати на розроблюваному веб-сайті, нам знадобиться відповідно веб-сервер. Нас цікавлять сервери, які повністю підтримують стек технологій JavaEE, та являються open source. В мережі існує багато безкоштовних веб-серверів (сервлет-контейнерів), таких як: GlassFish, WildFly, Geronimo, Tomcat, тощо. Та мій вибір впав на сервер додатків Glassfish.

Glassfish являється еталонною реалізацією JavaEE і таким чином підтримує EJB, JPA, JSF, JMS, RMI та інші технології, які потребуються для ентерпрайз-додатків. Це дозволяє розробникам створювати корпоративні додатки, які можна переносити і масштабувати, а також інтегрувати з успадкованими технологіями.

Отже завантажити сервер Glassfish можна з офіційного сайту Oracle. Далі заходимо в папку з розпакованим glassfish, далі в підпапку glassfish, далі підпапка domains, потім в папку domain1. Заходимо в папку config і знаходимо файл domain.xml. Там також шукаємо число 8080 (це число взагалі характерно як http-порту за замовчуванням для серверів додатків і контейнерів сервлетів) і міняємо його на що захочемо. Я залишив за замовчуванням.

Запустимо glassfish. Заходимо в папку з розпакованим glassfish, далі в підпапку glassfish, потім в папку bin. Запускаємо файл startserv.bat. У браузері вводимо адресу http://localhost: 8080. З'являється сторінка із заголовком GlassFish Server, знаходимо посилання go to the Administration Console і тиснемо на неї.

Для того, щоб наш додаток міг працювати, нам треба налаштувати роботу веб-сервера з базою даних. Для цього в консолі адміністратора потрібно зайти в розділ Resources – JDBC – JDBC Resources, та в JDBC Connection Pools (рис. 1.21).

Рисунок 1.21 – Меню підключень до бази даних в консолі адміністратора

Далі в меню JDBC Connection Pools нам потрібно налаштувати з'єднання з базою даних MySQL (рис. 1.22).

Рисунок 1.22 – Основні налаштування з’єднання

Із обов’язкових полей для заповнення являється Resource Type, Datasource Classname, та у вкладці Additional Properties (рис. 1.23) user, password, databaseName, portNumber та url. Інші поля можна заповнювати опціонально – все залежить від бажаної конфігурації сервера.

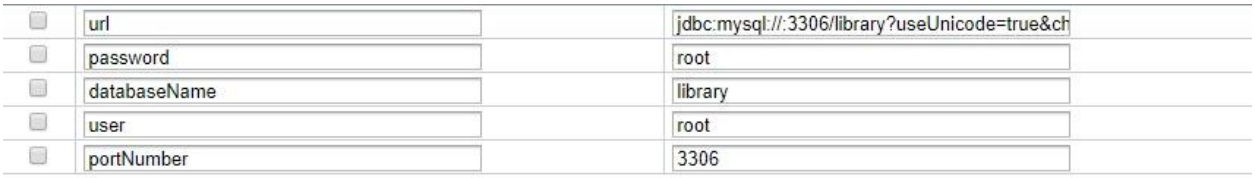

Рисунок – 1.23 Additional Properties

Так як в нашому додатку ми маємо систему автентифікації, то належним чином потрібно сконфігурувати сервер для роботи з системою безпеки. У Glassfish для цього потрібно перейти в розділ Configurations, Server-Config, Security, Realms (рис. 1.24).

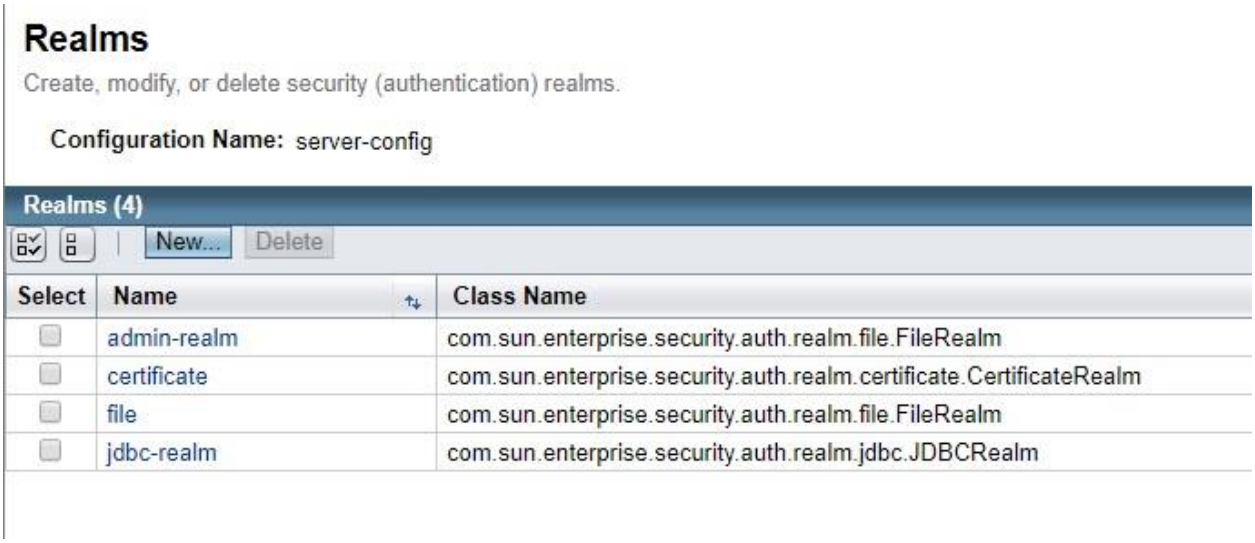

Рисунок 1.24 – Перелік реалмів для автентифікації

За замовчуванням розробнику доступно три реалми:

- admin realm – він же file realm – зберігає всю потрібну для автентифікації інформацію в файлі admin-keyfile;
- file realm – Glassfish зберігає дані для авторизації в файлі keyfile. File realm встановлений за замовчуванням;
- certificate – Glassfish зберігає облікові дані користувачів у базі даних сертифікатів. Під час використання сфери сертифікатів сервер використовує сертифікати з протоколом HTTPS для аутентифікації веб-клієнтів. Існують також LDAP realm, Digest, PAM тощо. Але в рамках даної роботи розглянуті вони не будуть.

Нас цікавить зберігання облікових даних користувачів в базі даних, отже нам потрібно сконфігурувати jdbc-realm (рис 1.25).

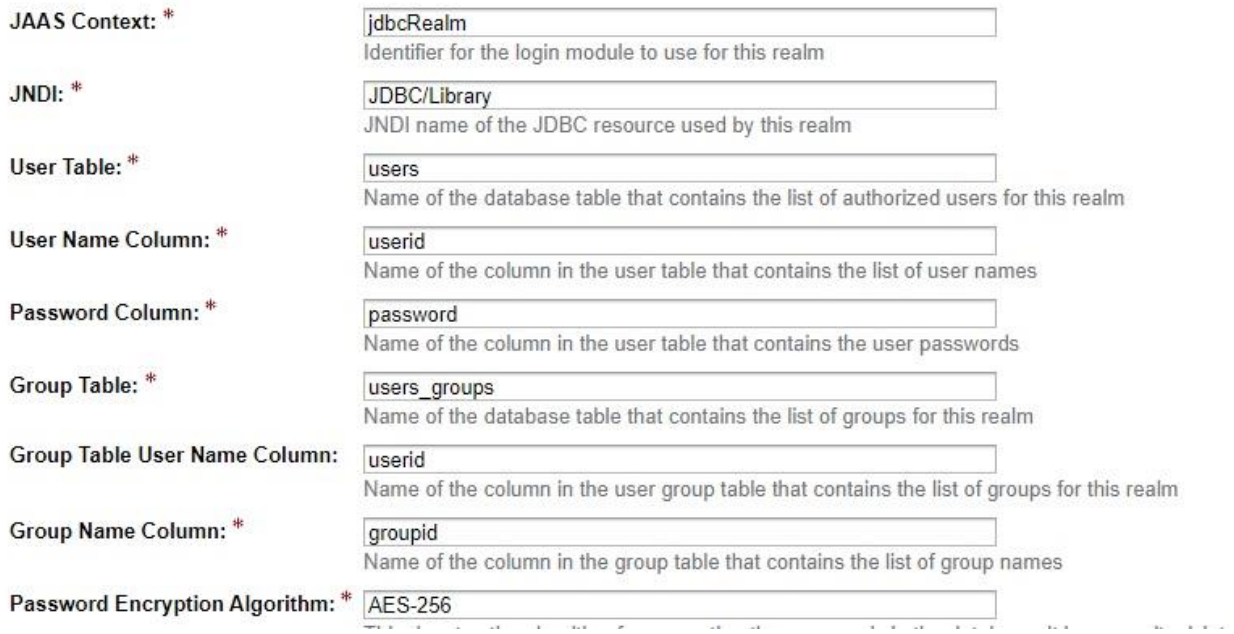

Рисунок 1.25 – Конфігурація jdbc реалму

- JAAS Context, JNDI – вказуємо довільну назву;
- User table – таблиця з користувачами в базі даних;
- User name column – колонка з унікальними ідентифікаторами користувачів;
- Password column – колонка для зберігання паролів;
- Group table – таблиця, яка зберігає групи користувачів (admins, users і т.д.);
- Group name column – колонка з унікальними ідентифікаторами груп користувачів;
- Password Encryption Algorithm – назва алгоритму шифрування паролів до облікових записів.