АНОТАЦІЯ

Магістерська робота на тему «Розробка програмного забезпечення адміністрування бібліотечних даних на основі використання системи Koha із вбудованим модулем каталогізації» Стадник Надії Іванівни. – Тернопільський національний технічний університет імені Івана Пулюя, Факультет комп'ютерно-інформаційних систем і програмної інженерії, Кафедра програмної інженерії, група СПм–62 // Тернопіль, 2019.

С. – 95, рис. – 20, табл. – 7, слайдів. – 10, додат. – 3, бібліогр. – 35.

Дипломна робота присвячена створенню програмного забезпечення адміністрування бібліотечних даних на основі використанням системи Koha з вбудованим модулем каталогізації.

Об’єктом дослідження є підвищення рівня швидкодії програмної системи для адміністрування бібліотечних даних.

Предметом дослідження: є програмна система керування даними, що використовує бази каталогів для удосконалення спеціалізованого програмного забезпечення.

Мета роботи: підвищити швидкодію програмного забезпечення, що використовується бібліотеками для адміністрування даних, використовуючи при проектуванні систему Koha з вбудованим методом каталогізації.

При створені програмного забезпечення використовувались такі технології розробки як об’єктно-орієнтоване програмування, середовище розробки Geany, для взаємодії з базами даних MySQL, веб-сервером вибрано Apache, для написання інтерфейсу мова XHTML.

Ключові слова: бібліотека, програмна система, АБІС, СУБД, Koha, автоматизація, адміністрування.

ABSTRACT

Master thesis «Development of software for administration of library data based on the use of Koha system with built-in cataloging module» by student Stadnyk Nadiia Ivanivna – Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Software engineering department, group SPm-62 // Ternopil, 2019.

Pages. – 95, pictures. – 20, tables. – 7, slides – 10, add. – 3, bibl.ref. – 35.

The diploma thesis is devoted to creation of software of administration of library data on the basis of use of Koha system with the built-in cataloging module.

The object of the study is to improve the performance of the software system for administering library data.

The subject of the study: is a data management software system that uses directory databases to improve specialized software.

Purpose: To increase the performance of software used by libraries to administer data using the Koha system with a built-in cataloging method when designing.

When the software was created, we used such development technologies as Object-Oriented Programming, Geany Development Environment, Apache was used to interact with MySQL databases, Web server was selected, and XHTML was written for the interface.

Keywords: library, software system, ABIS, DBMS, Koha, automation, administration.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення.

ПС – програмна система, комплекс програмного забезпечення.

ПК – персональний комп’ютер, робоча машина для розробки та виконання програм.

ООП – (Об’єктно-орієнтоване програмування) парадигма програмування, в якій основою є класи та об’єкти, які між собою взаємодіють.

RUP - Раціональний уніфікований процес (Rational Unified Process), який є ітеративним процесом розробки структури програмного забезпечення.

UML – (Unified Modeling Language) уніфікована мова графічного представлення та об’єктного моделювання в області розробки програмного забезпечення парадигми об’єктно-орієнтованого програмування.

ОП – охорона праці.

ВДТ – відео дисплейний термінал.

ДСанПіН - Державні санітарні правила і норми.

НПАОП - Нормативно-правовий акт з охорони праці.

ЗМІСТ

ВСТУП

Бібліотека займає велику роль у світі, забезпечує збереження, розповсюдження знань, та створення комфортних умов для навчання, спілкування та розвитку. Бібліотеки є важливими складовими будь-якого навчального закладу, який є центром викладацької та навчальної діяльності, де викладачі та студенти можуть досліджувати різні джерела інформації. В час розвитку інформаційних технологій для полегшення ведення обліку бібліотек, збільшення ефективності роботи працівників, підвищення якості обслуговування користувачів використовуються комп'ютери.

У епоху ІТ бібліотека кардинально змінилась відносно збору, організації та зберігання інформації. Також змінились і вимоги користувачів, які потребують актуальної, достовірної інформації швидко і в одному місці, що знаходиться постійно під рукою. Ця концепція дала поштовх для вирішення проблеми швидкого надання бібліотечних послух та інформації. Що наштовхнуло фахівців на ідею автоматизації бібліотек.

Автоматизацію бібліотеки можна визначити як застосування комп'ютерів для здійснення традиційних заходів з ведення бібліотеки, таких як придбання, обіг, каталогізація та контроль довідкових та серійних видань для автоматичної обробки даних.

Вперше над вирішенням цієї проблеми почали працювати ще в минулому столітті, так в 1999 році було створено автоматизовану бібліотечно-інформаційну систему Koha. Автоматизовані бібліотечно-інформаційні системи (АБІС) — системи планування ресурсів бібліотеки, які призначені для відстеження бібліотечних фондів, починаючи від замовлення та отримання до видачі відвідувачам бібліотек[1].

Бібліотеки та їх інформаційна діяльність розвивається в напрямку нового інформаційно-комунікаційного середовища. Зважаючи на ці обставини є важливим вибір програмного забезпечення, яке відповідає вимогам відносно основних функцій і завдань електронної бібліотеки. До

яких можна віднести керування і користування основними блоками до яких свою чергу відносяться електронний фонд і електронний каталог.

На даний момент створено безліч АБІС які орієнтовані на різні вимоги, проте найпопулярнішими залишаються системи з відкритим кодом. Програми з відкритим кодом є популярними через їх доступність, відносно недорогу ціну та досить великий функціонал. Проте більшість цих програм орієнтовані на операційні системи Linux і перехід програми під іншу операційну систему є дуже складним, а в деяких випадках недоцільним оскілки втрачається функціонал.

Програмна модернізація бібліотек є необхідною для сучасного розвиненого суспільства. Особливу увагу потрібно приділити вибору системи адміністрування даних. Проте перед використанням їх необхідно модернізувати, щоб вони відповідали всім вимогам, були зручними і зрозумілими у використанні.

Методи дослідження ґрунтуються на застосуванні аналізу програмних систем, об’єктно-орієнтованого проектування і методології функціонального моделювання.

За мету було поставлено удосконалення програмного забезпечення, що буде використовуватись бібліотеками для адміністрування даних, використовуючи при проектуванні систему Koha з вбудованим методом каталогізації. Програма є актуальною для невеликих бібліотек, до яких можна віднести шкільні та сільські, в яких відносно невелика технічна база.

Об’єктом дослідження є підвищення рівня швидкодії програмної системи для адміністрування бібліотечних даних.

Предметом дослідження є програмна система керування даними, що використовує бази каталогів для удосконалення спеціалізованого програмного забезпечення.

Наукова новизна отриманих результатів:

- отримала подальший розвиток спеціалізована програмна система автоматизації роботи бібліотеки яка призначена для підтримки бібліотечних технологічних процесів, що підвищує рівень швидкодії;
- запропоновано нову абстрактну модель системи та її архітектуру, вперше реалізовано і протестовано основні модулі системи, що дозволяє раціонально використовувати мережевий ресурс;
- удосконалено методологію адміністрування бібліотечних даних на основі використанням системи Koha з вбудованим модулем каталогізації, що дозволяє на практиці підвищити продуктивність програмної системи.

Представлена нижче дипломна робота описує процес та основні етапи дослідницької роботи та створення програмного забезпечення адміністрування бібліотечних даних на основі використання системи Koha із вбудованим модулем каталогізації.

1. РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

1.1. Аналіз вимог до предметної області

1.1.1. Аналіз предметної області

В рамках даної роботи виконується розробка системи для адміністрування бібліотечних даних в основу якої покладено систему Koha з вбудованим модулем каталогізації.

Об’єктом дослідження є підвищення рівня швидкодії програмної системи для адміністрування бібліотечних даних

Поняття електронної бібліотеки визначено не конкретно оскільки під цим поняттям розуміються колекції електронних документів. За сучасними підходами електронна бібліотека визначається як сукупність механізмів збору документів, а також пошуку та їх перегляду. Можна віднести також комп’ютерні мережі і певний набір сервісів, що направлені на вирішення завдань користувача.

При порівнянні електронних бібліотек з традиційними, електронні відзначаються такими перевагами, як:

- швидке отримання необхідної інформації в одному місці;
- збільшення можливостей пошуку та опрацювання отриманої інформації;
- отримання доступу до спільного використання інформації з унікальних джерел, для доступу до яких необхідно було раніше їхати в сховище, в якому воно зберігалось;
- можливість широкого доступу до фондів бібліотеки та архівів для користувачів з допомогою комп’ютерних мереж;
- надають можливість оновлень електронних версій документів для їх актуалізації даних;

– забезпечують цілодобовий доступ до ресурсів незалежно від місця знаходження;

– можливість відображення інформаційних матеріалів в різних форматах.

Автоматизована бібліотечна інформаційна система (АБІС) – це сукупність програмних модулів, що у поєднанні створюють складну інформаційну систему та призначенням яких є керування ресурсами бібліотеки[1].

В складі АБІС є такі модулі:

– модуль комплектування;

– модуль каталогізації та опрацювання фонду;

– модуль захисту;

– модуль обслуговування;

– інші модулі.

Модуль комплектування включає роботу яка відноситься до оформлення, документування, та скасування замовлень, обробкою нових надходжень, необхідними операціями аналізу фонду, підготовкою статистичної інформації.

Модуль каталогізації та опрацювання фонду – створює структуру фонду в бібліотеці. Головною перевагою автоматизації такого процесу надання можливості швидкого пошуку необхідної інформації, та можливість задання потрібного формату та впорядкувати отримані результати.

Модуль захисту – обов’язковий модуль, що надає право доступу користувача чи бібліотекаря до певної частини бази даних АБІС. Він має велику роль для супроводу програмної систем. Розподілена система доступу дає змогу запобігти несанкціонованому входженню в базу даних.

Модуль обслуговування користувачів – модуль, що призначений для розподілу та видачі/повернення літератури користувачам.

Додаткові модулі призначені розширити функціонал АБІС, доповнити новими функціями та можливостями для привернення уваги нових користувачів, а для існуючих розширюють ряд послуг та покращують ефективність роботи.

На рис. 1.1 представлено модель АБІС із обов'язковим мінімальним набором функцій, що вона повинна реалізовувати.

Рис. 1.1 – Структура АБІС з мінімальним набором функцій

Головною структурною одиницею АБІС вважається система автоматизації бібліотек (САБ), яка на думку конкретних бібліотек, відповідає за можливість користувачеві отримати комфортний доступ до фондів. САБ надає доступ користувачам до електронного каталогу (ЕК), та забезпечує функції його ведення, поповнення і лінгвістичні засоби каталогу. Підтримує бібліотечні стандарти і формати.

У випадку користування звичного АБІС користувач працює з перевіреним інформаційним та лінгвістичним забезпеченням (друкованими і електронними системами класифікацій, й ін.).

Досліджуючи предметну область було розглянуто і проаналізовано всі переваги і недоліки аналогічних програмних систем.

Однією з розглянутих при аналізі систем є LibraryWorld. Це веб-система для управління бібліотекою, що дозволяє створювати власний сайт, а також колекцію додатків для допомоги керування каталогом[2]. Дана система орієнтована на малі і середні бібліотеки, має простий та зручний для користувача інтерфейс, що дозволяє швидко і без особливих зусиль з ним працювати. LibraryWorld включає основні модулі до яких належать каталоги, тиражі, інвентаризація, послідовне відстеження, звіти по роботі а також додатки для інтернет-каталогу та мобільні програми.

Легка в експлуатації. Самостійна каталогізація в системі дозволяє легко класифікувати тексти за номерами та незалежними тегами у розділі коментарів.

До недоліків можна віднести недосконалу функцію пошуку, оскільки не приймає обмежених пошукових термінів або номерів ISBN. Ще одним недоліком можна виділити досить високу ціну програми, а також складність генерованого програмою звіту, що для відносно невеликого Інтернет – сховища повинен бути на порядок простішим.

ABCD (Автоматизація бібліотек та центрів документації) - це повна інтегрована система автоматизації бібліотек, заснована на ISIS- технології, як основи бази даних[3]. Вона охоплює всі основні функції АБІС, але додає модуль для легкого створення веб-сайту бібліотеки з інтегрованим мета-пошуком. Як особливість даної програми можна зазначити, що ABCD дуже гнучка та універсальна для використання з нестандартною структурою бази даних або не бібліографічних додатків.

Основні частини ABCD показані з їх ієрархічними взаємозв'язками на другому рівні на рис. 1.2.

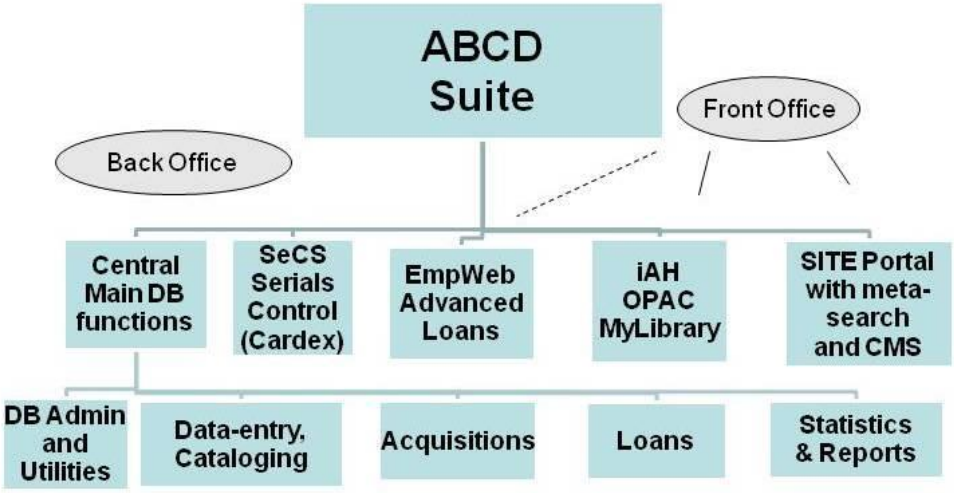

Рис. 1.2 - Основні частини інтегрованої системи ABCD

Програмне забезпечення працює повністю на веб-основі, тому ним можна користуватися та керувати ним із будь-якого поточного веб-браузера. Всі основні функції управління бібліотекою інтегровані за допомогою одного інтерфейсу та баз даних. Бібліографічні записи можна імпортувати із зовнішніх бібліотечних каталогів / серверів через засоби Z39.50. OPAC - простий пошук у Google, а також розширений пошук із булевими операторами, усічення, та обмеження поля для всіх типів баз даних, локально створених або зовнішніх.

Співробітники бібліотеки можуть визначати, копіювати або редагувати будь-яку нову структуру бази даних за допомогою існуючих програм ISIS. Доступна багатьма мовами, такими як англійська, французька, іспанська, португальська.

Evergreen - це ще один варіант при дослідженні АБІС з відкритим кодом. Evergreen був розроблений в США. Розробка розпочалася в червні 2004 року коли було прийнято рішення про створення власної системи автоматизації бібліотек. Інтерфейс бібліотеки представлено на рис. 1.3. Основною метою даної розробки було створення системи, що буде підходити спеціально під встановленні вимоги, але витрати на яку будуть менші за плату, що сплачується за альтернативні системи.

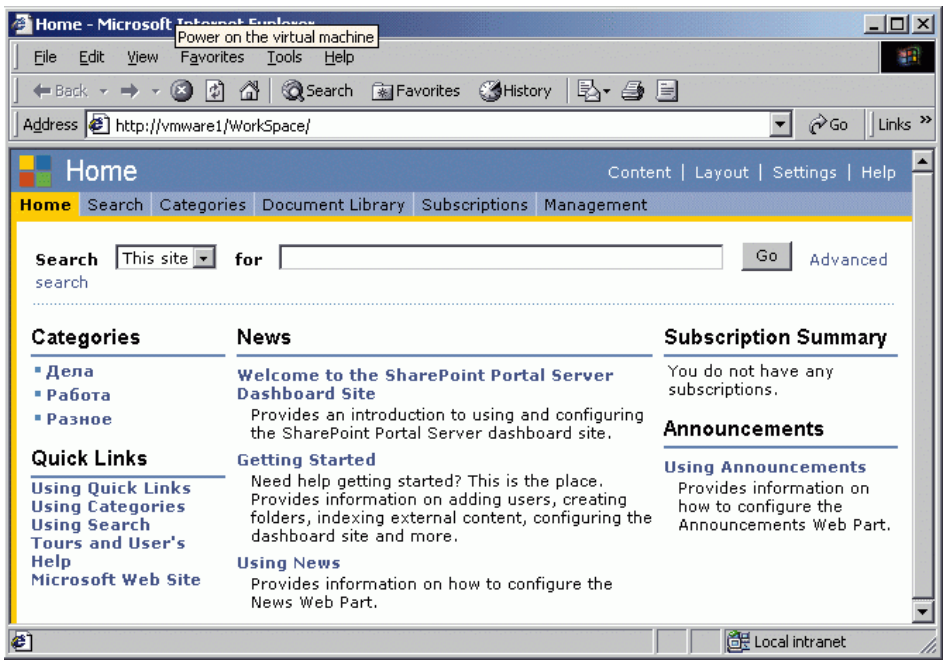

Рис. 1.3 - Інтерфейс бібліотеки Evergreen

Пріоритети розвитку Evergreen полягають у тому, щоб вона була стабільною, надійною, гнучкою, безпечною та зручною для користувачів[4].

Evergreen дозволяє відстежувати розташування будь-якого документу, що належить до її фонду. Недоліком є складна архітектура системи, що сильно ускладнює внесення будь-яких змін у її налаштування.

Koha - це інтегрована бібліотечна система з відкритим кодом, яка використовується у всьому світі державними, шкільними та спеціальними бібліотеками. Koha була створена в 1999 році компанією Katipo Communications для тресту бібліотек Horowhenua у Новій Зеландії, а перша установка розпочалася в січні 2000 року.

Koha - це веб-АБІС, що має базу даних SQL (надано перевагу MySQL) із каталогізацією даних, що зберігаються в MARC та доступних через Z39.50 або SRU. Інтерфейс системи дуже добре налаштовується і адаптується, а також була перекладена багатьма мовами[5]. Коха має більшість особливостей, які можна було б очікувати в АБІС, включаючи:

– Різні засоби Web 2.0, такі як тег, коментарі, спільний доступ до соціальних мереж та RSS-канали;

– Об'єкт каталогу об'єднання;
– Настроюваний пошук;
– Інтернет-тираж;
– Друк штрих-коду.

Серверна частина АБІС створена на мові Perl і для коректної роботи системи вимагає:

– Perl;
– Веб-сервер Apache;
– Сервер MySQL 5 ;
– Модулі Perl для деяких функцій.

Клієнтська сторона інтерфейсу, до якої відноситься і електронний каталог, написана мовою програмування XHTML з використанням графіки PNG, застосуванням CSS за рахунок чого повинна правильно відображатись у кожному браузері який сумісний з CSS 2.0. Для інтерфейсу бібліотекаря також необхідне виконання вище перерахованих вимог, але на відміну від інтерфейсу клієнта він додатково вимагає забезпечення коректної роботи браузера з Javascript.

1.1.2. Постановка завдання

Розробка любого програмного забезпечення спрямована на вирішення проблем або автоматизації певних задач, на виконання яких затрачався великий ресурс часу.

Програма може представляти собою ряд інструкцій, що описують зміни стану машини, але вона також може описувати обчислення і по-іншому.

Програмне забезпечення – це загальна інформація у вигляді набору інструкцій, реалізованих інтерфейсів і інтегрованих даних, призначених для комп'ютера для досягнення своїх цілей. Метою програмного забезпечення є

обробка даних у межах, визначених творцем. Прикладне програмне забезпечення має прямий контакт з користувачем та надає послуги цьому користувачеві через додаток, з технічної точки зору це програмне забезпечення, яке використовує послуги, для операційної системи

Цифрові бібліотеки - це бібліотеки, в яких зберігаються цифрові ресурси. Вони визначаються як організація, а не служба, яка надає доступ до цифрових творів, несуть відповідальність за збереження майбутнього доступу до матеріалів та надають ці елементи легко та доступно[6]. Визначення цифрової бібліотеки означає, що цифрова бібліотека використовує різноманітне програмне забезпечення, мережеві технології та стандарти для полегшення доступу до цифрового контенту та даних до визначеної спільноти користувачів. На доступ до цифрових бібліотек може впливати кілька факторів, окремо, або разом. Найбільш поширеними факторами, що впливають на доступ, є:

- зміст бібліотеки;
- характеристики та інформаційні потреби цільових користувачів;
- цифровий інтерфейс бібліотеки;
- цілі та завдання організаційної структури бібліотеки;
- стандарти та правила, що регулюють використання бібліотеки.

Доступ залежить від здатності користувачів виявити та отримати документи, які їх цікавлять, та які вони потребують, що, в свою чергу, є питанням збереження. Цифрові об'єкти не можна зберігати пасивно, вони повинні підтверджуватись друкованим виданням, щоб забезпечити довіру та цілісність цифрових об'єктів[7].

Одне з головних завдань для бібліотекарів цифрових технологій - необхідність забезпечення довгострокового доступу до своїх ресурсів. Для цього є необхідність у слідкуванні за: втратою ресурсів та застаріванням формату. У випадку відмови певний цифровий елемент є непридатним через певну помилку чи проблему. Застарівання формату - це коли цифровий формат витіснений новішими технологіями, і тому елементи в старому

форматі не читаються і не можуть бути використані. Усунення несправностей - це реактивний процес, оскільки щось робиться лише тоді, коли постає проблема. Бібліотека надає вам доступ до інтернет-ресурсів для вивчення предметів. Також допомагає розвинути навички, які необхідні під час навчання, на роботі чи в повсякденному житті.

Проаналізувавши предметну область та визначивши всі основні проблеми можна сформувати мету і завдання до виконання проектного рішення.

Мету можна визначити як удосконалення програмного забезпечення, що буде використовуватись бібліотеками для адміністрування даних, використовуючи при проектуванні систему Koha з вбудованим методом каталогізації.

Під час проектування системи управління бібліотекою було встановлено наступний набір вимог:

– Будь-який користувач бібліотеки повинен мати можливість шукати книги за назвою, автором, категорією предмета, а також за датою публікації.

– Кожна книга матиме унікальний ідентифікаційний номер та інші деталі, включаючи номер стелажа, який допоможе фізично знайти книгу.

– Книги може бути більше, ніж один примірник, і користувачі бібліотеки повинні мати можливість перевірити та зарезервувати будь-яку копію.

– Система повинна мати можливість отримувати інформацію, наприклад, хто взяв конкретну книгу або які книги перевіряв конкретний користувач бібліотеки.

– Повинно бути максимальне обмеження щодо того, скільки книг може взяти користувач.

– Повинен бути максимальний ліміт на скільки днів читач може зберігати книгу.

– Система повинна мати можливість стягувати штрафи за книги, повернені після закінчення строку.

– Користувачі повинні мати можливість резервувати книги, які наразі недоступні.

– Система повинна мати можливість надсилати сповіщення щоразу, коли зарезервовані книги стають доступними, а також коли книга не повертається у встановлений термін.

– Кожна книга та картка читача матимуть унікальний штрих-код. Система зможе читати штрих-коди з книг та бібліотечних карт читачів.

1.1.3. Пошук актантів та варіантів використання

Пошук основних акторів і варіантів використання системи виконувався на основі переліку поставлених завдань.

Місцем застосування системи передбачається бібліотека школи або села. Відповідно до вимог це система для автоматизованого ведення даних про роботу бібліотеки, що заснована на певній моделі.

Одним з головних актантів в системі можна виділити бібліотекаря, який працює з більшістю функцій системи і для покращення роботи якого система розробляється. Також можна виділити як актора читача, який користується системою для пошуку необхідної інформації.

Для бібліотекаря в програмі виділяється такий перелік функцій і можливостей, як:

– створення облікових записів для користувачів системи;

– ведення бібліографічних записів системи та їх редагуванні і перегляд за необхідності;

– керування каталогом бібліотеки, з можливістю внесення змін до нього;

– перегляд і керування обігом бібліотеки (видача і отримання повернених книг);

– пошук інформації про користувача та перегляд інформації про його дії в системі;

– генерування необхідних звітів для можливості аналізу роботи користувачів з програмою.

Схематично дані варіантів використання для актанта бібліотекар зображено на рис 1.4.

Рис 1.4 - Діаграма варіантів використання системи для актанта «Бібліотекар»

Для актанта Відвідувач системна програма надає такі можливості для роботи з нею, як:

– Авторизація в системі;

– Пошук необхідної інформації в каталозі;

– Перегляд вибраних джерел;

– Оформлення замовлення на отримання книги;

– Пошук інформації в Інтернеті;

– Створення власних збірок джерел;

– Перегляд історії пошуку та інформації про видані і повернені книги;

– Повернення книг в бібліотеку.

Наглядне зображення варіантів використання показано на рис. 1.5.

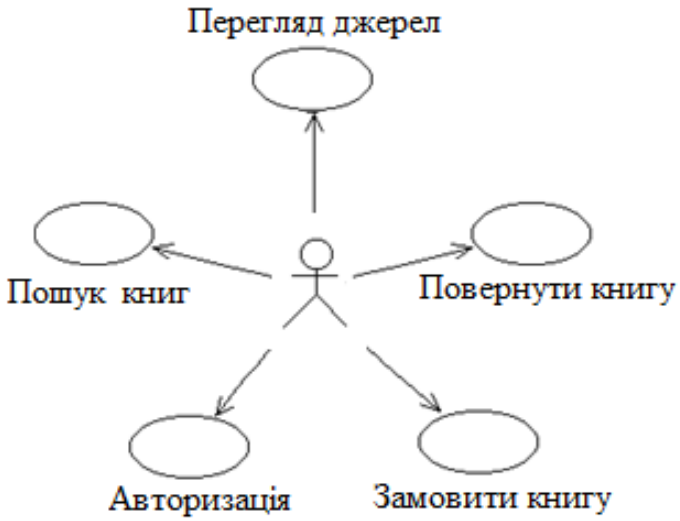

Рис 1.5 - Діаграма варіантів використання системи для актанта «Відвідувач»

Варіанти використання описують певні можливі послідовності дій користувача в системі, на яких побудована основна логіка системи. Тому деякі варіанти використання мають більше значення при роботі з системою і їх потрібно розглядати більш детально. Також варто зауважити, що деякі варіанти вимагають певного рівня доступу до даних і функціоналу системи.

Залежно від прав доступу для користувача і для бібліотекаря доступ до функціоналу системи відрізняється, адже для користувача він є обмеженим порівняно з бібліотекарем.

Одними з головних прецедентів які будуть описані більш детально є:

- Авторизація в системі;
- Створення облікових записів користувачів системи.

При вході в систему, користувач вводить логін і пароль за якими система його розпізнає і надає можливість подальшої роботи з системою. Якщо дані при вході було введено неправильно користувач отримує повідомлення про це, з проханням перевірити правильність логіну і пароля, та спробувати авторизуватись ще раз. Для користувача також передбачена можливість зміни паролю з використанням електронної пошити або телефону до якого прив'язаний обліковий запис. Детальний опис варіанту використання «Авторизація в системі» подано у таблиці 1.1.

Таблиця 1.1 – Опис варіанту використання «Авторизація в системі»

Прецедент: Авторизація в системі
Короткий опис: Відвідувач повинен авторизуватись в системі для розширення можливостей роботи з нею та отримання можливості перегляду своєї історії пошуку.
Головні актори: Відвідувач
Передумови: Користувач є зареєстрованим в системі
Основний потік: 1. Користувач заходить на сайт бібліотеки 2. В головному вікні інтерфейсу знаходиться форма для входу в систему. 3. Користувач заповнює поля для вводу логіну і пароля. 4. Система виконує авторизацію. 5. Система вносить дані в базу, про вхід користувача. 6. Відкривається інтерфейс користувача для роботи в системі.
Постумови: За умови вдалого завершення прецеденту в базу даних вносяться відмітки про вхід користувачі і фіксується час його входження в систему.
Альтернативні потоки: **1.1 Неправильно введений логін або пароль** Якщо виконуючи п.3 користувач вводить неправильне значення логіну або паролю система видає повідомлення, що дані були введені некоректно, просить перевірити і повторити спробу.

Відвідувач має можливість створити свій обліковий запис в системі за умови наявності читацького білету. Детальний опис створення облікового запису приведений у таблиці 1.2.

Таблиця 1.2 – Опис варіанту використання «Створення облікових записів для користувачів системи»

Прецедент: Створення облікових записів користувачів системи
Короткий опис: Користувач створює свій обліковий запис в системі для кращої роботи та розширення доступних можливостей
Головні актори: Відвідувач
Передумови: Відвідувач в системі не зареєстровани, але має № читацького білету
Основний потік: 1. Користувач заходить на сайт бібліотеки. 2. В головному вікні інтерфейсу біля форми для входу в систему знаходиться кнопка для реєстрації відвідувача. 3. Система переходить на сторінку реєстрації. 4. Користувач вводить данні про себе, заповнюючи обов'язкові поля. 5. Система зберігає данні і створює обліковий запис 6. Відкривається інтерфейс користувача для роботи в системі.
Постумови: За умови вдалого завершення прецеденту в базу даних вносяться дані про користувача, а також дані про його вхід в систему.
Альтернативні потоки: **1.2 Не заповнено всі обов'язкові поля** Якщо виконуючи п.4 користувач не вводить інформацію у всі обов'язкові поля його повертає назад до форми з повідомленням про необхідність їх заповнити.

Для створення облікового запису користувач на головному вікні програми натискає відповідну кнопку і переходить до форми створення облікового запису. Вводить необхідні дані про себе, створює пароль для можливості подальшої авторизації системі і заповнює всі обов'язкові поля.

Натиснувши кнопку зберегти відправляє запит на створення облікового запису, якщо система не знаходить помилок при заповнені форми обліковий запис створюється і користувачу відкривається вікно його облікового запису. При неправильному введені даних або не заповнені всіх обов'язкових полів форми, з'являється повідомлення про це і система повертається до форми заповнення даних з повідомленням про прохання перевірити заповнення всіх обов'язкових полів. Після перевірки користувач має можливість повторно спробувати зберегти дані.

1.2. Проектування системи

1.2.1. Вибір процесу розробки

Процес розробки програмного забезпечення являє собою процес поділу розробки ПЗ на окремі фази, щоб поліпшити дизайн, управління продуктом і управління проектами.

Методика може включати попереднє визначення конкретних результатів та артефактів, які створюються та доповнюються командою проекту для розробки або підтримки програми.

Основна ідея полягала в тому, щоб продовжувати розвиток інформаційних систем дуже обдумано, структуровано та методично, вимагаючи, щоб кожен етап життєвого циклу, від зародження ідеї до доставки кінцевої системи, здійснювався жорстко і послідовно в контексті застосовуваних рамок[8]. Основною метою є розробка широкомасштабних функціональних бізнес-систем.

Будь-яку методологію можна охарактеризувати за такими ознаками,як:

– Основні принципи, що впливають на ефективність. Зазвичай сформульовані досить коротко і легкі для розуміння;

– Встановленою відповідністю між множиною моделей та множиною методів, за допомогою яких реалізується методологія;

– Концепціями, за допомогою яких, методи визначаються максимально точно.

Як процес розробки для створюваного програмного забезпечення було обрано RUP. Раціональний уніфікований процес (Rational Unified Process), який є ітеративним процесом розробки структури програмного забезпечення.

RUP базується на наборі блоків, а також елементів вмісту, описуючи те, що має бути створено, з покроковим поясненням та зазначенням необхідних навиків, що описують, як потрібно досягти конкретних цілей розвитку. До основних блоків або змістових елементів можна віднести:

Ролі - які визначають сукупність пов’язаних функцій, можливостей та обов'язків.

Результат роботи – який є результатом виконання завдання, включаючи всі документи та моделі, виготовлені під час роботи в процесі.

Завдання – позначає одиницю роботи, призначену для Ролі, яка забезпечує вагомий результат.

Метою даної методології є надання гарантій якісного програмного забезпечення, що його розробка завершиться у вказані терміни, не вийде за рамки бюджету, а також буде відповідати потребам користувача.

RUP поділяє життєвий цикл проекту на чотири фази, які дозволяють представити процес на високому рівні. Кожна фаза має одну ключову мету і віху в кінці, яка позначає досягнуту мету[9].

Фаза початку.

Основна мета - адекватно розробити систему, як основу для перевірки початкових витрат і бюджетів. На цій фазі встановлюється економічне обгрунтування, що включає бізнес-контекст, фактори та фінансовий прогноз. Для доповнення економічного обгрунтування формується основна модель використання, план проекту, початкова оцінка ризику та опис проекту. Результати завершення фази перевіряються за критеріями:

– узгодження зацікавлених сторін стосовно визначення обсягу та оцінки витрат;

– основні вимоги, для визначення базового функціоналу продукту.

– достовірність оцінок витрат, пріоритетів, ризиків та процесу розвитку.

– встановлення базового плану, з яким будуть порівнювати фактичні видатки та заплановані витрати.

Після проходження даного етапу життєвого циклу приймається рішення щодо подальшої розробки програми або відправлення проекту на доопрацювання.

Етап розробки.

Метою якого є зменшення основних ризиків, що були визначені шляхом аналізу, до закінчення цієї фази. На етапі розробки проект формується, та проводиться аналіз проблемної області після чого архітектура проекту набуває своєї основної форми. Як результат етапу є:

– Спроектована і описана архітектура програмного забезпечення

– Побудовано модель варіантів використання

– Сформовано економічне обґрунтування та план розвитку загального проекту

– Створено прототипи, та мінімізовані всі технічні ризики, що було виявлено.

Якщо проект не може пройти цю віху, ще є час для його скасування чи перероблення. Однак після закінчення цієї фази проект переходить до операції з високим рівнем ризику, де зміни набагато складніші та затратні.

Етап побудови.

Основна мета - побудова програмної системи. У цій фазі основна увага приділяється розробці компонентів та інших особливостей системи. Це фаза, коли відбувається основна частина кодування. Для великих проектів є

характерним розроблення декількох ітерацій, намагаючись розділити варіанти використання на керовані сегменти, що дають наочні прототипи.

Фаза переходу.

Основна мета – перехід системи від розробки до впровадження. Завданням цього етапу є навчання користувачів та обслуговуючого персоналу роботі на створеному ПЗ, тестування системи для оцінки її відповідності вимогам користувачів, а також на відповідність рівня якості, зазначеному на фазі початку.

Після проходження даної фази проект приходить до впровадження і закінчення розробки ПЗ.

1.2.2. Моделювання архітектури системи

Архітектура програмного забезпечення відноситься до фундаментальних структур програмної системи та дисципліни створення таких структур і систем. Кожна структура містить програмні елементи, а також відносини між ними, та властивості як елементів, так і зв’язків[10]. Архітектура функціонує як концепція системи та проекту, що розробляється, визначаючи завдання, необхідні для виконання проектними командами.

Архітектура програмного забезпечення полягає у прийнятті фундаментальних структурних рішень, а її вибір залежить від структурних елементів та розроблюваних можливостей майбутнього програмного забезпечення.

Документування архітектури полегшує спілкування між зацікавленими сторонами, фіксує ранні рішення щодо дизайну високого рівня та дозволяє повторно використовувати компоненти дизайну між проектами.

За допомогою архітектури програми можна побачити:

– зацікавлені сторони системи, такі як: власники, користувачі та оператори, що мають свої задачі відносно системи. Врівноваження цих задач та демонстрація їх вирішення є частиною проектування системи[11].

– розділення задач для зменшення складності систем. Задачі розділяють шляхом моделювання та опису архітектури відносно конкретних можливостей, що виконують певний функціонал.

– оцінка якості: класичні підходи до проектування програмного забезпечення визначалися необхідною функціональністю та потоком даних через систему, що доводить тісний зв'язок архітектури програмної системи з характеристиками її якості.

– повторювані стилі: через повторюваність архітектури при вирішенні задач, було створено стилі архітектури.

– концептуальна цілісність: визначення загального бачення роботи програмної системи, що вона повинна робити і вказівок як це робити. Тобто показує наскільки зміни відповідають початковому баченню системи.

Архітектурою програмного забезпечення є інтелектуально зрозуміла абстракція складної системи, що надає ряд переваг:

– дає можливість перевірити, чи майбутня програмна система задовольняє потреби замовника, не будуючи її фактично, представляє значну економію коштів та зменшення ризиків.

– забезпечує використання окремих архітектурних стратегій та рішень, які можуть бути повторно застосовані в декількох системах, зацікавлені сторони яких вимагають аналогічних атрибутів якості або функціональності.

– підтримує ранні проектні рішення, які впливають на розвиток, розгортання та експлуатацію системи[10].

– архітектура програмного забезпечення допомагає зменшити ризики та ймовірність виходу системи з ладу.

– дозволяє зменшити витрати, оскільки є засобом управління ризиками та витратами в складних проектах.

При розробці програмного продукту велику роль у ефективності, зручності, безпеці та експлуатації покладають на правильний вибір саме архітектури програмного забезпечення. Для розроблюваної системи було вибрано трирівневу клієнт-серверну архітектуру. Прикладом даної архітектури є Koha, що організована в три шари, кожен з яких складається з ряду компонентів (рис. 1.6):

– рівень зберігання даних

– бізнес-логічний рівень

– рівень додатків.

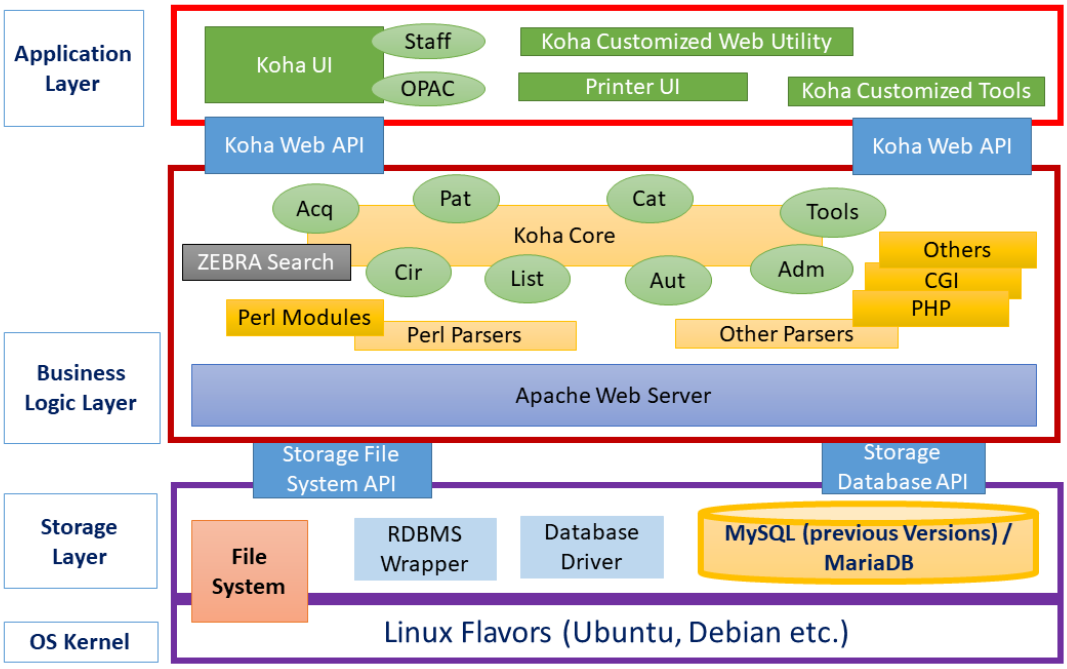

Рис. 1.6 – Схематичне зображення архітектури Koha

Шар зберігання: відповідає за фізичне зберігання бібліографічних даних або будь-якого файлу, завантаженого в систему. Він складається з драйверів баз даних, обгортки RDBMS та API файлової системи.

Бізнес-логічний шар: стосується бізнес-моделі Koha. Ядро koha разом з різними модулями Придбання, Каталог, Тираж, Меценат, Списки, Адміністрація, Інструменти, Повноваження, Серіали, Звіти тощо.

Рівень програми: містить компоненти, які спілкуються із зовнішнім світом. Два інтерфейси Koha - це найважливіший клієнтський персонал koha та koha opac.

Оскільки koha є програмним забезпеченням з відкритим кодом, воно також є відкритим для інтеграції інших індивідуальних інструментів у програму. Як, наприклад, PHP-код може використовуватися для написання користувацьких утиліт.

Кожен окремий шар викликає лише шар під ним; прикладний рівень може не використовувати шару зберігання безпосередньо, швидше компоненти шару програми використовують компоненти шарів бізнес-логіки, а для цього він використовує API рівня сховища.

Посилаючись на API, модулі написані з використанням мови скриптів perl, а архітектура залишається відкритою для реалізації іншої мови сценаріїв для розробки та розробки спеціалізованих інструментів чи утиліт.

На сьогодні вихідний код організований для дотримання тришарової архітектури, тому це спрощує реалізацію програмного забезпечення.

1.2.3. Абстрактна модель системи

Під час виконання проектування системи було обрано об’єктно-орієнтований підхід розробки, для якого необхідним є проведення аналізу класів і сутностей системи. Мовою для розробки інтерфейсу було обрано мову XHTML.

Опираючись на прийняті рішення та проведений аналіз було реалізовано програму з застосуванням класів, їх наслідування, а також інтерфейсів та об’єктів.

Для візуального представлення системи разом з її основними дійовими особами, ролями, діями, артефактами чи класами використовуються UML діаграми, які є схемами, що засновані на уніфікованій мові моделювання (UML). Діаграми UML створюються для кращого розуміння системи і можливості вчасно вносити зміни, підтримання та документування інформації про систему.

Складні програми потребують співпраці та планування з декількох команд, а отже, потрібен чіткий і стислий спосіб спілкування між ними. UML є важливим для спілкування замовниками відносно основних вимог, функціональних можливостей та процесами системи.

UML пов'язаний з об'єктно-орієнтованим дизайном та аналізом[12]. Використовує елементи та формує асоціації між ними для формування діаграм. Діаграми в UML можна широко класифікувати як:

– структурні - Захоплення статичних аспектів або структури системи. До них належать: діаграми компонентів, діаграми об'єктів, діаграми класів та діаграми розгортання.

– поведінкові - фіксують динамічні аспекти або поведінку системи. До яких можна віднести діаграми: випадків, стану, діяльності та взаємодії.

На рисунку 1.7 зображено діаграму побудови зв’язків між модулями системи. Основна увага на цій діаграмі приділяється відображенню обміну інформації між об’єктами. Побудова діаграми подібна до діаграми послідовностей, але немає хронологічного підходу зверху вниз, і не є вертикально структурованою.

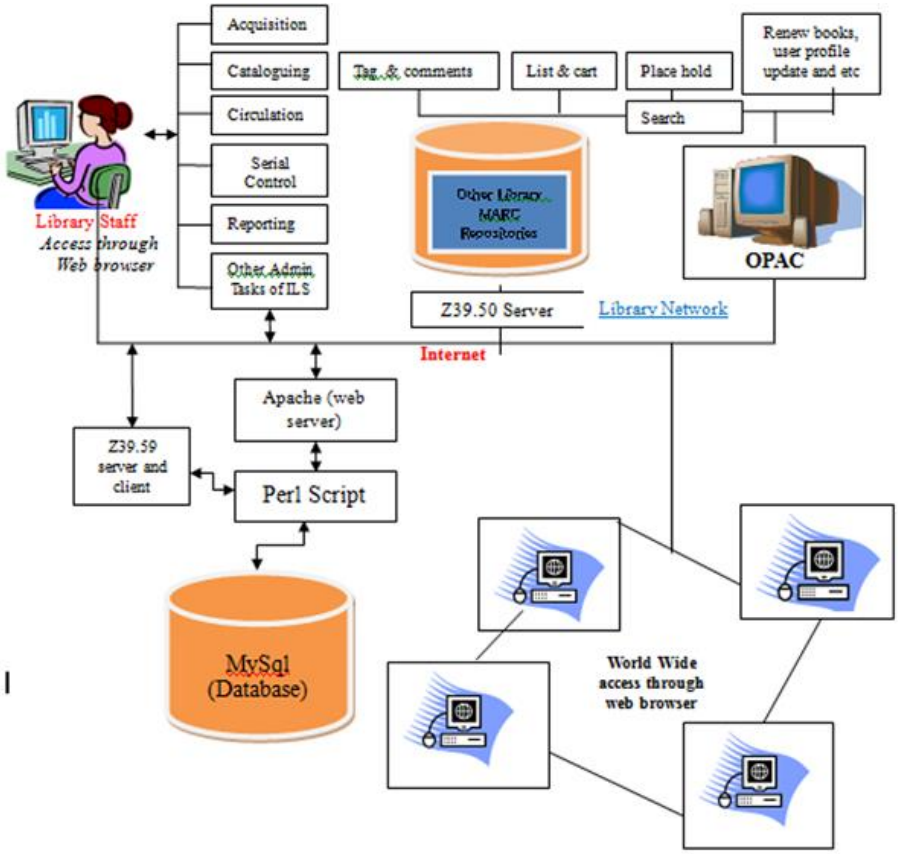

Рис.1.7 – Діаграма зв’язків між модулями програми

Діаграма активності є одною з найважливіших UML діаграм для моделювання бізнес-процесів. У розробці програмного забезпечення зазвичай використовується для опису потоку різних видів діяльності та дій. Дії можуть бути як послідовними, так і паралельними. А також описують об'єкти, які використовуються, споживаються або виробляються в результаті діяльності, і взаємозв'язок між різними видами діяльності. Діаграму загальної активності в програмі дивитись на рис. 1.8.

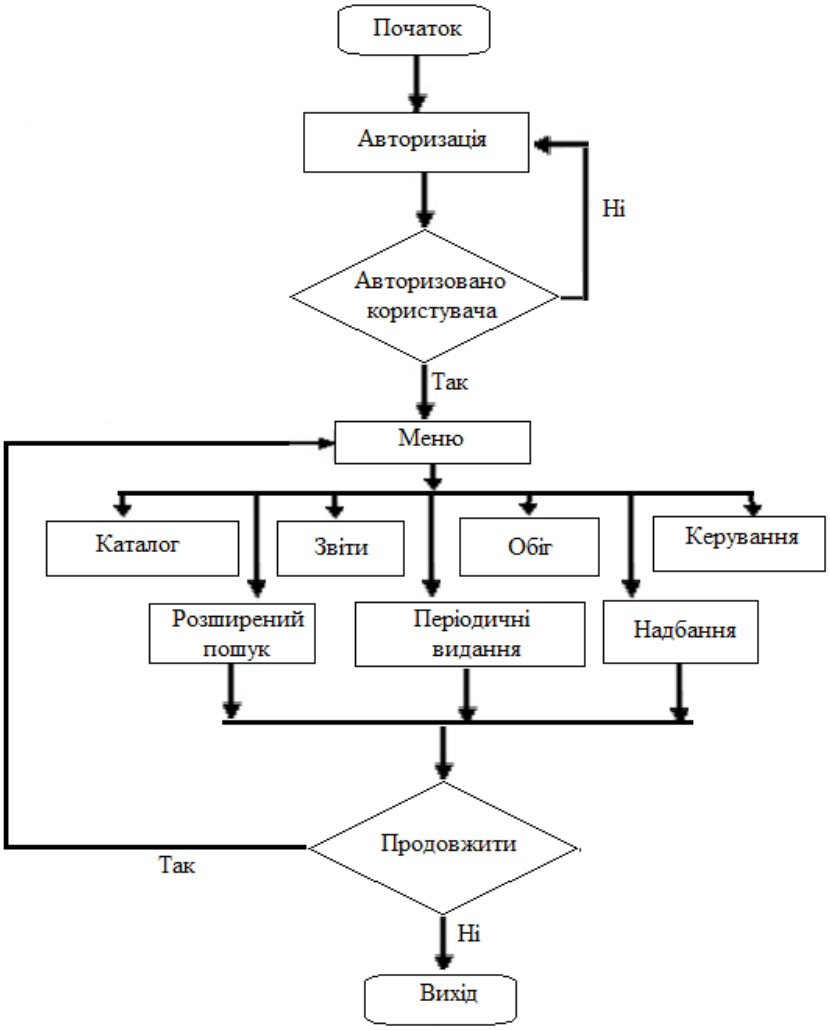

Рис.1.8 – Діаграма активності програми

На діаграмі представлено логічну схему активності в розроблюваній системі, яка має важливе значення в моделюванні бізнес процесів. На ній присутні точка входу в систему, з якої розпочинається виконання процесу ідентифікації користувача в системі. На схемі також присутнє позначення виходу з системи, що наступає в разі невдачі при одному з ключових кроків, або при успішному завершенні поставлених задач.

Також створена діаграма активності для перевірки книги (рис. 1.9), на можливість взяти її з бібліотеки.

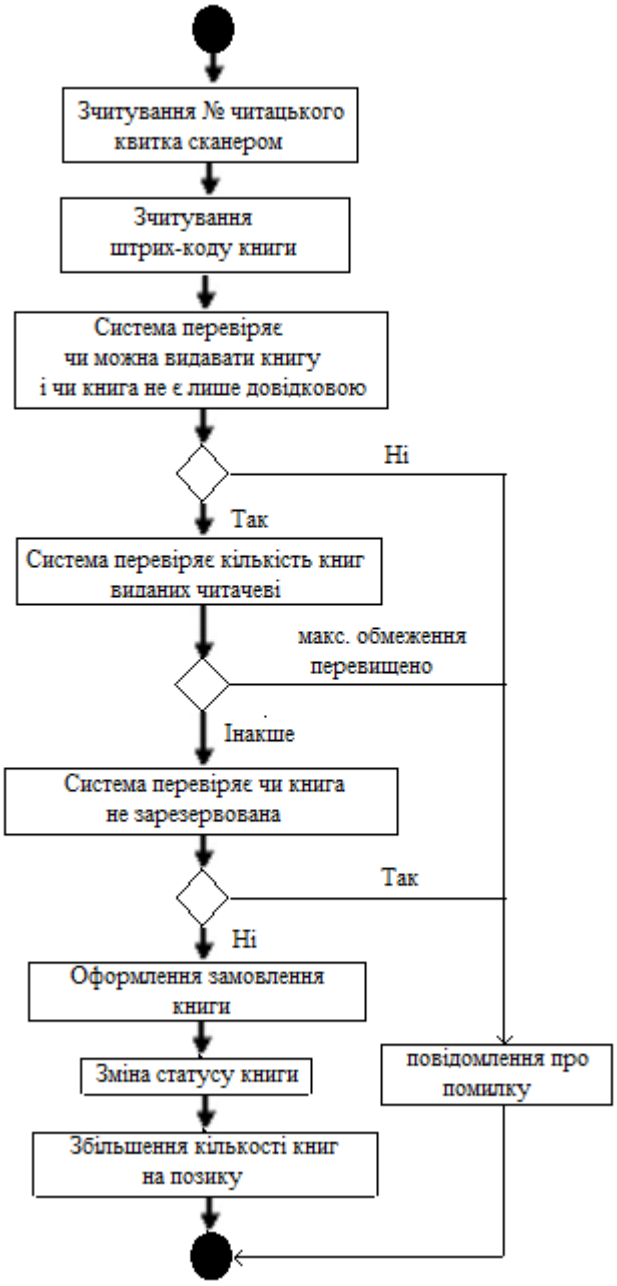

Рис. 1.9 - Діаграма активності для перевірки книги.

Перевірити книгу може будь який користувач системи, виконавши перераховані на діаграмі кроки. Основними критеріями перевірки книги є визначення типу книги, перевірка чи є книга заздалегідь зарезервованою іншим користувачем. Також здійснюється перевірка кількості вже виданих користувачеві книг. У разі перевищення максимального обмеження виданих книг появляється повідомлення про помилку і система виходить з форми. Якщо обмеження не перевищено то система переходить до подальшого

оформлення замовлення з подальшою зміною статусу книги та додаванням книги до списку запозичених користувачем книг.

Діаграма активності для повернення книги користувачем (рис. 1.10.), та перевірки чи вчасно повернена книга.

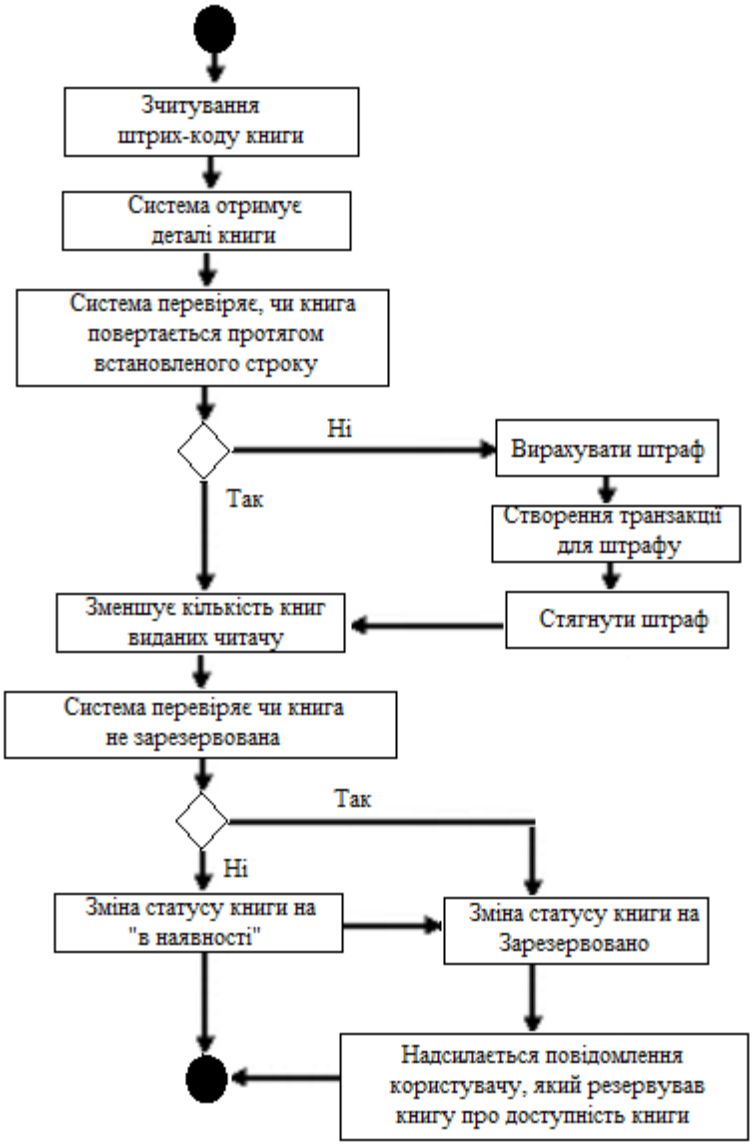

Рис. 1.10 - Діаграма активності для повернення книги.

При повернені книгу в систему користувач зчитує код книги, щоб система могла отримати детальну інформацію про неї. Також проводиться перевірка на те чи книга повертається вчасно, У разі закінчення терміну повернення користувачу нараховується штраф, який користувач повинен

оплатити. Тоді система видаляє книгу з списку виданих користувачу і зменшує їх кількість на одну. Далі слідує перевірка чи інші користувачі не резервували цю книгу. Якщо книга була зарезервована її статус отримує значення «зарезервовано» і користувачу, що стоїть першим в списку хто резервував цю книгу, надсилається повідомлення про те, що книга у вільному доступі. Якщо книга не була зарезервована, то їй присвоюється статус «у вільному доступі».

Діаграма послідовності показує взаємодію об'єктів, розташованих у часовій послідовності. Вона зображує об'єкти та класи, що беруть участь у сценарії, та послідовність повідомлень, що обмінюються між об'єктами, необхідні для виконання функціональності сценарію. Діаграми послідовності зазвичай пов'язані з реалізацією випадку використання в логічному представленні системи, що розробляється. Діаграми послідовності іноді називають діаграмами подій або сценаріями подій.

Діаграма послідовностей показує, як паралельні вертикальні лінії різні процеси або об'єкти, що живуть одночасно, і, як горизонтальні стрілки, повідомлення, що обмінюються між ними, у тому порядку, в якому вони відбуваються. Це дозволяє конкретизувати прості сценарії виконання графічно.

На рисунку 1.11 представлена діаграма послідовності для варіанту використання «Пошук книги» для відвідувача.

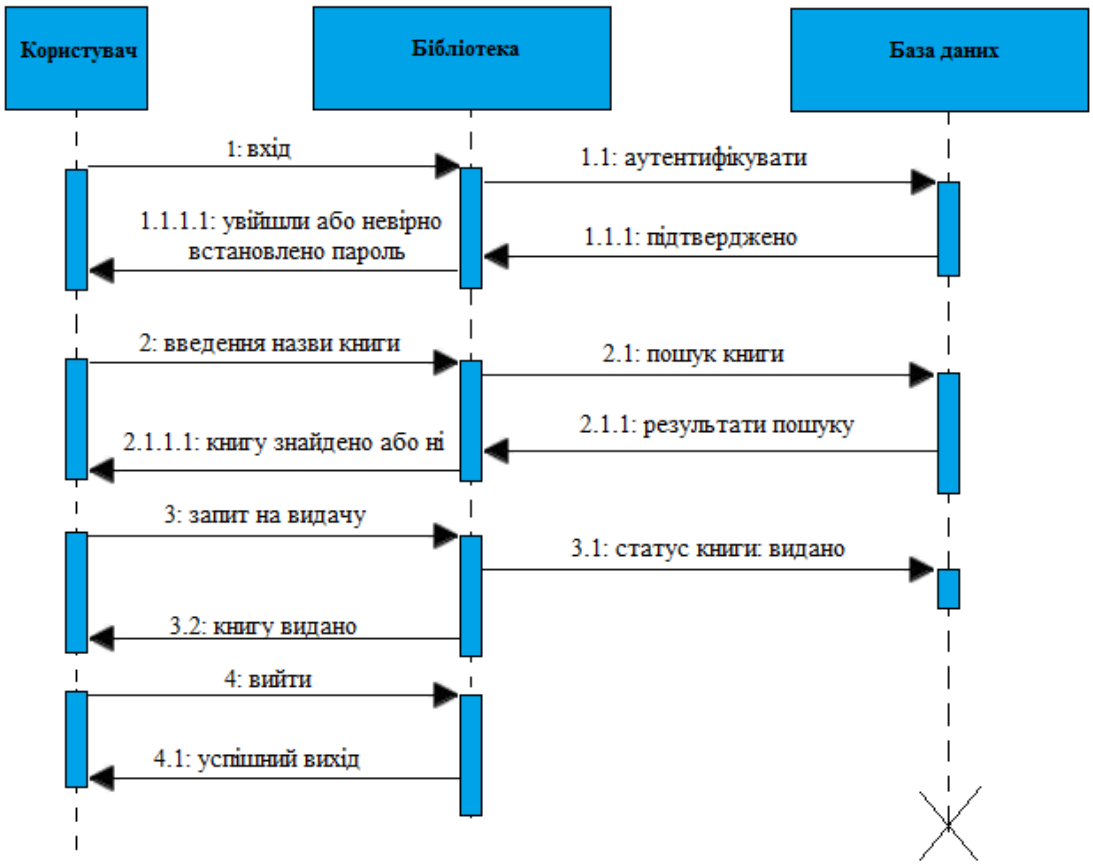

Рис. 1.11 - Діаграма послідовності «Пошук книги».

Для пошуку книги користувач входить в систему ввівши логін і пароль у спеціальну форму бібліотеки. Дані з форми перевіряються з даними користувача в базі, при підтверджені логіна і пароля в базі користувач входить в систему. Користувач вводить в поле пошуку назву книги, бібліотека виконує пошук в базі видаючи результат, що книга знайдена або ні. Наступним кроком є оформлення запиту на видачу книги, після чого відправляється запит на зміну статусу книги, після зміни якого книга видається користувачу. Після чого користувач виходить з системи і появляється інформація про успішний вихід з системи.

1.3. Конструювання програмної системи

1.3.1. Використані технологій для розробки

При створенні програмної системи адміністрації бібліотечних даних важливим є вибір мови якою буде написана система та середовища її розробки. Оскільки від цього вибору буде залежати швидкість розробки та якість результату.

Для виконання розробки інтерфейсу була обрана мова XHTML, яка є частиною сімейства мов розмітки XML, що відображає або розширює версії широко використовуваної мови розмітки HyperText (HTML), мовою, на якій сформульовані веб-сторінки. XHTML розроблений, щоб зробити HTML більш розширеним і збільшити сумісність з іншими форматами даних. Крім того, браузери прощали помилки в HTML, і більшість веб-сайтів відображалися, незважаючи на технічні помилки в розмітці; XHTML запровадив суворішу обробку помилок[13]. Використовуючи простори імен, документи XHTML можуть забезпечити розширення, включивши фрагменти з інших мов на основі XML, таких як масштабована векторна графіка та MathML. Існує розширена версія XHTML для підтримки RDF за допомогою колекції атрибутів та правил обробки у вигляді добре сформованих XML-документів. Ця хост-мова є однією з методик, що використовуються для розробки контенту Semantic Web шляхом вбудовування багатої семантичної розмітки. Документ XHTML, який відповідає специфікації XHTML, вважається дійсним, що забезпечує узгодженість коду документа, який, у свою чергу, полегшує обробку, але не обов'язково забезпечує послідовне відображення браузерами.

Для написання основних функцій використовується мова Perl, яка є мовою програмування загального призначення, котра на початку розроблялась, як інструмент для обробки текстів, а тепер застосовується для вирішення дуже широкого кола завдань, включно з системним

адмініструванням, веб-розробкою, розробкою мережевого програмного забезпечення, та програмного забезпечення з графічним інтерфейсом користувача.

Мова надає перевагу практичному (простота у використанні, ефективність, повнота) над гарним (крихітність, елегантність, мінімалістичність)[14]. Вона має багато можливостей, включно з підтримкою кількох парадигм програмування (процедурне програмування, об'єктно-орієнтоване програмування а також функціональне програмування), управлінням пам'яттю, вбудованою підтримкою системи обробки текстів і великим набором сторонніх модулів.

Веб-сервером при розробці вибрано Apache, який підтримує різноманітні функції, багато з яких реалізовані як складені модулі, що розширюють основні функціональні можливості. Вони можуть варіюватися від схем аутентифікації до підтримки мов програмування на стороні сервера, таких як Perl, Python, Tcl та PHP. В Apache реалізовано методи стиснення один з яких дозволяє зменшити розмір веб-сторінок, що обслуговуються через HTTP. Також забезпечує механізм виявлення та запобігання вторгнень з відкритим кодом для веб-додатків.

Віртуальний хостинг дозволяє одній установці сервера обслуговувати багато різних веб-сайтів, а також конфігурує повідомлення про помилки в СУБД.

Він підтримує автентифікацію пароля та аутентифікацію цифрових сертифікатів. Оскільки вихідний код є у вільному доступі, легкий до адаптації сервер під конкретні потреби, і має велику публічну бібліотеку додатків Apache[15].

Він дозволяє реалізовувати не одну архітектуру пропонуючи різноманітні модулі мультиобробки, які дозволяють йому запускатись або в режимі, заснованому на процесах або в режимі події-гібрид, щоб краще відповідати потребам кожної конкретної інфраструктури. Отже, вибір модулів мультиобробки та конфігурація є важливими. Там, де потрібно

досягти компромісів у роботі, Apache призначений для зменшення затримки та збільшення пропускної здатності відносно простого оброблення більшої кількості запитів, забезпечуючи тим самим послідовну та надійну обробку запитів у розумні часові рамки.

Використовувалась парадигма об'єктно-орієнтованого програмування (ООП), яка заснована на понятті об’єкта, що може містити дані у вигляді полів та коду у вигляді процедур. Особливістю об'єктів є їхні процедури, що можуть отримати доступ та часто змінювати поля даних об'єкта, з яким вони асоціюються. В ООП програми створюються з об’єктів і їх взаємодії між собою.

Підтримка модульного програмування забезпечує можливість групування процедур у файли та модулі для організаційних цілей. Модулі мають простір імен, тому ідентифікатори в одному не будуть конфліктувати з процедурою або змінною, яка спільно використовує те саме ім'я в іншому файлі.

Z39.50 - протокол пошуку та вилучення на прикладному рівні. Стандарт Z39.50 визначає спосіб для комп'ютерів спілкуватися в розподіленому середовищі клієнт / сервер з метою пошук інформації. Це забезпечує рівномірний доступ до великої кількості різноманітних та неоднорідних джерела інформації. Іншими словами, у сховищах інформації у великих базах даних можна шукати та отриманий незалежно від операційної системи серверів, менеджерів баз даних, клієнтів або користувачів інтерфейс.

У моделі взаємодії відкритих систем (OSI) це додаток протокол шару. Протокол є стаціонарним та орієнтованим на з'єднання. Протокол визначає взаємодії лише між двома машинами. Програми "Трансляція пошуку", які дозволяють клієнту здійснювати пошук декількох паралелів на цих серверах ці програми побудовані на версії Z39.50 і використовують декілька паралельних Z39.50 підключення до декількох машин[16]. Стандатр не визначає інтерфейс прикладної програми (API) для послуги протоколу або на

клієнті, або на сервері. Він стосується лише взаємодії між клієнтською та серверною машиною.

Z39.50, служба пошуку інформації визначення та специфікації протоколу для бібліотечних додатків - це стандарт, що складається із специфікацій для комп'ютера до комп'ютера, пов'язаного між різною системою пошуку інформації. Її мета – це кодувати повідомлення, необхідне для зв'язку між двома комп'ютерними системами для конкретних цілей пошуку та пошуку інформації[17].

Особливості використання Z39.50:

– Коли вперше встановлюється сеанс між клієнтом і сервером, він надає засоби для ініціювання параметрів, які слід використовувати через нагадування. Це включає набір символів за замовчуванням, за замовчуванням мову та протокол. Він також забезпечує засоби аутентифікації користувача.

– Надає засоби пошуку однієї або декількох баз даних за допомогою структурного запиту, використовуючи добре відомий формат пошуку. Запит може містити булеві оператори, пошукові терміни в полі, пошук близькості, терміни пошуку ваги, специфікація усічення, специфікатори відношення тощо.

– Надається широкий засіб доступу до інформації з набору результатів пошуку протоколу. Це включає запит конкретних діапазонів результатів пошуку, конкретних елементів у записах, конкретні варіанти записів, виділення пошукових термінів тощо.

– Z39.50 забезпечує можливість створення, називання, зберігання та повернення з одного або декількох наборів результатів пошуку. Це також забезпечує можливість клієнту застосувати критерій пошуку до створеного раніше набору результатів.

– надає можливість перегляду вікна індексу терміна або полів у базі даних.

– Z39.50 не тільки забезпечує аутентифікацію за сеансом, але і надає можливість автентифікації на основі операції у випадках, коли доступ до певних баз даних або записів контролюється.

– надає клієнтам можливість скасувати пошук або запит на презентацію в середині операції, продовжуючи підтримувати відкритий сеанс із сервером[18]. Він також дозволяє клієнтам запитувати звіти про ресурси, які містять облікову інформацію про кількість пошукових запитів, повторних пошуків тощо виконується користувачем.

– забезпечує готовність до операцій з обслуговування бази даних. Він також включає стійкі набори результатів, запити та запити періодів.

Протокол до веб-шлюзів Z39.50 існує вже кілька років. Вони дозволяють OPACS бути доступні через Інтернет. Протокол забезпечує доступ до будь-якої великої бібліотеки світу каталог або просто локаторні джерела за допомогою одного пошуку.

Z39.50 дозволяє бібліотекам в дисимілярних каталогах групуватися без фізичної необхідності копіювати свої бази даних. Користувач може сидіти на екрані OPAC та шукати декілька каталогів одночасно. Корисний матеріал та його розташування можуть відображатися без додаткових дій.

Узагальнюючи інформацію Z39.50 - це стандарт для архітектури клієнт / сервер, в якому знаходиться пошукова система та інтерфейс поділені на самостійні частини. Якщо і клієнт, і сервер відповідають стандарту, то клієнт може шукати будь-яку марку сервера. Широко розповсюджені бази даних у різних природних системах можна шукати з тим самим локальним клієнтом або інтерфейсом. Зв'язок бібліотечних систем із Інтернетом та дозрівання протоколу Z39.50 запропонували перспективу доступу до постійно зростаючого масиву бібліографічних баз даних та повнотекстових баз даних через локальну автоматизовану систему. Ця здатність безпосередньо зв’язувати користувачів з різними ресурсами обчислювальних платформ посилили привабливість протоколу Z39.50 для бібліотек.

Середовищем розробки для програми було вибрано Geany. Який є легким редактор тексту GUI [19], що використовує Scintilla та GTK +, включаючи основні функції IDE. Він призначений для короткого часу завантаження, з обмеженою залежністю від окремих пакетів або зовнішніх бібліотек в Linux. Він був перенесений на широкий спектр операційних систем, таких як BSD, Linux, macOS, Solaris та Windows. Серед підтримуваних мов програмування та мов розмітки є C, C++, C#, Java, JavaScript, PHP, HTML, LaTeX, CSS, Python, Perl, Ruby, Pasca, Haskell, Erlang, та багато інших[. Geany також включає підтримку корисних файлових типів, таких як конфігураційні файли стилю ini, Diff output, SQL-файли та багато іншого.

На відміну від традиційних редакторів на базі Unix, таких як Emacs або Vim, Geany більше нагадує редактори програм, поширені в Microsoft Windows, такі як Notepad ++, де також використовується Scintilla[20].

Це вільне програмне забезпечення, ліцензоване за умовами GNU GPL версії 2 або пізнішої. Особливостями Geany є:

- автозаповнення;
- багаторазова підтримка документу;
- просте управління проектами;
- підсвічування синтаксису;
- складання коду (частково);
- списки символів;
- навігація по коду;
- вбудований емулятор терміналу [21];
- побудувати систему для компіляції та виконання коду за допомогою зовнішніх інструментів;
- розширюється за допомогою плагінів;
- вибір стовпця / блоку / вертикальний (через SHIFT + CTRL + клавіші зі стрілками);

– дія, налаштована користувачем, на клавіатурі для відображення функції редактора.

Завдяки системі плагінів Geany користувачі можуть отримати більше можливостей у Geany, а розробники можуть легко додавати нові функції та / або вдосконалювати існуючі.

1.3.2. Вибір СУБД та опис її фізичної моделі

Для роботи з базами даних програми вибрано вільну систему керування базами даних MySQL, яка забезпечує дуже швидкий, багатопотоковий, багатокористувацький та надійний сервер бази даних. Призначена для критичних виробничих систем з великим навантаженням, а також для вбудовування в програмне забезпечення масового використання[22].

До переваг MySQL сервера відносять:

– використання багатошарової конструкції сервера з незалежними модулями;

– він є повністю багатопоточним з використанням потоків ядра, щоб легко використовувати декілька процесорів, якщо вони доступні;

– забезпечує транзакційні та нетранзакційні двигуни зберігання;

– призначений для того, щоб порівняно легко додавати інші двигуни для зберігання даних.

– використовує дуже швидку систему розподілу пам'яті на основі потоку;

– реалізує хеш-таблиці пам’яті, які використовуються як тимчасові таблиці.

– реалізує функції SQL, використовуючи високо оптимізовану бібліотеку класів, яка повинна бути максимально швидкою

Надає можливість використання сервера як окрему програму для використання в мережевому середовищі клієнт/сервер і як бібліотеку, яка може бути вбудована в окремі програми. Такі програми можна використовувати ізольовано або в середовищах, де нема мережі.

Потужна частина проектування реляційної бази даних - це поділ елементів даних на відповідні таблиці. Після чого важливу роль покладають на взаємозв'язки між таблицями, щоб об'єднати дані змістовно. Якщо нормалізовані таблиці є основою реляційних баз даних, то основою їх співпраці є відносини.

Взаємозв'язки бази даних дуже схожі тим, що вони є асоціаціями між таблицями. Існує три типи відносин:

– Один на один: Обидві таблиці можуть мати лише один запис з обох сторін відносин. Кожне значення первинного ключа стосується лише одного (або немає) запису у відповідній таблиці. Більшість стосунків один на один зумовлені діловими правилами і природним чином не надходять з даних[23]. За відсутності такого правила зазвичай можна поєднувати обидві таблиці в одну таблицю, не порушуючи жодних норм нормалізації.

– Один до багатьох: Таблиця первинного ключа містить лише один запис, що стосується жодної, однієї чи багатьох записів у відповідній таблиці.

– Багато-до-багатьох: кожен запис в обох таблицях може стосуватися будь-якої кількості записів (або відсутніх записів) в іншій таблиці. Відносини багато до багатьох потребують третьої таблиці, відомої як асоційована або пов’язуюча таблиця, оскільки реляційні системи не можуть безпосередньо вміщувати відносини.

Встановлення відносин

Система баз даних для формування взаємозв'язків покладається на відповідні значення, знайдені в обох таблицях. Коли знайдено збіг, система витягує дані з обох таблиць для створення віртуального запису. Наприклад, можена побачити всі книги, написані певним автором. У цьому випадку

система відповідатиме значенням між таблицями Книги та Авторами. Важливо пам’ятати, що більшу частину часу отриманий запис є динамічним, а це означає, що будь-які зміни, внесені у віртуальний запис, як правило, повертаються до нижньої таблиці.

Ці відповідні значення - це первинні та зовнішні ключові значення. Реляційна модель не вимагає, щоб відносини були засновані на первинному ключі. Можна використовувати будь-який кандидат-ключ у таблиці, але використання первинного ключа є прийнятим стандартом.

При створені програми було інтегровано базу даних з програмної системи Koha, яка є досить великою і об’ємною. Для опису в даній роботі було обрано таблиці, що мають пряме відношення до реалізації основних модулів. Одною з яких можна відзначити таблицю borrowers яка містить основні дані про відвідувача системи, такі як: номер відвідувача, номер карти, прізвище, ім’я, код категорії, id поручителя і користувача(рис. 1.9).

Таблиця має зв’язок один до багатьох з таблицею borrower_attributes, яка в свою чергу пов’язана з таблицею borrower_attributes_types також зв’язком багато до одного. Вони відповідають за збереження даних про відвідувачів та його атрибути. За допомогою цих сутностей реалізуються модулі системи відвідувач та обіг. Допомагають здійснювати пошук даних в системі про користувача, ділити їх на типи відносно рівня доступу і правил видачі книг. За замовчуванням доступна можливість ведення таких груп користувачів як: індивідуальні дорослі, колективні для організацій та відвідувачів для дітей. При настроюванні програми типи користувачів можуть змінюватись.

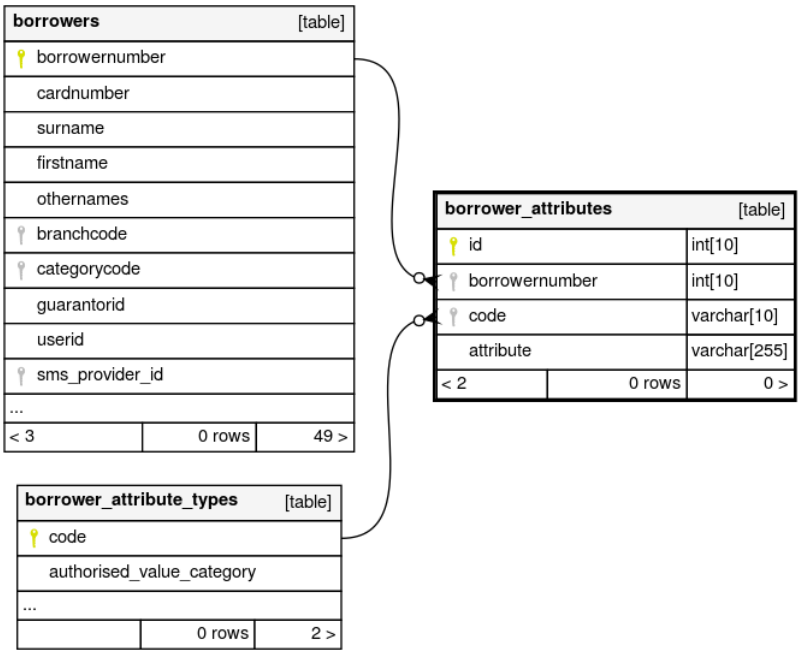

Рис. 1.9 – таблиці сутностей borrowers, borrower_attributes та borrower_attributes_types.

Сутність statistics збирає інформацію про роботу системи та про дії користувача в системі (див. рис 1.10). Має зв’язок багато до одного з сутністю account_offset_types, що є посередником і яка в свою чергу пов’язана з account_offset і search_history. Вважається, що один тип аканту може мати багато акаунтів користувача, а також багато даних з статистики і історії пошуку для кожного акаунту. За допомого цих сутностей стає можливим проводити аналіз дій користувача в системі відповідно до його типу і прав доступу, а також переглядати історію пошуку користувача в системі.

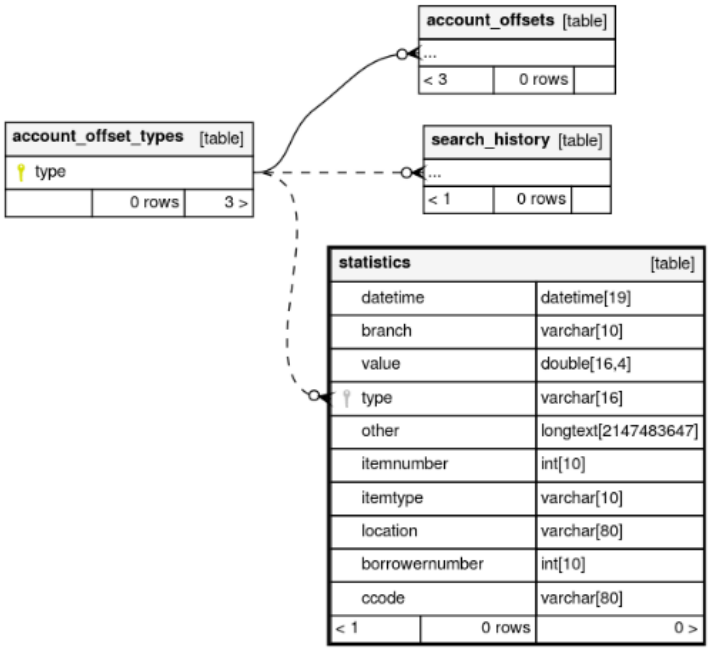

Рис. 1.10 – таблиці сутностей statistics, account_offset_types, account_offset та search_history.

Інформація про історію отримання і повернення книг до бібліотеки, а також про невчасне повернення книг та штрафи за це. Також сутність statistics приймає участь у формуванні звітів у системі. За допомогою функції майстра статистики створюються звіти про надходження книг до бібліотеки, про каталог системи, обіг дій у системі. Звіти про найбільш видавані примірники та втрачені екземпляри.

Функція збору статистики включає визначення загальної кількості читачів на місяць та загальний місячний приріст. Відображення статистики замовлень та завантажень наукової літератури по місяцях, загальна за місяцями та приріст за місяцями. Можливість експорту даних у форматі.csv.

Subscriptionroutinglist, subscription та borrowers (рис. 1.11) – сутності які реалізують абонементні списки користувачів стосовно їхніх підписок. Вони вказують маршрут розповсюдження серіалів, та списки рейтингу в яких вказано чергу конкретного користувача до доступу до серіалу. Реалізують можливості створення нових підписок для користувачів, оновлень вже існуючих, а також їх редагування або видалення для користувача.

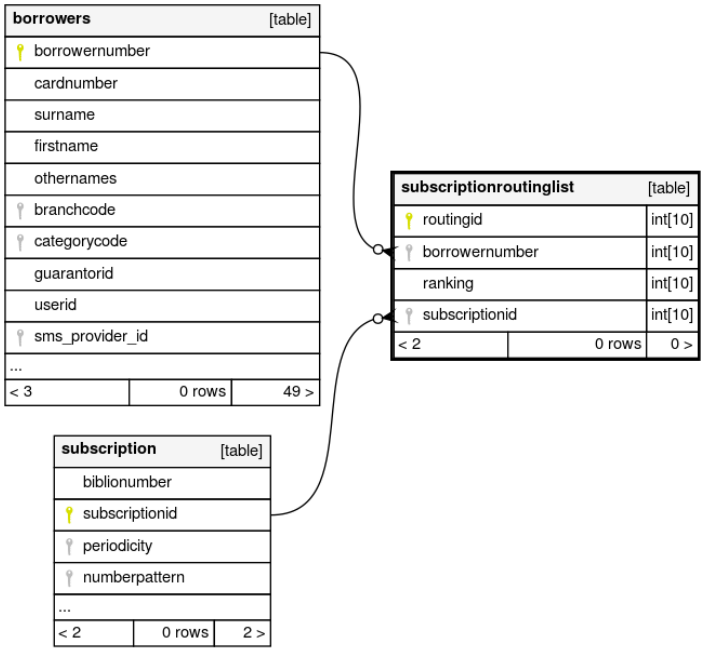

Рис. 1.11 – Таблиці сутностей subscriptionroutinglist, subscription та borrowers.

Один відвідувач може мати багато підписок стосовно різних видань, так само як і для отримання одного видання може бути створена черга з безлічі відвідувачів.

1.4. Реалізація основних класів та методів

Одним з головних методів при створенні програмного забезпечення була функція пошуку. В цьому розділі представлено опис для скрипту пошуку записів в системі.

Даний сценарій використовує новий API пошуку. Він створений так, щоб бути простим у використанні та налаштуваннях, але здатний виконувати такі подвиги, як стримування, зважування поля, релевантність ранжирування, підтримка форматів декількох мов запитів, повна підтримка набору атрибутів, розширений набір атрибутів, визначені в профілях Zebra, доступ до повного діапазону Z39.50 і варіанти запитів SRU, об'єднаний пошук за цілями Z39.50 / SRU тощо.

Цей сценарій виконує дві функції:

– взаємодіє з програмою для отримання та відображення результатів пошуку;

– завантажує сторінку розширеного пошукую.

Ці дві функції поділяють багато однакових змінних та модулів, тому першим завданням є завантаження спільних елементів, а тоді визначення, який з шаблонів необхідно відображати.

Після визначення, система приступає до завантаження лише необхідних змінних та процедур для цієї функції.

При здійсненні пошуку, цей сценарій виконує три основні операції:

– будує рядки запиту;

– виконує пошук і повертає масив результатів;

– формує HTML для виведення в шаблон.

Побудування радків запиту

Існує кілька типів запитів, необхідних у процесі пошуку для отримання:

$ query - повністю вбудований запит, переданий в Zebra. Це найскладніший запит, для побудови. Оригінальою метою створення дизайну було використано спеціальний аналізатор запитів CCL2PQF для перетворення вхідного запиту CCL в запит із кількома полями, який потрібно передати Zebra. Щоб рядок перетворити в поле, необхідно пройти багато зважувань, визначити рейтинг відповідності для конкретного коха та стримування. Цей запит містить профілі запитів, несумісні з більшістю не-Zebra. Z39.50 має на меті здійснити оцінку ваги та релевантності на місцях.

$ simple_query - простий запит, який не містить зважування поля, підходить для переходу до інших цілей пошуку. Цей запит - це лише запит користувача, виражений у CCL CQL або PQF для переходу до цілей Z39.50, не Zebra (та, яка не підтримує розширений профіль, який робить Zebra).

$ query_cgi - передається до шаблону / зберігається для подальшого уточнення запиту (користувачем). Це простий рядок, який повністю виражає

запит як рядок CGI, що може використовуватися для подальшого уточнення запиту або як частина функції історії.

$ query_desc - Опис пошуку людини - що бачить користувач у пошуку області зворотного зв’язку. Це простий рядок, який читається людиною. Він буде містити '=', ',' і т.д.

Виконання пошуку

Отримує рядки запиту та здійснює пошук на названих серверах, включаючи сервер Koha Zebra, зберігає результати в глибоко вкладеному об'єкті, будує результати та повертає ці об'єкти.

Лістинг 2.1. – Здійснення пошуку.

```
my $total = 0;
my $facets;
my $results_hashref;
eval {
        my $itemtypes = {map {$_->{itemtype} => $_} @{Koha::ItemTypes->search_with_localization->unblessed } };
        ($error, $results_hashref, $facets ) = $searcher->search_compat(
          $query,                $simple_query,      \@sort_by,      \@servers,
          $results_per_page,     $offset,            undef,          $itemtypes,
          $query_type,           $scan
        );
);
if ($@ || $error) {
        $template->param(query_error => $error.$@);
        output_html_with_http_headers $cgi, $cookie, $template->output;
        exit;
}
```

Побудувати HTML

Останній головний розділ цього сценарію займає об'єкти, зібрані під час пошуку, і будує HTML для виведення шаблону для користувача.

Лістинг 2.2. – Формування сторінки.

```
my $some_private_shelves = Koha::Virtualshelves->get_some_shelves(
        {
                borrowernumber => $borrowernumber,
                add_allowed    => 1,
                category       => 1,
         }
);
my $some_public_shelves = Koha::Virtualshelves->get_some_shelves(
        {
                borrowernumber => $borrowernumber,
                add_allowed    => 1,
                category       => 2,
         }
);
$template->param(
                add_to_some_private_shelves => $some_private_shelves,
                add_to_some_public_shelves  => $some_public_shelves,
);
output_html_with_http_headers $cgi, $cookie, $template->output;
```

Сценарій генерує сторінку зі змістом при запиті на нього, на основі інформації з бази даних. Зібрана інформація при пошуку формується в певний шаблон і відображається для користувача. При формуванні пошуку сторінка генерується один раз, надалі вона швидше підвантажується. Оновлення сторінки відбувається автоматично, після закінченні деякого терміну або при внесенні змін в певні розділи. Збереження певних інформаційних блоків на етапі редагування сайту і збірка сторінки з цих блоків при запиті відповідної сторінки користувачем.

2. ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ ПРОГРАМИ

2.1 Огляд основних функцій і інтерфейсу програми

Розглянемо основні функцій системи та інструкція по їх використанні. Після відкриття бібліотеки з’являється вікно головної сторінки на якому розміщене поле пошуку в каталозі бібліотеки (рис.2.1), Здійснюється широкий пошук за словом, словосполученнями або комбінацією слів.

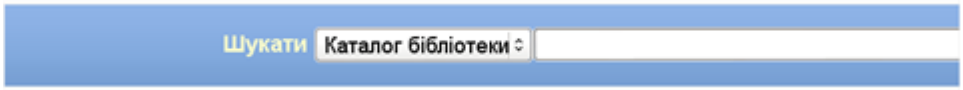

Рис.2.1 - пошуку в каталозі бібліотеки

Також можливий пошук за тематикою, заголовком, серією або автором. Для отримання більш точних результатів використовується функція розширеного пошуку, що дає можливість обмежити його за певними типами такими, як, наприклад, журнали і книги, а також здійснює пошук за трьома ключовими словами.

Пошук можливий без входу в обліковий запис системи користувача, проте для отримання більшої кількості можливостей необхідно ввійти в обліковий запис. Вхід здійснюється з головної сторінки, на якій в правому боці знаходиться форма для входу в обліковий запис(див. рис. 2.2).

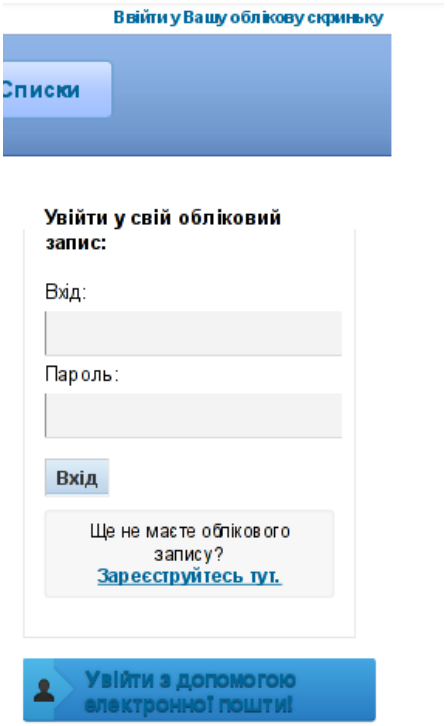

Рис. 2.2 – Форма входу в бібліотеку

За допомогою даної форми також є можливість реєстрації, якщо немає облікового запису у системі та передбачено вхід за допомогою електронної пошти. Обліковий запис дає додаткову можливість переглядати історію пошуків в системі, а також історію всіх видач і повернень літератури власником облікового запису. Передбачено також можливість створення списків літератури для подальшого перегляду, зміни назв списків, а також їхнього редагування та видалення.

Результати пошуку (рис 2.3) відображаються на екрані у вигляді списку. Ключові слова за яким здійснювали пошук виділяються іншим кольором відносно основного тексту. Є можливість додавати, переглядати, резервувати та сортувати результати за певними критеріями.

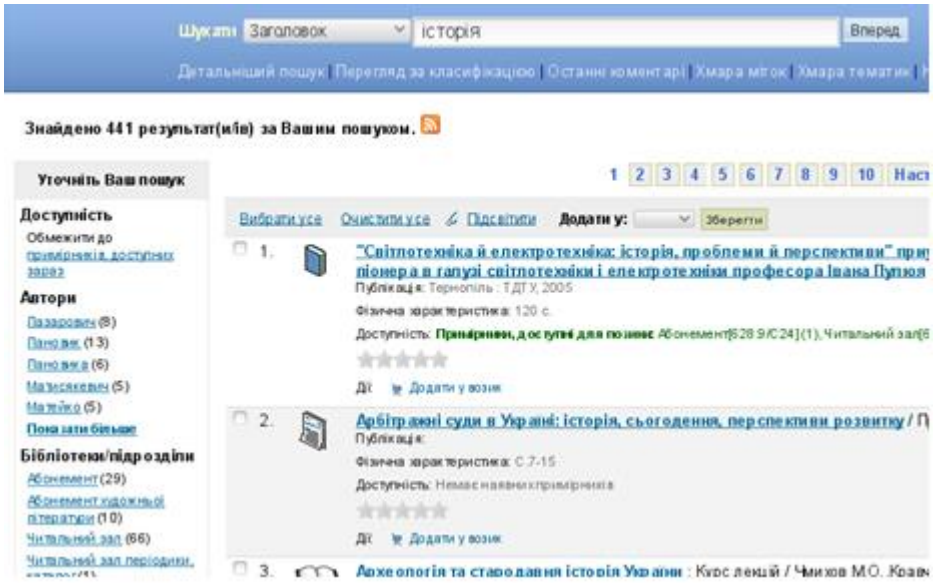

Рис. 2.3 – Список результати пошуку

Вибравши один з результатів система переходить до його бібліографічного запису, тобто відображається вся інформація про запис. Інформація включає: назву, місце публікації, мову, серії, короткий опис або резюме, а також інформацію про усі примірники та наявність їх у бібліотечному фонді (рис. 2.4) та його розташування.

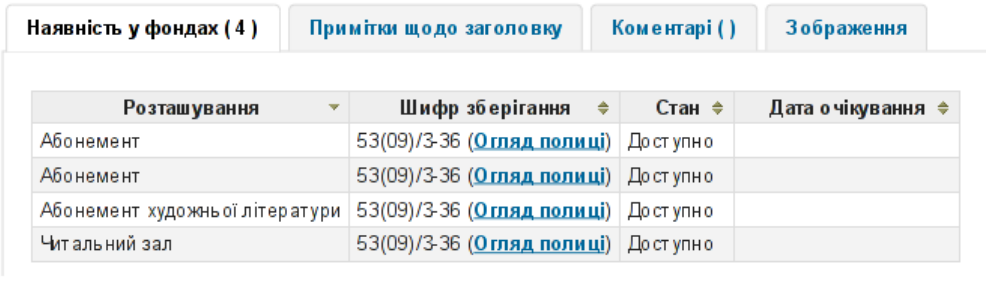

Рис. 2.4 - Інформація про примірники

Інтерфейс для користувач Бібліотекар, в якого є права доступу супер користувача, відрізняється від інтерфейсу простого користувача. Такий рівень доступу призначений для працівників бібліотеки. Щоб працювати з такими правами потрібно увійти в систему(рис.2.5).

Рис.2.5 – Вікно в ходу в систему бібліотекар

Після входу в систему бібліотекаря з'являється екран з усіма можливостями і функціями системи. Для деяких користувачів рівень доступу може бути менший в зв'язку з чим відображатись будуть не всі функції.

В верхній частині вказано основні модулі системи з якими працює бібліотекар. До них відносяться обіг, відвідувачі, пошук, каталогізація, звіти та надходження.

Обіг бібліотеки включає дані про видачу і повернення книг до бібліотеки, переміщення книг в бібліотечних фондах, можливість резервувати книгу дистанційно, а також за термінами повернення книг до бібліотеки. Основні функції модуля обіг представлено на рис. 2.6.

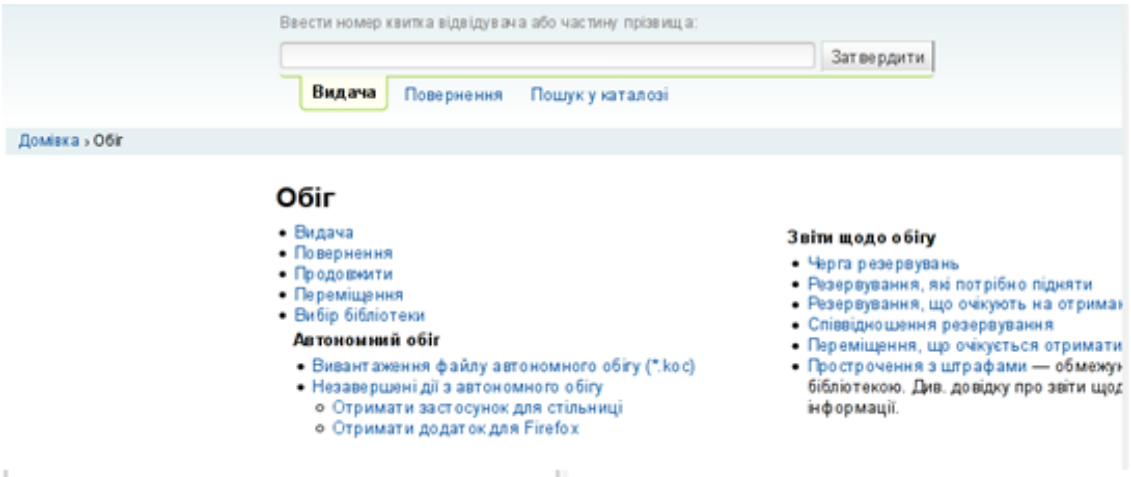

Рис. 2.6 – Функції модуля Обіг

Для відслідковування книг, кожній з них присвоюється власний номер штрих-коду, для користувачів таким ідентифікатором є номер читацького білету.

2.2. Тестування програмної системи

Системне тестування - це рівень тестування програмного забезпечення, на якому тестується повне та інтегроване програмне забезпечення. Мета цього тесту - оцінити відповідність системи визначеним вимогам.

Тестування системи бере в якості свого вкладу всі інтегровані компоненти, які пройшли інтеграційне тестування. Метою інтеграційного тестування є виявлення будь-яких невідповідностей між одиницями, які інтегруються разом. Тестування системи прагне виявити дефекти як у межах збірки, так і всередині системи в цілому[24]. Фактичним результатом є поведінка, яка виробляється або спостерігається під час тестування компонента чи системи.

Тестування системи проводиться у всій системі в контексті або функціональних специфікацій вимог, або специфікації системних вимог, або обох. Перевіряє не тільки дизайн, але й поведінку та навіть вірогідні очікування замовника. Він також призначений для випробування до меж, визначених у специфікаціях програмного забезпечення або обладнання.

Зазвичай використовується метод тестування чорної коробки, також відомий як тестування на поведінку(див. рис. 2.7), - це метод тестування програмного забезпечення, при якому внутрішня структура / дизайн / реалізація предмета, який тестується, не відомий тестеру[25]. Ці тести можуть бути функціональними або не функціональними, хоча зазвичай є функціональними.

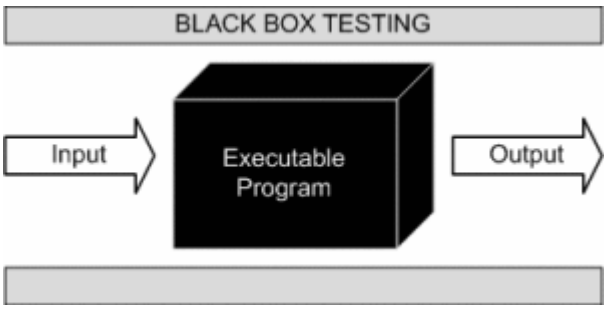

Рис. 2.7 - Зображення тестування чорного поля.

Цей метод названий так, тому що програмне забезпечення в очах тестера - це як чорний ящик; всередині якого не можна побачити. Цей метод намагається знайти помилки в наступних категоріях:

– неправильні або відсутні функції;

– помилки інтерфейсу;

– помилки в структурі даних або до доступу до зовнішньої бази даних;

– помилки в поведінці або виконанні;

помилки ініціалізації та припинення.

Техніка проектування чорної скриньки: Процедура отримання та / або відбору тестових випадків на основі аналізу специфікацій, або функціональних, або не функціональних, компонента чи системи без посилання на його внутрішню структуру.

Було проведено тестування інтерфейсу програми, а також всіх основних функцій системи. Вхід в систему виконується правильно. Дані в базу даних записуються коректно, запити з пошуку інформації виконуються швидко та без помилок. Каталоги, адміністрування та обіг бібліотеки повністю відповідають поставленим вимогам.

За результатами тестування програмної системи методом чорного ящика можна зробити висновок, що система працює коректно.

3. ОРГАНІЗАЦІЙНО-ЕКОНОМІЧНА ЧАСТИНА

3.1 Загальний підхід до визначення економічної ефективності розробки.

Головною метою розділу виступає встановлення економічної доцільності розробки удосконалення програмного забезпечення, що буде використовуватись бібліотеками для адміністрування даних.

Розробка нового програмного продукту вимагає свого управління і контролю з боку керівника. Таким чином, складання та організація економічної частини є актуальною проблемою сучасного менеджменту.

Планування потребує будь-яке підприємство, будь-яке виробництво, економіка в цілому. Спланувати – означає оцінити можливості, необхідність і обсяги випуску конкурентоспроможної продукції, визначити місткість ринку і його конкретного сегмента, оцінити попит на продукцію, що випускається підприємством, результативність його роботи на ринку.

Економічне обґрунтування доцільності розробки програмного забезпечення. Зокрема розраховується комплексний показник якості проектного рішення, який показує його переваги в порівнянні з аналогами. А також на основі показника якості та ціни споживання проектного рішення та його аналога визначається коефіцієнт конкурентоздатності, який показує спроможність даного проектного рішення конкурувати з аналогами.

Приступаючи до розробки такого програмного забезпечення перш за все потрібно обґрунтувати його з точки зору економічної доцільності. Дане обґрунтування необхідне для того, щоб розробка була економічно ефективною і визначити чи варто розробляти новий програмний продукт, чи вигідніше модернізувати застарілий.

Економічний ефект розробленого продукту визначається на основі економічних показників, які дають можливість прогнозувати результат від

впровадження даної програми. Враховуючи інтенсивний розвиток комп'ютерної техніки і широкий вибір програмного забезпечення, на сьогодні економічний аналіз є невід'ємною частиною попереднього аналізу, а кошти, що затрачаються, повинні бути еквівалентними тому ефекту, який принесе конкретне нововведення.

Оцінка вартості дослідницьких розробок базується на витратному підході: використанні первісної вартості об'єктів, виходячи з фактичних витрат на розробку та доведення до комерційного використання з урахуванням амортизації.

Згідно Статті 8 Закону № 3792-12 передбачено, що твори наукового характеру та комп’ютерні програми є об’єктами авторського права [26]. У разі реєстрації виконаної роботи як інтелектуальної власності та авторського права у державній службі інтелектуальної власності України необхідно буде сплатити збори за державну реєстрацію.

Для отримання відмінних результатів експериментів та доцільності розробки такого спеціалізованого ПЗ потрібні відповідні затрати на дослідження та розробку. Це і становитиме основу витрат, які будуть здійснені протягом підготовки та виконання реалізації даного рішення. Уявно модель витрат можна поділити на дві основні частини: витрати, пов’язанні на дослідження предметної області, побудову математичних моделей, отримання попередніх результатів експериментів, та частину реалізації програмної системи, архітектури та тестування.

Беручи до уваги залежність якості кінцевого продукту від кваліфікації програмістів, потрібно сконцентрувати увагу на якості та результативності розробки. Для цього потрібно провести ґрунтовний аналіз предметної області, залучити найновіші та інноваційні технології, провести ґрунтовне тестування та оцінку результатів розробки. Для забезпечення хорошої результативності при розробці доводиться йти на додаткові заходи заохочення та стимулювання у вигляді преміювання працівників.

До створення ПЗ можуть бути залучені позаштатні програмісти як зареєстровані, так і не зареєстровані підприємцями. В обох випадках співпраця з ними здійснюється на підставі цивільно-правового договору, найчастіше – договорами підряду. Щодо оподаткування виплат за договором підряду, то все залежить від того, чи зареєстрований виконавець підприємцем. Важливим етапом розробки ПЗ є його тестування, яке виконується тестувальником за певну винагороду. Якщо тестувальники не перебувають з підприємством у трудових відносинах, оплата виконується на основі договору підряду на виконання робіт з тестування ПЗ. Сам же результат розробки не оподатковується, адже не є комерційним проектом і не спрямований на продаж.

3.2 Розрахунок вартості процесу розробки та оцінка економічної ефективності проекту

Для оцінки нематеріальних активів використовують міжнародні стандарти оцінки розрахунку вартості об’єктів інтелектуальної власності, розроблені TIAVIS (The International Assets Vabulation Standarts Commitie).

Виконання розробки програмного забезпечення з огляду економічної моделі можна виконувати двома способами: процедурним та об’єктно-орієнтованим. Обидва підходи потребують залучення ресурсів у вигляді програмістів-розробників, тестувальників, керівника проекту, наукового ресурсу. Різниця виникає в самій схемі розробки, тривалості періоду розробки та відповідній вартості. Процедурний підхід для розробки ПЗ в основі якого лежать процедури і функції передбачає розробку ПЗ як монолітного композиту, що в подальшому, як правило, вимагає великих витрат на супровід та модернізацію. Об’єктно-орієнтований підхід, що ґрунтується на основі об’єктів певних класів, що описують певну область, описують певну поведінку (методи) та володіють властивостями

(атрибутами), орієнтовані на варіанти використання та покроковий процес розробки.

Для початку робіт необхідно скласти технічне завдання на розробку, яке є основним документом, що регламентує подальшу роботу, та містить докладний опис необхідних функцій програми, інтерфейс, технології, інше. Вартість складання технічного завдання переважно складає до 10% від планованої вартості розробки. Роботу зі складання технічного завдання веде керівник проекту разом із програмістами та консультуючись із замовником.

Усі програмісти, що працюють у штаті підприємства-розробника мають встановлено певний посадовий оклад. Місячний оклад, денна заробітна плата, трудомісткість (днів) і основна заробітна плата кожного учасника техпроцесу представлено у таблиці 3.1. Всі суми наведені в національній валюті – в гривні.

Таблиця 3.1 – Розрахункова вартість технологічного процесу розробки

Посада	Місячний оклад, грн.	Денна зар. плата, грн.	Об'єктно-орієнтований підхід		Процедурний підхід	
			Днів	Сума, грн.	Днів	Сума, грн.
Керівник проекту	13300,00	570,00	18	10260,00	19	10830,00
Програміст	12500,00	530,00	27	14310,00	31	16430 ,00
Інженер - тестувальник	10120,00	440,00	10	4400,00	10	4400,00
Всього			55	28970 грн.	60	31660 грн.

Витрати на науково-дослідницьку роботу та здійснення розробки програмних продуктів і об'єктно-орієнтованим, і процедурним способом включають:

Основна заробітна плата:

$$ЗП_{осн\ 1} = 28970 \text{ грн}; \quad ЗП_{осн\ 2} = 31660 \text{ грн}.$$

Додаткова заробітна плата обчислюється як:

$$ЗП_{дод} = 0{,}2 \cdot ЗП_{осн} \tag{3.1}$$

$$ЗП_{дод\ 1} = 0{,}2 \cdot 28970 = 5794 \text{ грн};$$

$$ЗП_{дод\ 2} = 0{,}2 \cdot 31660 = 6332 \text{ грн}.$$

Таким чином загальний фонд заробітної плати, що обчислюється за формулою:

$$ФЗП = ЗП_{осн} + ЗП_{дод}. \tag{3.2}$$

$$ФЗП_1 = 28970 + 5794 = 34764 \text{ грн.};$$

$$ФЗП_2 = 31660 + 6332 = 37992 \text{ грн.}$$

Крім того, слід визначити відрахування на соціальні заходи:

– єдиний соціальний внесок – 3,6 %;

– військовий збір – 1,5 %;

– ПДФО (прибутковий податок) – 15 %.

Отже, сума відрахувань на соціальні заходи буде становити:

$$Відр_{ЄСВ1} = 0{,}036 \cdot ФЗП = 1251{,}50 \text{ грн.};$$

$$Відр_{ЄСВ2} = 0{,}036 \cdot ФЗП = 1367{,}71 \text{ грн.};$$

$$Відр_{вз1} = 0{,}015 \cdot ФЗП = 521{,}46 \text{ грн.};$$

$$Відр_{вз2} = 0{,}015 \cdot ФЗП = 569{,}88 \text{ грн.}$$

$$Відр_{ПДФО1} = 0{,}15 \cdot ФЗП = 5214{,}6 \text{ грн.};$$

$$Відр_{ПДФО2} = 0{,}15 \cdot ФЗП = 5698{,}8 \text{ грн..}$$

Нарахування на фонд оплати праці, які включають відрахування до Пенсійного фонду, фонду з тимчасової втрати працездатності, фонду з безробіття і фонду страхування від нещасних випадків на виробництві; для бюджетної організації тариф на фонд оплати праці встановлено на рівні 36,3%.

Зокрема, видання програмного забезпечення – 36,77%.

Нарахування на Фонд оплати праці (ФОП): $ФОП_{ЄСВ} = 0{,}3677 \cdot ФЗП$

$$ФОП_{ЄСВ1} = 0{,}3677 \cdot ФЗП_1 = 12782{,}72 \text{ грн.};$$

$$\text{ФОП}_{\text{ЄСВ2}} = 0{,}3677 \cdot \text{ФЗП}_2 = 13969{,}66 \text{ грн}$$

Всього витрат:

$$\text{В}_{\text{ЗП1}} = \text{ФЗП}_1 + \text{ФОП}_{\text{ЄСВ1}} = 34764 + 12782{,}72 = 47546{,}72 \text{ грн.};$$

$$\text{В}_{\text{ЗП2}} = \text{ФЗП}_2 + \text{ФОП}_{\text{ЄСВ2}} = 37992 + 13969{,}66 = 51961{,}66 \text{ грн.};$$

Також важливою складовою витрат є матеріальні витрати. Матеріальні витрати визначаються як добуток кількості витрачених матеріалів та їх ціни:

$$M_{в.і} = q_{\text{і}} \cdot p_{\text{і}}\,, \qquad (3.3)$$

де: $q_{\text{і}}$ – кількість витраченого матеріалу і–го виду;

$p_{\text{і}}$ – ціна матеріалу і–го виду.

Звідси, загальні матеріальні витрати можна визначити:

$$З_{М.В} = \sum M_{В.і} \qquad (3.4)$$

Проведені розрахунки занесемо у таблицю:

Таблиця 3.2 – Зведені розрахунки матеріальних витрат

Найменування ресурсу	Кількість, шт.	Ціна одиниці, грн	Загальна сума, грн
Флешки	2	227,00	454,00
Папір для друку А4, арк	500	0,2	100,00
Тонер для принтера	1	60,00	60,00
Всього			614,00

Отже загальна сума матеріальних витрат становить 614 гривень.

Розрахунок витрат на електроенергію одиниці обладнання визначаються за формулою:

$$З_в = W * T * S, \qquad (3.5)$$

де W – необхідна потужність, кВт; T – кількість годин роботи обладнання; S – вартість кіловат-години електроенергії, S = 2,50 грн/кВт·год.

$З_{в1}$ =0.7 * 440 * 2.50 = 770 грн;

$З_{в2}$ =0.7 * 480 * 2.50 = 840 грн;

Розрахунок суми амортизаційних відрахувань. Комп'ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 %, вартість яких перевищує 1000 грн. і визначається:

$$A = \frac{C_Б \cdot N_A \cdot T_{ФАК}}{T_{ГОД}} \quad (3.6)$$

де $C_Б$ – балансова вартість обладнання, грн; N_A – норма амортизаційних відрахувань в рік, %; $T_{ФАК}$ – фактичний час роботи обладнання по написанню програми, год; $T_{ГОД}$ – річний робочий фонд часу, год.

У даній формулі норма відрахувань на амортизацію рівна N_A = 0,6. Балансова вартість обладнання вказана в таблиці 3.3 і рівна $C_Б$ = 16045 гривень. Річний робочий фонд часу приймемо за $T_{ГОД}$ = 2120 годин. З них реальний фактичний робочий час становить $T_{ФАК}$ = 440 години згідно об'єктно-орієнтованого підходу та $T_{ФАК}$ = 480 годин згідно процедурного підходу.

Таблиця 3.3 – Перелік необхідного обладнання

Найменування	Кількість, шт	Ціна, грн	Сума, грн
Комп'ютер	1	12545,00	12545,00
Принтер	1	3500,00	3500,00
Середовища розробки	2	безкоштовно	безкоштовно
Операційна система	2	безкоштовно	безкоштовно
Всього більше 1000 грн.			16045,00
Всього			16045 грн.

$$A_1 = (16045 \cdot 0{,}6 \cdot 440)/2120 = 1998{,}1 \text{ грн.}$$

$$A_2 = (16045 \cdot 0{,}6 \cdot 480)/2120 = 2179{,}7 \text{ грн.}$$

Накладні витрати пов'язані з обслуговуванням виробництва, утриманням апарату управління та створення необхідних умов праці та закупівлю ресурсів та обладнання для розробки наведені в таблиці 3.3.

Залежно від організаційно-правової форми діяльності господарюючого суб'єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників.

$$НВ = 0{,}5 \cdot ЗП_{осн} \qquad (3.7)$$

де $НВ$ – накладні витрати.

Отже, накладні витрати:

$$НВ_1 = 28970 \cdot 0{,}5 = 14485 \text{ грн.}$$

$$НВ_2 = 31660 \cdot 0{,}5 = 15830 \text{ грн.}$$

Проведемо розрахунок вартості створюваного програмного продукту. Вартість продукції включає у собі собівартість і планований прибуток. Найважливішим моментом для розробника, з економічної точки зору, є процес встановлення ціни.

Можна отримати загальні значення витрат на розробку та реалізацію проекту, враховуючи всі вище описані затрати та нарахування. Цей вид

витрат складається з сум витрат на оплату праці (всього витрати на оплату праці), матеріальні затрати, затрати на електроенергію, накладні витрати, витрати на обладнання, враховуючи амортизації обладнання на час виконання проекту.

Собівартість продукції – це сума грошових витрат підприємства (фірми) на виробництво і збут одиниці продукції, виконання робіт та надання послуг.[27]

Повна собівартість програмного продукту дорівнює сумі усіх витрат на його виробництво(таблиця 3.4):

$$Св1 = 59619{,}82 \text{ грн.};\quad Св2 = 65093{,}36 \text{ грн.}$$

Таблиця 3.4 – Розрахунок собівартості

Зміст витрат	Сума, грн	
	Об’єктно-орієнтований підхід	Процедурний підхід
Витрати на оплату праці	28970	31660
Відрахування на соціальні заходи	12782,72	13969,66
Матеріальні витрати	614	614
Витрати на електроенергію	770	840
Амортизаційні відрахування	1998,1	2179,7
Накладні витрати	14485	15830
Собівартість	59619,82	65093,36

Приймемо прибуток на рівні 30%. Для нових інноваційних продуктів, що користуються високим попитом на ринку, ринкову вартість Вр можна встановити вищу.

Отже, вартість розробленого програмного забезпечення:

$$Вр1 = Св1 + 0{,}3 \cdot Св1 = 59619{,}82 + 0{,}3 \cdot 59619{,}82 = 77505{,}8 \text{ грн.};$$

Вр2 = Св2 + 0,3 · Св2 = 65093,36+ 0,3 · 65093,36= 84621,4 грн.

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Економічна ефективність (Е) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E = \frac{П}{Св} \tag{3.8}$$

де *П* – прибуток, *П= В - Св; Св* – собівартість.

Плановий прибуток ($П_{пл}$) знаходимо за формулою:

$$П_{пл} = В_р - С_в \tag{3.9}$$

Розраховуємо плановий прибуток:

$$П_{пл1} = 77505,8 – 59619,82 = 17627,99 \text{ грн.}$$

$$П_{пл2} = 84621,4 – 65093,36 = 19528 \text{ грн.}$$

Отже, формула для визначення економічної ефективності набуде вигляду:

$$E_р = \frac{П_{пл}}{С_в}. \tag{4.10}$$

$$Е_{р1} = 17627,99 / 59619,82 = 0,3.$$

$$Е_{р2} = 19528 / 65093,36 = 0,3.$$

Якщо ринкова вартість програмного продукту рівна прийнятій, то економічна ефективність визначається встановленим рівнем прибутку. Поряд

із економічною ефективністю розраховують термін окупності капітальних вкладень ($Т_{ок}$):

$$Т_{ок}=\frac{1}{Е} \quad (3.10)$$

Термін окупності дорівнює:

$$Т_{ок} = 1 / 0,3 = 3,3 \text{ роки}$$

У нашому випадку $Т_{ок1} = Т_{ок2} = 1/0,30 = 3,33$ років, що є нормальним, оскільки допустимим вважається термін окупності до 5 років.

Даний розрахунок виконаний у розрахунку на 1 екземпляр програмного продукту без врахування його тиражування.

Чистий приведений дохід (ЧПД) визначається як різниця між сукупними доходами (сукупний грошовий потік) і сукупними витратами (сукупними інвестиціями) взятими за весь період життя інвестицій і дисконтова ними в кожному році на фактор часу. Дисконтування являє собою визначення вартості майбутніх грошових потоків у теперішній момент часу. Ефективним вважається той проект, який забезпечує максимум ЧПД, оскільки при цьому досягається найвища дохідність власників інвестицій.

Визначення ЧПД відбувається за формулою:

$$ЧПД = \sum_{i=1}^{t} ГП_i \alpha_{ТВi} - \sum_{i=1}^{t} ІК_i \alpha_{ТВi} \quad (3.11)$$

де $ГП_i$ – грошовий потік і-го розрахункового року;

$ІК_i$ – сума інвестицій і-го розрахункового року;

$\alpha_{ТВi}$ – коефіцієнт дисконтування (коефіцієнт приведення інвестицій і грошового потоку до теперішньої вартості).

Грошовий потік – це фінансовий показник, що характеризує ефект інвестицій у вигляді грошових коштів, які повертаються інвестору.

Коефіцієнт дисконтування показує, яку величину грошових коштів ми отримаємо з урахуванням фактору часу та ризиків. Він дозволяє перетворити майбутню вартість у вартість на даний момент. Для розрахунку коефіцієнта

дисконтування (коефіцієнта приведення) грошових потоків за роками періоду економічного життя інвестицій використовується формула:

$$\alpha = \frac{1}{(1+i)^n} \quad (3.12)$$

де i – ставка дисконтування або норма дисконту, i = 0,2;

n – час або кількість періодів (років), протягом якого планується отримання доходу.

α_0=1, $\alpha_1 = \frac{1}{(1+0.2)^n}$=0,83

Вважатимемо, що обидва програмних продукта одинаково забезпечують потреби і вимоги споживача, і тому придбання першої чи другої програми одинаково вплинуть на розмір його додаткових доходів на вкладений капітал.

Тому приймемо цю величину за постійну, а порівняння дохідності двох проектів проведемо тільки за витратами.

$ЧПД_1' = ГП$+0,83*$ГП$-76387,95 - 0,83*17627,99 = 1,83$ГП$ - 91019,19 грн.;

$ЧПД_2' = ГП$+0,83*$ГП$-83437,59- 0,83*19254,83 = 1,83$ГП$ - 99419,10 грн.

Чим менші витрати, тим більша дохідність проекту.

Економія витрат у випадку придбання, супроводу і одноразової модернізації програмного продукту, створеного за об’єктно-орієнтованим підходом, становить 8652,2 грн. Однак варто зауважити, що розробка спрямована на короткотривалу підтримку та не прогнозує модернізації. У разі ж необхідності розробки такого ж або суміжного ПЗ можна вважати за доцільно розпочинати розробку з початкових етапів, що потягне за собою нові витрати у повній мірі. Тобто у разі короткотермінової підтримки все ж за доцільніше обирати об’єктно-орієнтовану модель розробки, адже вартість короткотермінової підтримки згідно цієї моделі не вплине значно на сукупну вартість розробки.

Здана в експлуатацію система не завжди цілком завершена, її треба змінювати протягом терміну експлуатації. Внаслідок змін система стає більш складною і погано керованою. Об’єктно-орієнтоване представлення програми дозволяє навіть середньому програмісту швидко і ефективно супроводжувати і модернізувати програми, що значно скорочує подальші витрати на супровід і модернізацію [27].

Сумарні дані економічного розрахунку розробки даного проекту наведені в таблиці 3.5.

Таблиця 3.5 – Загальні витрати

Вид витрат	Об’єктно-орієнтований підхід, грн	Процедурний підхід, грн
Зарплата основна	28970	31660
Зарплата додаткова	5794	6332
Фонд заробітної плати, грн.	34764	37992
Відрахування на ФОП	12782,72	13969,66
Разом на оплату праці	82310,72	89953,66
Матеріальні витрати	614,00	614,00
Електроенергія	770	840
Амортизація	1998,1	2179,7
Накладні витрати	14485	15830
Разом на ін. витрати	17867,1	19403,7
Собівартість	59619,82	65093,36
Вартість розробленого ПЗ	77505,8	84621,4
Економічна ефективність	0,30	0,30
Термін окупності, років	3,33	3,33
Загальні витрати на розробку	95372,9	104025,1
Економія ($ЗВ_2$-$ЗВ_1$)	**8652,2**	

Загальна вартість пропонованих робіт становить 104025,1 гривень для процедурного і 95372,9 гривень для об’єктно-орієнтованого підходів розробки. В даному випадку реалізації проекту варто вибрати об’єктно орієнтований підхід для розробки даного ПЗ, адже фінансово це більш вигідно. Також у даній методиці розробки кращі часові рамки та перспективи підтримки і модернізації. У оцінці вартості продукту варто враховувати можливість фінансування та спонсорування проекту, що дозволить гнучко та ефективно підійти до розробки та організації праці.

4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці при роботі з персональними комп'ютерами

Результатом виконання даної магістерської роботи буде програмна система автоматизації роботи бібліотеки, яка буде використовуватись на комп’ютеризованому робочому місці.

Площа та об’єм для одного робочого місця визначають згідно з вимогами ДСанПіН 3.3.2-007-98[28]. Площа має бути не менше 6,0 кв. м, об’єм — не менше 20,0 куб. м.

При розміщенні робочих місць з персональними комп'ютерами відстань між робочими столами з екранами має бути не менше 2,0 м, а відстань між бічними поверхнями ВДТ - не менше 1,2 м.

Робочі місця з персональними комп'ютерами в приміщеннях з джерелами шкідливих виробничих факторів розміщуються в ізольованих кабінах з організованим повітрообміном. При виконанні творчої роботи, що вимагає значного розумового напруження або високої концентрації уваги, робочі місця з персональними комп'ютерами рекомендується ізолювати один від одного перегородками висотою 1,5-2,0 м.

Екран повинен знаходитися від очей користувача на відстані 600-700 мм, але не ближче 500 мм з урахуванням розмірів алфавітно-цифрових знаків і символів.

Відповідно до НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» конструкція робочого столу повинна забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання з урахуванням його кількості і конструктивних особливостей відносно характеру виконуваної роботи[29]. Проте допускається використання робочих столів різних конструкцій, які відповідають сучасним вимогам ергономіки. Коефіцієнт відбиття для поверхні робочого столу повинен становити 0,5-0,7.

Висота робочої поверхні столу для дорослих користувачів повинна регулюватися в межах 680-800 мм; при відсутності такої можливості висота робочої поверхні столу повинна складати 725 мм. Модульними розмірами робочої поверхні столу для ПК, на підставі яких повинні розраховуватися конструктивні розміри, слід вважати: ширину - 800, 1000, 1200 і 1400 мм, глибину - 800 і 1000 мм при нерегульованої його висоті, рівній 725 мм.

Робочий стіл повинен мати простір для ніг висотою не менше 600 мм, шириною - не менше 500 мм, глибиною на рівні колін - не менше 450 мм, на рівні витягнутої ноги - не менше 650 мм.

Конструкція робочого стільця (крісла) повинна забезпечувати підтримку раціональної робочої пози під час роботи на персональному комп'ютері, дозволяти змінювати позу з метою зниження статичного напруження м'язів шийно-плечової області і спини для попередження розвитку втоми. Тип робочого стільця (крісла) слід вибирати з урахуванням зростання користувача, характеру та тривалості роботи з ПК[30].

Робоче місце користувача ПК слід обладнати підставкою для ніг, має ширину не менше 300 мм, глибину не менше 400 мм, регулювання по висоті в межах до 150 мм і за кутом нахилу опорної поверхні підставки до 20 град. Поверхня підставки повинна бути рифленою і мати по передньому краю бортик висотою 10 мм.

Клавіатуру слід розташовувати на поверхні столу на відстані 100-300 мм від краю, зверненого до користувача, або на спеціальній, регульованою по висоті робочої поверхні, відокремленої від основної стільниці.

Профілактичні заходи для зниження нервовоемоційного напруження викладено у ДСанПіН 3.3.2.007-98. Раціональний режим праці та відпочинку передбачає дотримання певної тривалості безперервної роботи на персональному комп'ютері і перерв, регламентованих з урахуванням тривалості робочої зміни, виду трудової діяльності.

При виконанні протягом робочої зміни робіт, що відносяться до різних видів трудової діяльності, за основну роботу з ПК слід сприймати

таку, що займає не менше 50% часу протягом робочої зміни або робочого дня.

Для попередження передчасної стомлюваності користувачів ПК рекомендується організовувати робочу зміну шляхом чергування робіт з використанням персонального комп'ютера і без нього[31].

При виникненні у працюючих з ПК зорового дискомфорту та інших несприятливих суб'єктивних відчуттів, незважаючи на дотримання санітарно-гігієнічних і ергономічних вимог, рекомендується застосовувати індивідуальний підхід з обмеженням часу роботи з ПК.

У випадках, коли характер роботи вимагає постійної взаємодії з ВДТ (набір текстів або введення даних тощо) з напругою уваги та зосередженості, при виключенні можливості періодичного перемикання на інші види трудової діяльності, не пов'язані з ПК, рекомендується організація перерв на 10 -15 хвилин через кожні 45-60 хвилин роботи. Тривалість безперервної роботи з ВДТ без регламентованого перерви не повинна перевищувати однієї години[32].

При роботі з ПК в нічну зміну (з 22 до 6 год.) незалежно від категорії і виду трудової діяльності тривалість регламентованих перерв слід збільшувати на 30%.

Під час регламентованих перерв з метою зниження нервово-емоційного напруження, стомлення зорового аналізатора, усунення впливу гіподинамії та гіпокінезії, запобігання розвитку позотонічної (статичного) втоми доцільно виконувати спеціально розроблені комплекси вправ.

Користувачам ПК, виконують роботу з високим рівнем напруженості, показана психологічне розвантаження під час регламентованих перерв і в кінці робочого дня в спеціально обладнаних приміщеннях (кімната психологічного розвантаження).

Користувачі автоматизованої системи роботи бібліотеки повинні дотримуватися правил та норм поведінки з комп’ютерною технікою та портативними пристроями. У разі виникнення ситуацій, які суперечать

нормам охорони праці та можуть бути причиною негативних наслідків чи завдати шкоди, потрібно припинити роботу з системою та повідомити керівника роботи чи особу, що відповідає за охорону праці, про порушення та проблеми.

Під час розробки, тестування та впровадження автоматизованої системи були дотримані всі вимоги, норми та державні стандарти з охорони праці.

4.2. Освітлення виробничих приміщень для роботи з ВДТ

Однією із характерних особливостей сучасного розвитку суспільства є зростання сфер діяльності людини, в яких використовуються інформаційні технології, до цих сфер також відноситься автоматизація роботи бібліотеки. Широке розповсюдження отримали персональні комп'ютери. Однак їх використання загострило проблеми збереження власного та суспільного здоров'я, вимагає вдосконалення існуючих та розробки нових підходів до організації робочих місць, проведення профілактичних заходів для запобігання розвитку негативних наслідків впливу ПК на здоров'я користувачів.

Серед причин, що зумовлюють виникнення професійних захворювань користувачів ВДТ, значне місце посідають умови праці.

Перенапруження здорового аналізатора при роботі з ВДТ можуть бути з таких причин:

– неправильна орієнтація робочого місця відносно світлових отворів;

– неадекватні світлові характеристики світильників, неправильне їх просторове розташування відносно робочих місць;

– засліплююча дія яскравих предметів, що перебувають у полі зору користувача (пряма блискість);

– дзеркальне відбиття на екрані предметів з високою яскравістю, які за спиною користувача (відбита блискість):

– неправильний розподіл яскравості в полі зору користувача;

– засвітлення екрана прямим чи розсіяним світлом світильників або небосхилу через світлові отвори[32].

У запобіганні дискомфортним умовам важливе місце належить оптимізації кількісних та якісних показників освітлення.

Приміщення для роботи з ВДТ повинні мати природне та штучне освітлення відповідно до ДБН В.2.5-28-2006 (На заміну СНиП II-4-79).

Природне освітлення має здійснюватись через світлові прорізи, орієнтовані переважно на північ чи північний схід і забезпечувати коефіцієнт природної освітленості (КПО) не нижче ніж 1,5%. Розраховується КПО за методикою, викладеною в ДБН В.2.5-28-2006[33].

За виробничої потреби дозволяється експлуатувати ЕОМ у приміщеннях без природного освітлення за узгодженням з органами державного нагляду за охороною праці та органами і установами санітарно-епідеміологічної служби.

Вікна приміщень з ВДТ повинні мати регулювальні пристрої для відкривання, а також жалюзі, штори, зовнішні козирки тощо.

Штучне освітлення приміщення з робочими місцями, обладнаними ВДТ ЕОМ загального та персонального користування, має бути обладнане системою загального рівномірного освітлення. У виробничих та адміністративно-громадських приміщеннях, де переважають роботи з документами, допускається вживати систему комбінованого освітлення (додатково до загального освітлення встановлюються світильники місцевого освітлення).

Світловий клімат визначає зоровий дискомфорт. Істотне значення має те, в якому світловому поясі розміщується будівля, тому що природне освітлення залежить від кількості сонячних днів у році, снігового покриву взимку та інших факторів. Для врахування цих обставин використовується

поняття світлового клімату. Світловий клімат — сукупність умов природного освітлення в місцевості (освітленість і якість освітлення на горизонтальній та орієнтованих за сторонами горизонту вертикальних поверхнях, що створюються розсіяним світлом неба і прямим світлом сонця, тривалість сонячного сяйва, відбиваючі властивості земного покриву) за період понад десять років.

Запобігти шкідливому впливу освітлення можна шляхом правильного підбору системи освітлення, джерел світла (за їх спектрального складу випромінювання), світильників. Коли штучне світло змішується з природним, рекомендується використовувати лампи за спектральним складом найбільш близькі до сонячного світла. Світильники слід вибирати з розсіювачами, а блискучі деталі освітлювального обладнання, що можуть потрапити в поле зору оператора, повинні бути замінені на матові. Розташовувати робоче місце, обладнане дисплеєм, необхідно таким чином, щоб у полі зору оператора не потрапляли вікна або освітлювальні прилади; вони не повинні знаходиться і безпосередньо за спиною оператора. Світловий клімат може бути поліпшений шляхом встановлення спеціальних антивідблискових контрастних фільтрів, однак при виборі типу фільтра необхідно враховувати умови роботи з комп'ютером, оскільки оптимальні значення коефіцієнтів пропускання і дзеркального відображення фільтрів залежать від освітленості робочого місця і типу джерела світла.

Загальне освітлення має бути виконане у вигляді суцільних або переривчатих ліній світильників, що розміщуються збоку від робочих місць (переважно зліва) паралельно лінії зору працівників. Допускається застосовувати світильники таких класів світлорозподілу:

– світильники прямого світла – П;
– переважно прямого світла – Н;
– переважно відбитого світла – В.

При розташуванні відеотерміналів ЕОМ за периметром приміщення лінії світильників штучного освітлення повинні розміщуватися локально над робочими місцями.

Для внутрішнього оздоблення приміщень з ВДТ слід використовувати дифузно-відбивні матеріали з коефіцієнтами відбиття для стелі 0,7 - 0,8, для стін 0,5 - 0,6. Покриття підлоги повинне бути матовим з коефіцієнтом відбиття 0,3 - 0,5.

Для загального освітлення необхідно застосовувати світильники із розсіювачами та дзеркальними екранними сітками або віддзеркалювачами, укомплектовані високочастотними пускорегулювальними апаратами (ВЧ ПРА). Допускається застосовувати світильники без ВЧ ПРА тільки при використанні моделі з технічною назвою "Кососвітло". Застосування світильників без розсіювачів та екранних сіток забороняється.

Як джерело світла при штучному освітленні повинні застосовуватися, як правило, люмінесцентні лампи типу ЛБ. При обладнанні відбивного освітлення у виробничих та адміністративно-громадських приміщеннях можуть застосовуватися металогалогенні лампи потужністю до 250 Вт. Допускається у світильниках місцевого освітлення застосовувати лампи розжарювання.

Яскравість світильників загального освітлення в зоні кутів випромінювання від 50º до 90º відносно вертикалі в подовжній і поперечній площинах повинна складати не більше 200 кд/м^2, а захисний кут світильників повинен бути не більшим за 40º.

Коефіцієнт запасу ($K_з$) відповідно до ДБН В.2.5-28-2006 для освітлювальної установки загального освітлення слід приймати рівним 1,4.

Коефіцієнт пульсації повинен не перевищувати 5 % і забезпечуватися застосуванням газорозрядних ламп у світильниках загального і місцевого освітлення.

За відсутності світильників з ВЧ ПРА лампи багатолампових світильників або розташовані поруч світильники загального освітлення необхідно підключати до різних фаз трифазної мережі.

Рівень освітленості на робочому столі в зоні розташування документів має бути в межах 300…500 лк. У разі неможливості забезпечити даний рівень освітленості системою загального освітлення допускається застосування світильників місцевого освітлення, але при цьому не повинно бути відблисків на поверхні екрану та збільшення освітленості екрану більше ніж до 300 лк[34].

Світильники місцевого освітлення повинні мати напівпрозорий відбивач світла з захисним кутом не меншим за 40º.

Необхідно передбачити обмеження прямої блискості від джерела природного та штучного освітлення, при цьому яскравість поверхонь, що світяться (вікна, джерела штучного світла) і перебувають у полі зору, повинна бути не більшою за 200 кд/м^2.

Необхідно обмежувати відбиту блискість шляхом правильного вибору типів світильників та розміщенням робочих місць відносно джерел природного та штучного освітлення. При цьому яскравість відблисків на екрані відеотерміналу не повинна перевищувати 40 кд/м^2, яскравість стелі при застосуванні системи відбивного освітлення не повинна перевищувати 200 кд/м^2.

Необхідно обмежувати нерівномірність розподілу яскравості в полі зору осіб, що працюють з відеотерміналом, при цьому відношення значень яскравості робочих поверхонь не повинно перевищувати 3:1, а робочих поверхонь і навколишніх предметів (стіни, обладнання) – 5:1.

Необхідно використовувати систему вимикачів, що дозволяє регулювати інтенсивність штучного освітлення залежно від інтенсивності природного, а також дозволяє освітлювати тільки потрібні для роботи зони приміщення.

Для забезпечення нормованих значень освітлення в приміщеннях з відеотерміналами ЕОМ загального та персонального користування необхідно очищати віконне скло та світильники не рідше ніж 2 рази на рік, та своєчасно проводити заміну ламп, що перегоріли.

Виробниче освітлення. За висновками експертів ВООЗ, під час роботи на ВДТ найбільше навантаження припадає на зоровий аналізатор, тому в забезпеченні роботи користувача важливе місце займає раціональне освітлення робочих місць.

Аварійне освітлення ділиться на два види: освітлення для продовження роботи й для евакуації людей. Освітлення для продовження роботи обладнується у виробничих приміщеннях підприємств, у яких неприпустимі перерви в роботах при відключенні робочого освітлення. Найменша освітленість робочих місць в аварійному режимі повинна становити не менше 5% нормованої робочої освітленості[35]. Аварійне освітлення для евакуації людей встановлюється в місцях, небезпечних для проходу людей, коридорах, на сходових клітках, їдальні, конференц-залах і виробничих приміщеннях. Аварійне освітлення повинне забезпечувати освітленість не менше 0,5 лк на рівні підлоги основних проходів і сходів.

ВИСНОВКИ

В результаті виконання дипломної роботи було розроблено програмне забезпечення адміністрування бібліотечних даних на основі використання системи Koha із вбудованим модулем каталогізації. Методи дослідження предметної області ґрунтувалися на застосуванні аналізу програмних систем, об’єктно-орієнтованого проектування і методології функціонального моделювання.

На етапі проектування було проаналізовано предметну область, описано поставлені завдання, обґрунтовано архітектуру системи, описано варіанти використання та реалізацію програми. На завершальному етапі реалізації описано основні вимоги до використання розробленого ПЗ, виконано тестування готової розробки. В реалізації програми наявні всі заплановані функції та можливості для роботи, реалізовано поставлені задачі.

В якості системи адміністрування даними бібліотеки було спроектовано програмне забезпечення, що містить набір методів та функцій необхідні для автоматизації роботи. Наведено опис основних можливостей обчислення параметрів та функцій моделі. З метою перевірки вірності прийнятих рішень під час розробки, та точності отриманих результатів та загальної ефективності розроблюваної методики було проведено тестування програмного забезпечення методом чорного ящика. Для більш ґрунтовного підходу до дослідження та реалізації необхідних методів було розглянуто основні параметри системи.

Доцільність розробки опирається на обґрунтуванні техніко-економічних показників. Так було розраховано витрати на розробку та впровадження, підтримку проекту. З точки зору організації процесу реалізації у вигляді програмного продукту було обґрунтовано, що об’єктно-орієнтований підхід більш прийнятний для використання в такого типу проектах.

Задля дотримання державних стандартів та норм з Охорони праці у галузі розробки програмного забезпечення, було проаналізовано нормативно-правові акти, правила та норми праці при роботі з персональними комп'ютерами. Було створено та забезпечено всі необхідні умови освітлення приміщень при роботі з ВДТ для розробки та тестування програмного забезпечення.

Даний дослідницький проект та його реалізація виконувався з метою удосконалення програмного забезпечення, що буде використовуватись бібліотеками для адміністрування даних. Програмна модернізація бібліотек є необхідною для сучасного розвиненого суспільства., де особливу увага приділяється вибору системи адміністрування даних, перед використанням їх необхідно модернізувати, щоб вони відповідали всім вимогам, були зручними і зрозумілими у використанні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Автоматизовані бібліотечно-інформаційні системи - Вікіпедія [Електронний ресурс] - Режим доступу: URL: https://uk.wikipedia.org/wiki/Автоматизована_бібліотечна_інформаційна_система

2. Library Technology Guides, LibraryWorld – Librarytechnology [Електронний ресурс] - Режим доступу: URL: https://librarytechnology.org/

3. Library Technology Guides , ABCD: A New Open Source ILS Launched – Librarytechnology [Електронний ресурс] - Режим доступу: URL: https://librarytechnology.org/document/14607

4. About the Evergreen – Evergreen [Електронний ресурс] - Режим доступу: URL: https://evergreen-ils.org/about-us/

5. Koha (software) - Вікіпедія [Електронний ресурс] - Режим доступу: URL: https://en.wikipedia.org/wiki/Koha_(software)

6. Chowdhury, G.G.; Foo, C. (2012). Digital Libraries and Information Access: research perspectives. Chicago: Neal-Schuman. pp. 2–3. ISBN 978-1-55570-914-3. OCLC 869282690.

7. Borgman, Christine L. (2007). Scholarship in the digital age: information, infrastructure, and the Internet. Cambridge: MIT Press. p. 88. ISBN 978-0-262-02619-2. OCLC 181028448.

8. Software development process - Вікіпедія [Електронний ресурс] - Режим доступу: URL: https://en.wikipedia.org/wiki/Software_development_process

9. Rational Unified Process: Best Practices for Software development teams, Rational Software White Paper TP026B, Rev 11/01, IBM

10. Clements, Paul; Felix Bachmann; Len Bass; David Garlan; James Ivers; Reed Little; Paulo Merson; Robert Nord; Judith Stafford (2010). Documenting Software Architectures: Views and Beyond, Second Edition. Boston: Addison-Wesley. ISBN 978-0-321-55268-6.

11. Bass, Len; Paul Clements; Rick Kazman (2012). Software Architecture in Practice, Third Edition. Boston: Addison-Wesley. ISBN 978-0-321-81573-6.

12. Unified Modeling Language (UML), An Introduction – GeeksforGeeks [Електронний ресурс] - Режим доступу: URL: https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/

13. "XHTML 1.0 Specification, Section 1.1: Why the need for XHTML?". World Wide Web Consortium. 2000-01-26. Retrieved 2007-06-16.

14. Perl Language – Webopedia [Електронний ресурс] - Режим доступу: URL: https://en.wikipedia.org/wiki/Perl

15. Apache Web server – Webopedia [Електронний ресурс] - Режим доступу: URL: https://www.webopedia.com/TERM/A/Apache_Web_server.html

16. Demsey, Lorcan; Russell, Rosemary; Kirriemuir, John. (1996, January). Towards Distributed library systems: Z39.50 in a European context, 02, Retrieved December 20, 2002, from URL http://www.aslib.co.uk/program/1996/jan/02.html

17. Lynch, Clifford. (1997, April). The Z39.50 Information Retrieval Standard: Part1: A Strategic View of Its Past, Present and Future, Retrieved January 2, 2003, from URL http://www.dlib.org/dlib/april97/04lynch.html

18. Finnigan, Sonya; Ward, Nigel. Z39.50 Made Simple. Retrieved January 15, 2003, from http://archive.dstc.edu.au/DDU/projects/ZINC/zsimple.htm

19. Fast, powerful Geany editor offers IDE features". Linux.com. Retrieved . "Geany : All Filetypes". Geany. Retrieved 2016-04-03. [Електронний ресурс] - Режим доступу: URL: https://www.geany.org/about/filetypes/

20. "Notepad++ and Scintilla". Retrieved 2019-02-18.

21. Running Geany on Windows. [Електронний ресурс] - Режим доступу: URL: https://wiki.geany.org/howtos/win32/running

22. MySQL Reference Manual [Електронний ресурс] - Режим доступу: URL: https://dev.mysql.com/doc/refman/5.7/en/introduction.html

23. Типи зв'язків у таблицях. Створення зв'язків між елементами в таблицях. Запити. Створення запитів Testing [Електронний ресурс] - Режим доступу: URL: https://works.doklad.ru/view/JfSC0RtpoAY.html

24. Black Box Testing [Електронний ресурс] - Режим доступу: URL: http://softwaretestingfundamentals.com/black-box-testing/

25. "ISTQB Standard glossary of terms used in Software Testing". [Електронний ресурс] - Режим доступу: URL: http://glossar.german-testing-board.info/v3.21/

26. Закон України «Про збір та облік єдиного внеску на загальнообов'язкове державне соціальне страхування» №2464-VI від 08.07.2010

27. Методичні вказівки для виконання розділу дипломної роботи щодо техніко-економічного обґрунтування вибору проектного рішення розробки та оцінки якості програмного забезпечення / Упор. Петрик М.Р., Кінах Я.І., Головатий А.І., Рогатинська Л.Р. – Тернопіль: Вид-во ТНТУ ім. І. Пулюя. – 2013. – 34 с

28. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [Електронний ресурс] - Режим доступу: URL: https://zakon.rada.gov.ua/rada/show/v0007282-98

29. Наказ 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [Електронний ресурс] - Режим доступу: URL: https://zakon.rada.gov.ua/laws/show/z0508-18#n14

30. Охорона праці в галузі. Конспект лекцій - ELARTU [Електронний ресурс] - Режим доступу: URL http://elartu.tntu.edu.ua/bitstream/123456789/17891/1/KonspektOPG.pdf

31. Психологічне забезпечення складних систем діяльності [Електронний ресурс] - Режим доступу: URL http://maup.com.ua/assets/files/lib/book/psihol_zabezp_skl.pdf

32. Формування поняття про професійні захворювання користувачів ПК в процесі підготовки фахівців С.В. Дембіцька, М.С. Фурман. 2018.

33. Природне і штучне освітлення [Текст]: ДБН В.2.5-28-2006. - Київ : Мінбуд України, 2006. 96 с.

34. Освітлення виробничих приміщень [Електронний ресурс] - Режим доступу: URL: https://buklib.net/books/35234/

35. Аварійне освітлення[Електронний ресурс] - Режим доступу: URL: http://eet.ua/ua/publication/ua-avariyne-osvitlennya-vse-shho-varto-z/

ДОДАТКИ

Додаток А

Технічне завдання

Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп’ютерно-інформаційних систем і програмної інженерії

Кафедра програмної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедрою

програмної інженерії

“___ “_____ 2019 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської дипломної роботи

на тему: «Розробка програмного забезпечення адміністрування бібліотечних даних на основі використання системи Koha із вбудованим модулем каталогізації.»

Стадник Надії Іванівні <СПм-15-300> ТЗ

Керівник роботи:
к.т.н., доцент Кінах Я.І.
“___” _____ 2019р.
_____

Виконавець:
студентка групи СПм-62
Стадник Надія Іванівна
“___” _____ 2019р.
_____

м. Тернопіль – 2019

ЗМІСТ

1 ПІДСТАВИ ДО РОЗРОБКИ

Розробка проводиться у відповідності до графіку навчального плану на 2019 рік, та згідно наказу на виконання дипломної роботи студента-магістра.

Тема проекту: «Розробка програмного забезпечення адміністрування бібліотечних даних на основі використання системи Koha із вбудованим модулем каталогізації.».

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Дипломна робота присвячена створенню програмного забезпечення адміністрування бібліотечних даних на основі використанням системи Koha з вбудованим модулем каталогізації.

Предметом дослідження: є програмна система керування даними, що використовує бази каталогів для удосконалення спеціалізованого програмного забезпечення.

Мета роботи підвищити швидкодію програмного забезпечення, що використовується бібліотеками для адміністрування даних, використовуючи при проектуванні систему Koha з вбудованим методом каталогізації.

За результатами виконаної роботи необхідно отримати програмне забезпечення для автоматизованого ведення даних про роботу бібліотеки, що заснована на певній моделі

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Функціональні характеристики

Програмне забезпечення має виконувати наступні дії:

– створення облікових записів для користувачів системи;

– ведення бібліографічних записів системи та їх редагуванні і перегляд за необхідності;

– керування каталогом бібліотеки, з можливістю внесення змін до нього;

– перегляд і керування обігом бібліотеки (видача і отримання повернених книг);

– пошук інформації про користувача та перегляд інформації про його дії в системі;

– генерування необхідних звітів для можливості аналізу роботи користувачів з програмою.

3.2 Склад та параметри технічних засобів

ПК з 8 Гб оперативної пам‘яті, встановленою системою Windows, Linux. Не менше 1000 Мб вільного місця на жорсткому диску. Двоядерний процесор з тактовою частотою від 2.4 GHz і більше.

3.3 Інформаційна та програмна сполучність

Програмний продукт повинен коректно функціонувати в різних операційних системах. Розроблювана програмна система повинна бути пристосована до використання у різних бібліотеках. Розробку виконувати з використанням мови XHTML, середовище розробки Geany, базами даних MySQL, веб-сервер Apache.

4. СТАДІЇ РОЗРОБКИ

В ходів реалізації роботи проект повинен пройти крізь наступні стадії розробки:

- аналіз предметної області;
- проектування архітектури;
- реалізація класів і модулів;
- тестування результатів розробки;
- оформлення супровідної документації;
- здача роботи.

5. ПРОГРАМНА ДОКУМЕНТАЦІЯ

Для програмного продукту повинні бути розроблені наступні документи:

- Пояснювальна записка;
- Технічне завдання;
- Презентаційний матеріал;
- Інструкція користувача;
- Додатки.

6. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Розроблений програмний продукт має виконувати всі вимоги, що складаються з перерахованих у п. 3.1 характеристик.

Приймання проводиться спеціально створеною екзаменаційною комісією в термін до:

“23” грудня 2019р.

Додаток Б

Тези

УДК 004.422.83
Н.І. Стадник – магістрантка
Тернопільський національний технічний університет імені Івана Пулюя, Україна
Я.І. Кінах – кандидат технічних наук, доцент
Тернопільський національний технічний університет імені Івана Пулюя, Україна

УДОСКОНАЛЕННЯ ПРОГРАМНИХ СИСТЕМ АДМІНІСТРУВАННЯ БІБЛІОТЕЧНИХ ДАНИХ

N.I. Stadnyk – magistrate, I.I. Kinakh – Ph.D, Assoc. Prof.
IMPROVING SOFTWARE SYSTEMS FOR ADMINISTRATION OF LIBRARY DATA

Сучасна бібліотека – є віртуальною, де користувач має змогу знайти не тільки те, що складає фонд даної бібліотеки, але й дає можливість знайти будь яку інформацію із будь якої бібліотеки світу. Тому є необхідність в модернізації роботи сільської та шкільної бібліотеки, забезпеченні її сучасними комунікаційними технологіями. З розвитком інформаційних технологій з'явилось безліч програмних систем для адміністрування бібліотечних даних [1], в тому числі і вільного відкритого програмного забезпечення. Для прикладу розглянемо найпопулярніші з них. Програмна система типу Koha - перша вільна автоматизована бібліотечна інформаційна система (АБІС), призначена для підтримки традиційних бібліотечних технологічних процесів. Створена на основі бібліотечних стандартів та протоколів, що забезпечує можливість взаємодії між системою Koha та іншими бібліотечними системами і технологіями. Koha включає більшість очікуваних можливостей програмного забезпечення АБІС, зокрема: інтерфейс для бібліотекарів і читачів (відвідувачів), модуль каталогізації з вбудованим клієнтом Z39.50, модуль керування читачами, списки прочитаного для відвідувачів. АБІС Koha переважно використовується для ОС типу Linux. Теоретично вона може бути сумісною із ОС типу Windows, але це вимагає складного встановлення декількох додаткових програмних модулів [2]. Програмна система типу Evergreen - інтегрована бібліотечна система, розроблена консорціумом PINES, для застосування у великій публічній бібліотеці та мережі бібліотек корпоративного типу. Програмна система Evergreen забезпечує підтримку повсякденних бібліотечних операцій таких, як облік фондів, реєстрація читачів і надання доступу до онлайн каталогу. Evergreen має можливості обчислення і контролю термінів видачі і повернення документів. Система дає змогу відстежувати де знаходиться будь-яка книга, компакт-диск або будь-який інший документ бібліотеки. Evergreen має дружній веб-інтерфейс бібліотечного каталогу, який дозволяє читачеві знайти все, що йому потрібно незалежно від того, де це знаходиться.

Підсумовуючи весь матеріал можна дійти висновку, що програмна модернізація бібліотек є необхідною для сучасного розвиненого суспільства. Особливу увагу потрібно приділити вибору системи адміністрування даних. Проте перед використанням їх необхідно модернізувати, щоб вони відповідали всім вимогам, були зручними і зрозумілими у використанні та встановленні не тільки в професіоналів, але і в звичайних користувачів.

Література

1. Проблема програмного забезпечення для функціонування електронної бібліотеки / Іванова С. М. [Електронний ресурс]. – Режим доступу: http://nbuv.gov.ua/sites/default/files/method_mg/mfiles/201410_method/2009.pdf
2. Юдін О.К. Кодування в інформаційно-комунікаційних мережах: – Монографія. - К.:НАУ,. 2007.-308с.

Додаток В

Диск