НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра автоматики та управління в технічних системах

Курсова робота

На тему:

Книжковий каталог

З дисципліни «Розробка проблемно-орієнтованих та сервісно-орієнтованих систем»

Керівник Виконавець

Корнієнко Б.Я. ст. Козак О.С.

«Допущений до захисту» зал. Книжка №

гр. ІТ-з03мп

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Особистий підпис керівника) (Особистий підпис виконавця)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 р.

Захищений з оцінкою

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оцінка)

Члени комісії:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Особистий підпис) (Розшифровка підпису)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Особистий підпис) (Розшифровка підпису)

Київ - 2020

#### АНОТАЦІЯ

В курсовій роботі розглядається питання автоматизації ведення книжкового каталогу та пошуку потрібних матеріалів в книжковому каталозі. Графічний інтерфейс програми був реалізований таким чином, щоб користувачу було зручно її застосовувати. Програма розроблена як клієнт-серверний застосунок з використанням технології WCF Microsoft. Серверна частина застосунку розроблена як розподілена обчислювальна система Побудована об’єктна модель з чисельними зв’язками.

Програма реалізована мовою програмування високого рівня C# в середовищі Microsoft Visual Studio 2019 з використанням інтерфейсу Windows Presentation Foundation. Застосунок налічує у собі ряд зручностей, таких як: створення книги, пошук книги, створення замовлення.

Пояснювальна записка виконана на 51-ній сторінці, містить 3 додатки, 24 рисунки.

ЗМІСТ

ВСТУП

На сьогоднішній день існує незначна кількість електронних книжкових каталогів, які повною мірою можуть задовольнити всі потреби користувачів і при цьому залишатися зручними у використанні. Наразі проблема замовлення книги є доволі значною.

**Метою** курсової роботи є ознайомлення з можливостями взаємодії додатків на основі веб-сервісів та розробка власної автоматизованої системи ведення електронного книжкового каталогу та пошуку потрібних матеріалів в книжковому каталозі. На розробку застосунку в певній мірі вплинули, такі існуючі електронні книжкові каталоги, як електронний каталог Національної бібліотеки ім. В.І. Вернадського(див), електронний каталог Державної науково-технічної бібліотеки(див), Науково-технічної бібліотеки ім. Г.І. Денисенка (див) та електронного каталогу сервісу «ScienceDirect» (див). Ознайомившись із принципом їхньої роботи та узагальнивши основні аспекти цих сервісів, я використала набуті знання для створення своєї системи.

Реалізація проекту основана на використанні технології WCF Microsoft, використанні інтерфейсу Windows Presentation Foundation. Програма розроблена в середовищі Microsoft Visual Studio 2019.

ПОСТАНОВКА ЗАДАЧІ

Головною метою програми є створення автоматизованої системи ведення електронного каталогу та пошуку потрібних матеріалів в книжковому каталозі.

Також під час виконання курсової роботи планується виконати такі **задачі**:

* організувати зручний користувацький інтерфейс таким чином, щоб користування програмою було зручним і ефективним, використовуючи систему побудови клієнтських додатків Windows Presentation Foundation;
* реалізувати можливість додавання та редагування книг, статей, періодичних видань та дисертацій;
* реалізувати зручний пошук та доступ до бажаної користувачем інформації про документ, який шукає користувач;
* реалізувати зручне оформлення замовлень користувачем та відслідковування цих замовлень адміністратором;
* Використання технології WCF Microsoft;
* окрім того, реалізувати серверну частину застосунку у вигляді розподіленої системи;
* реалізувати таку архітектуру клієнтської частини застосунку, використовуючи шаблон MVVM (Model-View-ViewModel), таким чином, щоб логіка додатку була відокремлена від візуальної частини (представлення).

ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Аналіз аналогічних програм

В сучасному світі існує кілька сервісів пов’язаних веденням книжкового каталогу. Такі сервіси дають змогу знайти потрібну клієнту наукову або ж художню літературу. Скориставшись електронним пошуком користувач може з легкістю перевірити наявність книги в бібліотеці та оформити замовлення в читальний зал або ж на видачу. Деякі сервіси дають можливість лише переглянути наявність книги в бібліотеці для того, щоб скористатися нею в читальному залі, інші ж дозволяють замовити книгу для користування на певний період. Такі сервіси значно допомагають у пошуку інформації про книгу, або ж скорочують час, який витрачається на замовлення книги в бібліотеці. З переглянутих мною сервісів хочу зазначити такі:

1. Електронний каталог Національної бібліотеки імені В.І. Вернадського

Гарним прикладом електронного каталогу є каталог Національної бібліотеки імені В.І. Вернадського(рис. 3.1).

Головними перевагами даного каталогу є:

* дуже велика база документів, яка вміщує книжки, періодику, картографічні видання, ноти та музичні матеріали, рідкісні видання та ін.;
* зручний та інтуїтивний пошук;
* надається детальна інформація про книгу чи інший документ;
* пошук по категоріях;
* наявність великої кількості оцифрованих видань, доступ до яких можна мати безпосередньо на сайті;
* можливість використовувати сервіс з мобільних пристроїв;
* сайт перекладений надзвичайно великою кількістю мов.

Поряд із даними перевагами є недоліки:

* електронний пошук може здійснюватися лише а одним із критеріїв;
* зазвичай є можливість лише переглянути наявність книги в бібліотеці та переглянути інформацію або місце зберігання та немає опції замовлення книги;
* не інтуїтивна сторінка з отриманими результатами.



Рисунок 3.1 – Сторінка електронного каталогу Національної бібліотеки ім. В.І. Вернадського

2. Електронний каталог державної науково-технічної бібліотеки України

Цей сервіс дещо відрізняється від попереднього тим, що менш націлений на пошук всіх можливих тематик та жанрів книг і надає інформацію про науково технічну літературу, проте в інтерфейсі користувача у них є подібності.

Позитивні риси книжкового каталогу Державної наукової бібліотеки (рис. 3.2) в наступному:

* детальна та обґрунтована інформація;
* широка база даних науково-технічної літератури;
* розширений пошук дає можливість пошуку абсолютно будь-якої науково-технічної літератури, фільтруючи по галузях науки;
* зручний та швидкий пошук.

Проте наявні і значні недоліки:

• немає можливості замовити книгу онлайн.

• немає цифрової бібліотеки.

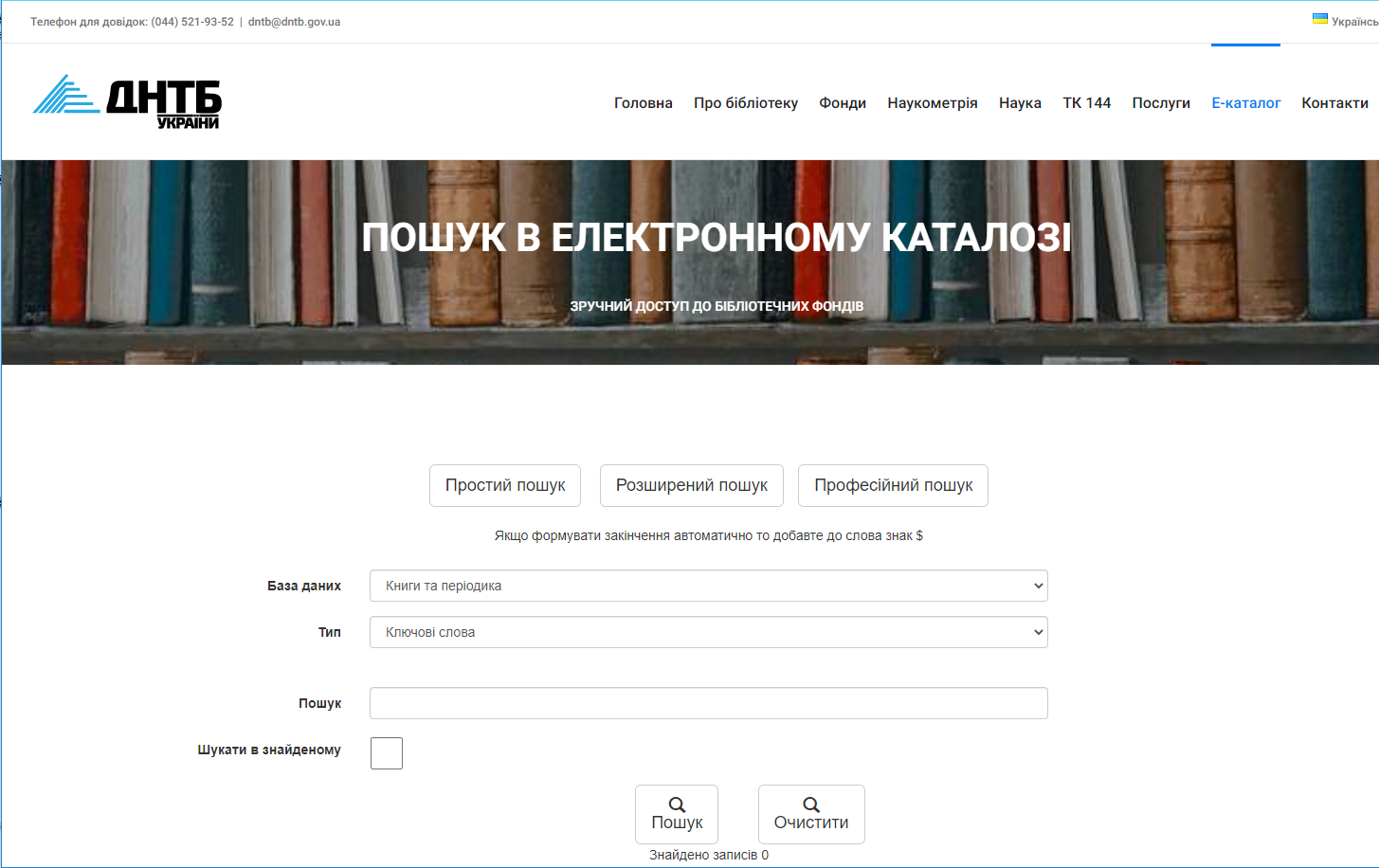


Рисунок 3.2 – Електронний каталог Державної науково-технічної бібліотеки

Електронний книжковий каталог Науково-технічної бібліотеки ім. Г.І. Денисенка

Позитивні риси книжкового каталогу Науково-технічної бібліотеки ім. Г.І. Денисенка (рис. 3.3) в наступному:

* безкоштовний та широкий доступ для користувача;
* зручність та легкість використання;
* зручний та інтуїтивний пошук по категоріях;
* можливість замовлення книги, зареєструвавшись, як читач бібліотеки;
* надзвичайно детальна та розширена інформація про книгу або інший документ.

Недоліками є те, що:

* немає оцифрованої бази книжок;
* читачем бібліотеки можна стати лише за наявністю студентського квитка.

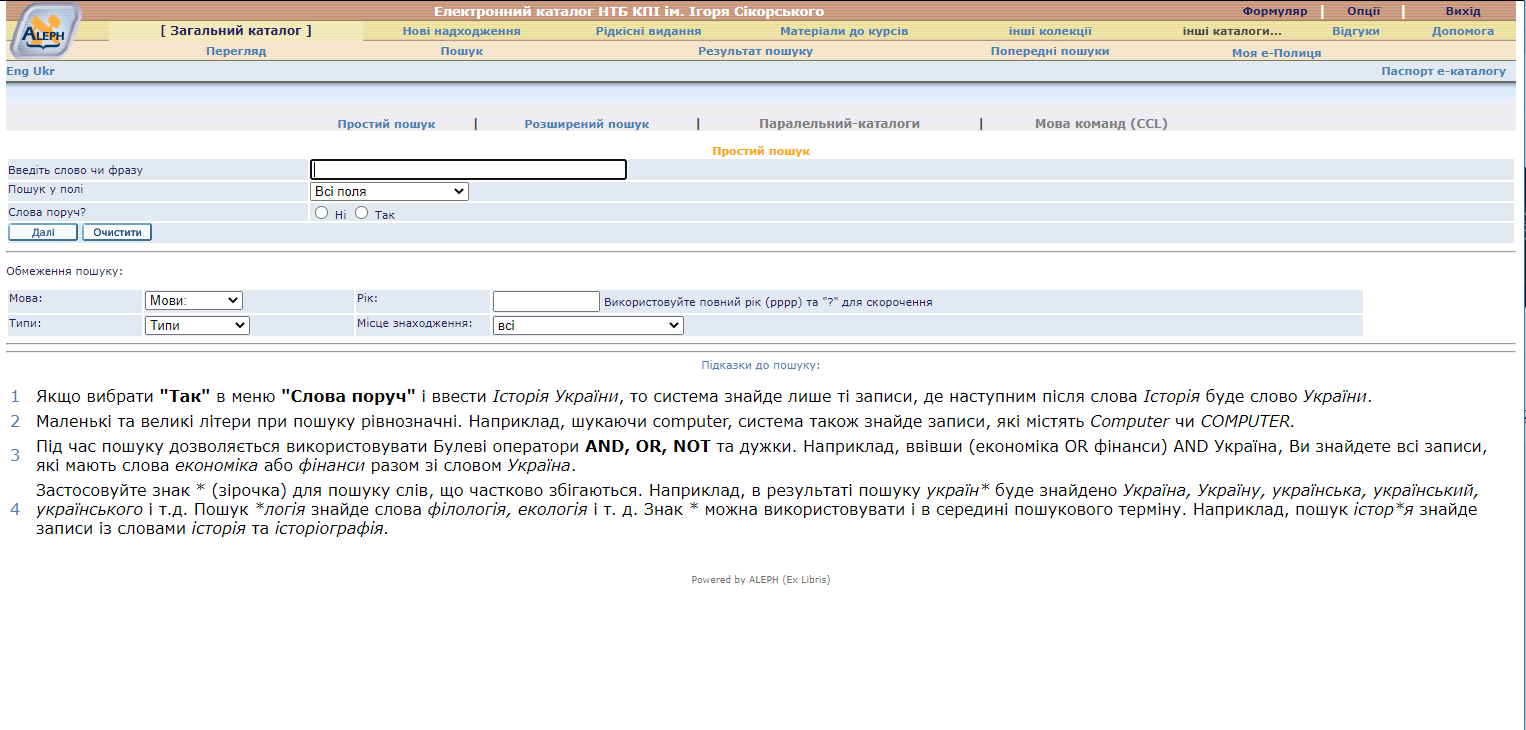


Рисунок 3.3 – Електронний каталог Науково-технічної бібліотеки ім. Г.І. Денисенка

Електронний каталог сервісу “ScienceDirect” є пошуком англомовної науково-технічної літератури.

Головними перевагами є:

* можливість доступу до науково-технічної літератури онлайн за умови реєстрації від певної організації або ж купивши платну підписку;
* велика база технічних та наукових зарубіжних джерел;
* зручний та інтуїтивний пошук по категоріях.

Недоліки:

* сервіс надає доступ лише до англомовних ресурсів;
* сервіс є платним та доволі дорогим.

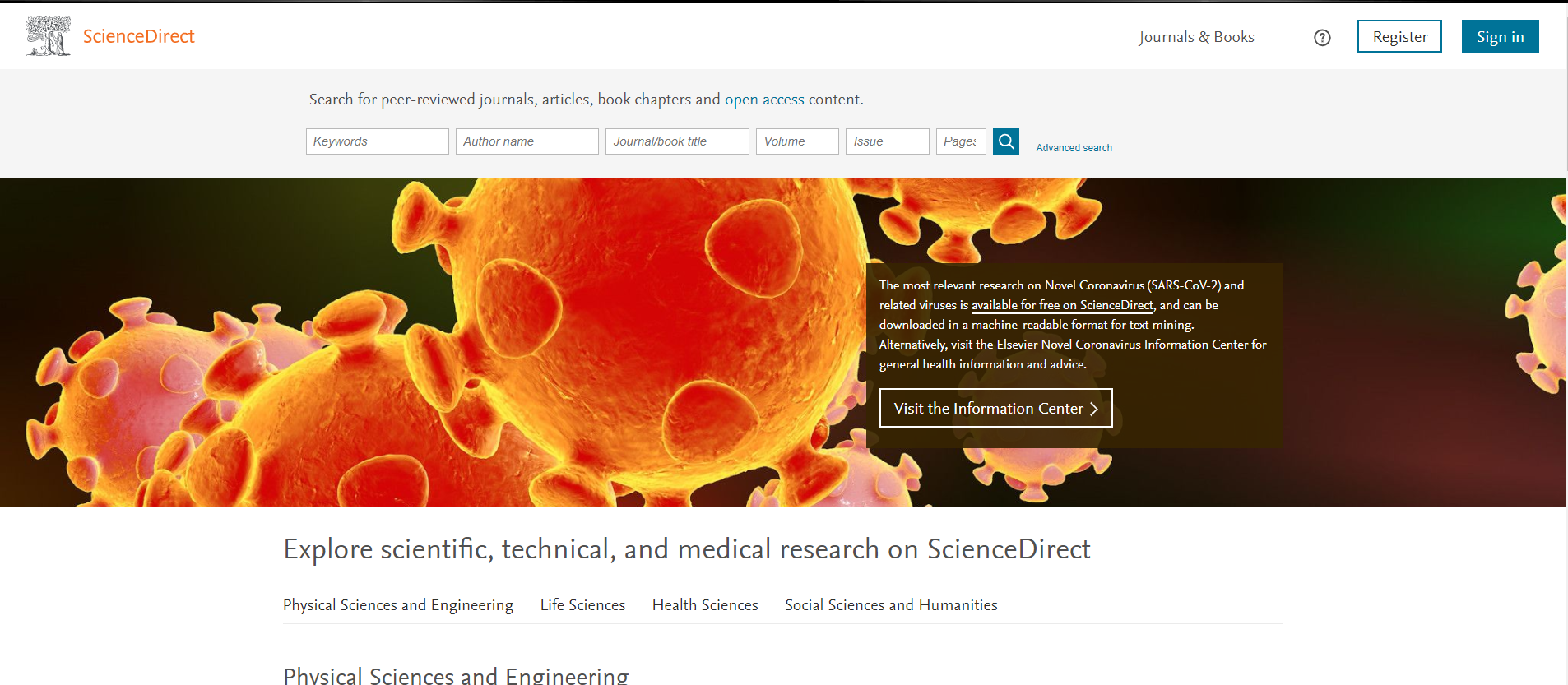


Рисунок 3.4 – Сторінка електронного каталогу “ScienceDirect”

СТРУКТУРА ЗАСТОСУНКУ ТА ЗАГАЛЬНИЙ ОПИС СЕРВЕРНОЇ ЧАСТИНИ ЗАСТОСУНКУ



Рисунок – Інфографічне представлення розподіленої обчислювальної системи(Взяв тут https://www.confluent.io/learn/distributed-systems/)

Серверна частина застосунку «» розроблена як розподілена обчислювальна система, набір незалежних комп'ютерів, що представляється їх користувачам єдиною об'єднаною системою. Кожен елемент розподіленої системи (мережі) – автономна обчислювальна машина єдиними вимогами від якої є можливість її виявлення (Service discovery) і створення каналу обміну даними по мережі з нею. Таким чином дана обчислювальна машина може бути як фізичною машиною так і віртуальною; бути розміщеною як на приватних (корпоративних) обчислювальних машинах (мережах) так і на хмарних сервісах. Розподілені обчислювальні системи надають змогу розробляти окремі її елементи незалежно, використовуючи діаметрально різні технологічні стеки (за необхідності), а також різними командами розробників. Розподілені обчислювальні системи відкривають можливість будувати відмовостійкі, надійні та доступності системи: у разі відмови обчислювальної машини решта елементів розподіленої системи буде активною, елемент системи що відмовив можна відновити, розгорнувши його на новій обчислювальній машині на постійній або тимчасовій основі (поки обчислювальна машина що відмовила не буде відновлена), даний процес у певних випадках може бути автоматизований, що підвищує доступність системи, адже не накладаються часові затримки на відновлення системи, пов’язані з застосуванням людських ресурсів.



Рисунок – Логотип Microsoft WCF. (Взяв тут https://blog.tonysneed.com/2016/01/06/wcf-is-dead-long-live-mvc-6/)

Кожен елемент розподіленої системи (надалі сервіс) розроблено з використанням технології Windows Communication Foundation (WCF) (<https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf> посилання якщо треба). WCF – це фреймворк для побудови розподілених сервісно-оріентованих застосунків. WCF значно прискорює розробку мережевих додатків. Фремворк дозволяє цілковито абстрагувати код застосунку від протоколів спілкування по комп’ютерній мережі, встановлення з’єднання підключення з клієнтом сервіса, форматів повідомлень, сереалізації та десереалізації вхідних/вихідних повідомленнь, криптографічного захисту даних, перевірці даних на справжність та коректність. Завдяки наданому функціоналу наведеному вище, взаємодія з клієнта з сервісом, з точки зору написання коду, зводиться до простого виклику методу. Також варто відзначити, що великою перевагою WCF є всеосяжна детальна і зрозуміла документація, як від постачальника технології (Microsoft) так і від спільноти спеціалістів що застосовують дану технологію.

Протоколом спілкування по комп’ютерній мережі був обраний протокол Simple Object Access Protocol (SOAP), використаний разом з протоколом прикладного рівня Hypertext Transfer Protocol (HTTP), як транспортним протоколом. Сервіси обмінюються між собою повідомленнями за специфікацією SOAP (https://www.w3.org/TR/soap/ посилання якщо треба) версії 1.2 (мал. №).

POST **/InStock** **HTTP**/1.1

Host: www.example.org

Content-Type: application/soap+xml; charset=utf-8

Content-Length: 299

SOAPAction: "http://www.w3.org/2003/05/soap-envelope"

<?xml version="1.0"?>

**<soap:Envelope** xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:m="http://www.example.org"**>**

**<soap:Header>**

**</soap:Header>**

**<soap:Body>**

**<m:GetStockPrice>**

**<m:StockName>**T**</m:StockName>**

**</m:GetStockPrice>**

**</soap:Body>**

**</soap:Envelope>**

мал. № . Приклад SOAP повідомлення-запиту надісланого по HTTP протоколу (взяв тут https://en.wikipedia.org/wiki/SOAP)

Дані повідомлення є ієрархічно структурованими за стандартом World Wide Web Consortium - розширювана мова розмітки (англ. Extensible Markup Language, скорочено XML) (<https://www.w3.org/TR/xml/> посилання якщо треба). Завдяки використанню XML, повідомлення набувають наступних якостей:

* Коректність – будь-яке XML повідомлення може бути перевірене на предмет вірності його побудови, відповідності його стандарту XML. Обробка некоректних повідомлення має бути автоматично відхилена.
* Практичність – у платформі .NET (а також у багатьох інших) доступні синтаксичні аналізатори (парсери, від англ. parser), програмні компоненти, що аналізують і валідують формат формат повідомлення XML і трансформують такі повідомлення у об’єктну модель.
* Функціональна сумісність – XML повністю відкритий стандарт, а отже споживати повідомлення такого формату може будь-яка комп’ютерна система, тобто досягається багатоплатформність кліентів.

# **Опис структури застосунку**

Розподілена обчислювальна система «» утворюється з наступних компонентів (див мал №): клієнтського застосунку (детальний опис у розділі «»), сервіс «IronMacbeth.UserManagement», сервіс «IronMacbeth.FileStore», сервіс «IronMacbeth.BFF».



Мал №. Структура та основні компоненти застосунку

## **Сервіс «IronMacbeth.UserManagement»**

Відповідальність сервісу «IronMacbeth.UserManagement» (надалі у цьому підрозділі – сервіс) полягає у автентифікації користувачів, реестрації нових користувачів у системі, а також у збереженні особистих даних користувача. Сервіс надає функціонал валідації відповідності користувацького логіна паролю.

### **Механізм валідації імені та паролю користувача**

Зберігати користувацькі паролі у чистому вигляді (plain text) небезпечно, а також не відповідає загальному регламенту про захист даних Європейського союзу (англ. General Data Protection Regulation, GDPR; Regulation (EU) 2016/679) (взяв тут <https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9_%D1%80%D0%B5%D0%B3%D0%BB%D0%B0%D0%BC%D0%B5%D0%BD%D1%82_%D0%BF%D1%80%D0%BE_%D0%B7%D0%B0%D1%85%D0%B8%D1%81%D1%82_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85>). У разі маловірогідної хакерської атаки або потрапляння ключів доступу до бази даних сервісу, зловмисники зможуть отримати прямий доступ до користувацьких паролів. Для вирішення даної проблеми у сервісі реалізований наступний фунціонал: отриманий при реєстрації користувача, пароль хешується з додаванням так званої солі (англ. Salt), отриманий хеш разом із сіллю зберігається у базі даних; при повторних авторизаціях користувача, наданий пароль хешується з сіллю, що була попередньо збережена у базі даних, отриманий хеш перевіряється з хешем у базі даних; якщо встановлена побітна відповідність двох хешів – користувач надав правильних пароль, якщо ні – надані ім’я та пароль користувача не відповідають один одному (див додаток «»).

Сіль – сгенерована випадковим чином послідовність даних, основною ціллю якої є приховування однакових паролів. Наприклад, два користувачі обрали один і той самий пароль, в такому разі в базі даних для них будуть збережені однакові хеші. Сіль нівелює дану вразливість системи, так як, для користувачів будуть сгенеровані різні послідовності, що унеможливить виявити однакові паролі у користувачів.

Хеш-функція – функція що перетворює вхідні дані будь-якого (як правило великого) розміру в дані фіксованого розміру, що не має зворотної (функція яка б перетворювала хеш у оригінальне значення). У сервісі застосована хеш функція SHA256. На момент написання роботи, дана хеш-функція вважається стійкою, та такою що не має вразливостей.

## **Сервіс «IronMacbeth.FileStorage»**

Сервіс «IronMacbeth.FileStorage» (надалі у цьому підрозділі – сервіс) відповідальний за збереження неструктурованих бінарних файлів. Даний сервіс абстрагує спосіб збереження файлів від решти розподіленої системи. Наприклад файли можуть зберігатись на дисковому носії інформацій або у хмарному сховищі; файли можуть зберігатись як заархівовані чи у оригінальному стані, чи комбінований варіант, коли часто використовувані файл архівуються і зберігаються у холодному сховищі (такому що має великий обсяг, проте невисоку швидкодію) або у гарячому (швидкодіючий носій інформації проте більш дорогий), тощо; зашифровані чи ні (наприклад файли з вразливою інформацією шифруються, а ті що були отримати з загальнодоступних джерел - ні).

Сервіс надає наступний прикладний програмний інтерфейс (англ. Аpplication Programming Interface, API) виражений інтерфейсом C# (див мал №.).

using System;

using System.IO;

using System.ServiceModel;

namespace IronMacbeth.FileStorage.Contract

{

[ServiceContract]

public interface IFileStorageService

{

[OperationContract]

Guid AddFile(Stream fileStream);

[OperationContract]

Stream GetFile(Guid fileId);

}

}

Рисунок – API Сервісу «IronMacbeth.FileStorage»

Унікальними ідентифікаторами файлів будо обрано GUID (англ. Globally Unique Identifier). GUID – ідентифікатор розмірністю 16 байт з загальною кількістю унікальних ключів рівною 2128. Ймовірність того, що у світі будуть незалежно сгенеровані два однакові ідентифікатори вкрай мала. Таким чином сервіс, в подальшому, за необхідності, можна піддати як вертикальному так і горизонтальному масштабуванню (мал №.).

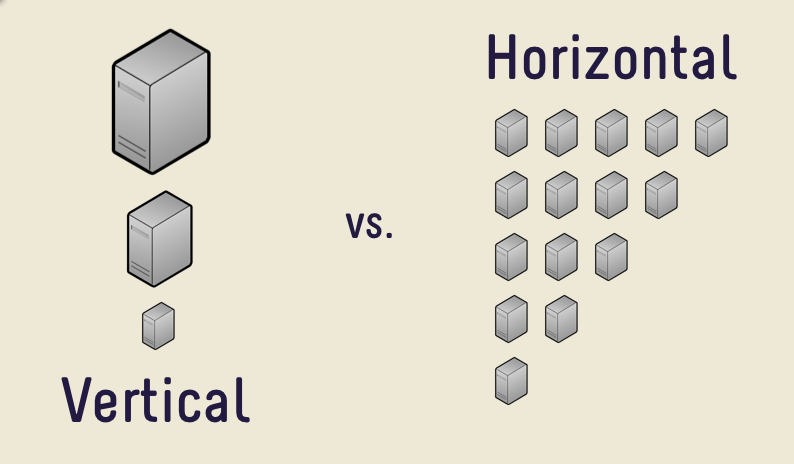


Рисунок – Інфографічне представлення вертикального і горизонтального масштабування.

Вертикальне масштабування – нарощування комп’ютерних ресурсів (процесора, оперативної пам’яті, носіїв інформації) певної обчислювальної машини.

Горизонтальне масштабування – нарощування комп’ютерних ресурсів розподіленої системи шляхом введення додаткових обчислювальних машин, з розподіленням навантаження між існуючими і доданими обчислювальними машинами. Даний тип масштабування дозволяє досягти більшої гнучкості у порівнянні з вертикальним масштабуванням, адже не тільки загальна обчислювальна здатність горизонтально масштабованих систем значно перевищує вертикально масштабовані, а ще й збільшує стійкість системи (якщо одна обчислювальна машина з низки горизонтально масштабованих вийде з ладу, користувачі лише відчують зниження швидкості виконання операцій, але не втратять можливість користуватись функціоналом, як це відбувається з вертикально масштабованими обчислювальними машинами); також можливо досягти ефекту підвищеної ефективності системи завдяки розміщенню додаткових горизонтально масштабованих обчислювальних машин безпосередньо у фізичному регіоні де швидкодія системи найбільш важлива.

## **Сервіс «IronMacbeth.BFF».**

Сервіс «IronMacbeth.BFF» (BBF - скорочено від Backend For Frontend) (надалі у цьому підрозділі – сервіс) виконує роль абстракції розподіленої системи від кліента, обслуговує запити кліента, переспрямовуючи їх до інших сервісів за необхідності.

Було прийнято рішення використовувати архітектурний підхід Backends For Frontends для можливості в подальшому розширяти додаток для підтримки веб та мобільної версії застосунку які також будуть потребувати абстрагування від розподіленої системи проте іншого інтерфейсу взаємодії (див мал №.).

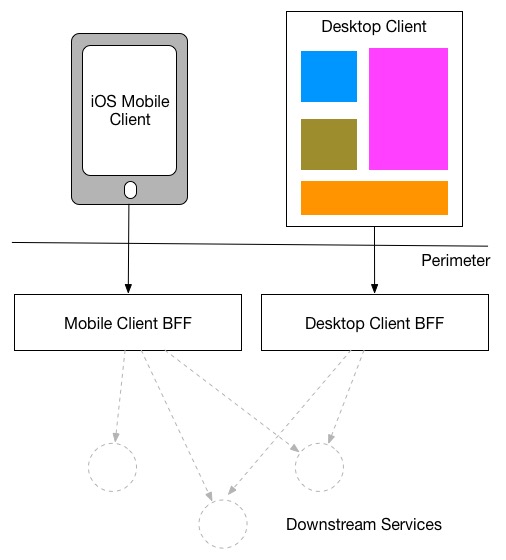


Рисунок – Інфографічне представлення архітектури розподіленого застосунку з використанням підходу Backends For Frontends.

Підхід Backends For Frontends дозволяє значно спростити розробку клієнтського додатку, адже клієнтська частина застосунку буде розвантажена від взаємодії з розподіленою системи, а також споживати прикладний програмний інтерфейс спеціально націлений на потреби клієнтського застосунку (BFF сервіс агрегує та трансформує дані, отримані від певного переліку інших обчислюваних машин розподіленої системи, у вигляд, який найбільш підходить під потреби кліента).

## **Методи забезпечення цілісності та шифрування даних, якими обмінюються компоненти**

У системах, що базуються на архітектурі розподілених застосунків, невід’ємною частиною штатного функціонування системи є спілкування по мережі. Повідомлення надіслане від одного мережевого пристрою, що проходить по мережі може бути зчитане третьою стороною, таким чином конфіденційні дані можуть бути скомпрометовані. Також третя сторона може виконати атаку «людина посередині» (англ. Man-in-the-middle attack), а саме перехопити повідомлення відправника, видозмінити його і відправити повідомлення отримувачу начебто від імені оригінального відправника.

Для запобігання негативних наслідків розподілених систем, наведених вище, у застосунку використовується шифрування даних та їх цифрове підписання, що засвідчує цілісність даних та їх відправника. Даний функціонал досягається завдяки використанню серверного сертифікату інфраструктури відкритого ключа (англ. Public Key Infrastructure Certificate) стандарту X.509 (посилання <https://tools.ietf.org/html/rfc5280>).

Фунціоналу захисту даних серверним сертифікатом було досягнуто шляхом використання стандартних механізмів, наданих фреймворком WCF.

## **Модель авторизації між компонентами**

У застосунку встановлена наступна модель авторизації між компонентами:

* IronMacbeth.Client – IronMacbeth.BFF. IronMacbeth.Client (надалі, у даному підрозділі, «кліент») має два типи з’єднання з IronMacbeth.BFF (надалі, даному підрозділі, «сервіс»): анонімне і авторизоване. Анонімне з’єднання надає користувачу значно обмежений функціонал і застосовується для того щоб користувач міг зареєструватись у системі. Авторизоване з’єднання вимагає від користувача коректну пару імені та паролю. Обидва з’єднання захищені серверним сертифікатом.
* Взаємодія між сервісами «IronMacbeth.UserManagement», «IronMacbeth.FileStore» та «IronMacbeth.BFF» (надалі, у даному підрозділі, «сервіси»). Взаємодія між сервісами захищена шляхом використання сертифікатів. Сервіс, що ініціює підключення до іншого сервісу, має надати свій сертифікат сервісу, до якого він ініціював підключення, наданий сертифікат має бути у списку сертифікатів яким довіряє сервіс до якого встановлюється підключення; сервіс, до якого встановлюється підключення, надає свій сертифікат сервісу, що ініціює підключення, для його перевірки. Якщо обидва сертифікати успішно проходять перевірку, встановлюється з’єднання, захищене сертифікатом сервісу який виступає у ролі сервера (сервіс до якого ініціюється підключення).

ЗАГАЛЬНИЙ ОПИС КЛІЄНТСЬКОЇ ЧАСТИНИ ЗАСТОСУНКУ

Компонент розподіленої системи, що відіграє роль клієнтського застосунку – «IronMacbeth.Client» (надалі, у цьому розділі, «клієнт»). Інформація щодо функціоналу та шляхів застосування вказана у розділі «».



Рисунок – Логотип Windows Presentation Foundation

Клієнт розроблений за технологією WPF (Windows Presentation Foundation) (мал № .). WPF – платформа для створення застосунків з графічним користувацьким інтерфейсом. WPF був розроблений на заміну технології Windows Forms. У порівнянні з Windows Forms, WPF дозволяє розробку сучасних графічних інтерфейсів з використанням адаптивної розмітки; має значну перевагу у тому, що дозволяє розробляти правильно структуровані програми, у яких відображення логічно відділене від моделі, бізнес-логіки і контенту (завдяки використанню прив’язок (англ. Binding) та контексту даних (англ. DataContext)); апаратне прискорення, підтримка мультисенсорного вводу; включає в себе підтримку декларативної мови розмітки XAML (англ. eXtensible Application Markup Language).

XAML дозволяє створювати витончені та комплексні графічні інтерфейси не вдаючись до імперативного програмування. XAML відкриває можливість описати складні графічні інтерфейси лаконічним і зрозумілим шляхом. XAML розроблений з урахуванням здатності розширюватись, таким чином додавання нових графічних елементів не завдає додаткових складнощів. Кожен елемент у мові розмітки відповідає класу у .Net Framework, що дозволяє безшовно взаємодіяти з елементами графічного інтерфейсу у C# класах (за необхідності).

Також у клієнтському застосунку було використано шаблон проектування MVVM (англ. Model-View-ViewModel). Даний підхід дозволяє цілковито відділити відображення (View) від логіки взаємодії (ViewModel), яка вносить зміни у стан об’єктів бізнес логіки (Model).Таким чином досягаються наступні переваги застосунку:

* Відображення (View) може розроблятись абсолютно відокремлено від логіки взаємодії (ViewModel), а логіка взаємодії та модель (Model) абсолютно незалежно від відображення.
* Тестованість. Адже логіка взаємодії не має жодних залежностей від відображення, така логіка легко піддається юніт-тестуванню, що підвищує кінцеву якість застосунку.
* Перевикористовуваність. Логіка взаємодії та відображення не має жорсткої зав’язки один на одного, а отже виникає можливість перевикористовувати логіку взаємодії або відображення з декількома різними відображеннями\різною логікою взаємодії.

ОПИС ПРОГРАМИ

Програма створена для ведення військового обліку та виконує такі функції:

* дозволяє користувачу легко та зручно здійснити пошук необхідної книги або іншого документа та інформації про них;
* швидко та зручно редагувати або додавати книги та інші документи та інформацію про них для адміністратора;
* захист входу на сторінку облікового запису адміністратора паролем та захист входу на сторінку облікового запису користувача;
* детальний перегляд всієї інформації будь-який документ;
* збереження всієї інформації в файли;
* досконалий функціонал програми, що включає в себе такі зручності як: швидке оновлення інформації, пошук документа та інформації про нього з будь-якого місця.
* продуманий інтерфейс, об’єднаний спільним дизайном для підвищення зручності;
* свого роду унікальна програма.

КЕРІВНИЦТВО АДМІНІСТРАТОРА

КЕРІВНИЦТВО КОРИСТУВАЧА

При запуску програми відкривається початкове вікно програми (**Рисунок)**, що містить кнопки: «Log in», «Register» та «Log out».

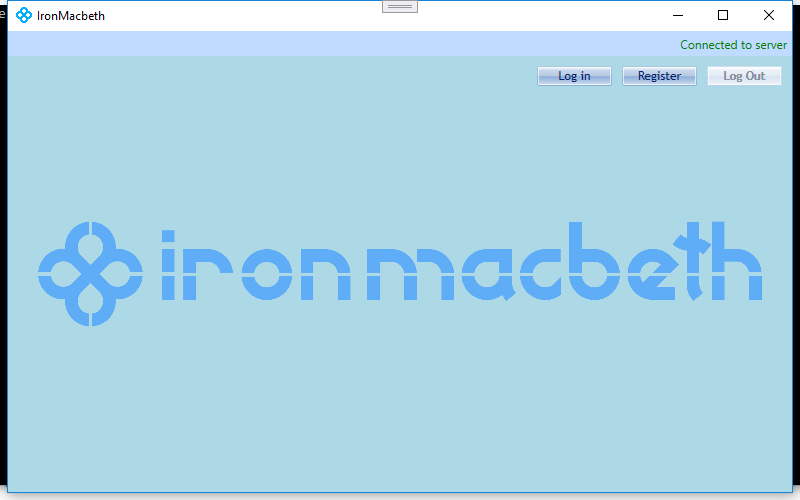


Рисунок – Початкове вікно програми

Кнопка «Log out» з’являється після реєстрації або ідентифікації та служить виходом з облікового засобу користувача.

Кнопка «Register» використовується для переходу у вікно реєстрації нового користувача(**Рисунок**). У цьому вікні користувач заповнює інформацію про себе для подальшої можливості користування книжковими ресурсами. Ці поля є обов’язковими до заповнення, оскільки ця інформація про користувача надалі відтворюється в оформлених замовленнях.

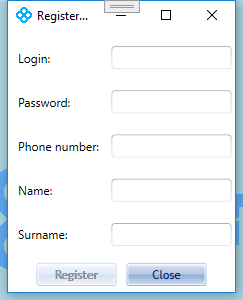


Рисунок – Вікно реєстрації нового користувача

Кнопка «Log in» переходить у вікно ідентифікації зареєстрованих користувачів, а також адміністратора (**Рисунок**). Для того, щоб перейти до наступного вікна, необхідно заповнити поля «Log in» та «Password» відповідно.

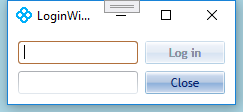


Рисунок – Вікно ідентифікації зареєстрованого користувача або адміністратора

**Сторінка користувача**

Після реєстрації або ідентифікації, зареєстрований або ж ідентифікований користувач потрапляє до головної сторінки гостя, де він може здійснити пошук та зробити замовлення книги або переглянути вже створені замовлення (**Рисунок**).



Рисунок – Головна сторінка користувача

Кнопка «Search» переходить на сторінку пошуку книги в електронному каталозі (**Рисунок**).



Рисунок – Сторінка пошуку книги

Пошук необхідного документу здійснюється за такими критеріями, як назва, автор, тематика та рік видання. Для того, щоб здійснити пошук необхідно заповнити одне з полів та вказати тип документу, який шукається. Кнопка «Search» у вікні пошуку переходить на сторінку результатів пошуку (**Рисунок**), а кнопка «Delete all» очищує всі заповнені поля для подальшого пошуку.

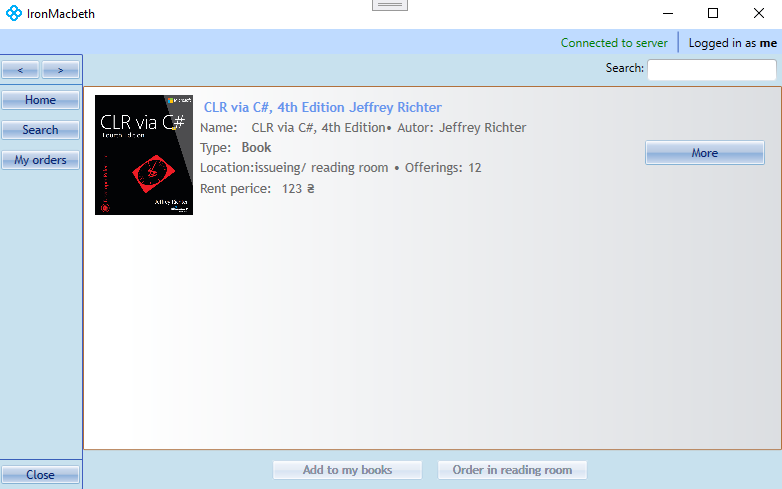


Рисунок – Сторінка результатів пошуку

Натиснувши на кнопку «More» можна перейти на сторінку з детальною інформацією про книгу (**Рисунок**).

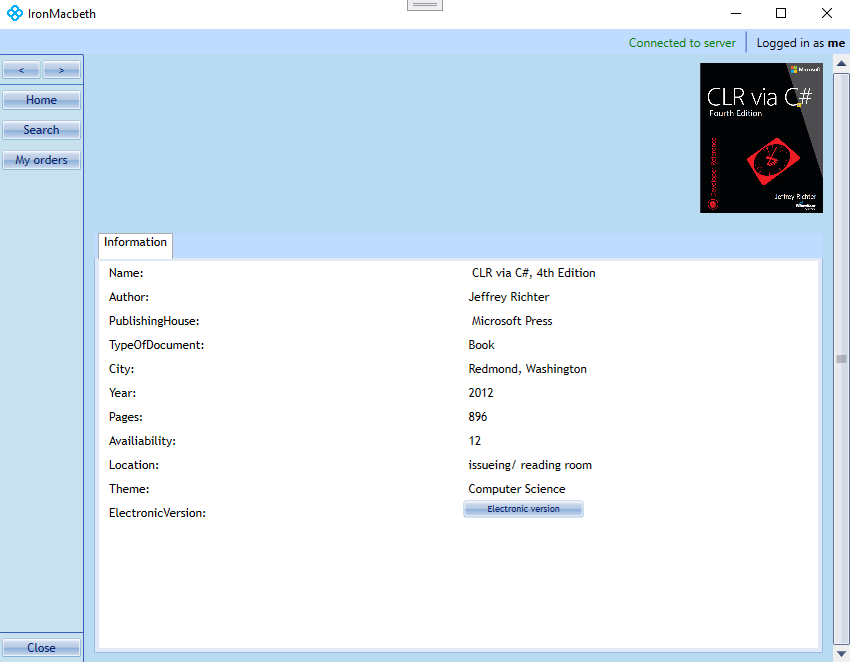


Рисунок – Сторінка детальної інформації про книгу

Обравши потрібну книгу можна додати її до замовлень на видачу, або ж для перегляду у читальному залі бібліотеки. Кнопка «Add to my books» створює замовлення на видачу книги користувачу (якщо є така можливість), а кнопка «Order in reading room» замовляє книгу для користування у читальному залі. Наявність книги показує поле «Offerings». Для успішного створення замовлення кількість доступних книг повинна бути не менше однієї.

Натиснувши на будь-яку з кнопок ми переходимо у вікно вибору бажаного часу отримання книги користувачем (**Рисунок**).

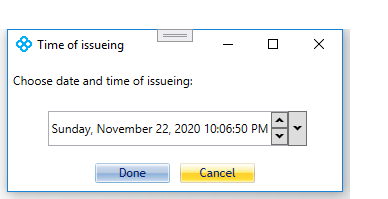


Рисунок – Вікно вибору дати та часу отримання

Користувач має змогу отримати книгу не менше, ніж за три години після замовлення (стандартний час, потрібний на пошук книги бібліотекару чи адміністратору) (**Рисунок**).

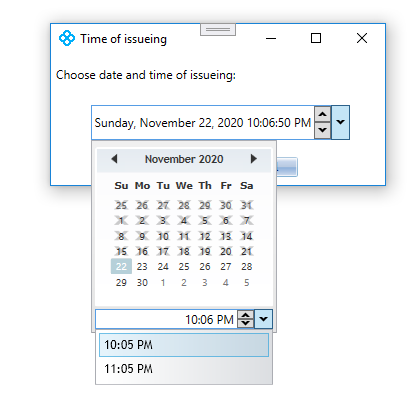


Рисунок – Вибір дати та часу отримання

Натиснувши на кнопку «Done» на екрані з’являється вікно, яке повідомляє користувачу про успішно доданий документ (**Рисунок**).

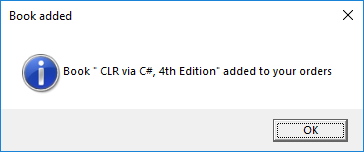


Рисунок – Результат успішного додавання книги до замовлень.

Якщо ж така книга, або інший документ вже є у замовленнях цього користувача то на екрані з’явиться вікно для отримання підтвердження того, що користувачу потрібна ще одна така книга або документ (**Рисунок**).

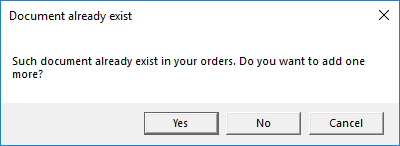


Рисунок – Вікно підтвердження повторного замовлення

Після того, як книга була успішно додана до замовлень, користувач може натиснути на кнопку «My orders», перейти на сторінку замовлень та переконатися у наявності даного замовлення (**Рисунок**). За необхідності, користувач може скасувати замовлення, обравши потрібне з переліку та натиснувши на кнопку «Cancel». Сторінка замовлень користувача показує нам детальну інформацію про користувача, який зробив замовлення, статус замовлення, час замовлення, дату видачі, а також дату повернення (книга видається на місяць).

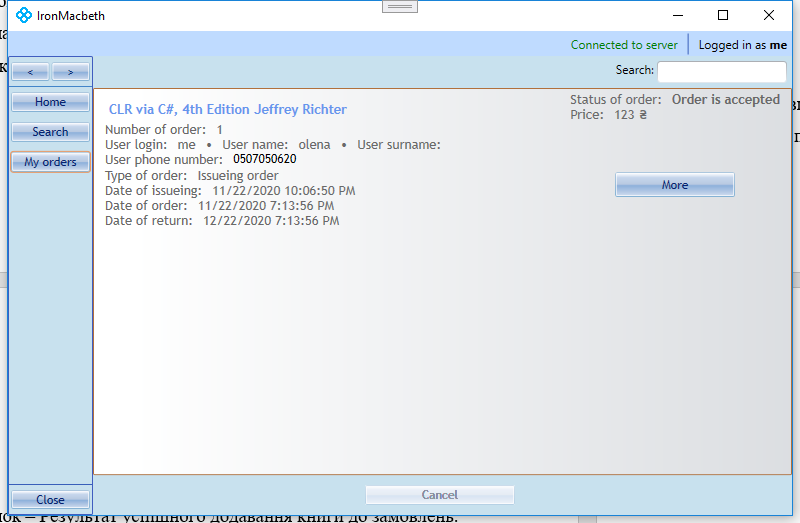


Рисунок – Сторінка оформлених замовлень

**Сторінка адміністратора**

Здійснивши ідентифікацію, довірена особа (адміністратор чи бібліотекар) потрапляє на головну сторінку адміністратора (**Рисунок**). Натиснувши на кнопку «Search» адміністратор, як і користувач, може перейти на пошукову сторінку.



Рисунок – Головна сторінка адміністратора

Саме адміністратор або бібліотекар додає до електронного каталогу книг нові книжки, статті або ж періодичні видання. Натиснувши на кнопку «Books», адміністратор потрапляє на сторінку додавання нових книг та перегляду, видалення чи зміни вже наявних у каталозі (**Рисунок**).

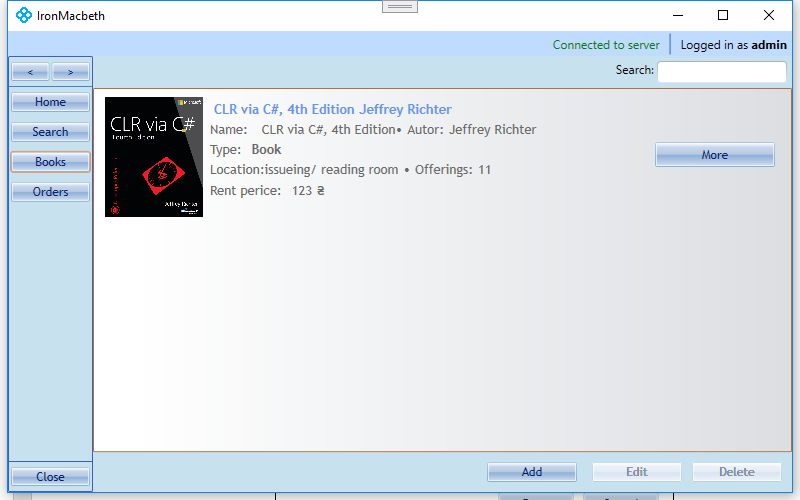


Рисунок – Сторінка додавання нових, зміни та видалення наявних книг

Кнопка «Add» дає можливість створити новий документ, натиснувши на неї, на екрані з’являється вікно вибору типу документу. Тип документа представлений у вигляді ComboBox, доступними типами якого є «Book», «Article», «Periodical», «Newspaper» та «Thesis».

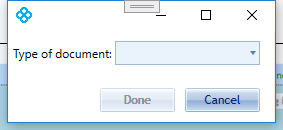


Рисунок – Вікно створення книги та інших документів

Обравши тип документу вікно збільшується та адміністратор, залежно від типу документу, повинен заповнити наявні поля для створення документа, а також додати обкладинку книги чи електронну версію (**Рисунок**). Видача книги для користування є платною, тому обов’язково потрібно вказати ціну оренди.

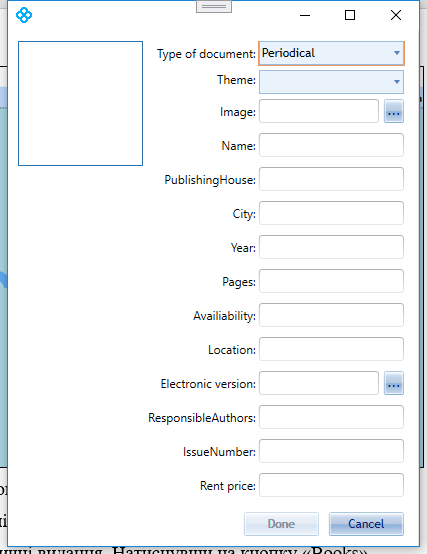


Рисунок – Вікно додавання журналу

У полі «Location» можна вказати лише дві опції «issueing» та «reading room» (**Рисунок**).



Рисунок – Заповнення поля «Location»

Поле «Theme» представлене у вигляді ComboBox та надає перелік всіх можливих тематик (**Рисунок**).

Якщо користувач залишає пусті поля на формі створення нових об’єктів, програма не дозволить створити новий об’єкт, так як не була надана необхідна інформація.

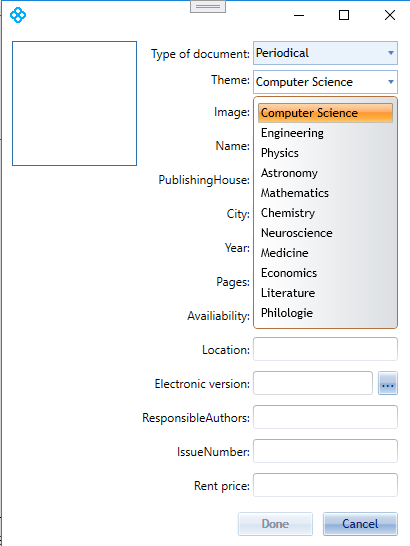


Рисунок – Заповнення поля з тематикою журналу

Заповнивши поля та натиснувши кнопку «Done», книга або інший тип документу будуть автоматично додані до каталогу (**Рисунок**).

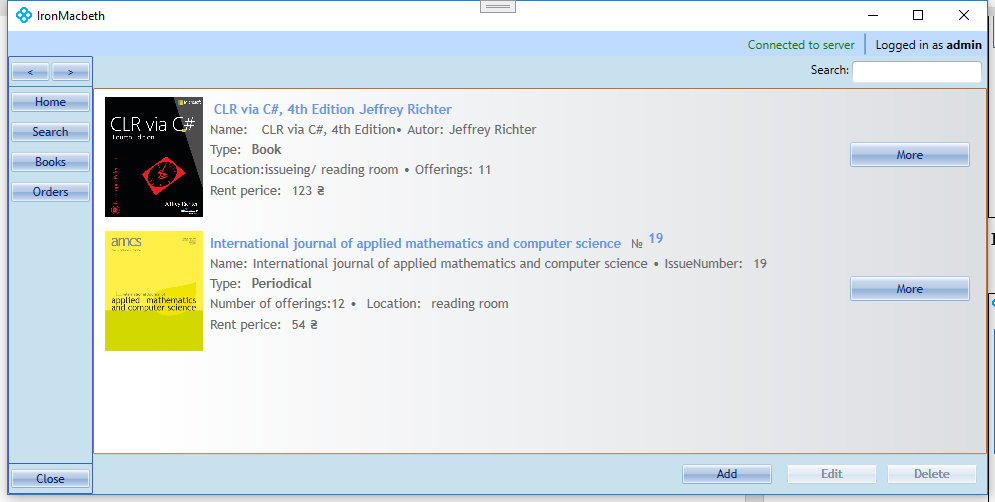


Рисунок – Додані книги до книжкового каталогу

Кнопка «Edit» дає можливість змінювати поля книги або документа (**Рисунок**), а кнопка «Delete» видаляє книгу з електронного каталогу.

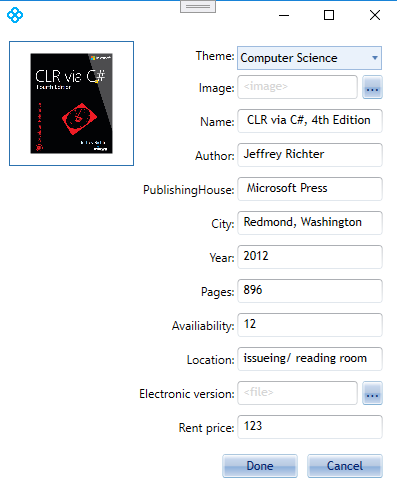


Рисунок – Зміна книги або іншого документу

Кнопка «Orders» надає інформацію про всіх отримані замовлення від користувачів, повну інформацію про користувача, а також повну інформацію про замовлення: дату та час отримання, статус.

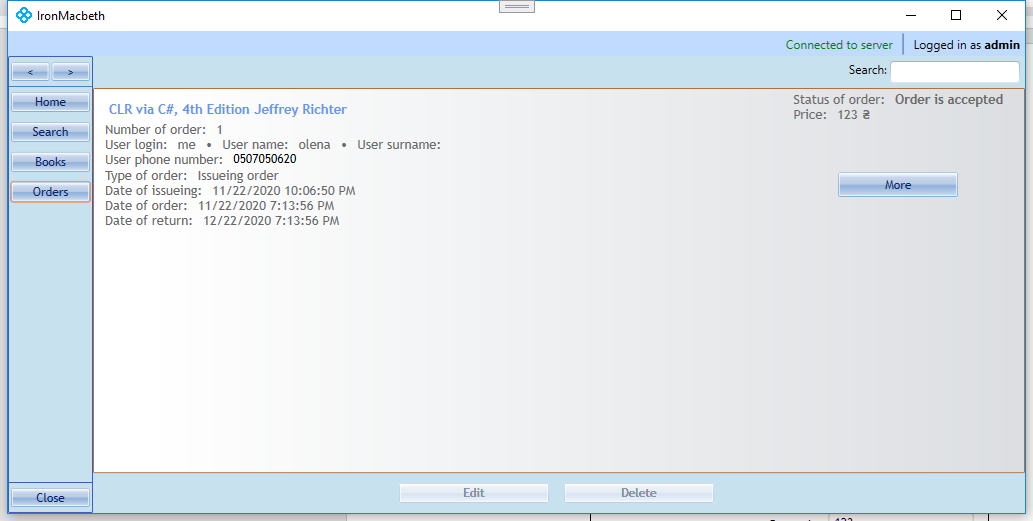


Рисунок – Сторінка із замовленнями

Користувач повинен звертати увагу на статус замовлення, оскільки замовлення вважається виконаним, якщо статус замовлення «Order is completed».

Адміністратор, отримавши замовлення, може змінити статус замовлення, час отримання та час повернення. Натиснувши на кнопку «Edit», на екрані з’являється вікно коригування замовлення (**Рисунок**).

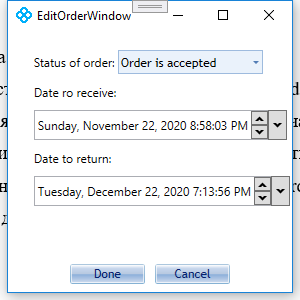


Рисунок – Вікно зміни замовлення

Адміністратор, отримавши замовлення, змінює статус замовлення на «Order is in processing», якщо книга ще готується до видачі, або «Order is completed», якщо замовлення є виконаним (**Рисунок**).

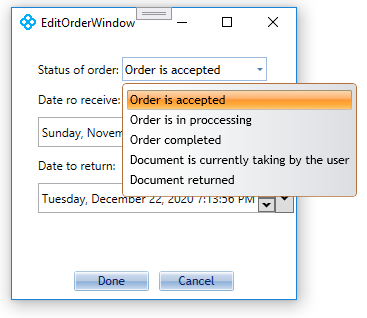


Рисунок – вікно зміни замовлення

Якщо замовлення було виконано раніше, то адміністратор може змінити час отримання замовлення (**Рисунок**).

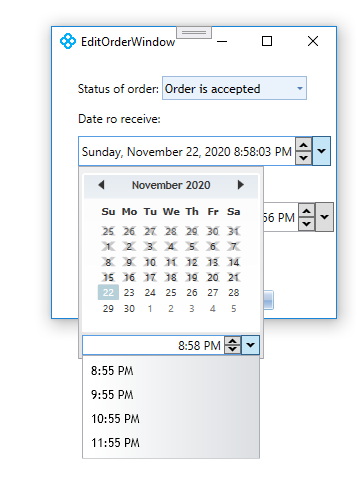


Рисунок – Зміна часу замовлення

Натиснувши «Done», ми отримуємо оновлене замовлення (**Рисунок**).

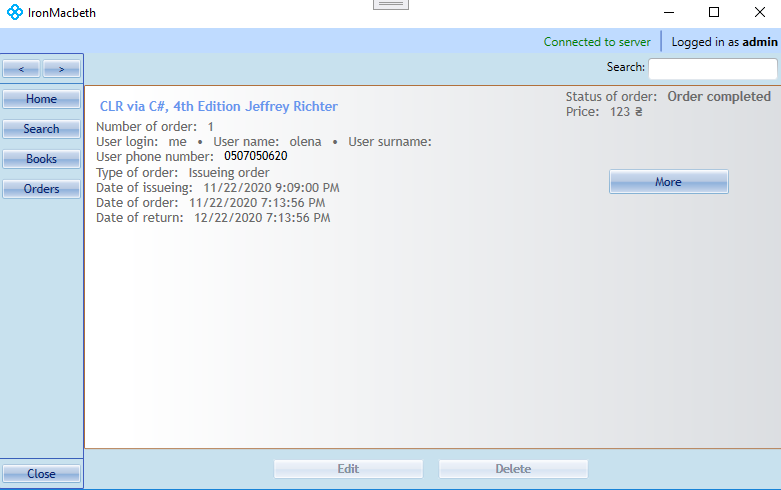


Рисунок – Сторінка із замовленнями

ВИСНОВКИ

В результаті написання курсової роботи була створена програма для автоматизації введення електронного книжкового каталогу . Програма написана на мові програмування C# у середовищі Visual Studio 2019.

В результаті мною досягнуті такі цілі:

• розглянуто подібні програми, які призначенні для пошуку книг та автоматизації ведення книжкового каталогу;

* організовано зручний користувацький інтерфейс таким чином, щоб користування програмою було зручним і ефективним;
* реалізовано можливість додавання та редагування книг, статей, періодичних видань та дисертацій;
* реалізовано зручний пошук та доступ до бажаної користувачем інформації;
* реалізувати зручне оформлення замовлень користувачем та відслідковування цих замовлень адміністратором;
* для створення програми було використано технології WCF Microsoft;
* реалізовано серверну частину застосунку у вигляді розподіленої системи;
* реалізована така архітектуру клієнтської частини застосунку, щоб логіка додатку була відокремлена від візуальної частини (представлення).
* надано широкий вибір інструментів для роботи з програмою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТОК А

Створення і верифікації хешу користувацьких паролів (~\IronMacbeth.UserManagement\IronMacbeth.UserManagement\SecurePasswordHasher.cs)

using System;

using System.Linq;

using System.Security.Cryptography;

namespace IronMacbeth.UserManagement

{

public static class SecurePasswordHasher

{

private const int SaltSize = 16;

private const int HashSize = 20;

private const int Iterations = 10000;

public static string Hash(string password)

{

// Create salt

byte[] salt = new byte[SaltSize];

new RNGCryptoServiceProvider().GetBytes(salt);

// Create hash

var pbkdf2 = new Rfc2898DeriveBytes(password, salt, Iterations, HashAlgorithmName.SHA256);

var hash = pbkdf2.GetBytes(HashSize);

// Combine salt and hash

var hashBytes = new byte[SaltSize + HashSize];

Array.Copy(salt, 0, hashBytes, 0, SaltSize);

Array.Copy(hash, 0, hashBytes, SaltSize, HashSize);

// Convert to base64

var base64Hash = Convert.ToBase64String(hashBytes);

// Format hash with extra information

return base64Hash;

}

public static bool Verify(string password, string hashedPassword)

{

// Get hash bytes

var hashBytes = Convert.FromBase64String(hashedPassword);

// Get salt

var salt = new byte[SaltSize];

Array.Copy(hashBytes, 0, salt, 0, SaltSize);

// Create hash with given salt

var pbkdf2 = new Rfc2898DeriveBytes(password, salt, Iterations, HashAlgorithmName.SHA256);

byte[] hash = pbkdf2.GetBytes(HashSize);

// Get result

var isMatch = hashBytes.Skip(SaltSize).SequenceEqual(hash);

return isMatch;

}

}

}