АННОТАЦІЯ

Дипломна робота на тему «Робоче місце бібліотекаря учбового закладу» складається з 100 стр, 14 діаграм, 8 ілюстрацій, використаних джерел 11, та ілюстраційний матеріал на 0 слайдів.

Мета – розробка програми зі зручним дизайном для управління бібліотекою учбового закладу та обслуговування читачів (цієї бібліотеки) ефективним та економічно-вигідним шляхом.

Теоретичне дослідження проводилося методом аналізу джерел в інтернеті, перевіркою роботи схожих програм та виявлення їх недоліків, ознайомлення зі стандартами проектування баз даних, створення документації, створення зрозумілого дизайну для користувача.

В роботі описується метод проектування та створення конкурентоздатної системи для автоматизації роботи бібліотекаря за стандартами розробки програмного забезпечення.

Було виконано такі етапи розробки: створення схеми бази даних, проектування архітектури програми – створення схеми роботи, визначення варіантів використання системи. Проектування компонентів програми та створення інтерфейсів для спілкування компонентів між собою та взаємодія з базою даних. Створення зрозумілого та простого дизайну для роботи з програмою. Перевірка на відповідання вимогам, визначеним на етапі проектування, перевірка коректності роботи – за стандартами тестування ПО.

Практична цінність дипломної роботи в тому, що програма готова до використання в бібліотеках учбових закладів, а з мінімальними адаптаційними змінами в учбових закладах інших типів, з метою покращення ефективності роботи. Також система має відкритий код і відкрита для масштабування.

ANNOTATION

Diploma work about “Workplace of librarian of the education institution” contain 100 pages, 14 diagram, 8 illustration, used sources 11, and demonstration materials of the 0 slides.

Goal is development of program that has useful design for manage of library of the education institution and efficient and cost effective service of users of the library.

Theoretical research was conducted with method of analysis internet sources. There are checking of work similar applications and finding them limitations, study of design data base standard, creating documentation, creating user friendly design.

In work described method of designing and creating competitive system according to software development standards of for automatization librarian work.

There were performed following steps of development: creating of database, design program architecture that includes creating schema of work and determining variants of system use. It is included component design of program and creating interface for component communication and interaction with database. There is creating clear and flat interface for interaction with program. There are requirements verification that was defined on design stage, validation of program work correspond to testing standard.

Practical value of diploma work is this program is ready for use in the library of University by way of improvements work. And with minimal adaptation also can be use in other education institution.This system also is open source and open for scaling in the future.

ЗМІСТ

[АННОТАЦІЯ 1](#_Toc532219054)

[ANNOTATION 2](#_Toc532219055)

[ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ 4](#_Toc532219056)

[ВСТУП 6](#_Toc532219057)

[1 АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ 8](#_Toc532219058)

[2 ТЕХНІЧНЕ ОБҐРУНТУВАННЯ І ПИТАННЯ ОРГАНІЗАЦІЇ РОЗРОБКИ 10](#_Toc532219059)

[3 АНАЛІТИЧНИЙ ОГЛЯД 12](#_Toc532219060)

[3.1 Формальна постановка вимог 12](#_Toc532219061)

[4 ВИБІР СЕРЕДОВИЩА РОЗРОБКИ 13](#_Toc532219062)

[ОСНОВНА ЧАСТИНА 14](#_Toc532219063)

[5 ПОСТАНОВКА ЗАДАЧІ 14](#_Toc532219064)

[5.1 Формалізація вимог до технічної розробки 14](#_Toc532219065)

[6 РОЗРОБКА ДІАГРАМ ТА СХЕМ 21](#_Toc532219066)

[6.1 Створення та опис діаграм 21](#_Toc532219067)

[6.2 Створення схеми бази данних 26](#_Toc532219068)

[6 РОЗРОБКА ПРОГРАММНОГО ЗАБЕСПЕЧЕННЯ 29](#_Toc532219069)

[8 РОЗРОБКА ЗАВДАННЯ ПО ОХОРОНІ ПРАЦІ 46](#_Toc532219070)

[1.1 Вимоги до мікроклімату комп'ютерних приміщень. Засоби його оптимізації 46](#_Toc532219071)

[1.2 Вимоги до виробничої меблі для робочих місць з ВДТ 50](#_Toc532219072)

[1.3 Засоби поліпшення функціонального стану спеціалістів ІТ 52](#_Toc532219073)

[ВИСНОВКИ 57](#_Toc532219074)

[ПЕРЕЛІК ПОСИЛАНЬ 58](#_Toc532219075)

[ДОДАТКИ 59](#_Toc532219076)

# ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ПК – персональний компьютер

АРМ – автоматизоване робоче місце

ISBN - (з англ. International Standard Book Number — міжнародний стандартний номер книги) універсальний ідентифікаційний номер, що присвоюється книзі з метою їх класифікації. ISBN призначений для ідентифікації окремих книг або різних видань та є унікальним для кожного видання книги. Присвоєння номера означает офіційнц реєстрації книги у Книжковій палаті України.

Windows – збіркова назва для операційних систем для ЕОМ, розроблених корпорацією Microsoft. Нині Windows встановлена на більшості персональних комп'ютерів і робочих станцій.

БД – (база даних) сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами;

СКБД – (Система керування базами данних) - комплекс програмного забезпечення, що надає можливості створення, збереження, оновлення та пошуку інформації в базах даних з контролем доступу до даних.

Microsoft SQL Server - комерційна система керування базами даних, що розповсюджується корпорацією Microsoft. Мова, що використовується для запитів — Transact-SQL.

Transact-SQL - є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями.

Docker for Windows – це платформа для розробки, розгортання та запуску контеризованних программ.

Microsoft.NET Framework - крос-платформова технологія для створення як звичайних програм, так і веб-застосунків.

CLR – (з англ. Common Language Runtime) - загальномовне виконуюче середовище - це компонент пакету Microsoft .NET Framework, віртуальна машина, на якій виконуються всі мови платформи .NET Framework.

C# - об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET.

Windows Forms — інтерфейс програмування додатків (API), відповідальний за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за допомогою створення обгортки для Win32 API в керованому коді.

API – (з англ. Application Programming Interface) - Прикладний програмний інтерфейс. Набір чітко визначених методів для взаємодії різних компонентів. API надає розробнику засоби для швидкої розробки програмного забезпечення.

IDE – (з англ. Integrated development environment) - комплексне програмне рішення для розробки програмного забезпечення. Зазвичай, складається з редактора початкового коду, інструментів для автоматизації складання та відлагодження програм.

Microsoft Visual Studio - серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів.

UML (англ. Unified Modeling Language) — уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення.

LMS – (з англ. Library Management System ) назва проекту для автоматизації робочого місця бібліотекаря.

# ВСТУП

В учбових закладах існує низька проблем, пов'язаних з підпорядкуванням книг під час взаємодії с читачами, розподілу книг по категоріях та предметам, отримання та списання книг, та ведення документації в цілому. Зазвичай використовується на перший погляд найпростіший метод ведення усієї документації в бібліотеці – написанням від руки або машинописом усієї звітної інформації, та зберігання архівів у вигляді паперових екземплярів.

Цей метод має низьку недоліків:

1) Усладнен процес оновлення інформації – якщо потрібно змінити один запис, який тягне за собою зміну ще в декількох.

2) Велика ймовірність пошкодження інформації – втрата свої цілісності. В тому випадку коли в різних місцях системи буде різна інформація.

3) Відсутність контролю за цілісністю та актуальністю даних в системі. Можливі дублікати, або навпаки, якщо якась інформація втратиться с часом й вже не існуватиме в системі.

4) Складний процес пошуку різноманітної інформації, що також ускладнює можливість отримання актуальних звітів роботи системи.

5) Економічно великі затрати на підтримання системи. Місце для звітів, папери для них та інші канцелярські вироби.

Всі вище перераховані проблеми частково вирішуються кваліфікованими фахівцями та плануванням упорядкування інформації. Але це лише зменшую ймовірність, але вона все ж існує, тому що люди здатні помилятися. До того ж все ще залишаються проблеми зберігання інформації – для цього потрібно багато місця. Також у випадку перевезення бібліотеки виникає багато труднощів.

Стрімке вторгнення цифрових технологій привело до радикальних змін усіх

напрямів діяльності бібліотеки: від традиційного інструментарію до методів

обслуговування, доступу до ресурсів, форм надання інформації. Відтепер роль і значення бібліотеки визначаються не тільки кількістю і різноманітністю її фонду,

але й комфортністю послуг, які вона може надати. Широковідома точка зору, що в Інтернеті «все є» часто трактується

помилково. Так, в Інтернеті дійсно є багато корисної інформації, але виключно

для того, хто володіє «секретами» її пошуку. Інтернет тільки за певних умов може стати могутнім інструментом науки та освіти, і в числі таких неодмінних умов є створення відповідного бібліотечноінформаційного середовища. Досить довго бібліотека вважалась лише «тихим місцем для читання», важливим джерелом надання інформації і сьогодні є нагальна потреба в підвищенні її консультативної функції, впровадженні навчальних технологій. З підвищенням ролі НБ ХНМУ в інформаційному середовищі університету

зазнає зміни вся сфера бібліотечно-інформаційної діяльності: ускладнюються

технології, впроваджуються нові послуги, розширюється кількісно і якісно склад

інформаційних ресурсів, набирають наукового темпу запити користувачів. Це

робить надзвичайно важливим призначення бібліотекаря і диктує нові вимоги до

бібліотечних працівників як посередників між користувачами та інформаційними

ресурсами.

Бібліотекар сьогодні виконує багато функцій, які не були йому властиві

раніше. Він – активний суб'єкт системи соціальних комунікацій, і господарник,

що освоює інноваційні напрямки економічної діяльності у невиробничій сфері, і

спеціаліст в області інформаційних технологій, і провідник інформаційної

культури. Повсякденна бібліотечна практика потребує зростання професійної

кваліфікації на всіх ділянках роботи: при наданні послуг, виконанні довідок,

проведенні комплексних заходів, чергуванні у залі каталогів та залах електронної

інформації і навіть просто у спілкуванні з відвідувачами бібліотеки, серед яких і

студенти, і викладачі, і науковці, і лікарі-практики.

# 1 АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

Вперше одержати більш-менш цілісну картину стану інформатизації регіональних бібліотек України, визначити рівень та потенціал кожної з бібліотек, побачити проблеми, що потребують вирішення, як на рівні кожної окремої бібліотеки, так і на рівні галузі дозволило дослідження «Електронні інформаційні ресурси бібліотек України». Воно проводилося у 2001 році Національною парламентською бібліотекою України на базі 27 бібліотек. Метою було дослідження інформаційних електронних ресурсів, кадрового складу, систем автоматизації бібліотек.

Процеси автоматизації бібліотек беруть свій початок ще у 80-х роках ХХ ст. Саме тоді у ОУНБ України були створені перші підрозділи автоматизації, структурне оформлення яких тривало близько 15 років. Кількість працюючих у цих підрозділах на момент збору даних була досить різною – від 1 до 9 співробітників, серед яких переважали фахівці з вищою технічною освітою, віком до 30 років. Рівень розвитку інформатизації бібліотек не залежав від часу створення підрозділів автоматизації, а якість та обсяги виконаних робіт не завжди залежали від кількості співробітників відділу. Можливо, більшу роль в цьому відігравали соціально-психологічна готовність керівників та підлеглих працювати по-новому. Вивчення комп’ютерного парку показало, що у 2001 році у ОУНБ було 384 ПК. 23 бібліотеки мали автоматизовані робочі місця (и), більшість з яких функціонувало у відділах обробки літератури, інформаційно-бібліографічних, комплектування, автоматизації. Для інформатизації бібліотек особливе значення мали проекти, тематика і масштаби яких були досить різноманітні – від створення або автоматизації одного відділу до програми інформатизації всієї бібліотеки.

В сучасному суспільстві великий вплив має модернізація пошуку та отримання нових знань. Широке розповсюдження електронних книг впливає в першу чергу на потреби читачів. Сучасне суспільство динамічно розвивається в політичній, економічній та соціальній сфері, й бібліотеки повинні також розвиватися відповідно до потреб читачів, які є частиною суспільства. Але, як було зазначено у дослідженні середній вік бібліотекаря більше 30 років – людина яка, як правило, має мало знань з компьютерів, базів данних та інше. Тож працювати з программами, з якими легко зможе розібратися людина 20-25, яка має більше досвід, та частіше зіткаеться з інформаційними технологіями навколо нас. Необхідність в модернізації бібліотечної діяльності полягає в тому, що необхідно знизити в першу чергу витрати на організацію простору для книг, їх підпорядкування та ведення документації. Наступний пункт – це модернізація взаємодії з читачами, та упорядкування даних які відносяться до звітності. Всі ці дії націлені на те щоб полегшити працю бібліотекарів, та зпростити їх взаємодію з чатачами. Вище перерахованих покращень можна досягти, якщо запровадити автоматизовану систему для контролю за книгами та по роботі з читачами. Ця програма повинна відповідати наступним вимогам:

* Опрацьовувати багато даних
* Зберігати їх за зазначеною схемою
* Мати внутрішні механізми для перевірки введених даних
* Мати зручний та зрозумілий дизайн
* Бути масштабованою

В нашому проекті, який має за свою мету створення модернізованної, автоматизованної бібліотечної системи, з можливістю масштабування у маймутньому, будуть використовуватся передові ідеї та механізми зі сфери інформаційних технологій та компьютерних наук. Для того щоб зберігати велику кількість інформації будуть використовуватися бази данних. Для нашего проекту ми будемо використовувати самий розповсюджений тип – реляційні бази данних. Щоб досягти універсальність зберігання інформації в схемі данних втілимо стандарт для інформації про книгу. Программа буде надавати її користувачам можливості які будуть поділені між категоріями користувачів. Для того щоб не перенавантажувати клієнтський інтерфейс непотрібними можливостями для конкретних користувачів.

Кінцева система підходить для використання в різноманітних учбових закладах. Таких як школи, університеті, наукові бібліотеки, та інші види біблиотек.

Специализована підготовка для працівників не потрібна. Система має короткий звіт правил та дій за допомогою яких можна виконати ту чи іншу дію. До того ж программа має інтуітивно зрозумілий дизайн, та низьку підзказок, які супроводжують користувача в продовж роботи.

# 2 ТЕХНІЧНЕ ОБҐРУНТУВАННЯ І ПИТАННЯ ОРГАНІЗАЦІЇ РОЗРОБКИ

Основна мета виконання роботи це досягти еффективності роботи бібліотеки. Для того щоб провести аналіз та наблизитися до чіткого формування критеріїв продукту та визначити необхідні для нього інструменти та технології, розглянемо наступний перелік аспектів роботи системи яких ми хочемо досягти:

1. Легкий та безпечний доступ до інформації;
2. Легкість зміни, додавання/видалення інформації про книги та читачів;
3. Забезпечення надійного збереження інформації;
4. Мінімізувати вартість зберігання даних;
5. Мінімізувати вартість обслуговування читачів;

В попередній частині ми з’ясували, що система автоматизації може покращити стан бібліотеки та підвищити її продуктивнітсть. Для початку виділимо процеси, які ми будемо автоматизовувати, з точки зору певних групп, які уснують в бібліотеці:

1. Комплектування
   1. облік документів у електронному вигляді
   2. управління існуючими в бібліотеці книгами
2. Каталогізування
   1. бібліографічний опис документів і створення ЕК, БД
   2. призначення інвентарних номерів для виданників
3. Обслуговування клієнтів
   1. Отримання інформації про поточний стан запису клієнта

Программа автоматизації для бібліотечної системи буде розроблятися для операційної системи Windows – найпоширенішої системою для компьютерів. До того ж вона буде мати зручний дизайн, зрозумілий на інтуєтивному рівні інтерфейс та багато інтструментів, які будуть використовуваться для нашої системи. Найчастіше саме під керуванням цієї операційної системи працюють компьютери в учбових закладах. Перед впровадженням такої системи необхідно поперше визначити які данні ми будемо зберігати. Визначити структуру - як вони будуть зберігатися – це етап проектування бази данних. Він має велике значення, тому що добре спроектована база даних дає можливість швидко її розширювати, без внесення великих змін у вже працююче рішення, також добре спроектована база данних надає можливість зберігати данні в одиничному єкземплярі – без непотрібного дублювання. Для оптимізації баз данних будемо використовувати методи нормалізації, та приводити нашу базу до першої, другої та третьої нормальної форми. Потім необхідно ввести всю інформації про конкретну бібіліотеку до бази данних. Для того щоб інформація була зв’язана між собою, та її було легко підтримувати будемо використовувати реляційні бази данних. З отриманням інформації про функціонування бібліотеки система може не тільки підтримувати працівників, а й контролювати введені ними дані. Для взаємодії з базою данних будемо використовувати программний інтерфейс, отже користувач не буде напряму змінювати інформацію у базі данних. А це у свою чергу дозволяє досягти легкості модифікування данних та зберегти їх цілістність, а також контролювати рівень доступу до даних. Доступ до даних у базі даних розмежован через программний інтерфейс для кожного типу облікових записів, він також має чітко розмежований доступ завдяки різним типам облікових записів. Доступ до кожного типу здійснюєтся за допомогою введення унікального логіну та, попереднью збереженого у базі, паролю. Тільки після проведення аутентифікації користувач може використовувати систему.

На цьому етапі ми визначаемо як багато даних будемо зберігати, та скільки це коштуватиме. Для того щоб зробити дані доступними, з урахуванням майбутніх розширень, будемо використовувати інструменти для віртуалізації. Це дасть змогу забеспечити одночасний доступ декількох кліентів до бази даних. Отже программний інтерфейс буде спілкуватися з системою управління базою данних, сервер якої буде працювати у ізольованному середовищі. Це буде забезпечувати безпеку данних та надавати можливість створювати скільки завгодно клієнтів. Алгоритми, які буде використовувати программа для валідації введенних данних будуть двох рівневими. Спочатку програмний інтерфейс буде перевіряти введені данні на помилки – відсітність даних, наявність заборонених символів для конкретного поля, або невідповідність мови, яка використовуется під час введення данних, та мови яка використовується в системі.

# 3 АНАЛІТИЧНИЙ ОГЛЯД

Перед тим як разпочати розробку продукту, необхідно визначити яким саме вимогам він має відповідати. Для початку це буде огляд вимог до кінцевого продукту з точки зору кінцевого замовника. Без внесення техничних деталей. Огляд лише з точки зору функціональності.

# 3.1 Формальна постановка вимог

1. Система повина надавати можливість авторизуватися під різними обліковими записами.
2. Система має надавати можливість адміністратрування облікових записів користувачів.
3. Система має надавати можливість працювати з обліковими записами читачів

та має доступ до інформації читачів.

1. Система має надавати доступ до поточної інформації щодо стану бібліотеки.
2. Система має бути застанована на міжнародних стандартах книгообертання.
3. Система повина надавати можливість отримувати різноманітні вибірки інформації у зручному форматі.
4. Система має надвати зручний інтерфейс пошуку книг.
5. Система має надавати можливість региструвати читачів у системі.
6. Система має надавати можливість змінювати інформацію про читачів у системі.
7. Система повина надавати зручний дизайн для взаємодії з программою. А також чітко розграничувати можливості різних типів користувачів.
8. Система має надавати можливість вносити інформацію про книги, або удаляти цю інформацію.
9. Система має надавати можливість змінювати інформацію про книги.
10. Система має надавати можливість зберігати інформацію про книги.

# 4 ВИБІР СЕРЕДОВИЩА РОЗРОБКИ

Для системи автоматизації, яка в свою чергу буде призначена для роботи на Windows, операційна система на якій буде відбуватися розробка – Windows 10 Education. В якость системи управління базою даних (СКБД) буде використовуватися Microsoft SQL Server Express. Дозволяє створювати невеликі до 10 Гб безкоштовно бази даних початкового рівня. Дає змогу працювати як служба в системі або як окремий процесс в программі. В нашому випадку – будемо використовувати сервер як окремий процес, який буде працювати в ізольованому середовищі.

Для розгортання сервера бази данних будемо використовувати Docker for Windows. Docker працює за принципом контеризації програм, яка в свою чергу дає змогу досягти гнучкості за рахунок того, что в контейнері може працювати будь-яка программа; портативності – робота программи не залежить від її розгортання та збору. Контейнерні програми використовують ресурси компьютера, але створюють ізольоване середовище вікинання – неможливо пошкоджити систему на якій це працює. Також контейнери дають велику перевагу в масштабованості – можна скільки завгодно створювати копії контейнерів.

Для контролю за виконанням коду, та для збереження крос-платформеності, будемо використовувати в проекті Microsoft.NET Framework (v4.7.2) – це програмна технологія, яка працює за принципом віртуального процессора. Має свої типи, компілятор, який компюлює код в машиний по потребі програми. Тобто код, який ви пишите та виконуете за допомогою платформи .NET є контрольованим зі сторони цієї ж платформи, до того ж він має багато оптимізацій, які дуже важко впроваджувати в середовищі, яким ви не можете керувати – наприклад операційна система. Це забезбечує захист системи на якій ви виконуете вашу програму, через те, що программа не використовуює ресурси систему напряму, а робить це через CLR. Платформа .NET підтримую багато різних мов. Для написання програми в якость мови програмування будемо використовувати C#. Для розробки призначенний для користувача інтерфейс для системи будемо використовувати частину .NET – Windows Forms, який являє собою зручний API для створення вікон та мнеджменту черги повідомлень на Windows.

Для проекту в якості IDE будемо використовувати Visual Studio 17 Community. Через те, що вона містить зручний менеджер пакетів для додавання нових компонентів до проекту, вбудованний статичний аналізатор коду, роботу з системами керування версіями, інтегровану роботу з базами.

# ОСНОВНА ЧАСТИНА

# ПОСТАНОВКА ЗАДАЧІ

# Формалізація вимог до технічної розробки

1. Система має надавати автоматизовану систему управління бібліотекою университету, яка складається з бази данних, та программи яка проводить валідацію вхідних данних для контролю взаємодії користувачів системи з базою данних бібіліотеки университету
2. Система повина надавати можливість авторизуватися під різними обліковими записами такими як:
   1. Адміністратор
   2. Бібліотекар читацького залу
   3. Комірник
3. Головний адмінистративний обліковий запис має попередньо заданий логін та пароль (login: root, password: root), подальші адміністративні записи можуть мати будь який пароль, заданий іншим адміністратором.
4. Обліковий запис типу адміністратор повинен надавати можливість створювати нових користувачів типу адміністратор, бібліотекар читацького залу та комірник.
5. Система повинна розграничувати доступ до будь-якої інформації за типом облікового запису.
6. Система повинна надавати можливість с початку своєї роботи вибирати тип обликового запису під яким ви почнете користуватися системою (див. п.1).
7. Система повинна надавати можливість під час своєї роботи змінювати активний обліковий запис на інший. В цьому випадку система також повинна зберегти усі валідні зміни, що були зроблені користувачем.
8. Система повинна зберігати у базі данних наступну інформацію:
   1. Про облікові записи:
      1. Логін
      2. Пароль
      3. ПІБ
      4. Дата народження
      5. Адреса проживання
      6. Телефон
      7. Посада
   2. Про читачів ­
      1. ПІБ
      2. Дата нарожденя
      3. Адреса проживання
      4. Телефон
      5. Номер читацького квитка
   3. Про книгооберт
      1. Читач
      2. Книга в наявності у читача
      3. Дата коли видана
      4. Дата коли була повернута
   4. Про книги
      1. Назва
      2. Повна назва
      3. Уникальній код (за стандартом ISBN)
      4. Формат
      5. Выдавництво
      6. Мова
      7. Дата видання
      8. Серія
      9. Аннотація
      10. Автори
      11. Предмет
   5. Примірник книги
      1. Інвентарний номер
      2. ISBN
   6. Про авторів книг
      1. Автор
      2. Уникальній код (ISBN)
   7. Про Видавництва
      1. Ім’я
      2. Місце знаходження
      3. Категорія
   8. Про предмет
      1. Назва
      2. Батьківський предмет
9. Система повинна зберігати лише ті данні до бази данних, що пройшли перевірку на корректність даних. Обмеження задаются у базі даних, на єтапі проектування бази данних
10. Система має підримувати наступні обмеження для полів бази данних.
    1. Про облікові записи:
       1. Логін – має складатися з символів латиниці, може містити цифри, не може починатися з цифри, максимальна довжина рядка не обмежена, мінімальна 6, поле обов’язкове для заповнення
       2. Пароль - має складатися з символів латиниці, може містити цифри та інші символи, обов’язково повинен мати різний реєстр, мінімальна довжина 6, максимальна 30, поле обов’язкове для заповнення
       3. ПІБ - має складатися з символів латиниці, максимальна довжина 50, поле обов’язкове для заповнення
       4. Дата народження – валідація для дати відносно календаря, поле обов’язкове для заповнення
       5. Адреса проживання - може складатися з будь-яких символів, максимальна довжина 150, поле необов’язкове для заповнення
       6. Телефон – має складатися з цифрів, мінімальна довжина 10, поле обов’язкове для заповнення
       7. Посада – має приймати наступні значення – {адміністратор, бібліотекар, комірник}, поле обов’язкове для заповнення
    2. Про читачів
       1. ПІБ - має складатися з символів латиниці, максимальна довжина 50, поле обов’язкове для заповнення
       2. Дата народження – валідація для дати відносно календаря
       3. Адреса проживання - може складатися з будь-яких символів, максимальна довжина 150, поле необов’язкове для заповнення
       4. Телефон – має складатися з цифрів, мінімальна довжина 10, поле обов’язкове для заповнення
       5. Номер читацького квитка - може складатися з будь-яких символів, максимальна довжина 124, поле обов’язкове для заповнення, поле бути унікальним
    3. Про книгооберт
       1. Читач – запис про читача (див п. 10.2), поле обов’язкове для заповнення
       2. Книга в наявності у читача – запис про книгу (див п. 10.4)
       3. Дата коли видана - валідація для дати відносно календаря, поле обов’язкове для заповнення
       4. Дата коли була повернута - валідація для дати відносно календаря, повинна бути більшою ніж дата коли видана (див. 10.3.3) поле обов’язкове для незаповнення
    4. Про книги
       1. Назва – має складатися з символів латиниці, максимальна довжина рядка 30 символів, мінімальна 10, поле обов’язкове для заповнення
       2. Повна назва – має складатися з символів латиниці, максимальна довжина рядка 150 символів, мінімальна 30, поле необов’язкове для заповнення
       3. Уникальній код (за стандартом ISBN) – може приймати числові значення, поле обов’язкове для заповнення
       4. Формат - має приймати наступні значення – {друкований, єлектронний}, поле обов’язкове для заповнення
       5. Выдавництво – приймає значення «Видавництво» (див. п.10.7) поле обов’язкове для заповнення
       6. Мова - має приймати наступні значення – {російська, українська, англійська, німецька}, поле обов’язкове для заповнення
       7. Дата видання - валідація для дати відносно календаря, мінімальне значення року 1880, поле обов’язкове для заповнення
       8. Серія - має складатися з символів латиниці, максимальна довжина рядка 50 символів, мінімальна 10, поле необов’язкове для заповнення
       9. Аннотація - має складатися з символів латиниці, максимальна довжина рядка 256 символів, мінімальна 10, поле необов’язкове для заповнення
       10. Автори – приймає значення «Автор книги» (див. п. 10.6), поле обов’язкове для заповнення
       11. Предмет - може приймати наступні значення {фізика, математика, англійська мова}, поле обов’язкове для заповнення
    5. Примірник книги
       1. Інвентарний номер – може приймати лише числові значення, повинно бути унікальним, поле обов’язкове для заповнення
       2. ISBN – поле обов’язкове для заповнення (див. 10.4.3)
    6. Про авторів книг
       1. Ім’я - має складатися з символів латиниці, максимальна довжина 50, поле обов’язкове для заповнення
       2. Книги – реалізується за допомогою додатковою таблицею зі зв’язком книга-автор.
    7. Про Видавництва
       1. Ім’я - має складатися з символів латиниці, максимальна довжина 100, поле обов’язкове для заповнення
       2. Місце знаходження - має складатися з символів латиниці, максимальна довжина 50, поле необов’язкове для заповнення
       3. Категорія - має складатися з символів латиниці, максимальна довжина 30, поле необов’язкове для заповнення
    8. Про предмет
       1. Назва - має складатися з символів латиниці, максимальна довжина 50, поле обов’язкове для заповнення
       2. Батьківський предмет – приймаю значення «Предмет» (див. п. 10.8), поле необов’язкове для заповнення
11. Для того щоб зареєструвати нового користувача потрібно ввести необхідну інформацію (див 7) та пройти процесс валідації данних
12. Обліковий запис типу бібліотекар читацького залу повинен надавати можливість реєструвати нових користувачів – читачів у системі (див 11)
13. Обліковий запис типу бібліотекар читацького залу повинен надавати можливість видавати книги зареестрованним читачам. Під час зміни значень полів потрібно дотримуватися заданих обмеженень
14. Обліковий запис типу бібліотекар читацького залу повинен надавати можливість читачам повертати книги до бібліотеки зареестрованним читачам.
15. Обліковий запис типу комірник повинен надавати можливість додавати книги до бази данних, шляхов заповнення зазначенних у п.7 полів (див п. 7.3, 7.4, 7.5, 7.6), та проведення валідаціі вхідних данних (див п.8)
16. Обліковий запис типу комірник повинен надавати можливість видаляти книги з бази данних, шляхов видалення зазначенних у п.7 полів (див п. 7.3)
17. Обліковий запис типу комірник повинен надавати можливість змінювати інформацію про книги, які вже внесені до бази данних, шляхов заповнення зазначенних у п.7 полів (див п. 7.3, 7.4, 7.5, 7.6), та проведення валідаціі вхідних данних (див п.8)

# 6 РОЗРОБКА ДІАГРАМ ТА СХЕМ

В попередньому розділі було визначено які задачі вирушує система, які інструменти вона буде для цього використовувати, та які дані вона буде зберігати. Початковим етапом при розробці програмного забезпечення є проектування майбутнього продукту. За допомогою проектування визначаємо архітектуру майбутнього продукту, щоб він відповідав функціональним можливостям програмного забезпечення які ми відили на єтапі визначення вимог. Проект складається з розробки діаграм, які в графічному вигляді дозволяють визначити усі складові програмного забеспечення.

Вимоги, які ми визначили до кінцевого продукту з боку замовника допоможуть нам в проектувані. Для проектування будемо використовувати UML, яка є спеціалізованою нотаціонною мовою, яку використовують розробники для графічного представлення програмного забезпечення. В якості середовища проектування будемо використовувати безкоштовний інтструмент з багатим функціоналом StarUML.

# 6.1 Створення та опис діаграм

На початку проектування, ми вже маємо маємо функціональні вимоги до ПП, іншими словами це варіанти використання нашої системи і ми можемо побудувати діаграму використання. Суть даної діаграми полягає в наступному: спроектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання використовують для описання послуг, які система надає актору. Хто є в нашій системі актор? Як біло зазначено вище система надає робочий простір для трьох разних типів користувачів: бібліотекар, адміністратор, та комірник. Ми визначили акторів и теперь побудуемо діаграмму використання для кажного з них. Загальний огляд роботи в системі:

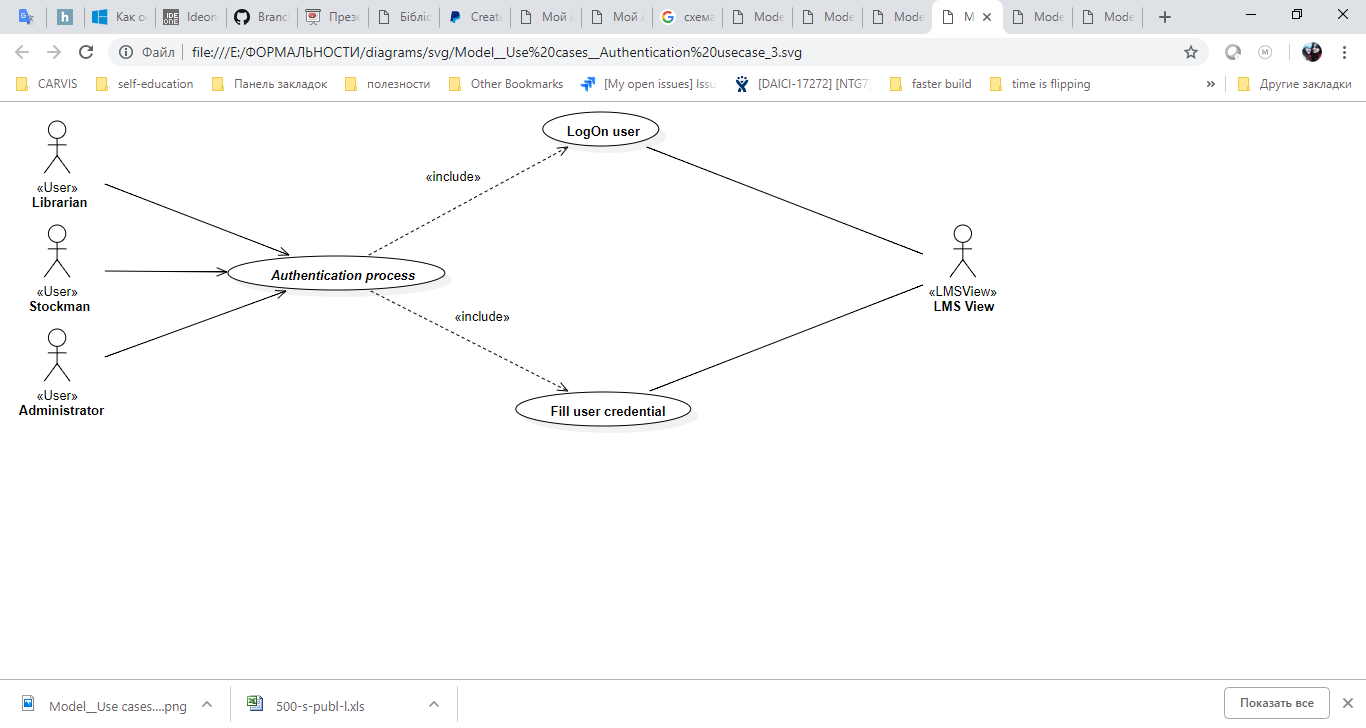


Рисунок 1.1 Діаграма варіантів використання загальна

На діаграмі ми можемо бачити, що відповідно до наших вимог, перед початком роботи з системою необхідно пройти процес Авторизації в системі.

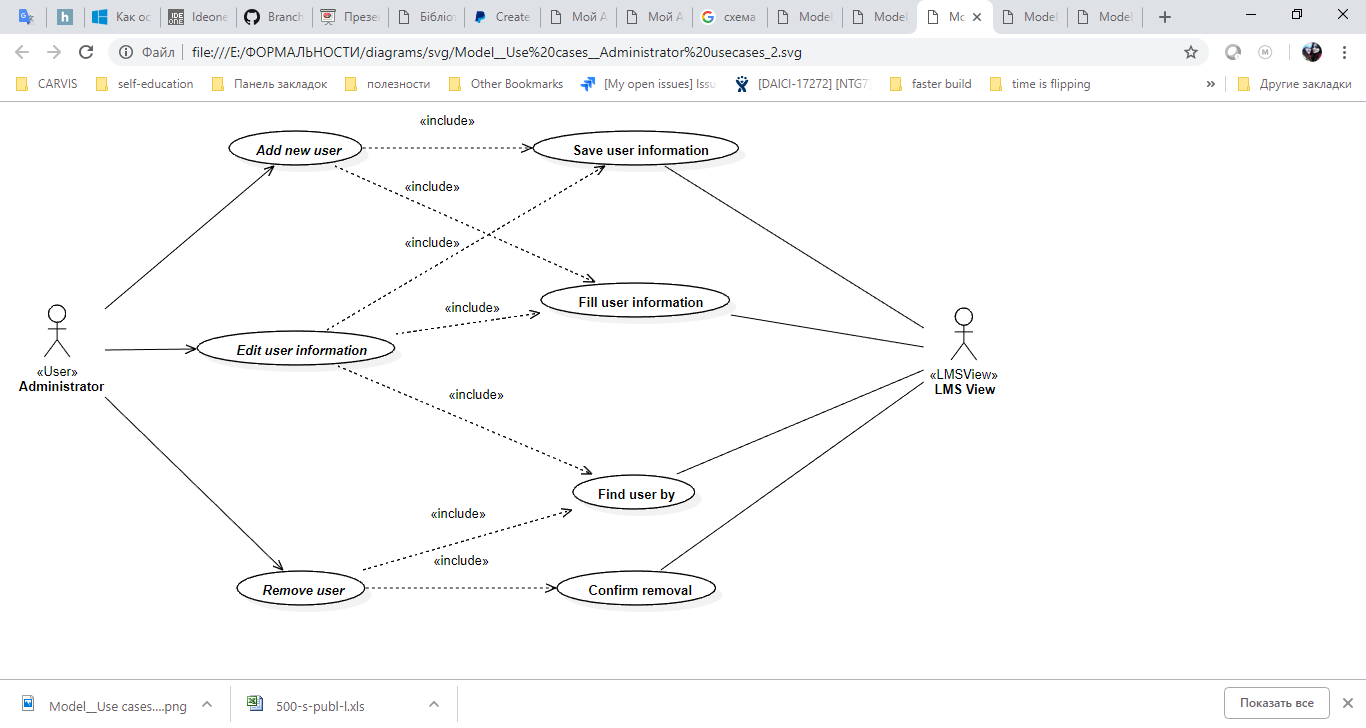


Рисунок 1.2 Діаграма варіантів використання для Актора «Адміністратор»

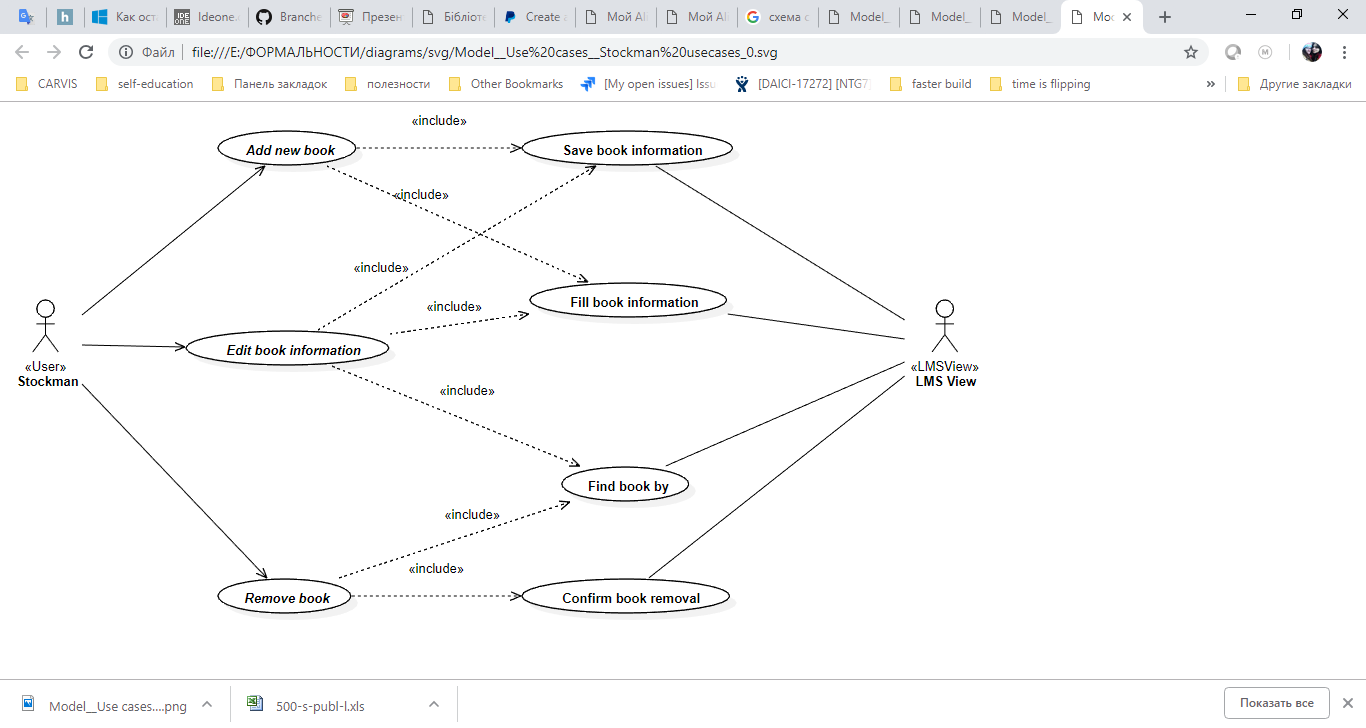
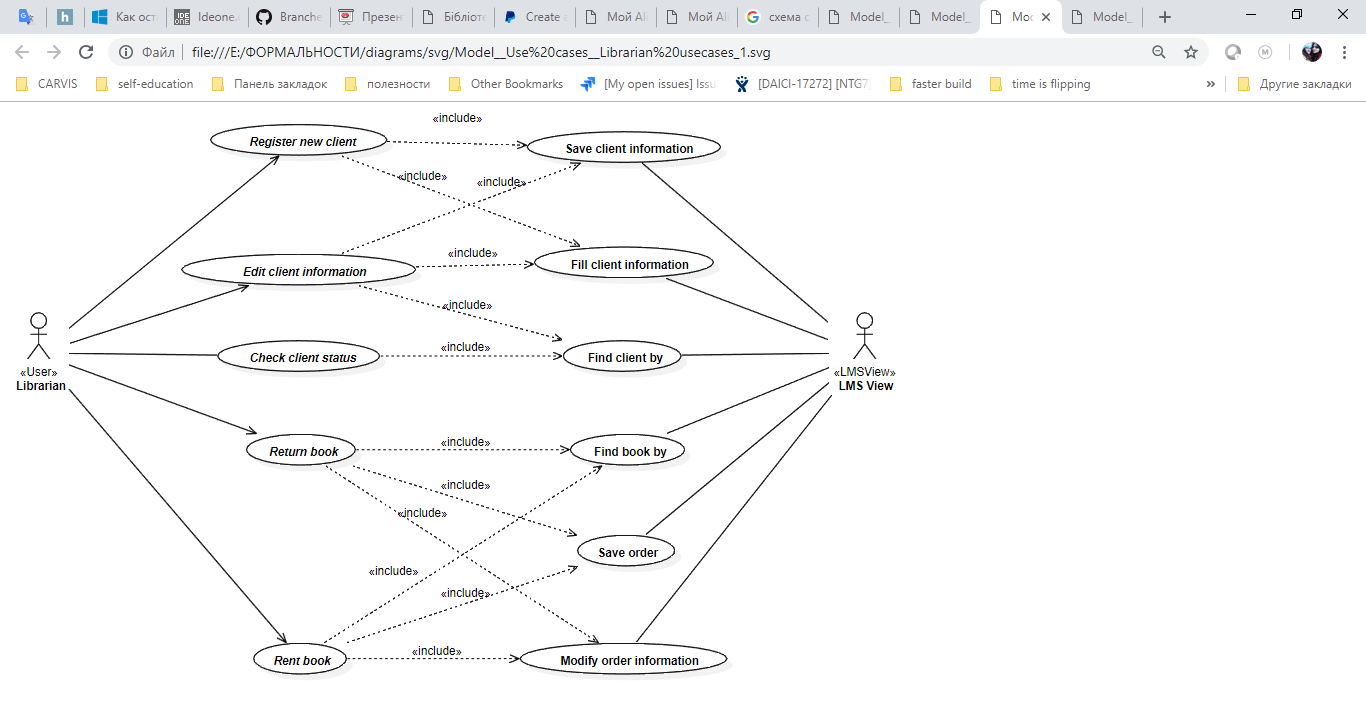


Рисунок 1.3 Діаграма варіантів використання для Актора «Комірник»

 Рисунок 1.4 Діаграма варіантів використання для Актора «Бібліотекар»

Ми завершили визначення варіантів використання та іллюстрування вимог та взаємозв’язків. Діаграма для кожного актора показує наглядним шляхом які windows вікна ми маємо розробити для кожного з них, відповідно до вимог. Тобто кожен варіант використання – це вікно для вводу інформації.

Добре обґрунтований проект, дозволяє виявити усі недоліки системи, та виправити їх, ще до початку кодінгу. Також, проектування є допоміжним засобом, для подальшого тестування, за рахунок того, що воно поєднує у собі усі особистості роботи системи, та допомагає проаналізувати схожість розробленої системи з проектом.

Кожен із варіантів використання має спеціалізоване призначення, і має бути описан індивідуально.

Authentication process – перевірка користувача на наявність у системі (вже зареєстрованний)

Log On user – авторизація та створення робочого простору в залежності від типу користувача

Fill user credential – перевірка валідності введених даних

Add new user – при реєстрації нового користувача системи, необхідно ввести персональну інформацію про нього, а також ввести логін та пароль для того щоб була можливість за допомогою цих даних авторизувати користувача.

Edit user information – відрегувати інформацію про існуючого користувача.

Remove user – видалити користувача із системи

Fill user information – заповнити всі необхідні поля для збереження нового користувача у системі

Save user information – записати у базу данних інформацюю про користувача

Find by user – пошук користувача по різним критерія

Confirm remove – підтвердити намір видалити користувача з бази даних

Add new book – додати нову книжку до бази данних

Edit book information – відрегувати інформацію про наявну системі книжку

Remove book information – видалити інформацію про книгу з системи

Save book information – зберегти нову книгу в базі данних

Fill book information – заповнити необхідну інформацію про книги, перед записом до бази даних

Find book by – пошук книги по різним критерія

Confirm book remove – підтвердити намір видалити книгу з бази даних

Register new client – зарееструвати нового клієнта в системі

Edit client information – відреагувати інформацію про клієнта, який вже збереж у базі даних

Check client status – перевірити книги в наявності у клієнта, та дати коли він їх взяв

Return book – повернути книгу до бібліотеки

Rent book – взяти книгу з бібліотеки.

Save client information – зберегти нову інформацію про клиєнта

Fill client information – заповнити всю необхідну інформацію про клієнта

Find client by – пошук клієнта по різним критеріям

Find book by – пошук книги по різним критеріям

Save order – зберегти нове замовлення клієнта

Modify order information – змінити інформацію про замовлення

# 6.2 Створення схеми бази данних

Для розробки додатку було визначено перелік таблиць, їх струтуру та зв’язки між ними. Що є відображенням вимог до зберігаємої інформації. Таким чином було створено наступні таблиці – Person, Account, Position, Reader, Bibliography, Book, Instance, PublishHouse, Author, Subject, Book\_related\_Author

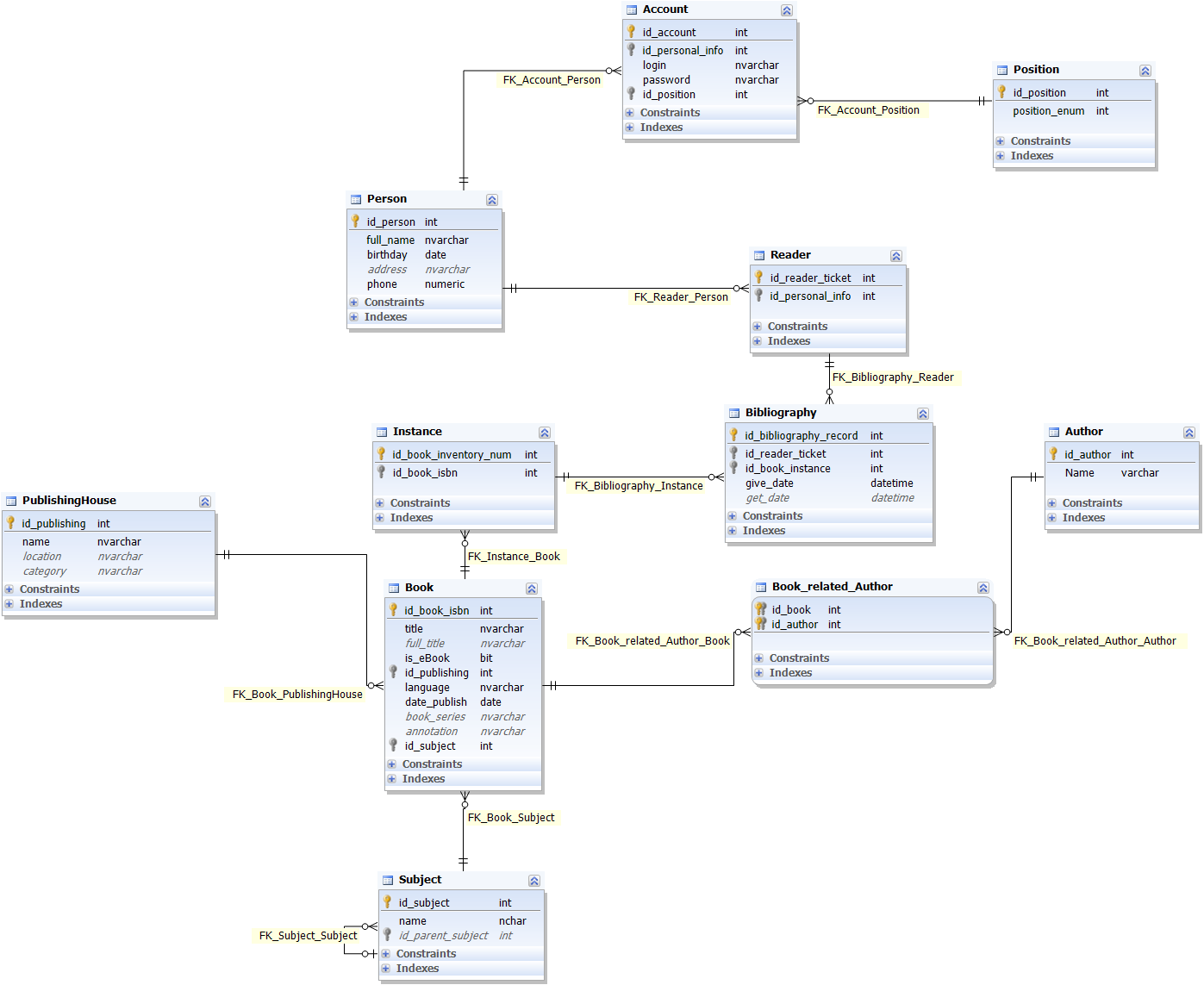


Рисунок 1.4 Структура бази даних для LMS

Структура та призначення кожної з таблиць

1. Person – складаеться з полів name, birth, phone, address. Необхідна для збереження особової інформації про користувачів и клієнтів. Ці данні винесені до окремої таблиця задля того, щоб уникнути зберігання однакових даних у базі даних
2. Account - складається з персональної інформація про користувача (name, birth, phone, address.) – ці дані вінесені до окремої таблиці, та логину та паролю для авторизації в системі. Також кожний запис містить поле position для того щоб ідентифікувати можливості користувача в системі. Ці данні винесені до окремої таблиці.
3. Position – складається з ідентиіфкаторів типу записів користувачів у системі. Ім’я цим ідентиіфкаторам присвоються у коді программи, задля збереження в майбутньому можливості розробити перемикач мови інтерфейсу та не втратити цілістність уже існуючих даних в системі.
4. Reader – складається з особової інформації про читачів - Ці данні винесені до окремої таблиця задля того, щоб уникнути зберігання однакових даних у базі даних - та номеру читацького квитка. Номер читацького квитка є унікальним полем для таблиці. Необхідний для однозначного пошуку читача у системі.
5. Bibliography – складається з інформації про бібліотечний запис. В собі зберігає інформацію про читата – тільки номер його читацького квитка – з таблиці Reader, інвентарний номер книги за таблиці Instance, дату коли читач взяв книгу, та коли повернув.
6. Book – складається з унікального коду книги – ISBN, з заголовку книги, повного заголовку книги до 150 символів, бінарне поле – чи є книжка в єлектронному форматі, видавництво (назва, місце розташування, та категорія книг) – винесено до окремої таблиці чтоб надати можливість в системі зберігати різні книги одного видавництва без копіювання даних; дати коли була опублікована книга, серія книги – чи є це окрема книга, або складається з серії книг, якщо вона є; мова якою написана книга; анотація до книги; предмет до якого відноситься книга – винесено до окремої таблиці, щоб уникнути копіювання даних.
7. Instance – містить в собі примірники книг. Складається з посилання на таблицю книг – на код ISBN, який є уніклальним для кожної книги, та зв’язаний з ним інвентарний код, значення якого можуть існувати для однієї книги в декількох примірників.
8. PublishHouse – містить в собі інформацію про видавництво. Це поля name, location, category
9. Author – містить інформацію про авторів. Їх ім’я.
10. Subject – містить інформацію про предмети, до яких віднясяться книги. Містить назву предмету, та якщо є то посилання до батьківського предмету.
11. Book\_related\_Author – містить в собі інформацію про книги та авторів. Необхідна для зоображення відношення – багато книг до декількох авторів. Та навпаки декулько авторів до однієї книги.

Приклад SQL коду для створення таблиці Book та налогодження зв’язків між іншими таблицями:

GO

/\*\*\*\*\*\* створення полів таблиці \*\*\*\*\*\*/

CREATE TABLE [dbo].[Book](

[id\_book\_isbn] [int] NOT NULL,

[title] [nvarchar](30) NOT NULL,

[full\_title] [nvarchar](150) NULL,

[is\_eBook] [bit] NOT NULL,

[id\_publishing] [int] NOT NULL,

[language] [nvarchar](50) NOT NULL,

[date\_publish] [date] NOT NULL,

[book\_series] [nvarchar](50) NULL,

[annotation] [nvarchar](256) NULL,

[id\_subject] [int] NOT NULL,

CONSTRAINT [PK\_dbo.Book] PRIMARY KEY CLUSTERED

(

/\*\*\*\*\*\* назначення унікального поля для ідентифікації книги \*\*\*\*\*\*/

[id\_book\_isbn] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* визначення зв’язків між таблицями. Поля які мають посилання на інші таблиці зв’язуються унікальними полями з цих полів \*\*\*\*\*\*/

GO

ALTER TABLE [dbo].[Book] WITH CHECK ADD CONSTRAINT [PublishingHouse\_Book] FOREIGN KEY([id\_publishing])

REFERENCES [dbo].[PublishingHouse] ([id\_publishing])

GO

ALTER TABLE [dbo].[Book] CHECK CONSTRAINT [PublishingHouse\_Book]

GO

ALTER TABLE [dbo].[Book] WITH CHECK ADD CONSTRAINT [Subject\_Book] FOREIGN KEY([id\_subject])

REFERENCES [dbo].[Subject] ([id\_subject])

GO

ALTER TABLE [dbo].[Book] CHECK CONSTRAINT [Subject\_Book]

# РОЗРОБКА ПРОГРАММНОГО ЗАБЕСПЕЧЕННЯ

Система розробляється в якості додатку до персонального компьютеру, та має складну архітектуру. Потрібно визначити необхідні компоненти для її працювання, та визначити зв’язки між різними частинами системи. Программа може працювати на різних операційних системах. В той час як Microsoft.NET Framework має можливість працювати на будь якій операційній системі де встановлена CLR для виконання программи, Microsoft SQL Server є системо залежним. Для вирішення цієї проблеми ми в якості середовища для розгортання серверу бази данних використовуємо ізольовану систему Docker. Також необхідно забеспечити відносно легке розгортування для того щоб почати працювати з системою. На наступній діаграмі розміщеня (deployment diagram), рис. 1.5, відображає фізичні взаємозв'язки між програмними і апаратними компонентами системи. Вона є хорошим засобом для того, щоб показати маршрути переміщення об'єктів і компонентів в розподіленій системі.

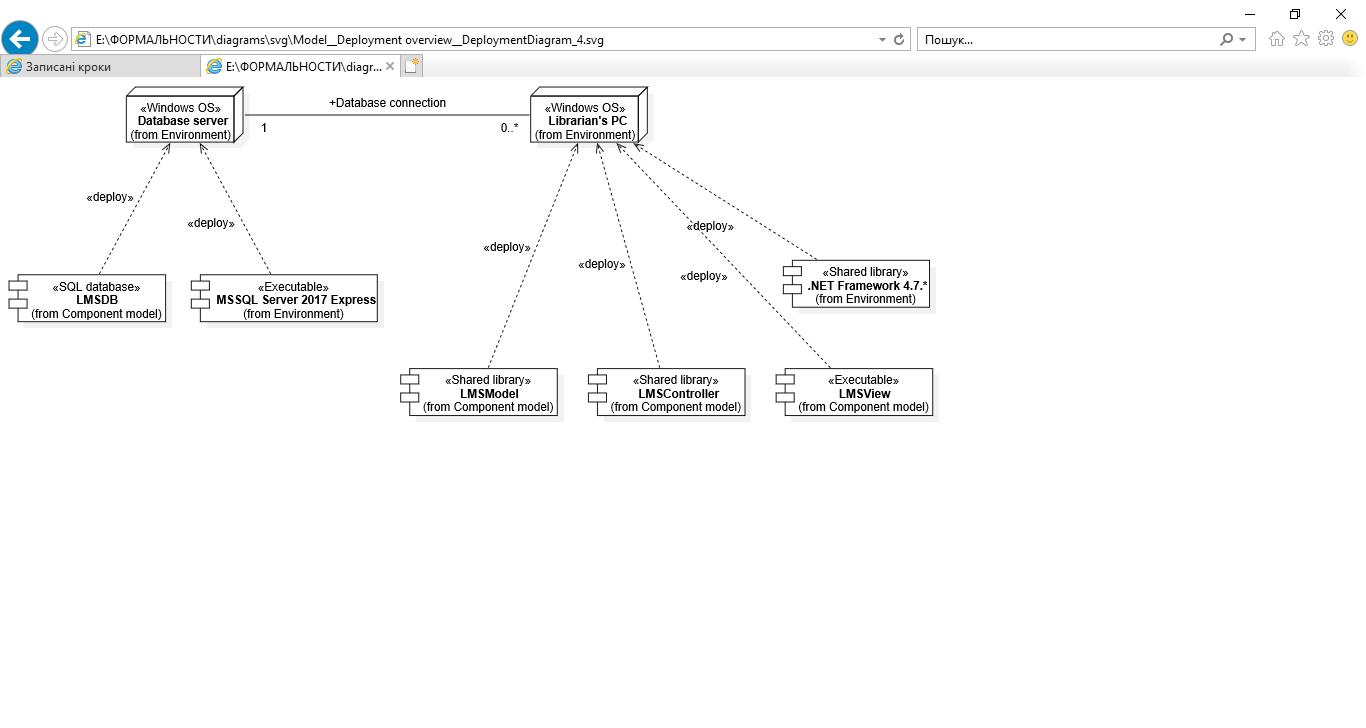


Рисунок 1.5 Діграма розміщення для системи

Кожен вузол на діаграмі відображає одну з цільових систем, або ж кінцевих.

Так компонети LMSView, LMSController, LMSModel – є частинами системи.

Які відображають собою підхід до программування з використанням відомого шаблону MVC – Model, View, Controller. В цьому підході до побудування архітектури необхідно розділяти обов’язки між компонентами. Так компонент LMSView – являє собою відображення View – легкої частини без важкої бізнес логіки. Ця частина системи відповідаю за інтерфейс та дизайн програми. Тої частини з якою взаємодіє користувач. LMSController – є Controller в орігінальному шаблоні. Це найважчий компонент, який відповідає за різні обробки данних, та переправляння їх між View та Model. Це є основна частина програми, де знаходиться бізнес логіка додатку. LMSModel виступає в ролі Model з орігінального шаблону. Ця частина відповідальна за роботу з даними, в нашому випадку за роботу з базою даних.

Microsoft.NET Framework 4.7.\* - це версія платформи яка використовувалася під час розробки. Але .NET Framework має одну важливу особливість – зворотню сумістність коду написаного під більш ранні версії .NET Framework.

Database Server – розгорнут в контейнері. Тож для цого запуску необхідно лише виканати зібраний контейнер. В якому в свою чергу буде виконуватися LMSDB – проект нашої бази данних, MS SQL Server 2017 Express – та сам сервер, який буде працювати в якості служби. Express видання дозволяє безкоштовно розміщувати бази даних до 10 ГБ.

З діаграми ми бачимо, що користувач нашої системи буде по мережі підключатися до серверу бази даних, та очікувати виконання запитів, які буде відпрвляти цому додаток за своєю внутрішньою логукою в залежності від дій користувача системи.

Інформаційна система розроблена в якості додатку до операційних систем на персональному компьютері з використанням сучасних технологічних рішень в програмуванні. Система побудована таким чином, що використовує три типи користувачів, які мають розрізнений доступ та різні функціональні можливості в системі. Інтерфейсна частина виконана в віконному стилі, де кожне вікно відповідає за відображення однієї можливості варіанту використання. При першому вході до інформаційної системи, користувач має авторизуватися, якщо він ще не заєстрований, то адміністратор повинен виконати реєстрацію користувача в системі, а вже потім авторизуватися в інформаційній системі та отримати доступ до її можливості в залежності від наданих прав.

На рис. 1.6, зображено вікно авторизації користувача

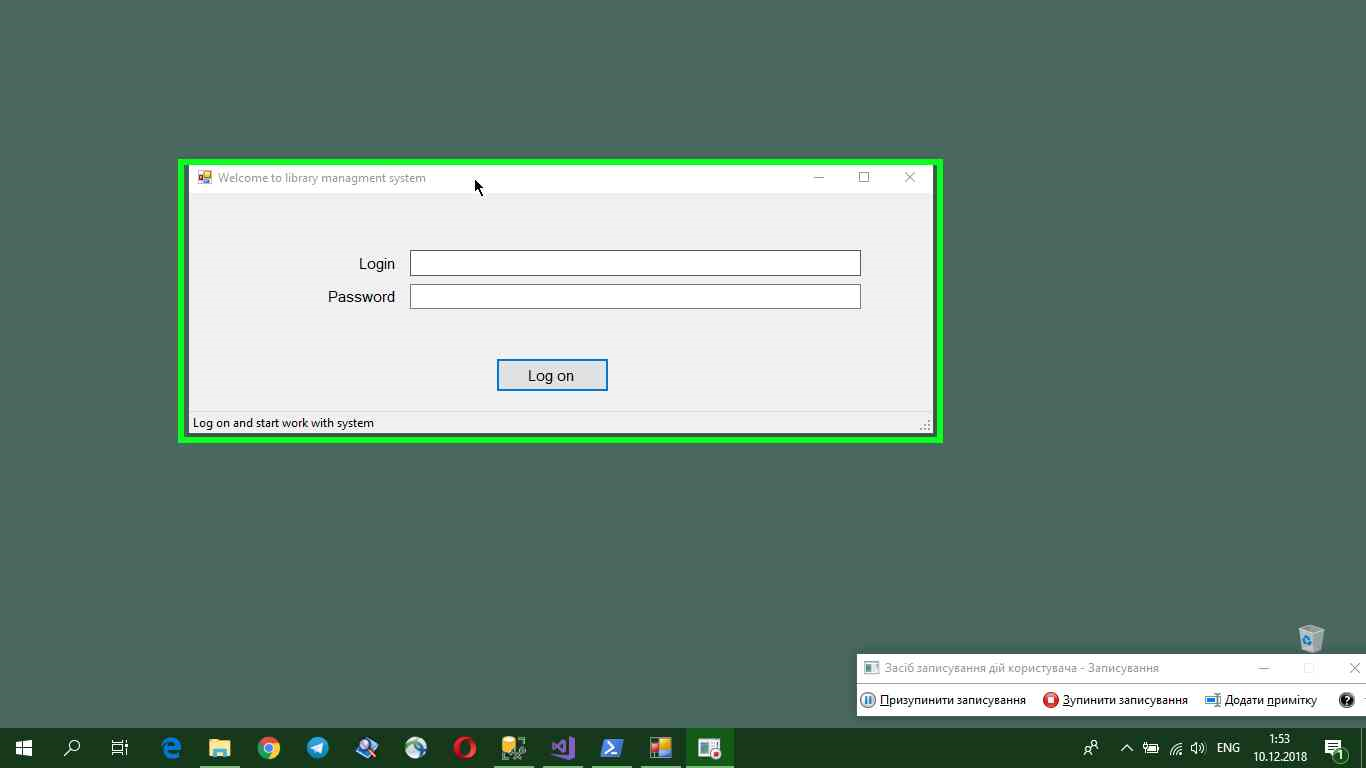


Рисунок 1.6 Вікно авторизації до системи

Форма авторизації користувача в ситемі включає в себе введення наступних обов’язкових полів. На рис. 1.7, та рис. 1.8, наведено приклад авторизації шляхом введення стандартних логіну та паролю (завжди є в системі за замовчуванням)

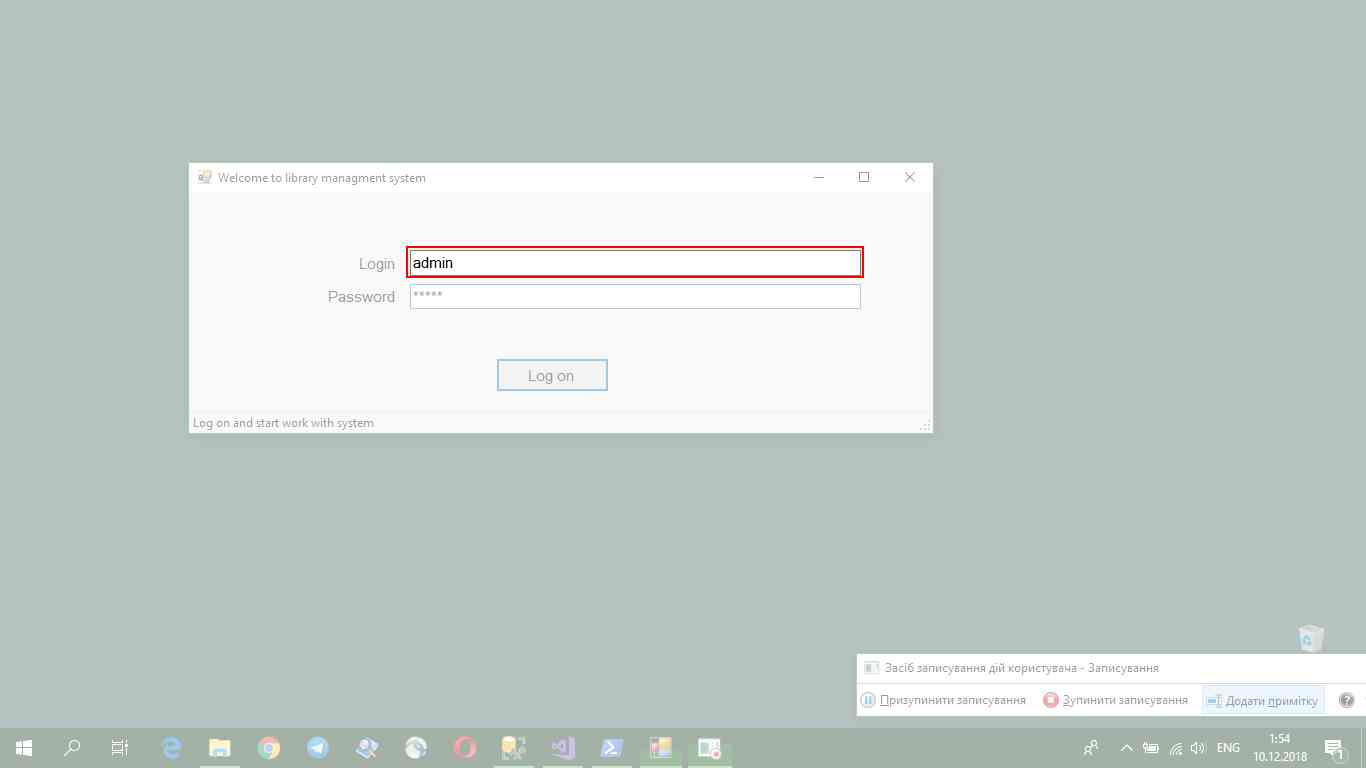


Рисунок 1.7 Поле для введення логіну користувача

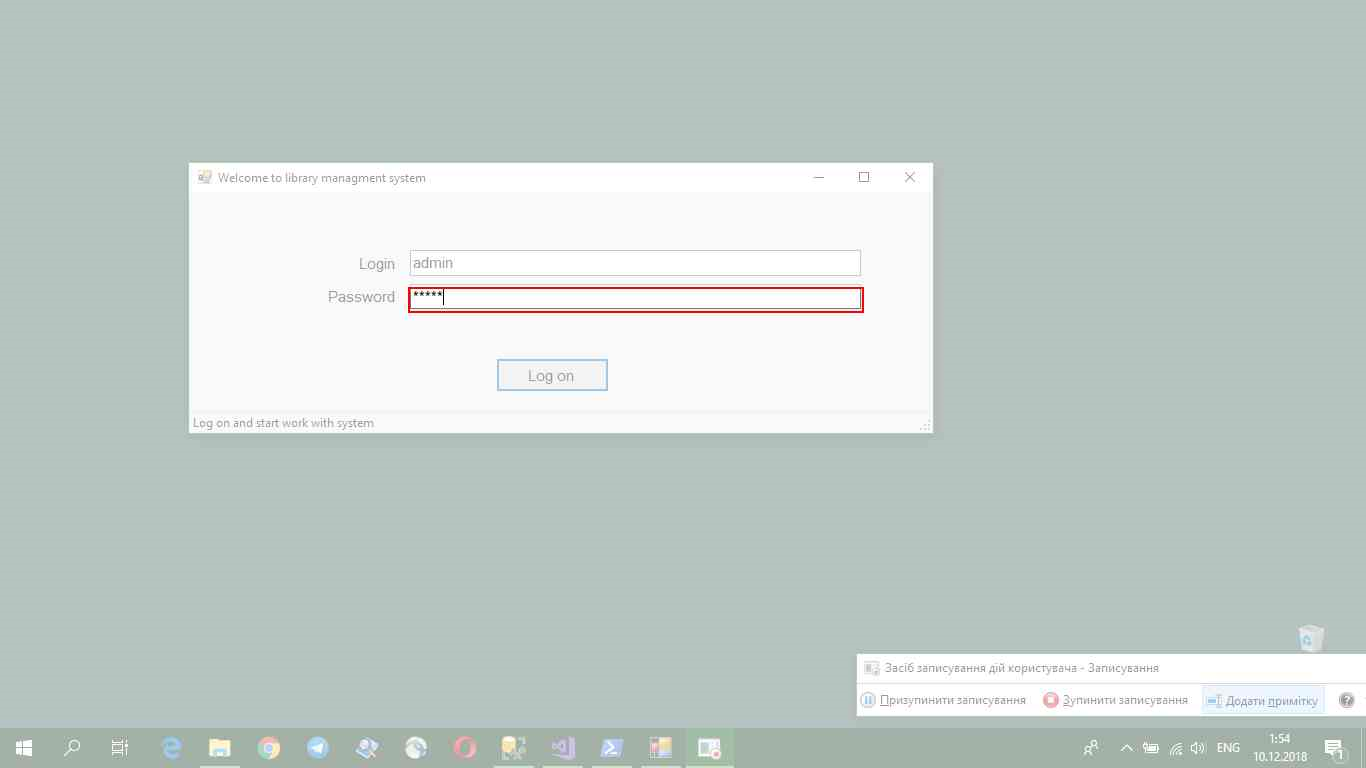


Рисунок 1.8 Поле для введення паролю користувача

На рис. 1.9, зображено натиск на кнопку підвердження авторизації до системи

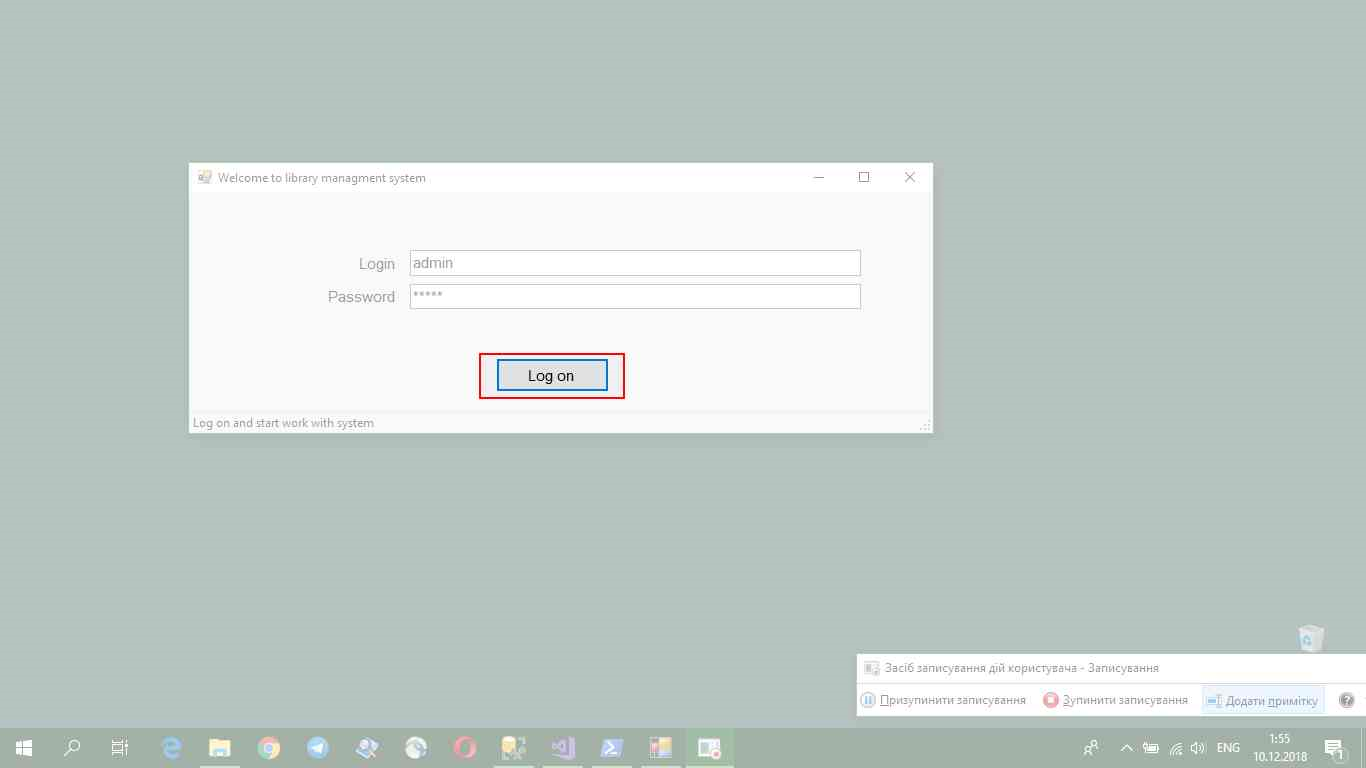


Рисунок 1.9 Кнопка авторизації

Далі наведенно код обробник кнопки авторизації, та обробка данних LMSController-ом.

LMSView:

//поле яке інкапсулює в собі інтерфейс авторизації

private IAuthentication authenticationProcess;

//конструктор форми

public UserLogOnForm()

{

InitializeComponent();

authenticationProcess = new AuthenticationProcess();

//підписування на події в контролері в випадку успішної авторизації кожного з типу //користувачів, і у випадку помилки

authenticationProcess.OnAdminLogOn += AuthorizeAdmin;

authenticationProcess.OnLibrarianLogOn += AuthorizeLibrarian;

authenticationProcess.OnStockmanLogOn += AuthorizeStockman;

authenticationProcess.OnFailedLogOn += OnFailedLogOn;

//код методів, що виконується у випадку успішної авторизації бібліотекаря

public void AuthorizeLibrarian(IWorkspaceLibrarian workspaceLibrarian)

{

Hide();

using (LibrarianControlForm mLibraryManageWin = new LibrarianControlForm(workspaceLibrarian))

{

mLibraryManageWin.ShowDialog();

}

Show();

}

//код методів, що виконується у випадку успішної авторизації адмінастратора

public void AuthorizeAdmin(IWorkspaceAdmin workspaceAdmin)

{

Hide();

if (workspaceAdmin != null)

{

using (UserInfoSearchForm mAdminManageWin = new UserInfoSearchForm(workspaceAdmin.UserInformationRegister))

{

mAdminManageWin.ShowDialog();

}

}

Show();

}

//код методів, що виконуються у випадку успішної авторизації комірника

public void AuthorizeStockman(IWorkspaceStockman workspaceStockman)

{

Hide();

using (BookInfoSearchForm mStockmanManageWin = new BookInfoSearchForm(workspaceStockman))

{

mStockmanManageWin.ShowDialog();

}

Show();

}

//код обробника кнопки авторизації

private void ButtonlogOn\_Click(object sender, System.EventArgs e)

{

authenticationProcess.Authorize(new UserCredential() { Name = textBoxLogin.Text, Password = textBoxPassword.Text });

ClearTextFields();

Activate();

}

LMSController:

// метод авторизації

public void Authorize(UserCredential userCredential)

{

if (userCredential == null)

{

throw new ArgumentNullException(nameof(userCredential));

}

//пошук в базі данних зареєстрованного користувача

Account account = Accounts.FindByUserNameAndPassword(userCredential.Name, userCredential.Password);

if (null != account)

{

//створення робочого простору в залежності від відповіді бази даних

switch (account.Position.position\_enum)

{

case UserPosition.Librarian:

OnLibrarianLogOn(new LibrarianWorkspace());

break;

case UserPosition.Administrator:

OnAdminLogOn(new AdminWorkspace());

break;

case UserPosition.Stockman:

OnStockmanLogOn(new StockmanWorkspace());

break;

default:

throw new InvalidOperationException(account.Position.position\_enum.ToString());

}

}

На діграмі класів функціональність авторизації виглядає наступним чином:

LMSView:

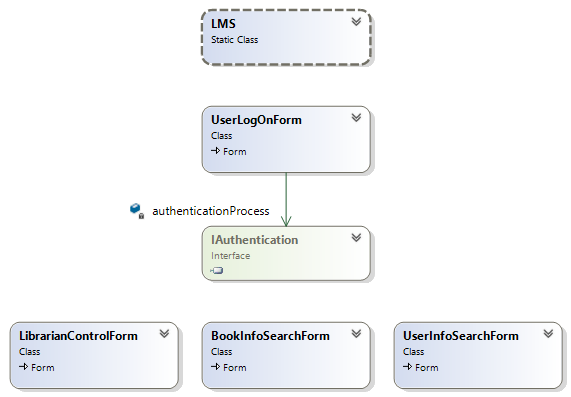


Рисунок 1.10 Діаграма класів для процессу авторизації LMSView

На приведеній діагрмі видно відношення з боку LMSView до інтерфейсів LMSController-а. Так IAuthentication – відновідальний за подальшу активацію вікна, в залежності від результату.

LMSController:

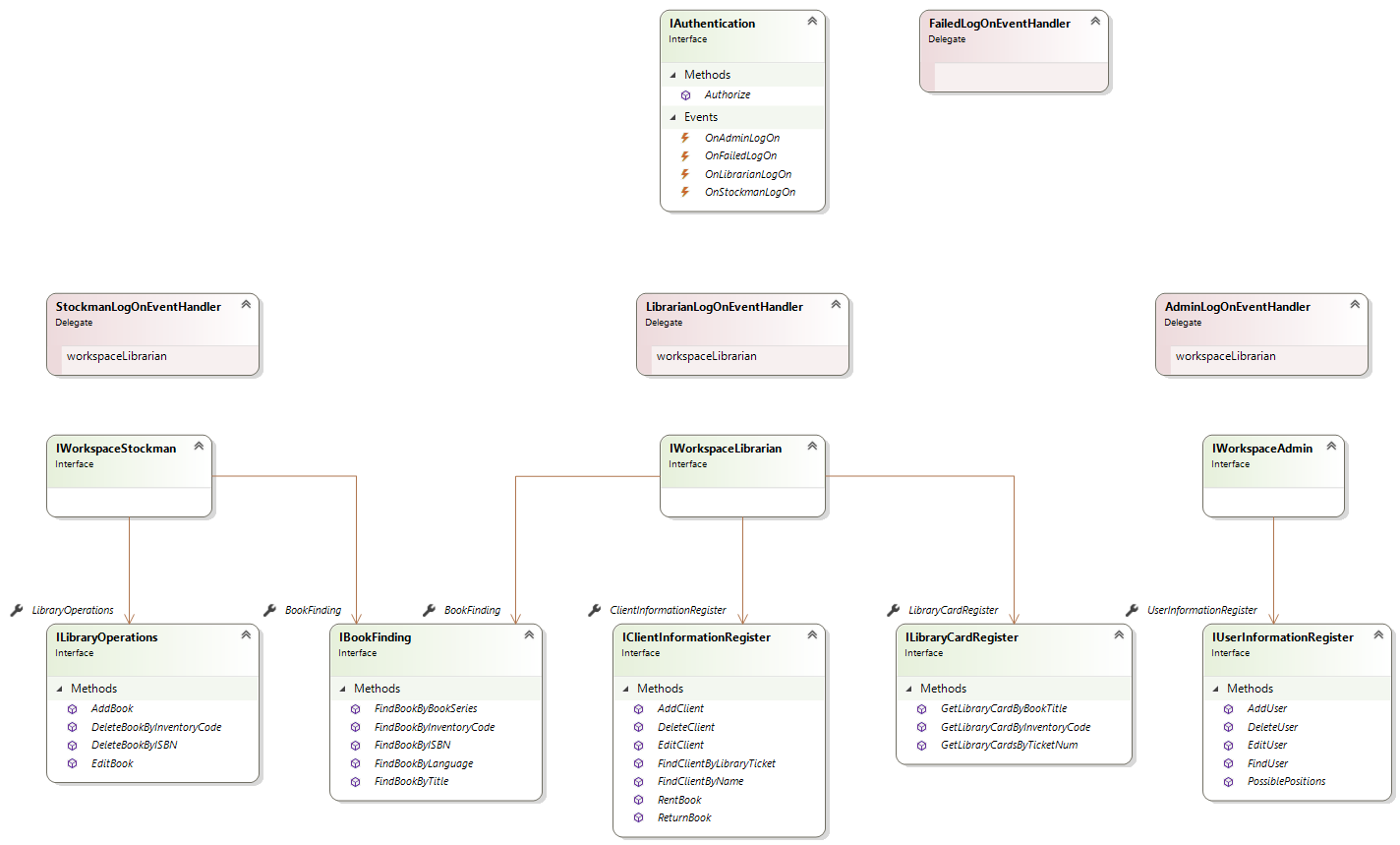


Рисунок 1.11 Діаграма класів для процессу авторизації LMSController

Відповідно до введених нами данних на попередніх вікнах, ми авторизувалися під обліковим записом адміністратора. З цим обліковим записом ми можемо шукати інформацію про користувачів в системі, переглядати її, редагувати, а також створювати нових користувачів. Приклад вікна для перегляду та редагування інформації наведено на рис. 1.12. Таке ж саме вікно ми бачимо у випадку створення нового користувача, але його поля є пустими. Для зручності користувача – адміністратору, обов’язкові поля для заповнення позначені \*.

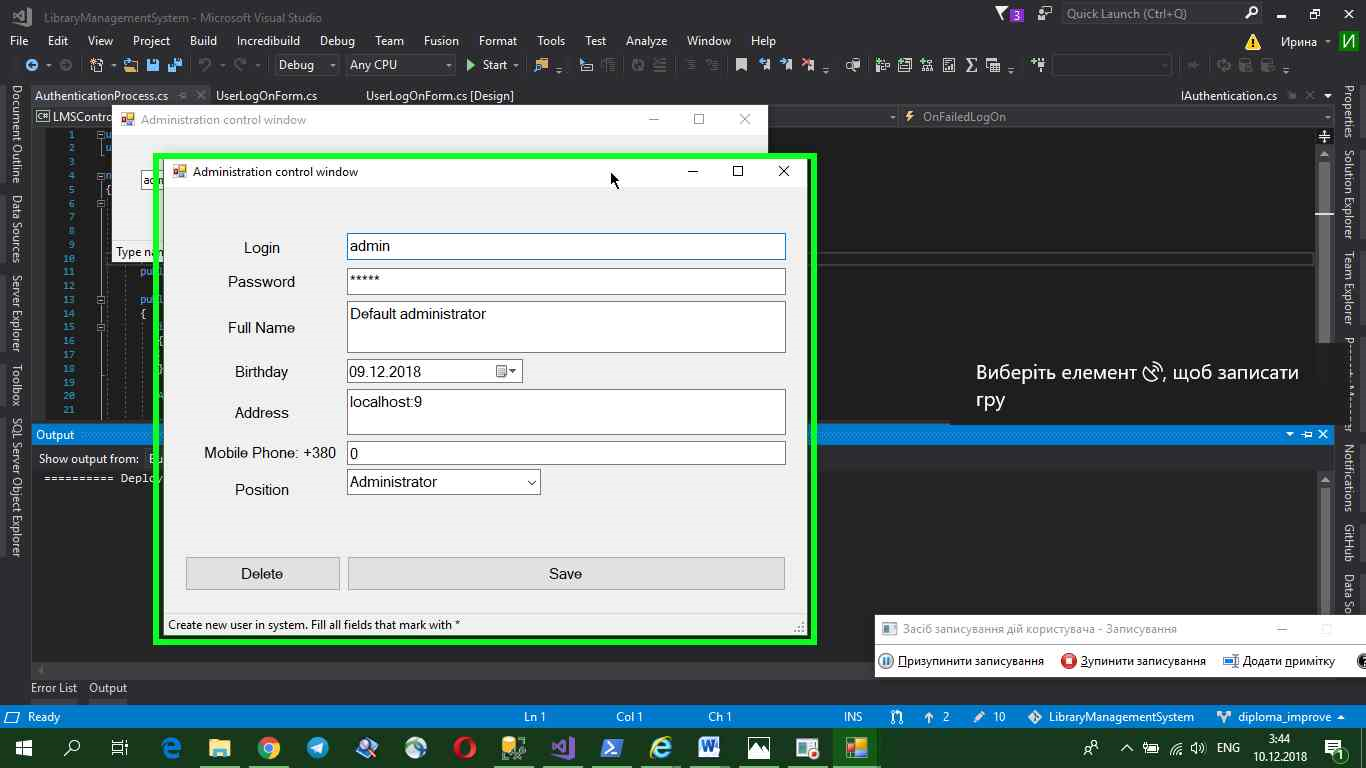


Рисунок 1.12 вікно редагування інформації користувачів

На настпних діаграмах класів зображені інтерфейси для спілкування між LMSView та LMSController та за допомогою зв’язків асоціації можна бачати взаємовз’язок інтеріфейсів, в випадку авторизації адміністратора. На діаграмі вказано, що View використовує AdminWorkspace через IAdminWorkspace для комунікації та передачі даних.

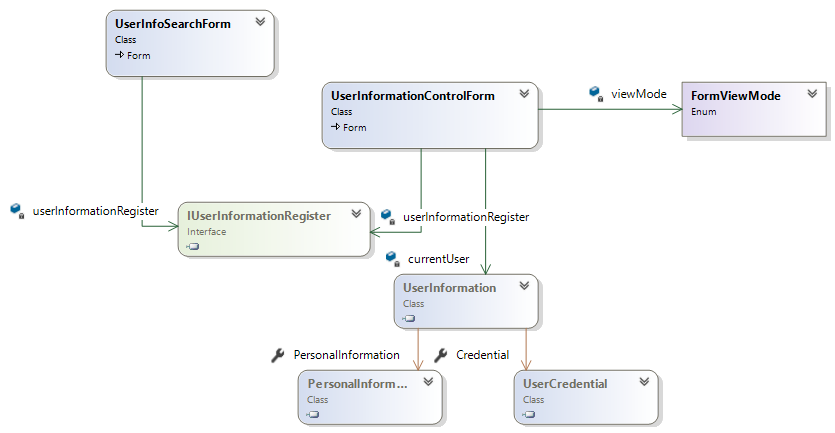


Рисунок 1.13 Діаграма класів для робочого простору адміністратора LMSView

На діаграмі ми бачимо що точка входу до - є UserInformationSearchForm – вікно для пошуку користувача. В випадку якщо користувач знайден, показати – UserInformationControlForm –це форма для відображення інформації про користувача, наведена на Рис. 1.12. Оперує UserInformation, яку передає надалі до контролера. У випадку будь-якої роботи з інформацією про користувача – створення нового, редагування існуючого або його видалення.

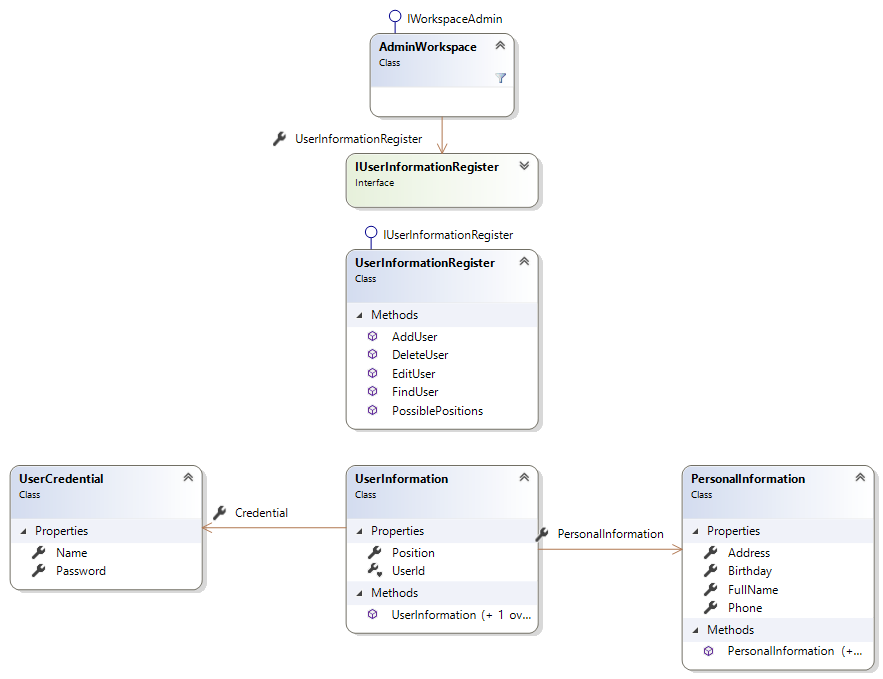


Рисунок 1.14 Діаграма класів для робочого простору адміністратора LMSController

Якщо користувач авторизувався під обліковим записом комірника бібліотеки, то він має наступні можливості: знайти книгу, наприклад по її назві, змінити інформацію про книгу, додати примірників до конкретного ISBN коду, або видалити іх. Також в свою чергу комірник може переглянути наявні в бібліотеці книги, та яких примірників зараз немає. Процесс авторизації комірника такий самий як і адманастратора, лише необхідно ввести інформацію про вже зареєстрованного користувача з типом облкового запису – комірник. Якщо такого не існує, то створити новий такий запис за допомогою адміністративного облікового запису. На рис. 1.15., зображено вікно для редагування інформації про книгу. Поля, які наведені на формі відповідають даним, що ми зберігаемо в базі даних.

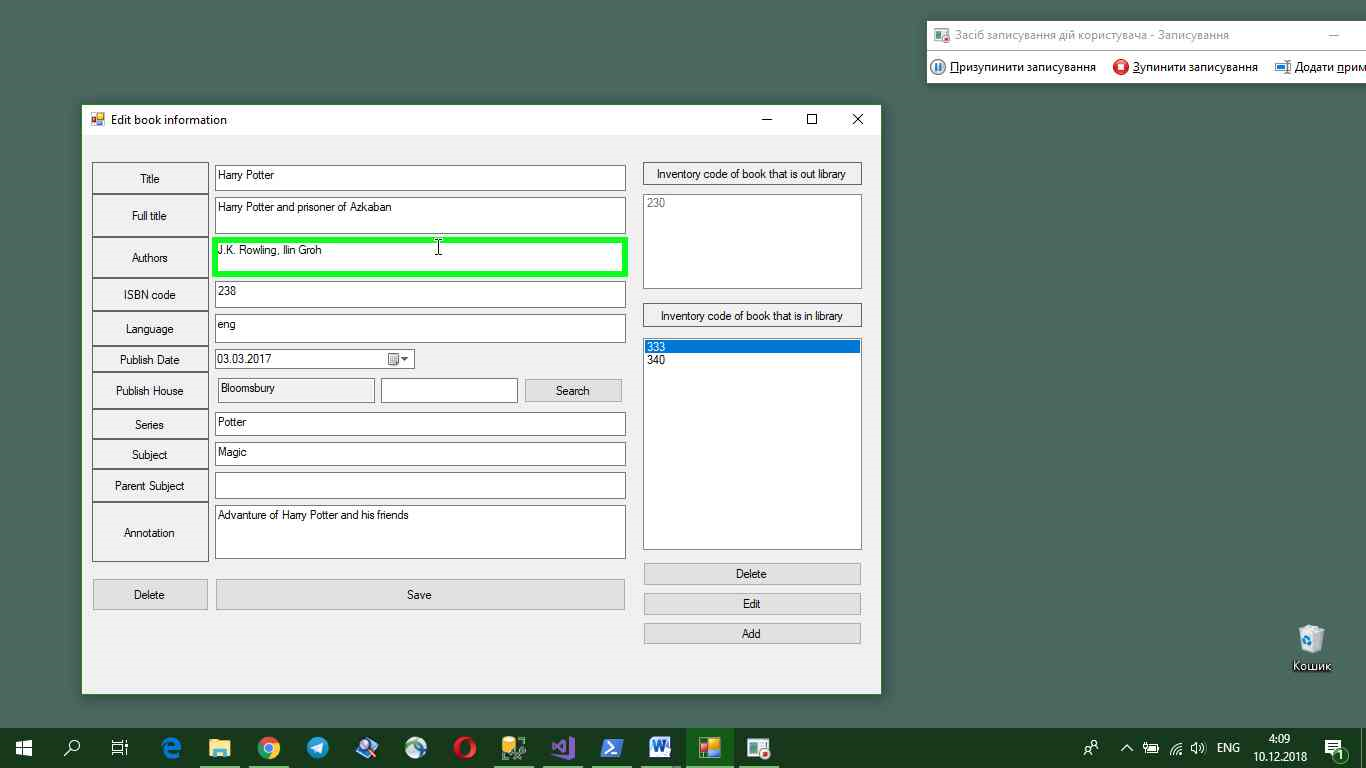


Рисунок 1.15 вікно редагування інформації про книги та їх примірники

Далі наведено приклад коду, пакування данних з форми та відправлення їх до контролеру задля того щоб записати в базу даних

//обробник кнопки Save

private void ButtonSave\_Click(object sender, EventArgs e)

{

if (FormViewMode.Create == viewMode)

{

currentBook = new BookInformation();

}

//передача запаковоного єкзампляра стркуктури данних до подальшої обробки в контроллер

libraryManagment.AddBook(PackageBookInformation(currentBook)

}

//пакування інформації до внутрішньої структури данних BookInformation

private BookInformation PackageBookInformation(BookInformation book)

{

Dictionary<int, bool> mapInventoryCodes = new Dictionary<int, bool>();

foreach (var code in listBoxInventoryCodeAvailable.Items)

{

mapInventoryCodes.Add(Convert.ToInt32(code, CultureInfo.CurrentCulture), true);

};

foreach (var code in listBoxInventoryCodeNotAvailable.Items)

{

mapInventoryCodes.Add(Convert.ToInt32(code, CultureInfo.CurrentCulture), false);

};

book.InventoryCode = mapInventoryCodes;

book.Title = textBoxTitle.Text;

book.FullTitle = textBoxFullTitle.Text;

book.ISBN = Convert.ToInt16(textBoxIsbn.Text, CultureInfo.CurrentCulture);

book.Language = textBoxLanguage.Text;

book.PublishHouse = publishHouseRegister.FindByName(textBoxPublishHouseResult.Text);

book.PublishDate = dateTimePickerPublish.Value;

book.Annotation = textBoxAnnotation.Text;

string[] authorsNames = textBoxAuthors.Text.Split(new char[] { ',', ' ' });

List<AuthorInformation> authors = new List<AuthorInformation>();

foreach (string name in authorsNames)

{

authors.Add(new AuthorInformation(){Name = name});

}

book.Authors = authors.ToArray();

book.Subject = new SubjectInformation() { Name = textBoxSubject.Text };

return book;

}

На діаграмі класів функціональність пошуку та редагування книг виглядає наступним чином:

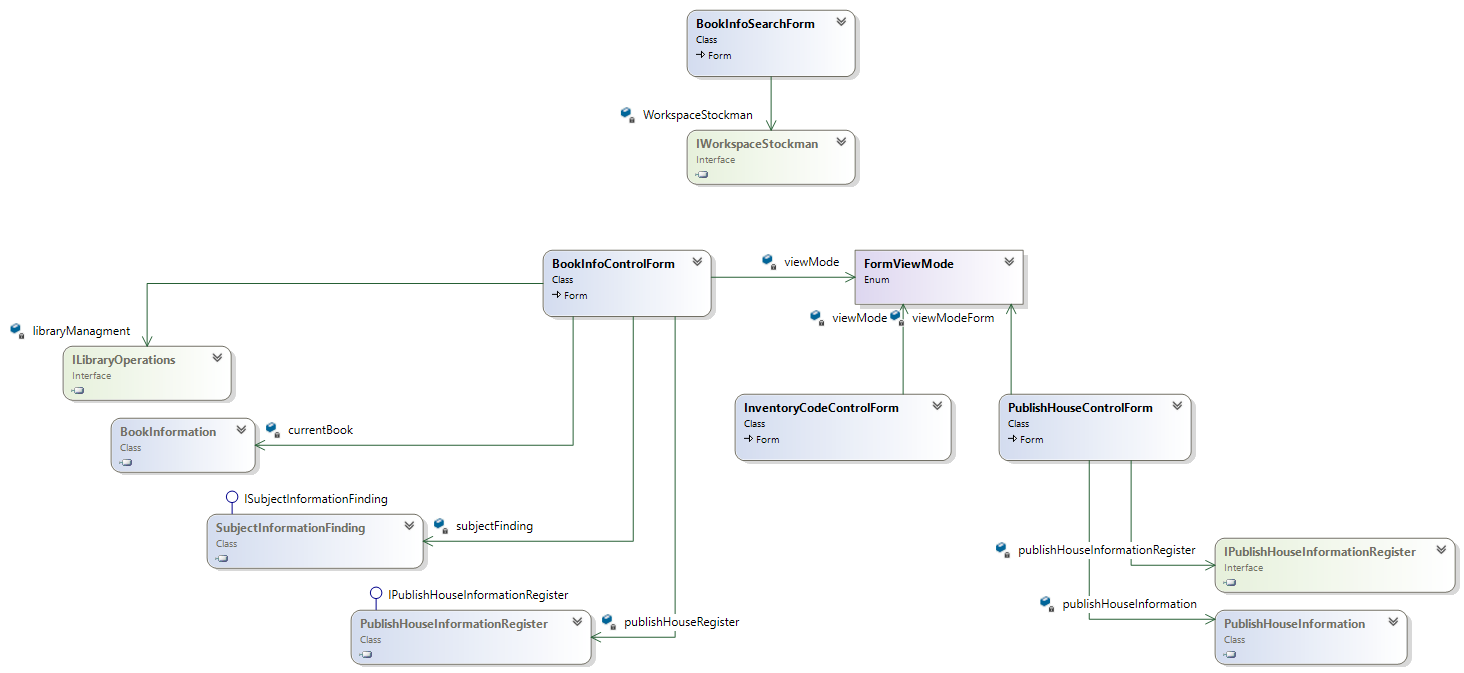


Рисунок 1.16 Діаграма робочого простору для комірника LMSView

На Рис. 1.15 зображенно вікно яке відповідає BookInfoControlForm. Воно використовує FormViewMode для визначення режиму його відображення, Edit – якщо редагування. Має обробник малювання форми з заповнення інформації в текстові поля на формі. В випадку малювання форми в режимі Create усі поля залишаються пустими. Вікно InventoryCodeControlForm, призначенне для призначення інвентарних номерів книгам по ISBN коду. В правій частині BookInfoControlForm, розташовані листи для інвентарних номерів для книг, що зараз в бібіліотеці, та поза її межами – видані читачам. Обидва листи підписані. Також під ними розташовані кнопки для управління інвентарними номерами.

На Рис.1.17., наведено вікно для редагування та створення інвентарних номерів для книги. Далі можна зберегти приміврник книги, або вийти з вікна, та видалити примірник, виділивши в листі код, та натиснувши кнопку Delete.

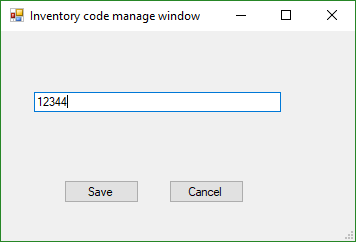


Рисунок 1.17 Вікно для редагування або створення інвентарних номерів книги

На Рис.1.18., наведено вікно для редагування та створення видавництв, які ще не існують в системі - PublishHouseControlForm. Далі можна зберегти іх, або вийти з вікна створення та редагування. Про те нове видавництво так само як інвентарний код потрапить до бази даних лише після того, як на основній формі BookInfoControlForm буде підтвердженно зберігання інформації.

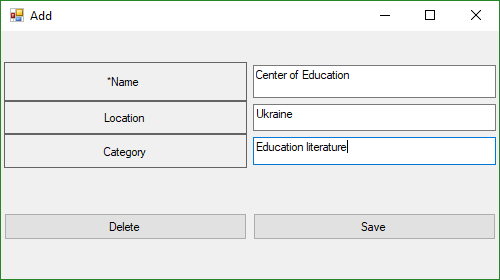
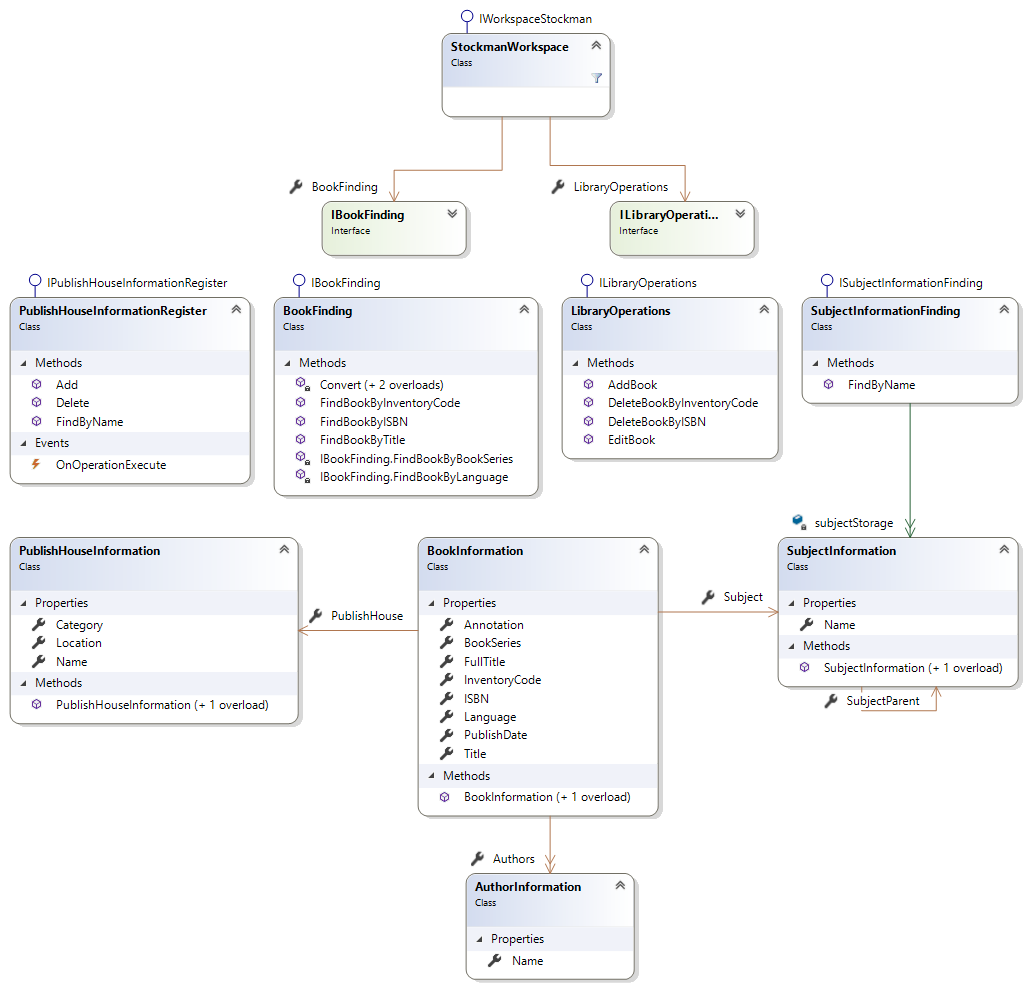


Рисунок 1.18 Вікно для редагування або створення Видавництв

За боку Контроллера діаграма класів для комірника має наступний вигляд:

Рисунок 1.19 Діаграма робочого простору для комірника LMSController



У собі Контроллер реалізує інтерфейс для пошуку книг IBookFinding, інтерфейс для працювання з видавництвами IPublishHouseInformationRegister, для пошуку предметів ISubjectInformationFinding, та бібліотечні операції ILibraryOperation – додати книги до системи, видалити чи модіфукувати інформацію. Робочий простір для комірника оперує такими структурами даних : SubjectInformation, PublishHouseInformation, BookInformation.

Робочий простір для Бібліотекаря на діаграмі класів має наступні форми, класи та компоненти для взаємодії:

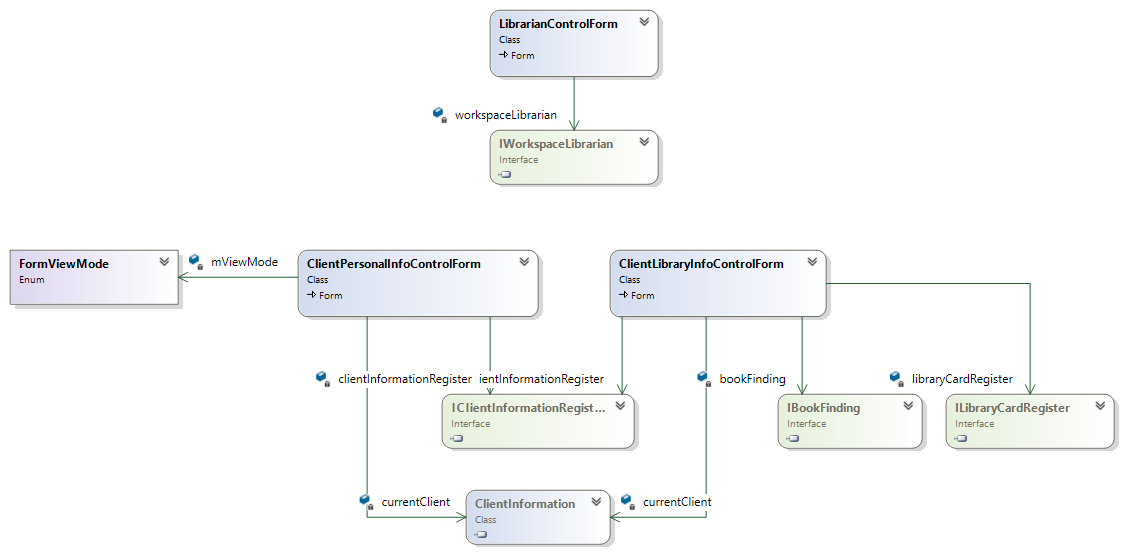


Рисунок 1.20 Діаграма робочого простору для бібіліотекаря LMSView

View компонент Має такі форми для взаємодії з бібілотекарем LibrarianControlForm – основна форма, на якій бібліотекар має можливість шукати книгу, шукати читача (клієнта в межах системи). Форма ClientPersonalInforControlForm необхідна для роботи з данними користувача – це є персональні данні та номер читацького квитка. Форма ClientLibraryInfoControlForm необхідна для роботи з записами в біліотецу. Які книги взяв поточний читача.  
Робочий простір бібліотекаря надає можливість працювати з інтерфейсом IBookFinding для пошуку кни у системі, з інтерфейсрм ILibraryCardRegister для роботи з бібіліотечними записами в бібліотеці. Інтерфейс IClientInformationRegister потрібен для роботи з записами клієнтів. Створити, модифікувати, видалити клієнта. Робочий простір бібліотекаря оперує ClientInformation – структура даних для зберігання інформації про клієнта. З боку контроллера робочий простір бібіліотекаря має такі діаграми класу:

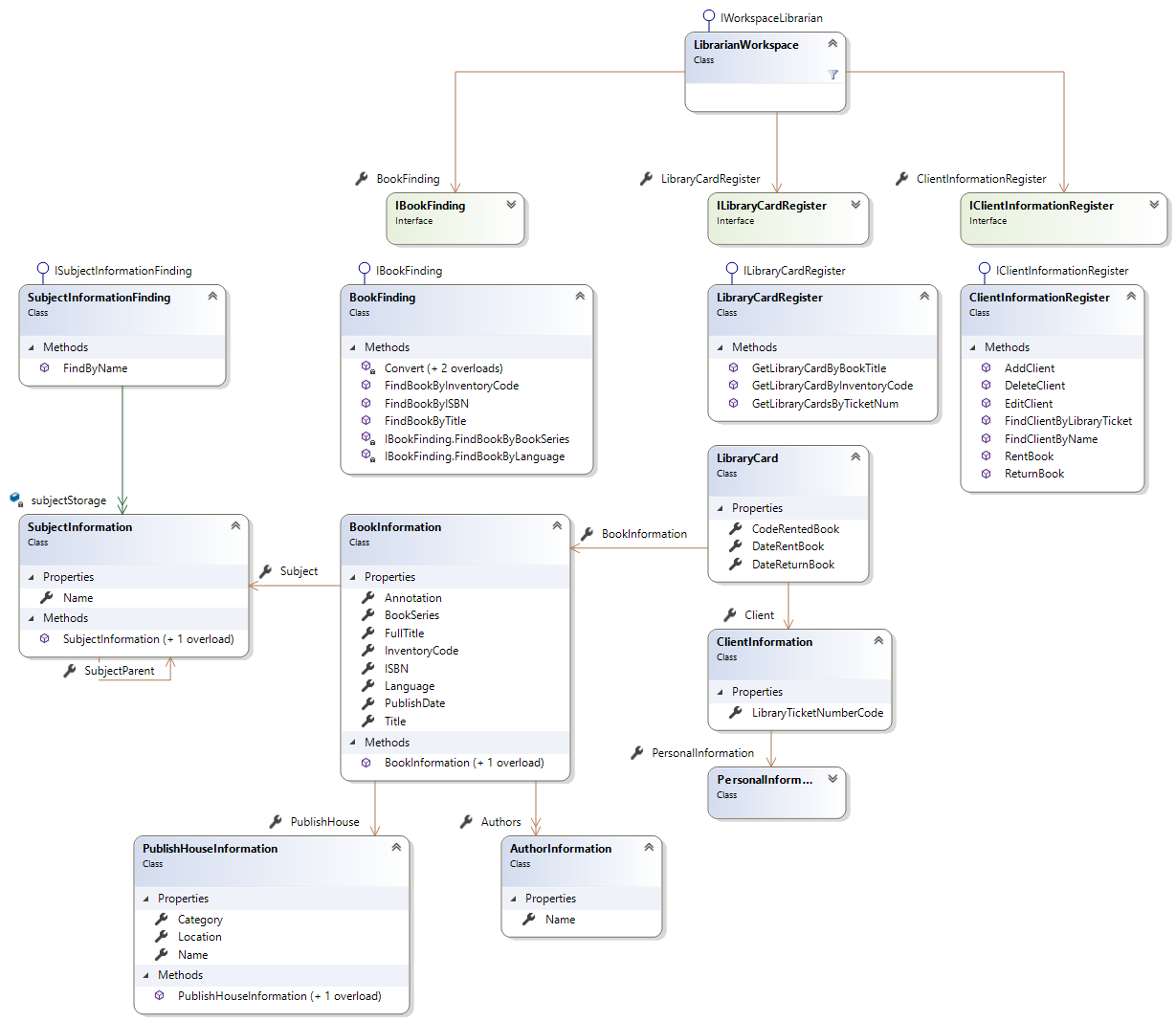


Рисунок 1.21 Діаграма робочого простору для бібіліотекаря LMSController

# 8 РОЗРОБКА ЗАВДАННЯ ПО ОХОРОНІ ПРАЦІ

* 1. Вимоги до мікроклімату комп'ютерних приміщень. Засоби його оптимізації

Санітарні норми мікроклімату комп'ютерних приміщень регламентируються ДСН 3.3.6.042-99.

Санітарні норми поширюються на умови мікроклімату в межах робочої зони комп'ютерних приміщень підприємств, закладів, установ тощо, незалежно від їх форми власності та підпорядкування.

Цей документ регламентує нормативні величини оптимальних та допустимих показників мікроклімату та встановлює вимоги до методів вимірювання мікрокліматичних параметрів та їх оцінки [19].

Вимоги до параметрів мікроклімату комп'ютерних приміщень

Мікрокліматичні умови комп'ютерних приміщень характеризуються такими показниками:

- температура повітря;

- відносна вологість повітря;

- швидкість руху повітря;

- інтенсивність теплового (інфрачервоного) опромінення.

За ступенем впливу на тепловий стан людини мікрокліматичної умови поділяють на оптимальні та допустимі. Для робочої зони комп'ютерних приміщень встановлюються оптимальні та допустимі мікрокліматичні умови з урахуванням важкості виконуваної роботи та періоду року. При одночасному виконанні в робочій зоні робіт різної категорії важкості рівні показників мікроклімату повинні встановлюватись з урахуванням найбільш чисельної групи працівників [19].

Оптимальні умови мікроклімату.

Оптимальні умови мікроклімату встановлюються для постійних робочих місць. Показники температури повітря в робочій зоні по висоті та по горизонталі, а також протягом робочої зміни не повинні виходити за межі нормованих величин оптимальної температури для даної категорії робіт.

Температура внутрішніх поверхонь робочої зони (стіни, підлога, стеля), технологічного обладнання (екрани і т. ін.), зовнішніх поверхонь технологічного устаткування, огороджуючи конструкцій не повинна виходити більш ніж на 2 грд.

При виконанні робіт операторського типу, пов'язаних з нервово-емоційним напруженням в кабінетах, пультах і постах керування технологічними процесами, в залах обчислювальної техніки та інших приміщеннях повинні дотримуватися оптимальні умови мікроклімату (температура повітря 22 – 24 грд. С, відносна вологість 60 - 40 %, швидкість руху повітря не більш 0,1 м/сек.).

Допустимі величини мікрокліматичних умов встановлюються у випадках, коли на робочих місцях не можна забезпечити оптимальні величини мікроклімату за технологічними вимогами виробництва, технічною недосяжністю та економічно обґрунтованою недоцільністю.

Величини показників, які характеризують допустимі мікрокліматичні умови, встановлюються для постійних і непостійних робочих місць, які наведені в табл. 3.1.

Перепад температури повітря по висоті робочої зони при забезпеченні допустимих умов мікроклімату не повинен бути більше 3 грд. С для всіх категорій робіт, а по горизонталі робочої зони та протягом робочої зміни - виходити за межі допустимих температур для даної категорії роботи. Рівні іонізації повітря приміщення при роботі на ВДТ та ПЕОМ в табл. 3.2.

Температура внутрішніх поверхонь приміщень (стіни, підлога, стеля), а також температура зовнішніх поверхонь технологічного устаткування або його захисних обладнань (екранів і т. ін.) не повинна виходити за межі допустимих величин температури повітря для даної категорії робіт.

Таблиця 3.1 Оптимальні величини температури, відносної вологості та швидкості руху повітря в робочій зоні комп'ютерних приміщень

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Пора року | Категорія робіт | Температура повітря | Відносна вологість | Швидкість руху повітря |
| оптимальна | оптимальна | оптимальна |
| Холодна | легка-1 а | 22 - 24 | 40 - 60 | 0,1 |
|  | легка-1 б | 21 - 23 | 40 - 60 | 0,1 |
| Тепла | легка-1 а | 23 - 25 | 40 - 60 | 0,1 |
|  | легка-1 б | 22 - 24 | 40 - 60 | 0,1 |

Таблиця 3.2 Рівні іонізації повітря приміщення при роботі на ВДТ та ПЕОМ

|  |  |  |
| --- | --- | --- |
| Рівні | Кількість іонів в 1 см куб. повітря | |
| +n | -n |
| Мінімальна необхідність | 400 | 600 |
| Оптимальні | 1500 - 3000 | 3000 - 5000 |
| Максимально допустимі | 50000 | 50000 |

Інтенсивність теплового опромінення працюючих від нагрітих поверхонь технологічного устаткування, освітлювальних приладів, інсоляція від засклених огороджень не повинна перевищувати 35,0 Вт/м2 - при опроміненні 50 % та більше поверхні тіла, 70 Вт/м 2 - при величині опромінюванної поверхні від 25 до 50 %, та 100 Вт/м2 - при опроміненні не більше 25 % поверхні тіла працюючого.

При наявності джерел з інтенсивністю 35,0 Вт/м 2 і більше температура повітря на постійних робочих місцях не повинна перевищувати верхніх меж оптимальних значень для теплого періоду року, на непостійних - верхніх меж допустимих значень для постійних робочих місць [20].

У комп'ютерних приміщеннях, які розташовані в районах з середньою максимальною температурою найбільш жаркого місяця вище 25 грд. С згідно з БНіП "Будівельна кліматологія" допускаються відхилення від величин показників мікроклімату, вказаних в табл. 3.2, для даної категорії робіт, але не більше ніж на 3 грд. С. При цьому швидкість руху повітря повинна бути збільшена на 1,1 м/сек., а відносна вологість повітря знижена на 5 % при підвищенні температури на кожний градус вище верхньої межі допустимих температур повітря.

У комп'ютерних приміщеннях, в яких не можна встановити допустимі величини мікроклімату через технологічні вимоги до виробничого процесу, технічну недосяжність або економічно обґрунтовану недоцільність передбачаються заходи щодо захисту від можливого перегрівання та охолодження.

Основні вимоги до засобів нормалізації мікроклімату та теплозахисту

Для профілактики перегрівань та переохолоджень робітників використовуються засоби індивідуального захисту, медико-біологічні тощо [19].

Формовані параметри мікроклімату на робочих місцях повинні бути досягнені, в першу чергу, за рахунок раціонального планування комп'ютерних приміщень і оптимального розміщення в них устаткування з тепло-, холодо- та вологовиділеннями. Для зменшення термічних навантажень на працюючих передбачається максимальна механізація, автоматизація та дистанційне управління технологічними процесами і устаткуванням.

У приміщеннях із значними площами засклених поверхонь передбачаються заходи щодо захисту від перегрівання при попаданні прямих сонячних променів в теплий період року (орієнтація віконних прорізів схід - захід, улаштування жалюзі та ін.), від радіаційного охолодження - в зимовий (екранування робочих місць). При температурі внутрішніх поверхонь огороджуючи конструкцій, засклення нижче або вище допустимих величин робочі місця повинні бути віддалені від них на відстань не менше 1 м.

У комп'ютерних приміщеннях з надлишком (явного) тепла використовують природну вентиляцію (аерацію). Аераційні ліхтарі та шахти розташовують безпосередньо над основними джерелами тепла на одній осі. У разі неможливості або неефективності аерації встановлюють механічну загальнообмінну вентиляцію.

При наявності одиничних джерел тепловиділень оснащують обладнання місцевою витяжною вентиляцією у вигляді локальних відсмоктувачів, витяжних зонтів та ін.

Найбільш сучасний і ефективний метод це установка систем кондиціонування з клімат контролем , це позволить підтримувати заданні оптимальні або допустими параметри мікроклимату впроовж усієї робочої зміни.

* 1. Вимоги до виробничої меблі для робочих місць з ВДТ

Вимоги до конструкції робочого столу, стільця, підставки для ніг на робочих місцях з вдт визначаються ДСТУ8604:2015 Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги

Конструкція робочого столу має відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні всього обладнання та допоміжних засобів, що використовуються, з урахуванням їх розмірів та конструктивних особливостей. Висота робочої поверхні столу для вдт має бути в межах 680—800 мм, а ширина і глибина — забезпечувати можливість виконання операцій у зоні досяжності моторного поля (рис. 3.4). Рекомендовані розміри столу: висота — 725 мм, ширина — 600—1400 мм, глибина — 800— 1000 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 [20].

Робочий стіл для вдт, як правило, має бути обладнаним підставкою для ніг шириною не менше 300 мм та глибиною не менше 400 мм, з можливістю регулювання по висоті в межах 150 мм та кута нахилу опорної поверхні — в межах 20°. Підставка повинна мати рифлену поверхню та бортик на передньому краї заввишки 10 мм. Застосування підставки для ніг тими, у кого ноги не дістають до підлоги, коли робоче сидіння знаходиться на висоті, потрібній для забезпечення оптимальної робочої пози, є обов'язковим.

Хід ступінчатого регулювання елементів сидіння має становити для лінійних розмірів 15—20 мм, для кутових — 2—5°.

Ширина та глибина сидіння повинні бути не меншими за 400 мм. Висота поверхні сидіння має регулюватися в межах 400—500 мм, а кут нахилу поверхні

Робочий стілець користувача вдт повинен мати такі основні елементи: сидіння, спинку та стаціонарні або знімні підлокітники. 600 400 200 0 200 400 600 мм

Керування: 1 — оптимальна зона моторного поля; 2 — зона легкої дсяжності моторного поля; з — зона досяжності моторного поля.

Від 15° вперед до 5° назад. Поверхня сидіння має бути плоскою, передній край заокругленим.

Конструкція робочого стільця має забезпечувати підтримання раціональної робочої пози під час виконання основних виробничих операцій, створювати умови для зміни пози. Тому стілець повинен бути підйомно- поворотним і регулюватися по висоті та кутах нахилу сидіння і спинки, а також відстані спинки від переднього краю сидіння, висоті підлокітників [19].

Висота спинки сидіння має становити 300±20 мм, ширина — не меньше 380 мм, радіус кривизни в горизонтальній площині — 400 мм. Кут нахилу спинки повинен регулюватися в межах 0—30° відносно вертикального положення. Відстань від спинки до переднього краю сидіння повинна регулюватись у межах 260—400 мм.

Для зниження статичного напруження м'язів рук необхідно застосовувати стаціонарні або знімні підлокітники довжиною не менше 250 мм, шириною — 50— 70 мм, що регулюються по висоті над сидінням у межах 230±30 мм та по відстані між підлокітниками у межах 350—500 мм.

Сидіння, спинка та підлокітники стільця мають бути напівм'якими, з неслизьким, таким, що не електризується та повітропроникним покриттям, матеріал якого забезпечує можливість легкого очищення від забруднення.

Конструкція виробничих меблів для користувача вдт має бути такою, щоб забезпечувати йому підтримання оптимальної робочої пози з такими ергономічними характеристиками: ступні ніг — на підлозі або на підставці для ніг; стегна — в горизонтальній площині; верхні (плечові) частини рук — вертикальні; кут ліктьового суглоба (між плечем та передпліччям) — 70—90°; зап'ястки зігнуті під кутом не більше 20° відносно горизонтальної площини, нахил голови вперед в межах 15—20° до вертикалі.

* 1. Засоби поліпшення функціонального стану спеціалістів ІТ

Підвищення працездатності людини є одним із важливих факторів раціоналізації трудової діяльності.

Ефективність трудової діяльності багато в чому визначається рівнем працездатності людини. Дослідженням цього питання займалося багато вчених.

Під працездатністю розуміють здатність людини виконувати роботу протягом певного часу при збереженні кількісних та якісних показників. Високій працездатності сприяє раціональний режим праці та відпочинку.

Зміни працездатності протягом робочої зміни, доби, тижня тощо, що відображають зв’язок організму людини з умовами і характером трудової діяльності, є вихідною базою для побудови раціональних режимів праці та відпочинку у ці проміжки часу [20].

Фізіолого-гігієнічні питання, зважаючи на їх прямий зв’язок із працездатністю, продуктивністю праці, є важливою соціальною проблемою. Тому боротьба з втомою та перенапруженням має здійснюватися комплексно, включаючи технічні, організаційні, гігієнічні та психофізіологічні заходи. Основними з них є такі:

* широка механізація та автоматизація виробничих операцій;
* організація раціонального режиму праці та відпочинку;
* оптимізація санітарно-гігієнічних умов праці;
* удосконалення робочих рухів та робочої пози;
* обладнання раціонального робочого місця та устаткування з урахуванням антропофізіологічних особливостей організму;
* правильне, раціональне виробниче навчання;
* послаблення несприятливої дії монотонності;
* заходи щодо запобігання гіподинамії;
* формування сумлінного ставлення до праці у трудових колективах (усвідомлення корисності праці, сприятливий психологічний мікроклімат, широка гласність, матеріальна зацікавленість, змагання тощо);
* естетичне оформлення робочого місця; організація та проведення профвідбору та профорієнтації.

Эффективним методом. поліпшення функціонального стану спеціалістів компьютерників є розробка і дотримання регламенту робочого дня

При організації праці, пов'язаної з використанням ВДТ ЕОМ і ПЕОМ, для збереження здоров'я працюючих, запобігання професійним захворюванням і підтримки працездатності передбачаються внутрішньозмінні регламентовані перерви для відпочинку. Внутрішньозмінні режими праці і відпочинку містять додаткові нетривалі перерви в періоди, що передують появі об'єктивних і суб'єктивних ознак стомлення і зниження працездатності. При виконанні робіт, що належать до різних видів трудової діяльності, за основну роботу з ВДТ слід вважати таку, що займає не менше 50% робочого часу [19].

Розробники програм (інженери-програмісти) виконують роботу переважно з відеотерміналом та документацією при необхідності інтенсивного обміну інформацією з ЕОМ і високою частотою прийняття рішень. Робота характеризується інтенсивною розумовою творчою працею з підвищеним напруженням зору, концентрацією уваги на фоні нервово-емоційного напруження, вимушеною робочою позою, загальною гіподинамією, періодичним навантаженням на кисті верхніх кінцівок.

Робота виконується в режимі діалогу з ЕОМ у вільному темпі з періодичним пошуком помилок в умовах дефіциту часу;

Оператори електронно-обчислювальних машин виконують роботу, пов'язану з обліком інформації, одержаної з ВДТ за попереднім запитом, або тієї, що надходить з нього, супроводжується перервами різної тривалості, пов'язана з виконанням іншої роботи і характеризується напруженням зору, невеликими фізичними зусиллями, нервовим напруженням середнього ступеня та виконується у вільному темпі;

Оператор комп'ютерного набору виконує одноманітні за характером роботи з документацією та клавіатурою і нечастими нетривалими переключеннями погляду на екран дисплея, з введенням даних з високою швидкістю.

Робота характеризується як фізична праця з підвищеним навантаженням на кисті верхніх кінцівок на фоні загальної гіподинамії з напруженням зору (фіксація зору переважно на документи), нервово-емоційним напруженням. Правилами встановлюються такі внутрішньозмінні режими праці та відпочинку при роботі з ЕОМ при 8- годинній денній робочій зміні в залежності від характеру праці [20].

Для розробників програм із застосуванням ЕОМ слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за ВДТ; - для операторів із застосуванням ЕОМ слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години; - для операторів комп'ютерного набору слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожної години роботи за ВДТ.

У всіх випадках, коли виробничі обставини не дозволяють застосувати регламентовані перерви, тривалість безперервної роботи з ВДТ не повинна перевищувати 4 години. При 12-годинній робочій зміні регламентовані перерви повинні встановлюватися в перші 8 годин роботи аналогічно перервам при 8-годинній робочій зміні, а протягом останніх 4-х годин роботи, незалежно від характеру трудової діяльності, через кожну годину тривалістю 15 хвилин.

Для зниження нервово-емоційного напруження, втомлення зорового аналізатора, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, запобігання втомі доцільно деякі перерви використовувати для виконання комплексу вправ, які наведені у Державних санітарних правилах і нормах роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98.

Діяльність що пов’язана з використанням ВТД характеризуется високою напруженність зору. Виробнича гімнастика дозволяє компенсувати спеціальних нестачу рухової активності. Виробнича гімнастика має на меті перш за все усунення несприятлиого впливу одноманітного впливу навантаження на одній і ті самій групи м’язів шляхом залучення о роботи раніше бездіяльних груп м’язів.

Вправи шо входят до комплексу виробничої гімнастики, час і методику їх проведення вибирають з урахуванням особливостей праці, зміни функціонального стану організму прцівника протягом робочого дня.

Поліпшення функціонального стану спеціалістів - комп'ютерників також можливо при проведенні сеансів психологчного розвантаження.

При проведенні сеансів психофізіологічного розвантаження рекомендується використовувати деякі елементи методу аутогенного тренування, який ґрунтується на свідомому застосуванні комплексу взаємопов'язаних прийомів психічної саморегуляції й виконанні нескладних фізичних вправ із словесним самонавіюванням. Головна увага при цьому приділяється набуванню й закріпленню навичок м'язового розслаблення (релаксації).

У рекомендованому сеансі, який має проводитися в кімнаті психофізіологічного розвантаження з відповідним інтер'єром та кольоровим оформленням, виділяються три періоди, що відповідають фазам відновлювального процесу. Перший період - абстрагування працівників від виробничої обстановки - відповідає фазі залишкового збудження. Лунають повільна мелодійна музика, пташиний спів [19].

Обравши зручну позу, працівники адаптуються і психологічно готуються до наступних періодів. Другий - заспокоєння - відповідає фазі відновлювального гальмування. Пропонується показ фотослайдів із зображеннями квітучого лугу, березового гаю, гладенької поверхні ставка тощо.

Як функціональне освітлення застосовують зелене світло. Яскравість світла має поступово знижуватись протягом періоду, а наприкінці його світло вимикається зовсім на 1 -2 хвилини. Екран теж гасне. Третій період - активізація - відповідає фазі підвищеної збудженості. На початку періоду світло вимкнене, через певний час на екрані з'являється червона пляма, розміри і яскравість якої поступово збільшуються. Наприкінці періоду лунає бадьора музика. Тричі вимовляються мобілізуючі формули аутогенного тренування, яким мають передувати глибоке вдихання та довге глибоке видихання

Сеанси психологічного розвантаження можуть проводитись за єдиною програмою через індивідуальні навушники і складатись із двох періодів по 5 хвилин кожний: 1) повне розслаблення; 2) активізація працездатності. У разі потреби, на фоні музичних програм можуть вимовлятися окремі фрази навіювання відпочинку, гарного самопочуття і, на заключному етапі, бадьорості. Після сеансів психофізіологічного розвантаження у працівників зменшується відчуття втоми, з'являються бадьорість, гарний настрій. Загальний стан відчутно поліпшується.

Дотримання правил охорони праці які були описані в даному розділі забезпечать комфортне та безпечні умови праці фахівців інформаційних технологій та допоможе уникнути виробничого травматизму та професійних захворювань.

# ВИСНОВКИ

В результаті отриманих знань, і додатково вивчених матеріалів були виконані всі вимоги щодо розробки інформаційної системи з автоматизації робочого місця бібліотекаря. Розроблений додаток дозволяє додавати нові книги до бібілотеки, реєструвати в системі нові примірники цих книг, редагувати інформацію при книги та примірники, а також відаляти її з системи. Також надає можливість видавати примірники книг клієнтам, які зареєстровані в системі. Під час написання дипломної роботи, було виконано:

* аналіз та обґрунтування предметної області;
* порівняння тенденції розвитку інформаційних систем для автоматизації робочого місця бібліотекаря;
* графічне проектування програмного забезпечення засобами мови проектування UML;
* опис бази даних;
* опис функціональної складової системи автоматизації.

Розробку програмного комплексу виконано з використанням мови програмування C# для Microsoft.NET 4.7.\*. В якості системи керування баз данних використано реляційну Microsoft SQL Server 2017 Express, сервер якої працює в Docker.

# ПЕРЕЛІК ПОСИЛАНЬ

<http://profy.nplu.org/articles.php?lng=uk&pg=71>

<https://cyberleninka.ru/article/v/integrirovannye-bibliotechnye-sistemy-v-zhizni-sovremennoy-biblioteki>

<https://cyberleninka.ru/article/v/analiz-pokazateley-deyatelnosti-biblioteki-vuza-sovremennoe-sostoyanie-i-perspektivy>

<https://uk.wikipedia.org/wiki/Microsoft_Windows>

<https://uk.wikipedia.org/wiki/Common_Language_Runtime>

<https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio>

https://ru.wikipedia.org/wiki/Microsoft\_SQL\_Server

<https://support.microsoft.com/ru-ru/help/283878/description-of-the-database-normalization-basics>

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/>

<https://ru.m.wikipedia.org/wiki/Международный_стандартный_книжный_номер>

[StarUML documentation. Introduction. URL: https://docs.staruml.io/](https://docs.staruml.io/)

# ДОДАТКИ

LMSView.UserLogOnForm

public partial class UserLogOnForm : Form

{

private IAuthentication authenticationProcess;

public UserLogOnForm()

{

InitializeComponent();

authenticationProcess = new AuthenticationProcess();

authenticationProcess.OnAdminLogOn += AuthorizeAdmin;

authenticationProcess.OnLibrarianLogOn += AuthorizeLibrarian;

authenticationProcess.OnStockmanLogOn += AuthorizeStockman;

authenticationProcess.OnFailedLogOn += OnFailedLogOn;

}

private void ClearTextFields()

{

textBoxLogin.Clear();

textBoxPassword.Clear();

}

private void UserLogOnForm\_Load(object sender, System.EventArgs e)

{

Text = Properties.Resources.welcome;

buttonlogOn.Text = Properties.Resources.logon;

labelLogin.Text = Properties.Resources.login;

labelLogin.Dock = DockStyle.Fill;

labelLogin.TextAlign = System.Drawing.ContentAlignment.MiddleRight;

labelPassword.Text = Properties.Resources.password;

labelPassword.Dock = DockStyle.Fill;

labelPassword.TextAlign = System.Drawing.ContentAlignment.MiddleRight;

toolTipsLogon.SetToolTip(textBoxLogin, Properties.Resources.tipsLogin);

toolTipsLogon.SetToolTip(textBoxPassword, Properties.Resources.tipsPassword);

toolStripStatusLabelLogOn.Text = Properties.Resources.statusStripLogOn;

}

private void ButtonlogOn\_Click(object sender, System.EventArgs e)

{

authenticationProcess.Authorize(new UserCredential() { Name = textBoxLogin.Text, Password = textBoxPassword.Text });

ClearTextFields();

Activate();

}

public void AuthorizeLibrarian(IWorkspaceLibrarian workspaceLibrarian)

{

Hide();

using (LibrarianControlForm mLibraryManageWin = new LibrarianControlForm(workspaceLibrarian))

{

mLibraryManageWin.ShowDialog();

}

Show();

}

public void AuthorizeAdmin(IWorkspaceAdmin workspaceAdmin)

{

Hide();

if (workspaceAdmin != null)

{

using (UserInfoSearchForm mAdminManageWin = new UserInfoSearchForm(workspaceAdmin.UserInformationRegister))

{

mAdminManageWin.ShowDialog();

}

}

Show();

}

public void AuthorizeStockman(IWorkspaceStockman workspaceStockman)

{

Hide();

using (BookInfoSearchForm mStockmanManageWin = new BookInfoSearchForm(workspaceStockman))

{

mStockmanManageWin.ShowDialog();

}

Show();

}

public void OnFailedLogOn()

{

MessageBox.Show(Properties.Resources.errorLoginEmpty, Properties.Resources.logonFailed,

MessageBoxButtons.OK, MessageBoxIcon.Information, MessageBoxDefaultButton.Button1, MessageBoxOptions.ServiceNotification);

}

}

LMSView.UserInfoSearchForm

public partial class UserInfoSearchForm : Form

{

private readonly IUserInformationRegister userInformationRegister;

public UserInfoSearchForm(IUserInformationRegister userRegister)

{

userInformationRegister = userRegister;

InitializeComponent();

}

private void ButtonSearch\_Click(object sender, EventArgs e)

{

try

{

string search = textBoxSearch.Text;

if (false == String.IsNullOrEmpty(search))

{

UserInformation user = userInformationRegister.FindUser(search);

if (user != null)

{

using (UserInformationControlForm accountForm = new UserInformationControlForm(NViewHelper.FormViewMode.Edit, userInformationRegister, user))

{

accountForm.ShowDialog();

}

}

else

{

DialogResult result = MessageBox.Show(search + Properties.Resources.userSearchFailed, Properties.Resources.failed,

MessageBoxButtons.YesNoCancel, MessageBoxIcon.Error, MessageBoxDefaultButton.Button1, MessageBoxOptions.ServiceNotification);

if(DialogResult.Yes == result)

{

using (UserInformationControlForm accountForm = new UserInformationControlForm(NViewHelper.FormViewMode.Create, userInformationRegister))

{

accountForm.ShowDialog();

}

}

}

Activate();

}

else

{

MessageBox.Show(Properties.Resources.userSearchEmpty, Properties.Resources.failed,

MessageBoxButtons.OK, MessageBoxIcon.Information, MessageBoxDefaultButton.Button1, MessageBoxOptions.ServiceNotification);

}

}

catch (InvalidOperationException)

{

MessageBox.Show(Properties.Resources.userSearchFailed, Properties.Resources.failed,

MessageBoxButtons.OK, MessageBoxIcon.Error, MessageBoxDefaultButton.Button1, MessageBoxOptions.ServiceNotification);

}

}

private void ButtonCreate\_Click(object sender, EventArgs e)

{

using (UserInformationControlForm accountForm = new UserInformationControlForm(NViewHelper.FormViewMode.Create, userInformationRegister))

{

accountForm.ShowDialog();

Activate();

}

}

private void UserInfoSearchForm\_Load(object sender, EventArgs e)

{

Text = Properties.Resources.adminWelcome;

toolStripStatusLabelListControl.Text = Properties.Resources.statusStripAdminMain;

buttonSearch.Text = Properties.Resources.search;

buttonCreate.Text = Properties.Resources.create;

}

}

LMSView.UserInformationControlForm

public partial class UserInformationControlForm : Form

{

private NViewHelper.FormViewMode viewMode;

private readonly IUserInformationRegister userInformationRegister;

private UserInformation currentUser;

public UserInformationControlForm(NViewHelper.FormViewMode mode,

IUserInformationRegister register,

UserInformation findedUser = null)

{

viewMode = mode;

userInformationRegister = register;

currentUser = findedUser;

InitializeComponent();

}

private void PrintForm()

{

Text = Properties.Resources.adminWelcome;

buttonSave.Text = Properties.Resources.save;

buttonDelete.Text = Properties.Resources.delete;

if (comboBoxPosition.Items.Count == 0)

{

comboBoxPosition.Items.AddRange(userInformationRegister.PossiblePositions());

}

labelLogin.Text = Properties.Resources.login;

labelPassword.Text = Properties.Resources.password;

labelBirth.Text = Properties.Resources.birth;

labelName.Text = Properties.Resources.fullName;

labelPhone.Text = Properties.Resources.phone;

labelAddress.Text = Properties.Resources.address;

labelPosition.Text = Properties.Resources.position;

}

private void ApplyCreateViewMode()

{

labelLogin.Text = MarkFieldAsImportant(labelLogin.Text);

labelPassword.Text = MarkFieldAsImportant(labelPassword.Text);

labelBirth.Text = MarkFieldAsImportant(labelBirth.Text);

labelName.Text = MarkFieldAsImportant(labelName.Text);

labelPhone.Text = MarkFieldAsImportant(labelPhone.Text);

labelPosition.Text = MarkFieldAsImportant(labelPosition.Text);

}

private void ApplyEditViewMode()

{

textBoxLogin.Text = currentUser.Credential.Name;

textBoxPassword.Text = currentUser.Credential.Password;

textBoxName.Text = currentUser.PersonalInformation.FullName;

dateTimeBirth.Value = currentUser.PersonalInformation.Birthday;

textBoxPhone.Text = currentUser.PersonalInformation.Phone.ToString(CultureInfo.CurrentCulture);

textBoxAddress.Text = currentUser.PersonalInformation.Address;

comboBoxPosition.Text = currentUser.Position;

}

static private string MarkFieldAsImportant(string field)

{

return Properties.Resources.importantFieldMark + field;

}

private void ButtonDelete\_Click(object sender, EventArgs e)

{

DialogResult result;

if (NViewHelper.FormViewMode.Edit == viewMode)

{

userInformationRegister.DeleteUser(currentUser);

result = MessageBox.Show(Properties.Resources.successfull, Properties.Resources.delete,

MessageBoxButtons.OK, MessageBoxIcon.Information, MessageBoxDefaultButton.Button1, MessageBoxOptions.ServiceNotification);

if (DialogResult.OK == result)

{

Close();

}

}

else

{

result = MessageBox.Show(Properties.Resources.userDeleteFailed, Properties.Resources.delete,

MessageBoxButtons.OK, MessageBoxIcon.Error, MessageBoxDefaultButton.Button1, MessageBoxOptions.ServiceNotification);

}

}

private void ButtonSave\_Click(object sender, EventArgs e)

{

try

{

if (NViewHelper.FormViewMode.Edit == viewMode)

{

userInformationRegister.EditUser(ExtractInputInformation(currentUser));

}

if (NViewHelper.FormViewMode.Create == viewMode)

{

userInformationRegister.AddUser(ExtractInputInformation(new UserInformation()));

}

MessageBox.Show(Properties.Resources.successfull, Properties.Resources.save,

MessageBoxButtons.OK, MessageBoxIcon.Information, MessageBoxDefaultButton.Button1, MessageBoxOptions.ServiceNotification);

Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, Properties.Resources.failed,

MessageBoxButtons.OK, MessageBoxIcon.Error, MessageBoxDefaultButton.Button1, MessageBoxOptions.ServiceNotification);

}

}

private UserInformation ExtractInputInformation(UserInformation userInformation)

{

userInformation.Credential = new UserCredential()

{

Name = textBoxLogin.Text,

Password = textBoxPassword.Text

};

userInformation.Position = comboBoxPosition.SelectedItem.ToString();

userInformation.PersonalInformation = new PersonalInformation

{

FullName = textBoxName.Text,

Birthday = dateTimeBirth.Value,

Phone = Convert.ToDecimal(textBoxPhone.Text, CultureInfo.CurrentCulture),

Address = textBoxAddress.Text

};

return userInformation;

}

private void UserInformationControlForm\_Paint(object sender, PaintEventArgs e)

{

toolStripStatusLabelAccounts.Text = Properties.Resources.statusStripControlAdmin;

PrintForm();

if (NViewHelper.FormViewMode.Create == viewMode)

{

ApplyCreateViewMode();

}

if (NViewHelper.FormViewMode.Edit == viewMode)

{

ApplyEditViewMode();

}

}

}

LMSView. PublishHouseControlForm

public partial class PublishHouseControlForm : Form

{

private NViewHelper.FormViewMode viewModeForm;

private IPublishHouseInformationRegister publishHouseInformationRegister;

private PublishHouseInformation publishHouseInformation;

public PublishHouseControlForm(NViewHelper.FormViewMode mode,

IPublishHouseInformationRegister publishHouseRegister,

PublishHouseInformation publishHouseInfo = null)

{

viewModeForm = mode;

publishHouseInformationRegister = publishHouseRegister;

if (publishHouseInformationRegister != null)

{

publishHouseInformationRegister.OnOperationExecute += PrintMessageForUser;

}

publishHouseInformation = publishHouseInfo;

InitializeComponent();

}

private void PrintMessageForUser(bool result)

{

if (result == true)

{

MessageBox.Show(Properties.Resources.operationOK, Properties.Resources.successfull,

MessageBoxButtons.OK, MessageBoxIcon.Information, MessageBoxDefaultButton.Button1, MessageBoxOptions.ServiceNotification);

Close();

}

else

{

MessageBox.Show(Properties.Resources.operationNOK, Properties.Resources.failed,

MessageBoxButtons.OK, MessageBoxIcon.Information, MessageBoxDefaultButton.Button1, MessageBoxOptions.ServiceNotification);

}

}

private PublishHouseInformation PackagePublishHouse(PublishHouseInformation targetPublishHouse)

{

targetPublishHouse.Name = textBoxName.Text;

targetPublishHouse.Location = textBoxLocation.Text;

targetPublishHouse.Category = textBoxCategory.Text;

return targetPublishHouse;

}

private void ButtonDelete\_Click(object sender, EventArgs e)

{

if (NViewHelper.FormViewMode.Edit == viewModeForm)

{

publishHouseInformationRegister.Delete(publishHouseInformation);

}

}

private void ButtonSave\_Click(object sender, EventArgs e)

{

publishHouseInformationRegister.Add(PackagePublishHouse(new PublishHouseInformation()));

}

private void PublishHouseControlPanel\_Paint(object sender, PaintEventArgs e)

{

buttonSave.Text = Properties.Resources.save;

buttonDelete.Text = Properties.Resources.delete;

labelName.Text = Properties.Resources.publishHouseName;

labelLocation.Text = Properties.Resources.location;

labelCategory.Text = Properties.Resources.category;

if (NViewHelper.FormViewMode.Create == viewModeForm)

{

Text = Properties.Resources.add;

labelName.Text = NViewHelper.ViewHelper.MarkFieldAsImportant(labelName.Text);

}

else

{

Text = Properties.Resources.edit;

}

}

}

LMSView. NViewHelper

public enum FormViewMode

{

Edit,

Create

}

public enum AccountMode

{

Admin,

Library,

Stockman

}

public static class ViewHelper

{

static public string MarkFieldAsImportant(string field)

{

return Properties.Resources.importantFieldMark + field;

}

}

LMSView. InventoryCodeControlForm

public partial class InventoryCodeControlForm : Form

{

private NViewHelper.FormViewMode viewMode;

private int inventoryCode;

public event SetInventoryCode AddNewBookExample;

public event SetInventoryCode EditNewBookExample;

public InventoryCodeControlForm(int selectedCode) : this(NViewHelper.FormViewMode.Edit, selectedCode)

{ }

public InventoryCodeControlForm() : this(NViewHelper.FormViewMode.Create, 0)

{ }

private InventoryCodeControlForm(NViewHelper.FormViewMode view, int code)

{

viewMode = view;

inventoryCode = code;

InitializeComponent();

}

private void InventoryCodeControlForm\_Load(object sender, EventArgs e)

{

Text = Properties.Resources.inventoryCodeWelcome;

buttonCancel.Text = DialogResult.Cancel.ToString();

buttonSave.Text = Properties.Resources.save;

if (NViewHelper.FormViewMode.Edit == viewMode)

{

textBoxInventoryCode.Text = inventoryCode.ToString(CultureInfo.CurrentCulture);

}

}

private void ButtonSave\_Click(object sender, EventArgs e)

{

if (false == String.IsNullOrEmpty(textBoxInventoryCode.Text))

{

int newCode = Convert.ToInt32(textBoxInventoryCode.Text, CultureInfo.CurrentCulture);

if (NViewHelper.FormViewMode.Edit == viewMode)

{

EditNewBookExample(inventoryCode);

}

AddNewBookExample(newCode);

Close();

}

}

private void ButtonCancel\_Click(object sender, EventArgs e)

{

Close();

}

}

LMSView. ClientPersonalInfoControlForm

public partial class ClientPersonalInfoControlForm : Form

{

private NViewHelper.FormViewMode mViewMode;

private readonly IClientInformationRegister clientInformationRegister;

private ClientInformation currentClient;

public ClientPersonalInfoControlForm(NViewHelper.FormViewMode mode,

IClientInformationRegister register,

ClientInformation client = null)

{

mViewMode = mode;

clientInformationRegister = register;

currentClient = client;

InitializeComponent();

}

private void ReaderControlPanel\_Paint(object sender, PaintEventArgs e)

{

labelLibTicketNum.Text = Properties.Resources.libraryNumTicket;

labelName.Text = Properties.Resources.fullName;

labelBirth.Text = Properties.Resources.birth;

labelPhone.Text = Properties.Resources.phone;

labelAddress.Text = Properties.Resources.address;

buttonSave.Text = Properties.Resources.save;

buttonDelete.Text = Properties.Resources.delete;

if (NViewHelper.FormViewMode.Edit == mViewMode)

{

Text = Properties.Resources.clientEdit;

}

if (NViewHelper.FormViewMode.Create == mViewMode)

{

Text = Properties.Resources.clientCreate;

labelLibTicketNum.Text = NViewHelper.ViewHelper.MarkFieldAsImportant(labelLibTicketNum.Text);

labelName.Text = NViewHelper.ViewHelper.MarkFieldAsImportant(labelName.Text);

labelBirth.Text = NViewHelper.ViewHelper.MarkFieldAsImportant(labelBirth.Text);

labelPhone.Text = NViewHelper.ViewHelper.MarkFieldAsImportant(labelPhone.Text);

}

}

private void ButtonSave\_Click(object sender, EventArgs e)

{

if (NViewHelper.FormViewMode.Edit == mViewMode)

{

clientInformationRegister.EditClient(PackageNewClientInfo(currentClient));

}

if (NViewHelper.FormViewMode.Create == mViewMode)

{

clientInformationRegister.AddClient(PackageNewClientInfo(new ClientInformation()));

}

}

private ClientInformation PackageNewClientInfo(ClientInformation client)

{

client.LibraryTicketNumberCode = Convert.ToInt32(textBoxLibTicketNum.Text, CultureInfo.CurrentCulture);

PersonalInformation personInfo = new PersonalInformation()

{

FullName = textBoxName.Text,

Birthday = dateTimePickerBirth.Value,

Phone = Convert.ToDecimal(textBoxPhone.Text, CultureInfo.CurrentCulture)

};

client.PersonalInformation = personInfo;

return client;

}

}

LMSView. ClientLibraryInfoControlForm

public partial class ClientLibraryInfoControlForm : Form

{

private readonly ILibraryCardRegister libraryCardRegister;

private readonly IClientInformationRegister clientInformationRegister;

private readonly IBookFinding bookFinding;

private ClientInformation currentClient;

public ClientLibraryInfoControlForm(ILibraryCardRegister libraryRegister,

IClientInformationRegister clientRegister,

IBookFinding finding,

ClientInformation client)

{

libraryCardRegister = libraryRegister;

clientInformationRegister = clientRegister;

bookFinding = finding;

currentClient = client;

InitializeComponent();

}

private void ClientLibraryInfoControlForm\_Load(object sender, EventArgs e)

{

labelClientTicketNum.Text = Properties.Resources.libraryNumTicket;

textBoxTicketNum.Text = currentClient.LibraryTicketNumberCode.ToString(CultureInfo.CurrentCulture);

labelClientName.Text = Properties.Resources.bookNum;

textBoxClientName.Text = currentClient.PersonalInformation.FullName;

labelBooksList.Text = Properties.Resources.bookList;

buttonClientPersonal.Text = Properties.Resources.personalInfo;

buttonReturnBook.Text = Properties.Resources.returnBook;

dateTimePickerReturn.Visible = false;

dateTimePickerGiveBook.Visible = false;

ICollection<LibraryCard> libraryCards = libraryCardRegister.GetLibraryCardsByTicketNum(currentClient.LibraryTicketNumberCode);

foreach (var card in libraryCards)

{

BookInformation book = bookFinding.FindBookByInventoryCode(card.CodeRentedBook);

listBoxClientBookCollection.Items.Add(book.Title);

}

listBoxClientBookCollection.SelectedIndex = 0;

}

private void ButtonReturnBook\_Click(object sender, EventArgs e)

{

if (listBoxClientBookCollection.SelectedItem != null)

{

BookInformation selectedBook = (BookInformation)listBoxClientBookCollection.SelectedItem;

clientInformationRegister.ReturnBook(currentClient, selectedBook);

}

else

{

MessageBox.Show(Properties.Resources.uncheckedBook, Properties.Resources.failed,

MessageBoxButtons.OK, MessageBoxIcon.Information, MessageBoxDefaultButton.Button1, MessageBoxOptions.ServiceNotification);

}

}

private void ButtonClientPersonal\_Click(object sender, EventArgs e)

{

using (ClientPersonalInfoControlForm info = new ClientPersonalInfoControlForm(NViewHelper.FormViewMode.Edit,

clientInformationRegister, currentClient))

{

info.ShowDialog();

Activate();

}

}

private void ListBoxClientBookCollection\_SelectedValueChanged(object sender, EventArgs e)

{

string title = listBoxClientBookCollection.SelectedValue as string;

BookInformation book = bookFinding.FindBookByTitle(title);

LibraryCard currentCard = libraryCardRegister.GetLibraryCardByBookTitle(book.Title);

dateTimePickerGiveBook.Visible = true;

dateTimePickerGiveBook.Value = currentCard.DateRentBook;

}

}

LMSView. BookInfoSearchForm

public partial class BookInfoSearchForm : Form

{

private IWorkspaceStockman WorkspaceStockman;

public BookInfoSearchForm(IWorkspaceStockman workspaceStockman)

{

WorkspaceStockman = workspaceStockman;

InitializeComponent();

}

private void BookInfoSearchForm\_Load(object sender, EventArgs e)

{

Text = Properties.Resources.stockmanWelcome;

toolStripStatusLabelBook.Text = Properties.Resources.statusStripStockmanMain;

buttonSearch.Text = Properties.Resources.search;

buttonCreate.Text = Properties.Resources.create;

}

private void ButtonSearch\_Click(object sender, EventArgs e)

{

BookInformation book = WorkspaceStockman.BookFinding.FindBookByTitle(textBoxSearch.Text);

if(book != null)

{

using (BookInfoControlForm bookForm = new BookInfoControlForm(NViewHelper.FormViewMode.Edit,

WorkspaceStockman.LibraryOperations, book))

{

Hide();

bookForm.ShowDialog();

Show();

}

}

}

private void ButtonCreate\_Click(object sender, EventArgs e)

{

using (BookInfoControlForm bookForm = new BookInfoControlForm(NViewHelper.FormViewMode.Create,

WorkspaceStockman.LibraryOperations))

{

Hide();

bookForm.ShowDialog();

Show();

}

}

}

LMSView. BookInfoControlForm

partial class BookInfoControlForm : Form

{

private FormViewMode viewMode;

private BookInformation currentBook;

private ILibraryOperations libraryManagment;

PublishHouseInformationRegister publishHouseRegister = new PublishHouseInformationRegister();

SubjectInformationFinding subjectFinding = new SubjectInformationFinding();

public BookInfoControlForm(FormViewMode mode, ILibraryOperations libraryOperations, BookInformation book = null)

{

viewMode = mode;

currentBook = book;

libraryManagment = libraryOperations;

InitializeComponent();

}

private void ButtonSave\_Click(object sender, EventArgs e)

{

if (FormViewMode.Create == viewMode)

{

currentBook = new BookInformation();

}

libraryManagment.AddBook(PackageBookInformation(currentBook));

}

private void ButtonDelete\_Click(object sender, EventArgs e)

{

if (FormViewMode.Edit == viewMode)

{

libraryManagment.DeleteBookByISBN(currentBook.ISBN);

}

}

private void BookControlPanel\_Paint(object sender, PaintEventArgs e)

{

PrintForm();

if (FormViewMode.Edit == viewMode)

{

Text = Properties.Resources.editBook;

foreach (var code in currentBook.InventoryCode)

{

if (code.Value == false)

{

listBoxInventoryCodeAvailable.Items.Add(code.Key);

}

else

{

listBoxInventoryCodeNotAvailable.Items.Add(code.Key);

}

}

textBoxTitle.Text = currentBook.Title;

AuthorInformation[] authors = currentBook.Authors;

foreach (var name in authors)

{

textBoxAuthors.Text += Properties.Resources.separator + name.Name;

}

textBoxFullTitle.Text = currentBook.FullTitle;

textBoxIsbn.Text = currentBook.ISBN.ToString(CultureInfo.CurrentCulture);

textBoxLanguage.Text = currentBook.Language;

dateTimePickerPublish.Value = currentBook.PublishDate;

textBoxPublishHouseResult.Text = currentBook.PublishHouse.Name;

textBoxSeries.Text = currentBook.BookSeries;

textBoxSubject.Text = currentBook.Subject.Name;

if (currentBook.Subject.SubjectParent != null)

{

textBoxSubjectParent.Text = currentBook.Subject.SubjectParent.Name;

}

textBoxAnnotation.Text = currentBook.Annotation;

}

if (FormViewMode.Create == viewMode)

{

Text = Properties.Resources.createBook;

labelTitle.Text = ViewHelper.MarkFieldAsImportant(labelTitle.Text);

labelInventoryCodeNotAvailable.Text = ViewHelper.MarkFieldAsImportant(labelInventoryCodeNotAvailable.Text);

labelAuthors.Text = ViewHelper.MarkFieldAsImportant(labelAuthors.Text);

labelIsbn.Text = ViewHelper.MarkFieldAsImportant(labelIsbn.Text);

labelLanguage.Text = ViewHelper.MarkFieldAsImportant(labelLanguage.Text);

labelPublishDate.Text = ViewHelper.MarkFieldAsImportant(labelPublishDate.Text);

labelPublishHouse.Text = ViewHelper.MarkFieldAsImportant(labelPublishHouse.Text);

}

}

private BookInformation PackageBookInformation(BookInformation book)

{

Dictionary<int, bool> mapInventoryCodes = new Dictionary<int, bool>();

foreach (var code in listBoxInventoryCodeAvailable.Items)

{

mapInventoryCodes.Add(Convert.ToInt32(code, CultureInfo.CurrentCulture), false);

};

foreach (var code in listBoxInventoryCodeNotAvailable.Items)

{

mapInventoryCodes.Add(Convert.ToInt32(code, CultureInfo.CurrentCulture), true);

};

book.InventoryCode = mapInventoryCodes;

book.Title = textBoxTitle.Text;

book.FullTitle = textBoxFullTitle.Text;

book.ISBN = Convert.ToInt32(textBoxIsbn.Text, CultureInfo.CurrentCulture);

book.Language = textBoxLanguage.Text;

book.PublishHouse = publishHouseRegister.FindByName(textBoxPublishHouseResult.Text);

book.PublishDate = dateTimePickerPublish.Value;

book.Annotation = textBoxAnnotation.Text;

book.BookSeries = textBoxSeries.Text;

string[] authorsNames = textBoxAuthors.Text.Split(new char[] { ',', ' ' });

List<AuthorInformation> authors = new List<AuthorInformation>();

foreach (string name in authorsNames)

{

authors.Add(new AuthorInformation() { Name = name });

}

book.Authors = authors.ToArray();

book.Subject = new SubjectInformation() { Name = textBoxSubject.Text };

return book;

}

private void PrintForm()

{

listBoxInventoryCodeNotAvailable.Enabled = false;

buttonSave.Text = Properties.Resources.save;

buttonDelete.Text = Properties.Resources.delete;

buttonSearchPublishHouse.Text = Properties.Resources.search;

buttonAddInventoryCode.Text = Properties.Resources.add;

buttonDeleteInventoryCode.Text = Properties.Resources.delete;

buttonEditInventoryCode.Text = Properties.Resources.edit;

labelInventoryCodeAvailable.Text = Properties.Resources.inventoryCodeAvailable;

labelInventoryCodeNotAvailable.Text = Properties.Resources.inventoryCodeNotAvailable;

labelTitle.Text = Properties.Resources.title;

labelFullTitle.Text = Properties.Resources.fullTitle;

labelAuthors.Text = Properties.Resources.author;

labelIsbn.Text = Properties.Resources.isbn;

labelLanguage.Text = Properties.Resources.language;

labelPublishDate.Text = Properties.Resources.publishDate;

labelSeries.Text = Properties.Resources.series;

labelPublishHouse.Text = Properties.Resources.publishHouse;

labelSubject.Text = Properties.Resources.subject;

labelParentSubject.Text = Properties.Resources.parentSubject;

labelAnnotation.Text = Properties.Resources.annotation;

}

private void ButtonSearchPublishHouse\_Click(object sender, EventArgs e)

{

PublishHouseInformation publishHouse = publishHouseRegister.FindByName(textBoxPublishHouseSearch.Text);

if (null == publishHouse)

{

DialogResult userChoice = MessageBox.Show(Properties.Resources.searchPublishHouseFailed, Properties.Resources.failed,

MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button1, MessageBoxOptions.ServiceNotification);

if (DialogResult.Yes == userChoice)

{

using (PublishHouseControlForm publishControl = new PublishHouseControlForm(FormViewMode.Create, publishHouseRegister))

{

publishControl.ShowDialog();

}

}

Activate();

}

else

{

textBoxPublishHouseResult.Text = publishHouse.Name;

}

}

private void RemoveCodeFromList(int lastCode)

{

listBoxInventoryCodeAvailable.Items.Remove(lastCode);

libraryManagment.DeleteBookByInventoryCode(lastCode);

}

private void AddNewCodeToList(int code)

{

listBoxInventoryCodeAvailable.Items.Add(code);

}

private void ButtonDeleteInventoryCode\_Click(object sender, EventArgs e)

{

if (listBoxInventoryCodeNotAvailable.SelectedItem != null)

{

RemoveCodeFromList(Convert.ToInt32(listBoxInventoryCodeNotAvailable.SelectedItem, CultureInfo.CurrentCulture));

}

}

private void ButtonEditInventoryCode\_Click(object sender, EventArgs e)

{

if (listBoxInventoryCodeNotAvailable.SelectedItem != null)

{

int currentCode = Convert.ToInt32(listBoxInventoryCodeNotAvailable.SelectedItem, CultureInfo.CurrentCulture);

using (InventoryCodeControlForm inventoryCodeControl = new InventoryCodeControlForm(currentCode))

{

inventoryCodeControl.AddNewBookExample += AddNewCodeToList;

inventoryCodeControl.EditNewBookExample += RemoveCodeFromList;

inventoryCodeControl.ShowDialog();

Activate();

}

}

}

private void ButtonAddInventoryCode\_Click(object sender, EventArgs e)

{

using (InventoryCodeControlForm inventoryCodeControl = new InventoryCodeControlForm())

{

inventoryCodeControl.AddNewBookExample += AddNewCodeToList;

inventoryCodeControl.ShowDialog();

Activate();

}

}

private void ListBoxInventoryCodeNotAvailable\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

LMSController:

internal class AdminWorkspace : IWorkspaceAdmin

{

private static readonly IUserInformationRegister userInformationRegister = new UserInformationRegister();

public IUserInformationRegister UserInformationRegister => userInformationRegister;

}

}

public class AuthenticationProcess : IAuthentication

{

public event LibrarianLogOnEventHandler OnLibrarianLogOn;

public event AdminLogOnEventHandler OnAdminLogOn;

public event StockmanLogOnEventHandler OnStockmanLogOn;

public event FailedLogOnEventHandler OnFailedLogOn;

public void Authorize(UserCredential userCredential)

{

if (userCredential == null)

{

throw new ArgumentNullException(nameof(userCredential));

}

Account account = Accounts.FindByUserNameAndPassword(userCredential.Name, userCredential.Password);

if (null != account)

{

switch (account.Position.position\_enum)

{

case UserPosition.Librarian:

OnLibrarianLogOn(new LibrarianWorkspace());

break;

case UserPosition.Administrator:

OnAdminLogOn(new AdminWorkspace());

break;

case UserPosition.Stockman:

OnStockmanLogOn(new StockmanWorkspace());

break;

default:

throw new InvalidOperationException(account.Position.position\_enum.ToString());

}

}

else

{

OnFailedLogOn();

}

}

}

public class AuthorInformation

{

public string Name { get; set; }

}

internal class BookFinding : IBookFinding

{

private static BookInformation[] Convert(Book[] books)

{

if (books == null)

{

return null;

}

BookInformation[] bookInformations = new BookInformation[books.Length];

for (int i = 0; i < books.Length; ++i)

{

bookInformations[i] = Convert(books[i]);

}

return bookInformations;

}

private static BookInformation Convert(Book book)

{

BookInformation bookInformation = null;

if (book != null)

{

bookInformation = new BookInformation()

{

BookSeries = book.book\_series,

Annotation = book.annotation,

FullTitle = book.full\_title,

ISBN = book.id\_book\_isbn,

Language = book.language,

PublishDate = book.date\_publish,

PublishHouse = new PublishHouseInformation()

{

Category = book.PublishingHouse.category,

Location = book.PublishingHouse.location,

Name = book.PublishingHouse.name

},

Subject = new SubjectInformation()

{

Name = book.Subject.name,

},

Title = book.title,

InventoryCode = Books.FindInventoryCodesByISBN(book.id\_book\_isbn),

Authors = Convert(Books.FindAutorsByISBN(book.id\_book\_isbn))

};

}

return bookInformation;

}

private static AuthorInformation[] Convert(string[] names)

{

AuthorInformation[] authorInformations = new AuthorInformation[names.Length];

for (int i = 0; i < authorInformations.Length; ++i)

{

authorInformations[i] = new AuthorInformation() { Name = names[i] };

}

return authorInformations;

}

public BookInformation FindBookByInventoryCode(int inventoryCode) => Convert(Books.FindByInventoryCode(inventoryCode));

public BookInformation FindBookByISBN(int isbn) => Convert(Books.FindByISBN(isbn));

public BookInformation FindBookByTitle(string title) => Convert(Books.FindByTitle(title));

BookInformation[] IBookFinding.FindBookByBookSeries(string bookSeries) => Convert(Books.FindByBookSeries(bookSeries));

BookInformation[] IBookFinding.FindBookByLanguage(string language) => Convert(Books.FindByLanguage(language));

}

public class BookInformation

{

public IDictionary<int, bool> InventoryCode { get; set; }

public string Title { get; set; }

public string FullTitle { get; set; }

public int ISBN { get; set; }

public AuthorInformation[] Authors { get; set; }

public string Annotation { get; set; }

public PublishHouseInformation PublishHouse { get; set; }

public DateTime PublishDate { get; set; }

public string Language { get; set; }

public string BookSeries { get; set; }

public SubjectInformation Subject { get; set; }

public BookInformation() { }

public BookInformation(IDictionary<int, bool> inventoryCode, string title, string fullTitle, int isbn,

AuthorInformation[] authors, string annotation, PublishHouseInformation publishHouse,

DateTime publishDate, string language, string bookSeries, SubjectInformation subject)

{

InventoryCode = inventoryCode;

Title = title;

FullTitle = fullTitle;

ISBN = isbn;

Authors = authors;

Annotation = annotation;

PublishHouse = publishHouse;

PublishDate = publishDate;

Language = language;

BookSeries = bookSeries;

Subject = subject;

}

}

public class ClientInformation

{

public int LibraryTicketNumberCode { get; set; }

public PersonalInformation PersonalInformation { get; set; }

}

public class ClientInformationRegister : IClientInformationRegister

{

public void AddClient(ClientInformation clientInformation)

{

throw new NotImplementedException();

}

public void DeleteClient(ClientInformation clientInformation)

{

throw new NotImplementedException();

}

public void EditClient(ClientInformation clientInformation)

{

throw new NotImplementedException();

}

public ClientInformation FindClientByLibraryTicket(int numberLibraryTicket)

{

throw new NotImplementedException();

}

public ClientInformation FindClientByName(string name)

{

throw new NotImplementedException();

}

public void RentBook(ClientInformation client, BookInformation book)

{

throw new NotImplementedException();

}

public void ReturnBook(ClientInformation client, BookInformation book)

{

throw new NotImplementedException();

}

}

public delegate void LibrarianLogOnEventHandler(IWorkspaceLibrarian workspaceLibrarian);

public delegate void AdminLogOnEventHandler(IWorkspaceAdmin workspaceLibrarian);

public delegate void StockmanLogOnEventHandler(IWorkspaceStockman workspaceLibrarian);

public delegate void FailedLogOnEventHandler();

public interface IAuthentication

{

void Authorize(UserCredential userCredential);

event LibrarianLogOnEventHandler OnLibrarianLogOn;

event AdminLogOnEventHandler OnAdminLogOn;

event StockmanLogOnEventHandler OnStockmanLogOn;

event FailedLogOnEventHandler OnFailedLogOn;

}

public interface IBookFinding

{

BookInformation FindBookByInventoryCode(int inventoryCode);

BookInformation FindBookByTitle(string title);

BookInformation FindBookByISBN(int isbn);

BookInformation[] FindBookByLanguage(string language);

BookInformation[] FindBookByBookSeries(string bookSeries);

}

public interface IClientInformationRegister

{

ClientInformation FindClientByLibraryTicket(int numberLibraryTicket);

ClientInformation FindClientByName(string name);

void AddClient(ClientInformation clientInformation);

void EditClient(ClientInformation clientInformation);

void DeleteClient(ClientInformation clientInformation);

void RentBook(ClientInformation client, BookInformation book);

void ReturnBook(ClientInformation client, BookInformation book);

}

public interface ILibraryOperations

{

void AddBook(BookInformation bookInformation);

void EditBook(BookInformation bookInformation);

void DeleteBookByISBN(int isbn);

void DeleteBookByInventoryCode(int inventoryCode);

}

public interface ILibraryCardRegister

{

ICollection<LibraryCard> GetLibraryCardsByTicketNum(int clientTicketNum);

LibraryCard GetLibraryCardByInventoryCode(int inventoryCode);

LibraryCard GetLibraryCardByBookTitle(string title);

}

public interface IPublishHouseInformationRegister

{

event OperationEventHandler OnOperationExecute;

PublishHouseInformation FindByName(string name);

void Add(PublishHouseInformation publishHouse);

void Delete(PublishHouseInformation publishHouse);

}

public interface ISubjectInformationFinding

{

SubjectInformation FindByName(string name);

}

public interface IUserInformationRegister

{

UserInformation FindUser(string userName);

string[] PossiblePositions();

void AddUser(UserInformation userInformation);

void EditUser(UserInformation userInformation);

void DeleteUser(UserInformation userInformation);

}

public interface IWorkspaceAdmin

{

IUserInformationRegister UserInformationRegister { get; }

}

public interface IWorkspaceLibrarian

{

IBookFinding BookFinding { get; }

IClientInformationRegister ClientInformationRegister { get; }

ILibraryCardRegister LibraryCardRegister { get; }

}

public interface IWorkspaceStockman

{

IBookFinding BookFinding { get; }

ILibraryOperations LibraryOperations { get; }

}

public class LibrarianWorkspace : IWorkspaceLibrarian

{

private static readonly IBookFinding bookFinding = new BookFinding();

public IBookFinding BookFinding => bookFinding;

public IClientInformationRegister ClientInformationRegister => throw new NotImplementedException();

public ILibraryCardRegister LibraryCardRegister => throw new NotImplementedException();

}

public class LibraryCard

{

public ClientInformation Client { get; set; }

public int CodeRentedBook { get; set; }

public BookInformation BookInformation { get; set; }

public DateTime DateRentBook { get; set; }

public DateTime DateReturnBook { get; set; }

}