Технически университет – Варна, 2020г.

Факултет ФИТА

Катедра СИТ

Специалност СИТ

Информационна система за библиотека

Проект по ООП 2

Документация

Мариян Желязков, ф.н. 18621641

Яна Илиева, ф.н. 18621762

Contents

[Задание на проекта 3](#_Toc58360691)

[Анализ на проблема 4](#_Toc58360692)

[Функционални изисквания 4](#_Toc58360694)

[Структура на проекта 5](#_Toc58360695)

[Дефиниция на модулите в системата 5](#_Toc58360697)

[Обзор на използваните технологии 6](#_Toc58360698)

[Проектиране на системата 7](#_Toc58360699)

[Проектиране на база данни 7](#_Toc58360700)

[Проектиране на GUI 7](#_Toc58360701)

[Проектиране на бизнес логика 7](#_Toc58360702)

[Проектиране на слой за достъп до база данни 7](#_Toc58360703)

[UML диаграми 7](#_Toc58360704)

[Activity диаграма за добавяне на нова книга 9](#_Toc58360705)

[Концептуален модел на базата данни 14](#_Toc58360706)

[Реализация на системата 20](#_Toc58360707)

[Реализация на базата данни 20](#_Toc58360708)

[Реализация на слоя за работа с база данни 21](#_Toc58360709)

[Реализация на бизнес логика и графичен интерфейс 22](#_Toc58360710)

[Реализация на модул за регистриране на събития в системата 25](#_Toc58360711)

[Конфигурация 25](#_Toc58360712)

[Сигурност 26](#_Toc58360713)

[Валидация 26](#_Toc58360714)

[Тестови резултати 27](#_Toc58360715)

[Screenshots 28](#_Toc58360716)

# Задание на проекта

Да се разработи информационна система – Библиотека. Програмата съхранява и обработва данни за книги и читатели. Системата позволява множествен достъп.

Системата поддържа два вида потребители – администратор и клиенти (оператори и читатели) с различни роли за достъп до функционалностите в системата.

Операции за работа с потребители:

* Създаване на оператори от администратор
* Създаване и отписване на читатели от оператор
* Формуляр за създаване на читателски профил
* Отдаване на книги

Системата поддържа операции за работи с книги:

* Добавяне на нови книги (Инвентарен номер, заглавие, автор, жанр…)
* Заемане на книги (различно ниво на сигурност при заемането на книги (читалня, изнасяне))
* Връщане на книги
* Бракуване на повредени книги
* Архивиране на стари издания (ползвани само в читалня)

Системата поддържа справки за:

* Подадени формуляри(дата, статус, съдържание на формуляра)
* Книги (Състояние на книгата, информация за книгата)
* Потребители (дата на одобрение, списък с книги, потребителска информация)
* Рейтинг на потребителите (лоялни и нелоялни)

Системата поддържа известия за събития:

* Заявка за откриване на читателски профил
* Известяване на необходимост за архивиране на книга
* Просрочване връщането на книги

# Анализ на проблема

# Трябва да се създаде информационна система за библиотека. Програмата трябва да обработва и съхранява данни за книги и за читатели. Също така, тя трябва да позволява множествен достъп през различни потребителски роли – администратор, оператор(библиотекар) и читател.

## Функционални изисквания

Системата трябва да поддържа следните общи операции за администратор и оператор:

* Създаване на читателски профил
* Деактивиране на читателски профил
* Отдаване на книга
  + За дома
  + В читалня
* Добавяне на нова книга
* Бракуване на книга
* Архивиране на книга
* Маркиране на книга като върната

Администраторът трябва да има допълнително възможността да:

* Създава профил на оператор

Операторът трябва да има допълнително възможността да:

* Получава известия за:
  + Нови формуляри за създаване на читателски профил
  + Книги, чието връщане е просрочено
  + Книги, които трябва да се архивират

Читателят трябва да може да:

* Се логне системата
* Да вижда книгите, които е заел към текущия момент
* Да вижда личните си данни, съхранени в базата данни

Анонимната роля трябва да може да:

* Изпраща формуляр-заявление за създаване на читателски профил

## Структура на проекта

# Системата е построена на принципа на многослойната архитектура. Отличават се 4 отделни слоя - Презентационен, Бизнес, Пърсистънс и База данни. Използва се подходът с dependency injection за осигуряване на независимост на компонентите.

## Дефиниция на модулите в системата

Системата съдържа следните модули:

* GUI – графичен интерфейс (JavaFX)
* Контролер за GUI – методи, „слушащи“ за действия от графичния интерфейс и предаващи действието на бизнес слоя.
* Бизнес логика – Отговаря за проверките, изчисленията и програмната логика като цяло.
* Слой за достъп до база данни (Hibernate) – Слой, предоставящ абстракция при работата с базата данни, за да се постигне независимост и необвързаност между приложението и лежащата под него конкретна база за данни.
* База данни (Oracle) – съхраняване на данни
* Сигурност – хеширане на пароли
* Валидация – автентикация и оторизация

# Обзор на използваните технологии

В проекта се използват следните технологии:

* Maven – средство за автоматизация на изграждане на проекти. Предлага лесно включване на всички външни библиотеки, които са необходими на един проект и се грижи за тяхната актуалност, ако не е зададена изрично желана версия на дадената зависимост.
* Hibernate – ORM (Object Relational Mapping) инструмент, чието предназначение е да осигури абстрактен слой между приложението и базата данни, така че то да е напълно независимо от нея. Hibernate се явява и улеснение на програмиста, който има възможността чрез Java синтаксис да прави заявки към базата, без да е директно запознат със специфичните езикови правила на съответния SQL.
* Oracle XE 18c – База данни на Oracle Corporation. Тя съхранява данните, който са генерирани от приложението или са преминали през него, дори когато то е изключено.
* JavaFX – GUI библиотека за създаване на десктоп, интернет и мобилни приложения на Java.
* Scene Builder – визуално drag-and-drop средство за създаване на изгледи на графичния интерфейс, без писане на код на Java. То генерира xml файлове с .fxml разширение.
* Weld – референтна имплементация на JSR-299 API (CDI – Context and Dependency Injection) за Java SE. Представлява контейнер за dependency injection, което позволява класовете да не се интересуват от своите зависимости, а да ги получават наготово. Този метод терминира директната зависимост между обектите, постига се low coupling. Той позволява също така лесно заменяне на имплементацията на дадена зависимост.

# Проектиране на системата

За проектиране на системата е използвана n-tier architecture – n-слойна архитектура, като всеки слой комуникира само и единствено със слоя над и под него. Тази архитектура има следните предимства:

* Сигурност – всеки слой може да се подсигури отделно, по различни методи
* Лесна за управление – всеки слой може да се управлява, модифицира, подменя независимо от останалите, защото другите слоеве нямат знание за вътрешната му структура.
* Мащабируемост – може да се добавят допълнителни ресурси, без да има ефект върху останалите слоеве.
* Гъвкавост – един слой може да се разширява според нови изисквания за системата.

## Проектиране на база данни

При проектирането е използван подхода database-first – entity-тата се създават по образец на базата данни.

## Проектиране на GUI

За графичния интерфейс е избран прост дизайн, който не набляга на естетичността, а по-скоро на показване на функционалностите на системата.

## Проектиране на бизнес логика

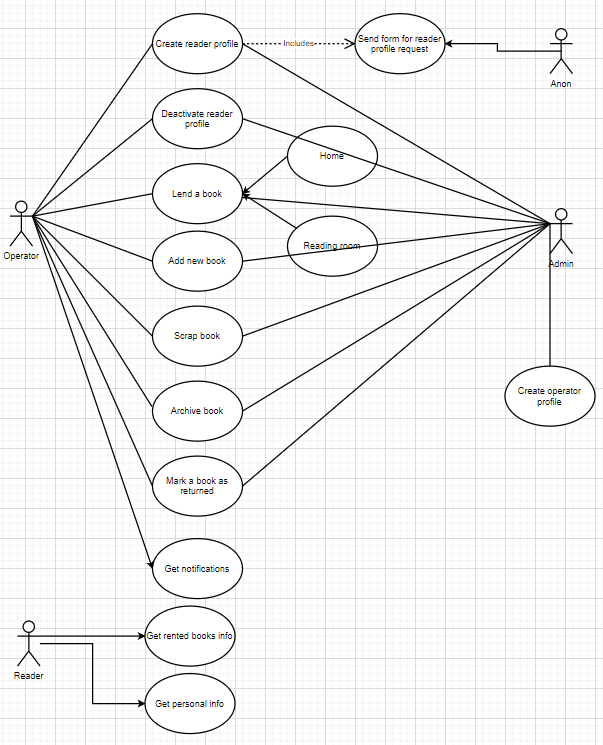
Този слой е изграден подобно слоя за достъп до база данни, тъй като има директно съответствие между методите им.

## Проектиране на слой за достъп до база данни

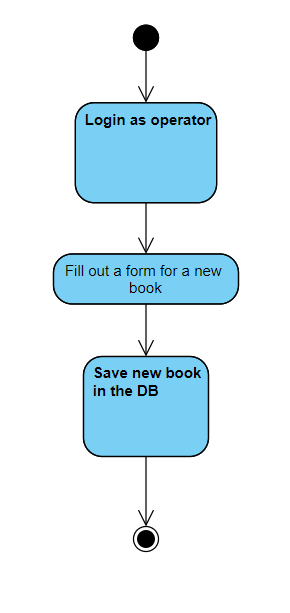
В слоя за достъп до базата данни се използва ORM инструментът Hibernate. Класовете от този слой са разделени според ролята на потребителите.

## UML диаграми

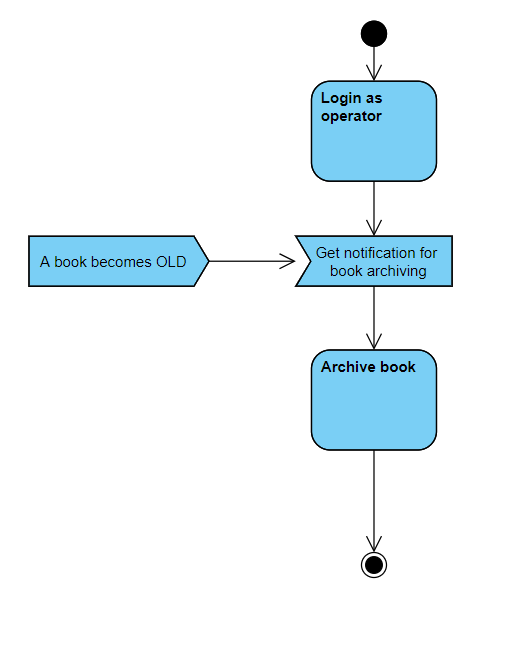
Use case диаграма



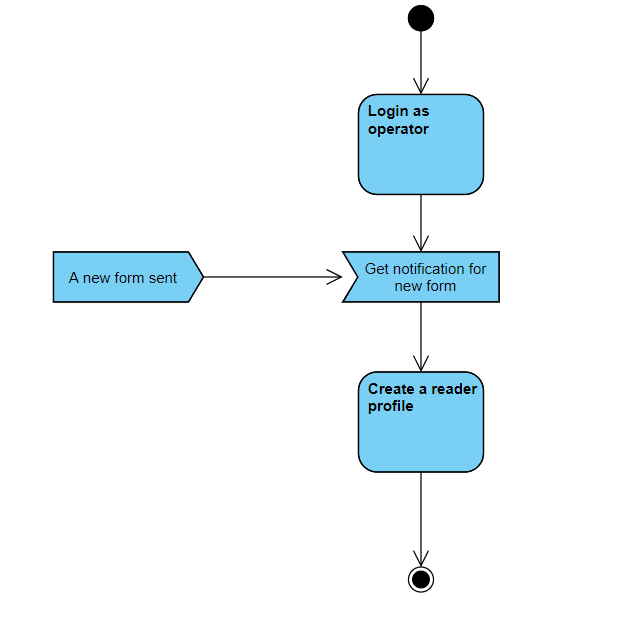
## Activity диаграма за добавяне на нова книга



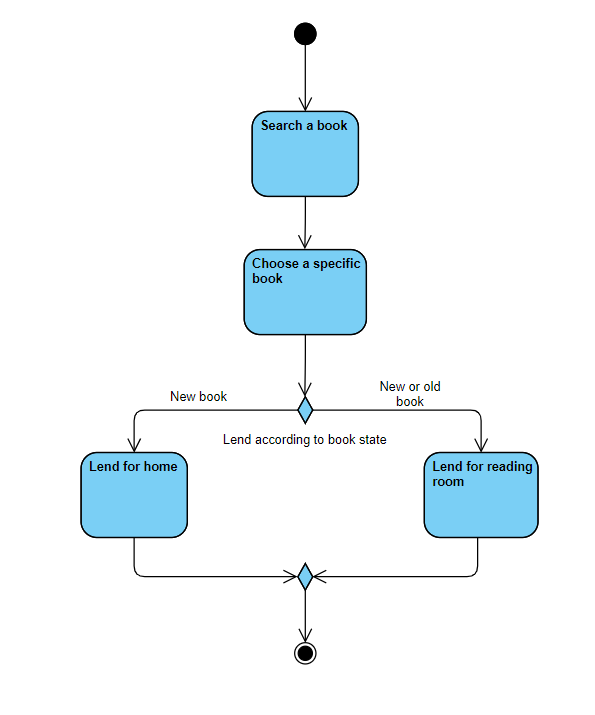
Activity диаграма за архивиране на книга



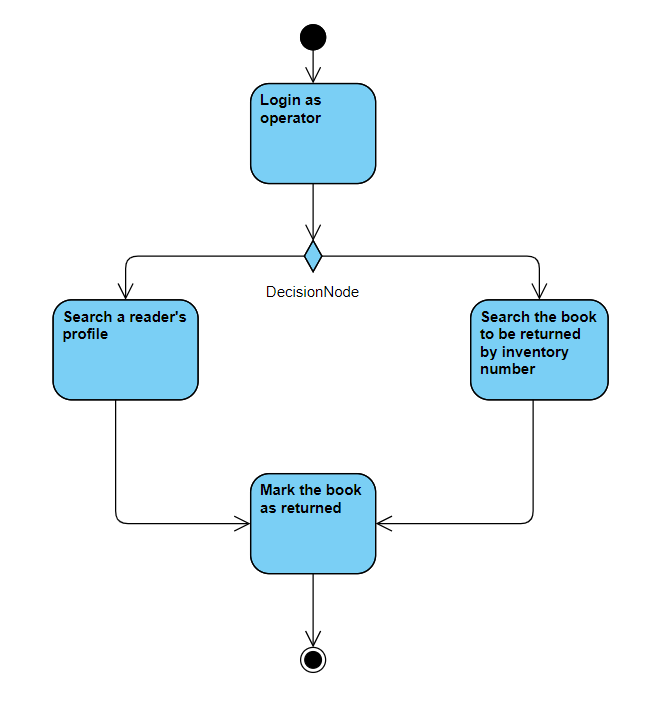
Activity диаграма за създаване на нов читател



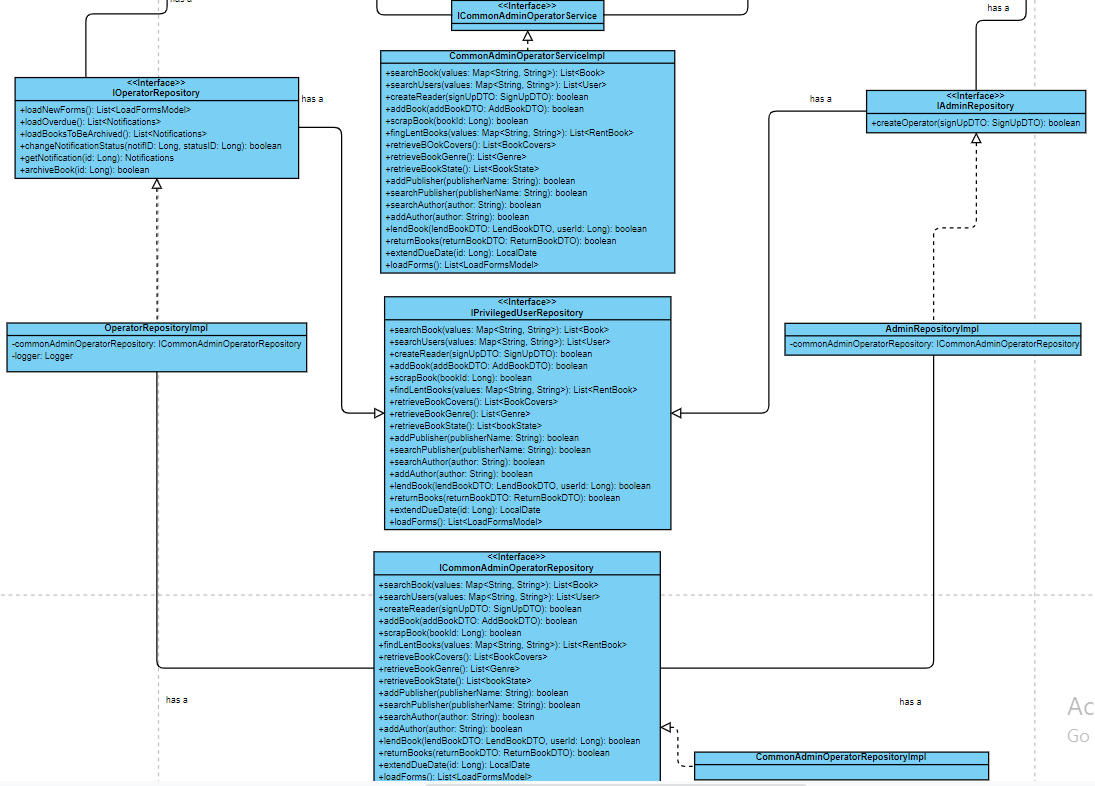
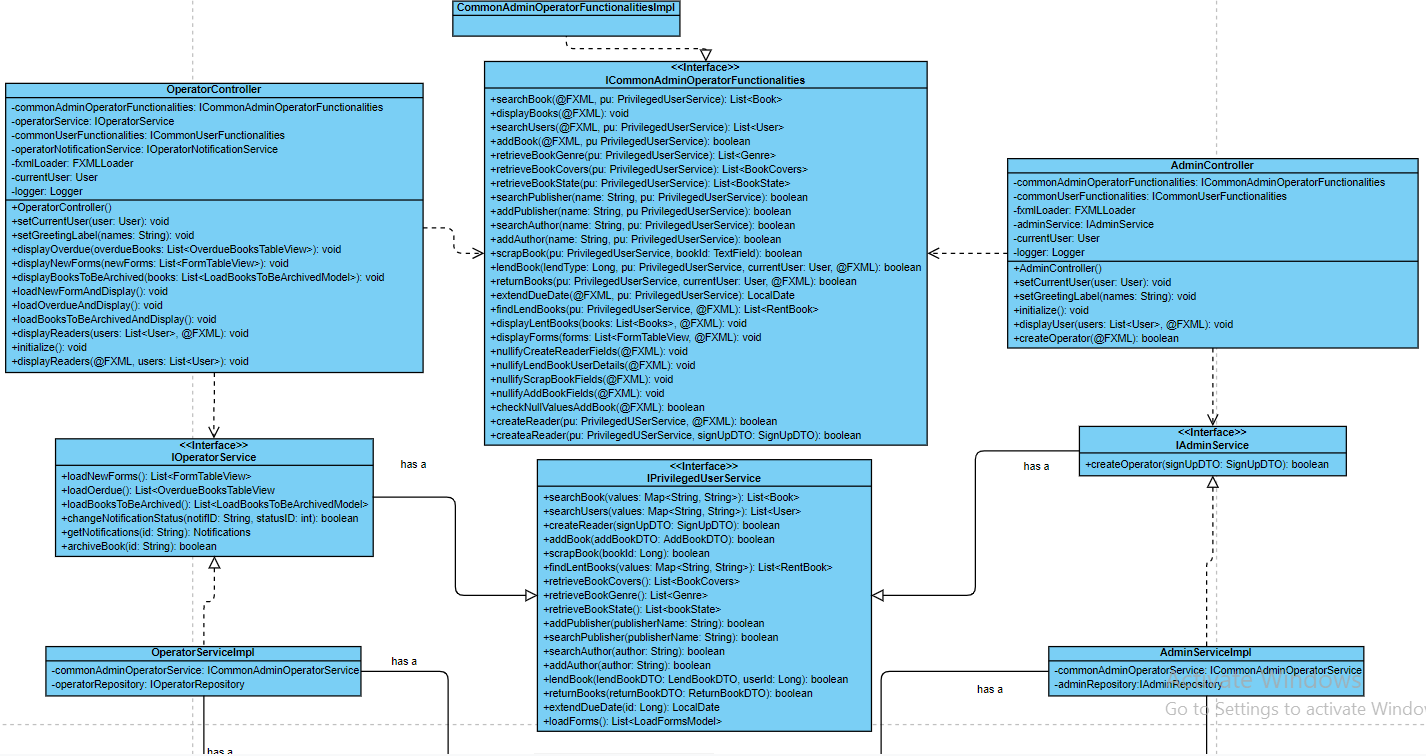
Activity диаграма за отдаване на книга



Activity диаграма за връщане на книга

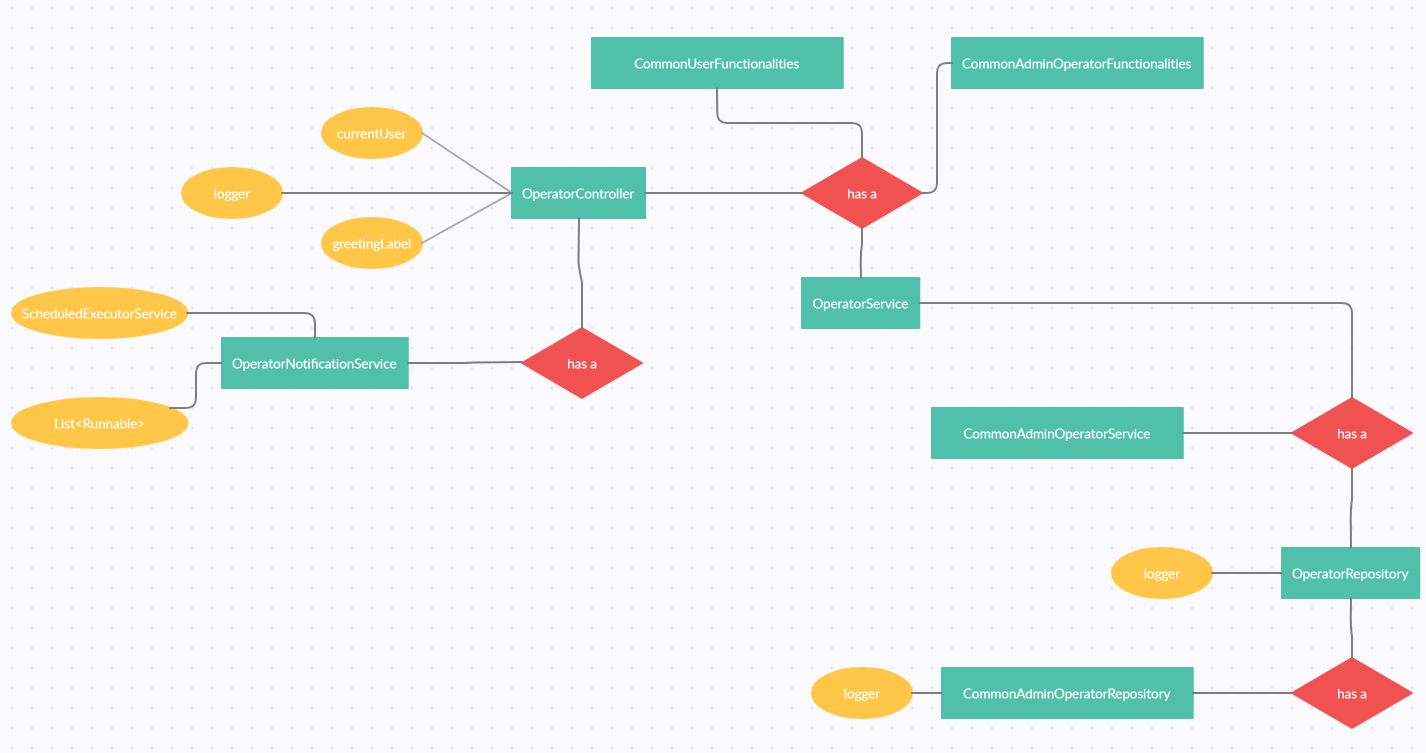


Клас диаграма

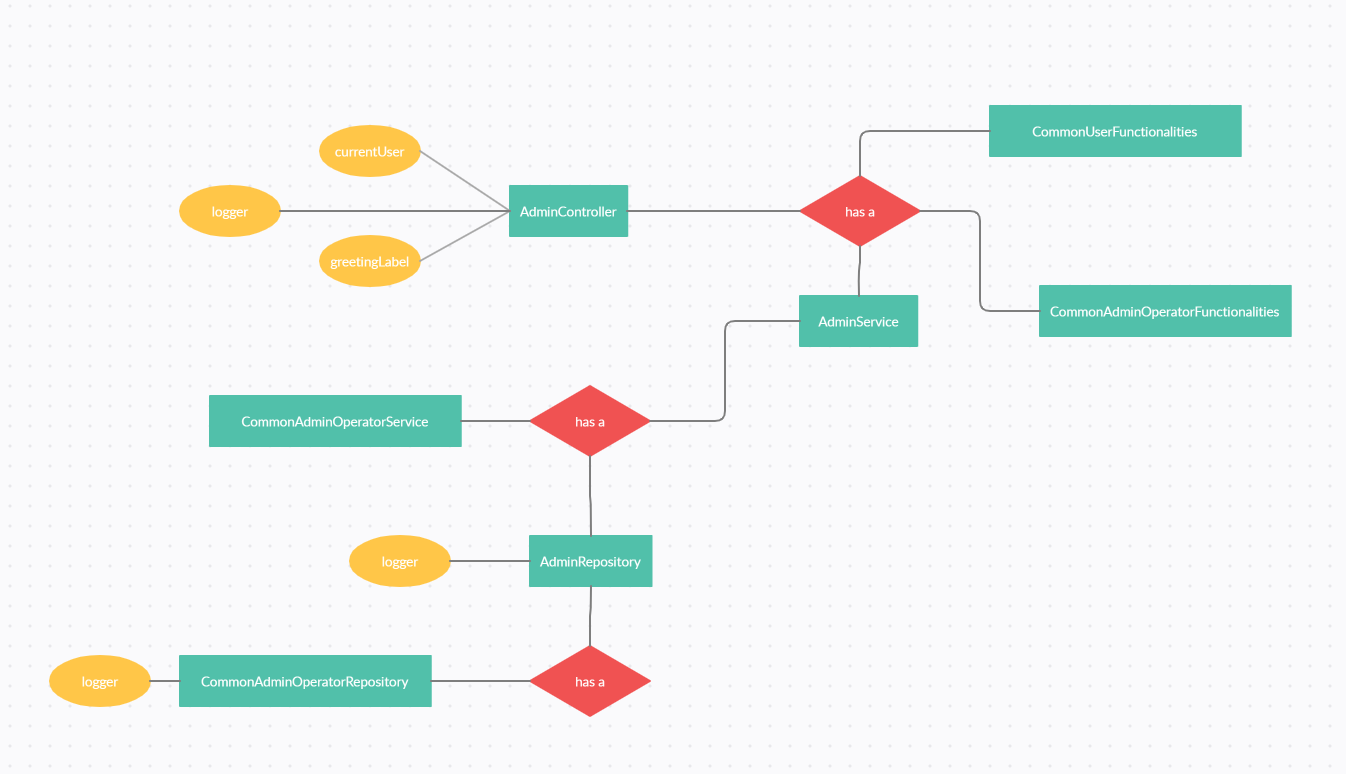


## Концептуален модел на базата данни

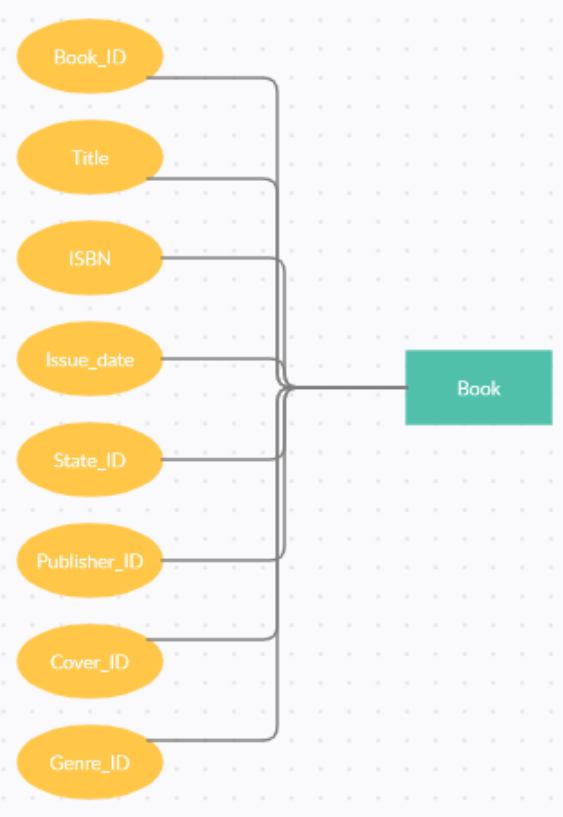
Модел на Чен за структурата на функционалностите за оператор



Модел на Чен за структурата на функционалностите за администратор



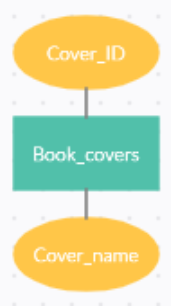
Модел на Чен за ентити Book



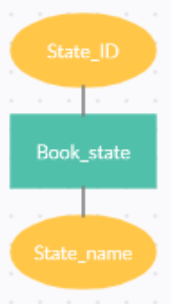
Модел на Чен за ентити Author



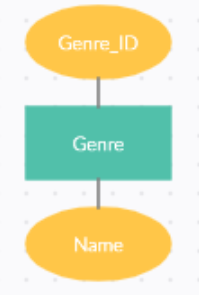
Модел на Чен за ентити BookCovers



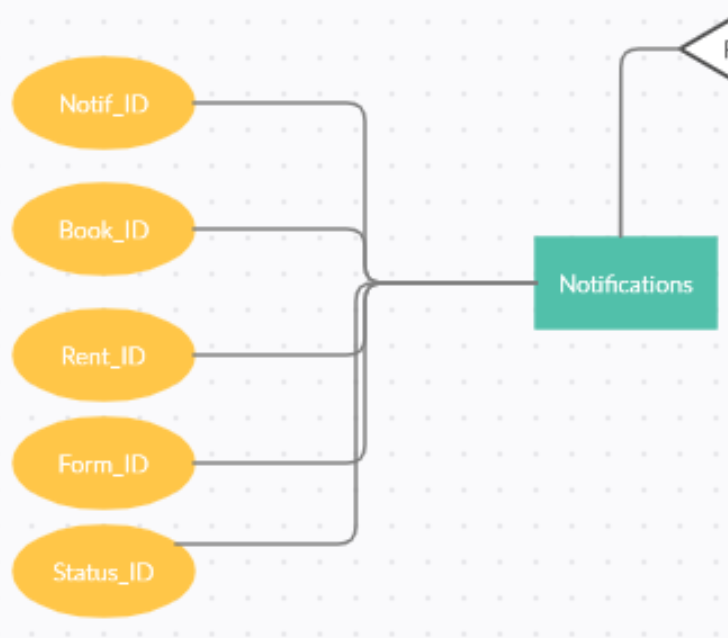
Модел на Чен за ентити BookState



Модел на Чен за ентити Genre



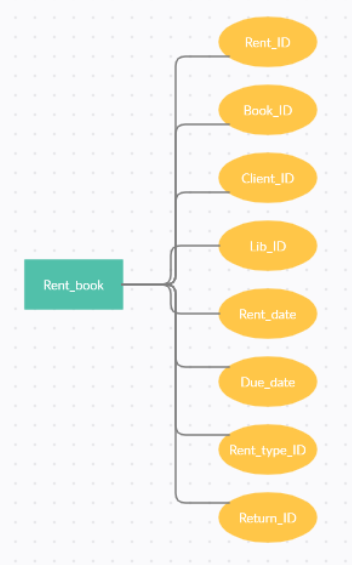
Модел на Чен за ентити Notifications



Модел на Чен за ентити Publisher



RentBook

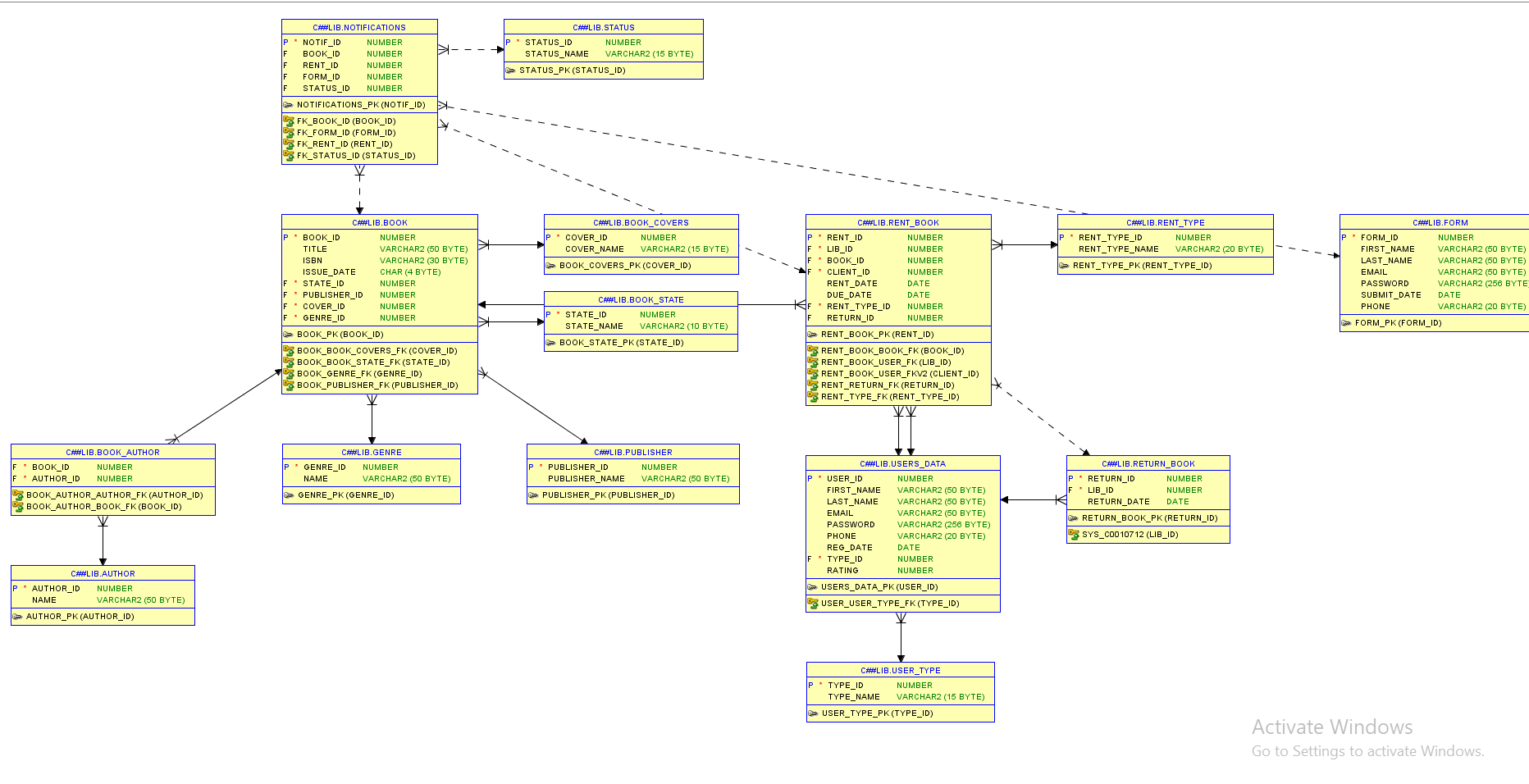


# Реализация на системата

Цялата система използва технологията на dependency injection за освобождаване от тесни зависимости между модулите и. Това се реализира чрез референтната имлементация на JSR-299 API (още известно като CDI – Contexts and Dependency Injection) – Weld.

## Реализация на базата данни

Релационният модел на базата данни изглежда по следния начин:



Тук има 15 таблици:

* notifications – тук се съхраняват известия за събитията просрочване на книга, нов формуляр за читател, необходимост от архивиране на книга
* status – това са статусите, в които едно известие може да се намира, напр. new, in processing, processed
* book – таблица за книга
* author – таблица за имена на автори
* book\_author – междинна таблица, в която много автори могат да са написали една книга и много книги могат да са написани от един автор. Това е many-to-many връзка.
* genre – таблица за жанр на книга
* book\_covers – таблица за видовете корица на книгата
* book\_state – таблица за възможните състояния на книгата
* publisher – таблица за издателствата
* rent\_book – таблица, в която се съхранява информация за дадено заемане на книга
* users\_data – таблица за потребителска информация
* user\_type – таблица за тип потребител
* rent\_type – таблица за вид заемане
* return\_book – таблица за връщане на книга
* form – таблица за формуляри-заявления, изпратени от анонимен бъдещ читател

## Реализация на слоя за работа с база данни

Слоят за работа с база данни се състои от няколко класа. Това са:

* AdminRepositoryImpl – това е имплементация на интерфейса AdminRepository. В него се съдържат имплементациите на методите, налични само за администратор. Това е методът: createOperator(), който създава нов оператор - библиотекар;
* OperatorRepositoryImpl – тук се съдържат имплементациите на методите, които са уникални само за оператор. Те са методите, свързани с получаване на известия:
  + loadNewForms() – сканира таблицата notifications селектира тези записи(Form), където Status е New, т.е. необработено известие.
  + loadOverdue() - сканира таблицата notifications селектира тези записи(RentBook), където Status е New, т.е. необработено известие.
  + loadBooksToBeArchived() - сканира таблицата notifications селектира тези записи(Book), където Status е New, т.е. необработено известие.

Също така методи, свързани с обработване на известията:

* getNotification()
* archiveBook()
* changeNotificationStatus()

Още една имплементация се съдържа в този клас:

* searchUsers() – тя специфицира, че операторът може да вижда единствено данните за читатели, но не и за други оператори и администратори.
* CommonAdminOperatorRepositoryImpl – в този клас се намират имплементациите на общите функции за администратор и оператор. Те са:
  + createReader() – създава читателски профил
  + searchBook() – търсене на книги по параметри
  + addBook() – добавяне на нова книга
  + retrieveBookCovers() – получаване на видовете корици на книга
  + retrieveBookGenre() – получаване на жанрове на книга
  + retrieveBookState() – получаване на състояния на книга
  + addPublisher() – добавяне на издателство
  + searchPublisher() – търсене на издателство
  + searchAuthor() – търсене на автор
  + addAuthor() – добавяне на автор
  + scrapBook() – бракуване на книга
  + findLentBooks() – намиране на заетите книги за даден читател
  + returnBooks() – връщане на книга
  + extendDueDate() – удължаване на срока за връщане на книга с един месец
  + loadForms() – зареждане на всички формуляри
  + lendBook() – отдаване на книга на даден читател
* ReaderRepositoryImpl – съдържа единствен метод loadBooks(), който селектира от базата данни, книгите които са заети от логнатия читател към текущия момент.
* LoginRepositoryImpl – съдържа следните методи:
  + findByEmailAndPass() – този метод служи за влизане в система. Той проверява дали съществува потребител с даденото потребителско име и парола. Ако да, методът връща съответният потребител, ако не – връща null.
  + persistNewReaderForm() – метод за регистрация. Той запазва въведените в регистрационната форма данни в таблицата form, като формуляр. Също така този методът автоматично създава нов запис в таблица notifications, за известие за новия формуляр.
* NotificationRepositoryImpl – това е клас, който принадлежи на отделен контейнер в контекста на Weld. Този клас съдържа методи за сканиране на таблиците book и rent\_book:
  + checkForOverdue() – проверява дали на някоя книга липсва дата на връщане, при условие, че посочената дата за връщане е в миналото. Ако има – създава нов запис в таблицата notifications.
  + checkBooksToBeArchived() - проверява дали някоя книга е взета прекалено много пъти, като се взима предвид и годината и на издаване. Формулата за пресмятане на лимита на броя на взимане е:

-0.8\*(2020 – година на издаване) + 60 -> тази формула дава като резултат максималният брой взимания позволени за дадена книга, преди да има нужда тя да се архивира. Колкото по-стара е една книга, толкова по-малък брой взимания са разрешени за нея.

## Реализация на бизнес логика и графичен интерфейс

Слоят за бизнес логиката се състои от следните класове:

* AdminServiceImpl
* OperatorServiceImpl
  + loadBooksToBeArchived() – получава от репозитори слоя List<Notifications> , преобразува го в List<LoadBooksToBeArchivedModel> и го връща в този вид на контролера.
  + loadOverdue() - получава от репозитори слоя List<Notifications> , преобразува го в List<OverdueBooksTableView> и го връща в този вид на контролера.
  + loadNewForms() - получава от репозитори слоя List<LoadFormsModel> , преобразува го в List<FormTableView> и го връща в този вид на контролера.
* CommonAdminOperatorServiceImpl
  + searchUsers() – преобразува рейтинга на потребител, който няма рейтинг(null) в -1, за да може да се вземе стойността в контролера.
  + addBook() – при въвеждане на нова книга в системата, проверява дали съществува такова издателство и автори като въведените данни. Ако не – създава нов запис в базата данни.
* ReaderServiceImpl
  + loadBook() – извиква метод loadBook() от репозитори
* LoginServiceImpl
  + Login()– извиква метод login() от репозитори
  + Signup() – извиква метод signup() от репозитори
* NotificationServiceImpl
  + Initialize() – за стартиране на ScheduledExecutorService
  + Stop() – за спиране на ScheduledExecutorService
* OperatorNotificationServiceImpl
  + Initialize() – стартиране на тасковете чрез ScheduledExecutorService
  + setTask() – добавя нова задача от тип Runnable към списък от задачи за изпълнение

Слоят за управление на графичния интерфейс се състои от следните класове:

* LoginController
  + login() – създава обект за трансфер на данни (DTO) с въведените в логин формата стойности.
  + signup() - създава обект за трансфер на данни (DTO) с въведените в регистрационната форма стойности. В базата данни това отговаря на един запис в таблица form.
  + Initialize() - слуша за действия върху контролите на GUI от страна на потребителя
* OperatorController
  + setCurrentUser() – сетър за атрибут currentUser;
  + setGreetingLabel() - сетър за елемент на javafx
  + displayOverdue() – записва в javaFX таблица данните на просрочените книги
  + displayNewForms() - записва в javaFX таблица данните на новопостъпилите формуляри за регистрация на читател
  + displayBooksToBeArchived() - записва в javaFX таблица данните на книгите, които са достигнали своя лимит на заемания и трябва да бъдат архивирани
  + loadNewFormsAndDisplay() – помощен метод за получаване на нови формуляри и тяхното изобразяване в GUI
  + loadOverdueAndDisplay() - помощен метод за получаване на просрочени книги и тяхното изобразяване в GUI
  + loadBookToBeArchivedAndDisplay() - помощен метод за получаване на книги за архивиране и тяхното изобразяване в GUI
  + displayReaders() – записва в javaFX таблица данните на заявените читатели
  + initialize() – слуша за действия върху контролите на GUI от страна на потребителя
* AdminController
  + createOperator() – създава нов профил на оператор
  + setCurrentUser() – сетър за атрибут currentUser;
  + setGreetingLabel() - сетър за елемент на javafx
  + displayUsers() – записва в javaFX таблица данните на заявените потребители(всички типове)
  + initialize() – слуша за действия върху контролите на GUI от страна на потребителя
* ReaderController
  + setCurrentUser() – сетър за атрибут currentUser;
  + setGreetingLabel() - сетър за елемент на javafx
  + displayBooks() - записва в javaFX таблица данните на текущо заетите книги от логналия се читател
  + displayPersonalData() – записва в javaFX listView личните данни на логналия се читател
  + initialize() – слуша за действия върху контролите на GUI от страна на потребителя
* CommonUserFunctionalitiesImpl – методи, общи за всички потребители на системата
  + logout() – метод за излизане от профила
* CommonAdminOperatorFunctionalitiesImpl – методи, общи за привилегированите потребители на системата
  + searchBook() – търсене на книги по параметри
  + addBook() – добавяне на нова книга
  + retrieveBookCovers() – получаване на видовете корици на книга
  + retrieveBookGenre() – получаване на жанрове на книга
  + retrieveBookState() – получаване на състояния на книга
  + addPublisher() – добавяне на издателство
  + searchPublisher() – търсене на издателство
  + searchAuthor() – търсене на автор
  + addAuthor() – добавяне на автор
  + scrapBook() – бракуване на книга
  + findLentBooks() – намиране на заетите книги за даден читател
  + returnBooks() – връщане на книга
  + extendDueDate() – удължаване на срока за връщане на книга с един месец
  + loadForms() – зареждане на всички формуляри
  + lendBook() – отдаване на книга на даден читател
  + displayForms() – извежда в javaFX таблица данните за всички формуляри
  + nullifyCreateReaderFields() – изчиства всички полета, използвани за създаване на читател
  + nullifyLendBookUserDetails() - изчиства всички полета, използвани за заемане на книги
  + nullifyCrapBookFields() - изчиства всички полета, използвани за бракуване на книги
  + nullifyAddBookFields() - изчиства всички полета, използвани за създаване на книга
  + createReader() – създаване на нов читателски профил
  + updateAfterReturn() – обновяване на таблицата след като се отбележи дадена книга за върната (тя изчезва от списъка с книги за връщане)
  + updateAfterExtendDueDate() - обновяване на таблицата след като се удължи срокът за връщане на дадена книга (датата на връщане се променя)
  + checkNullValuesAddBook() – проверява дали има празни стойности в javaFX контролите за добавяне на нова книга

Графичният интерфейс се състои от следните файлове:

* admin.fxml – изглед на администратор
* Initial.fxml – изглед за логин и регистрация
* operator.fxml – изглед на оператор(библиотекар)
* reader.fxml – изглед на читател

## Реализация на модул за регистриране на събития в системата

Log4j.properties – конфигуриране на log4j logger. Логовете се записват в текстов файл log.txt.

## Конфигурация

* ConfigurationSessionFactory – клас за създаване на SessionFactory обект, който да се създава един-единствен път и да се преизползва, когато е необходимо да се създаде обект Session.
* FXApplicationConfig – настройва FXMLLoader да изведе началния изглед на приложението.
* FXMLLoaderProducer – създава обект FXMLLoader
* NotificationConfiguration – входна точка на сървиса за създаване на известия в системата

## Сигурност

За сигурност на потребителите, паролите се складират в базата данни като 256-битов стринг. Този стринг е „посолен“ и кодиран, използвайки PBKDF2 с Hmac SHA-512 алгоритъм.

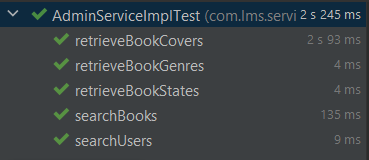
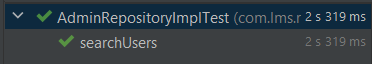
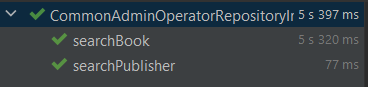
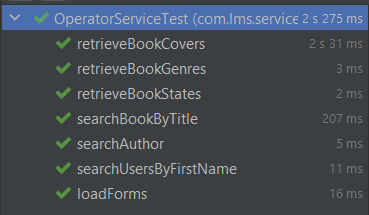
* Password – отговаря за хеширането на пароли.

## Валидация

За валидация на входните данни е използвана Декоратор архитектура. Това позволява информацията да бъде проверявана по различни критерии. Заради този подход е много лесно добавянето на нови критерии за валидация.

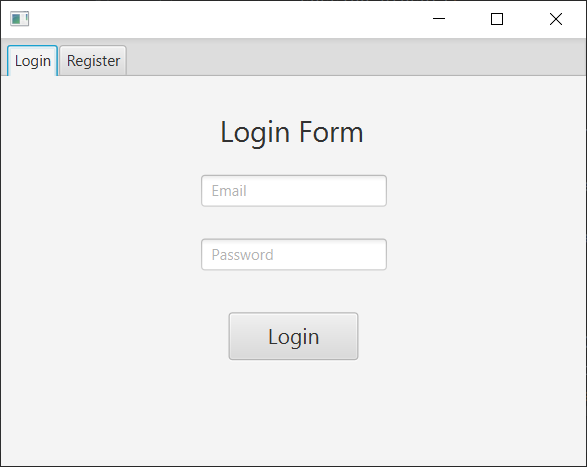
# Тестови резултати

Тестовете са unit и функционални тестове. Това са резултатите от тях:

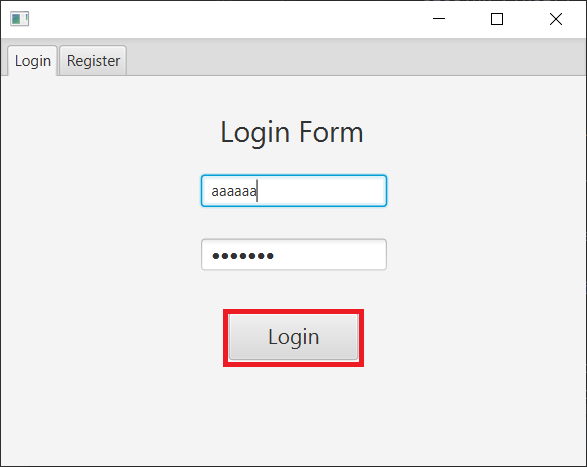


# Screenshots

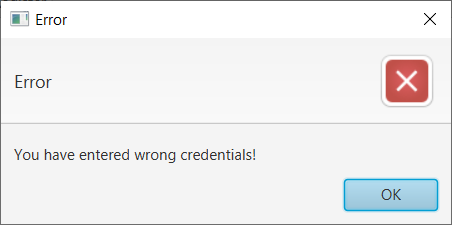
Login форма



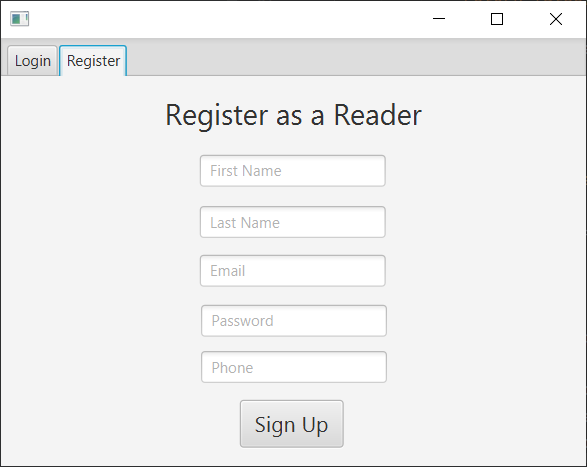
Въвеждане на грешни данни



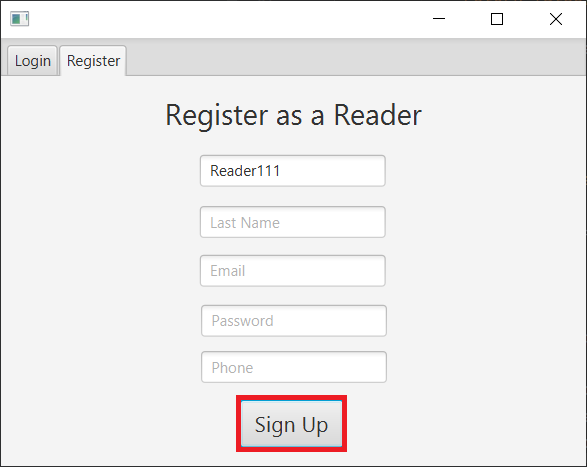
Login Error alert

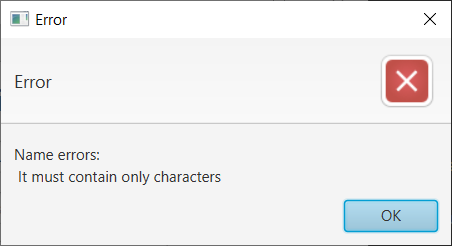


Регистрационна форма

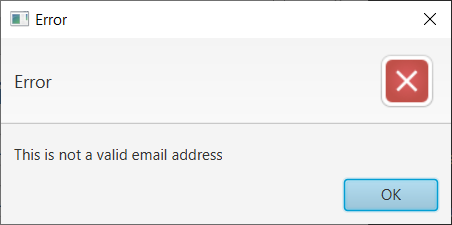
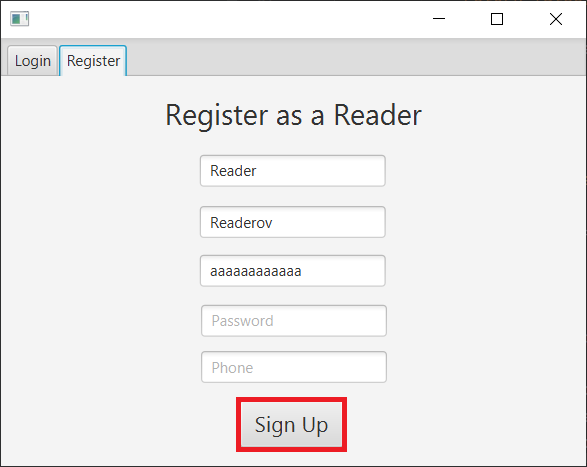


Въвеждане на невалидно име и съобщение за грешка

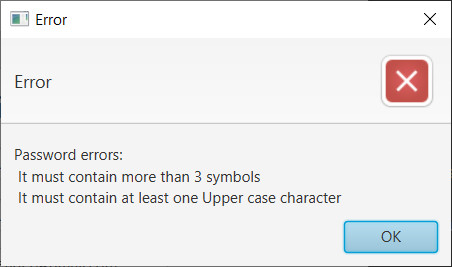
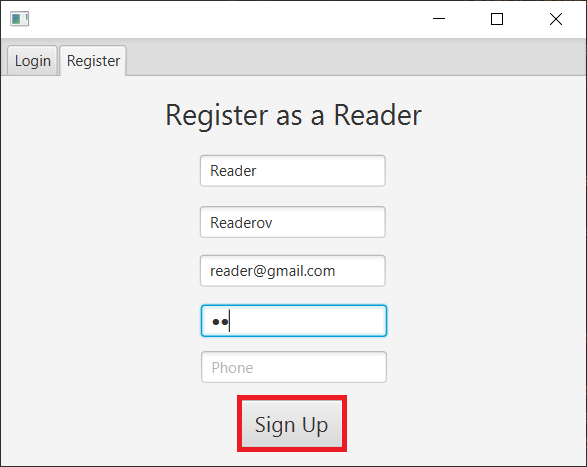




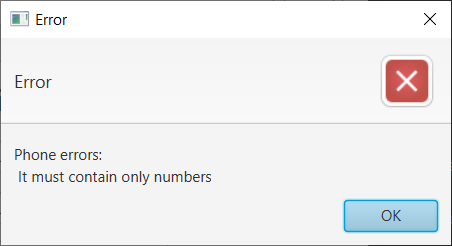
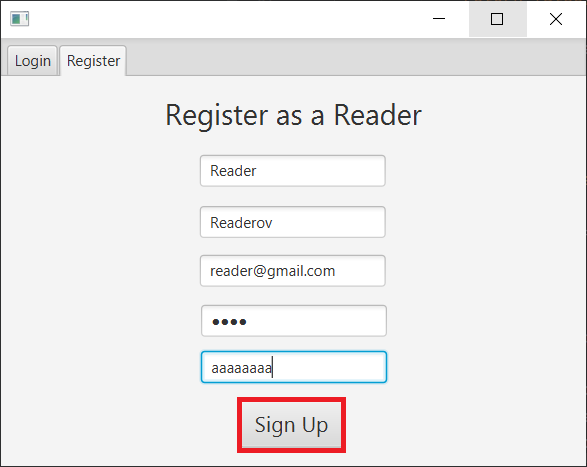
Въвеждане на невалиден имейл и извеждане на съобщение за грешка



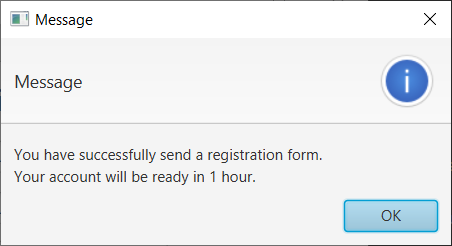
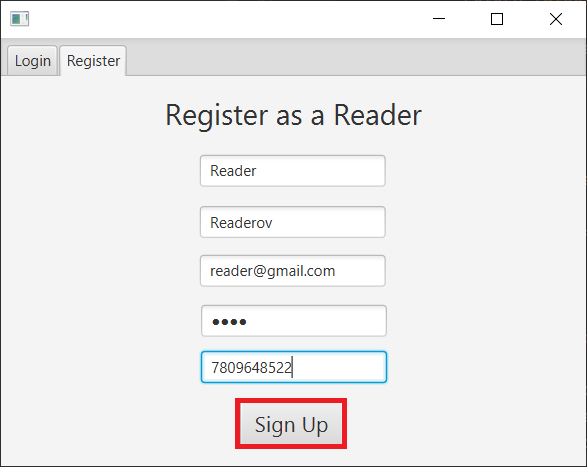
Въвеждане на невалидна парола и извеждане на съобщение за грешка



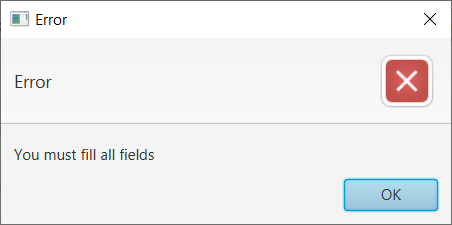
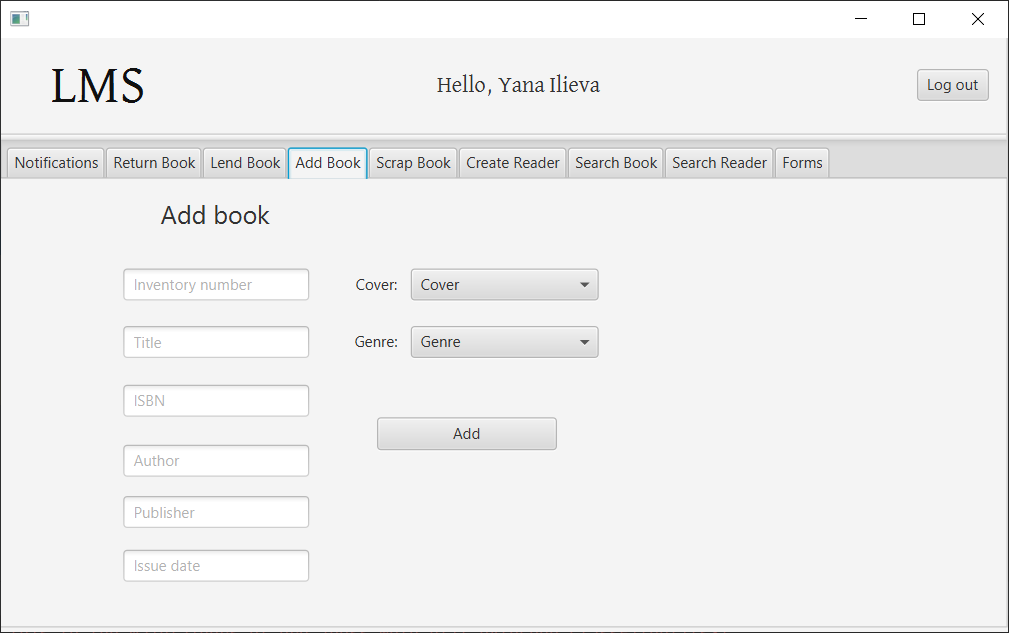
Въвеждане на невалиден телефонен номер и извеждане на съобщение за грешка



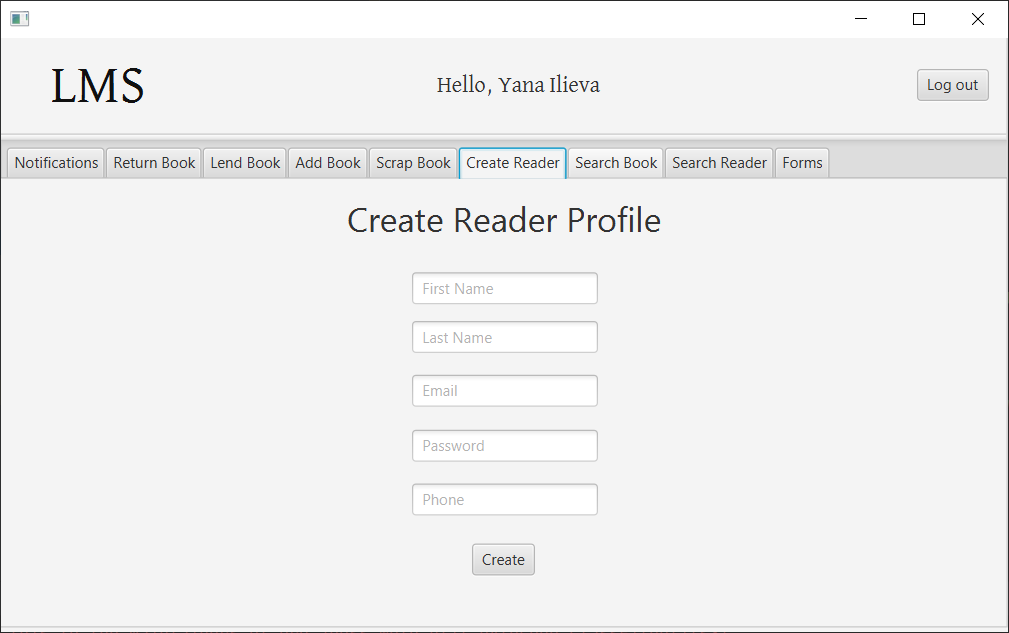
Въвеждане на правилни данни и извеждане на съобщение за успех



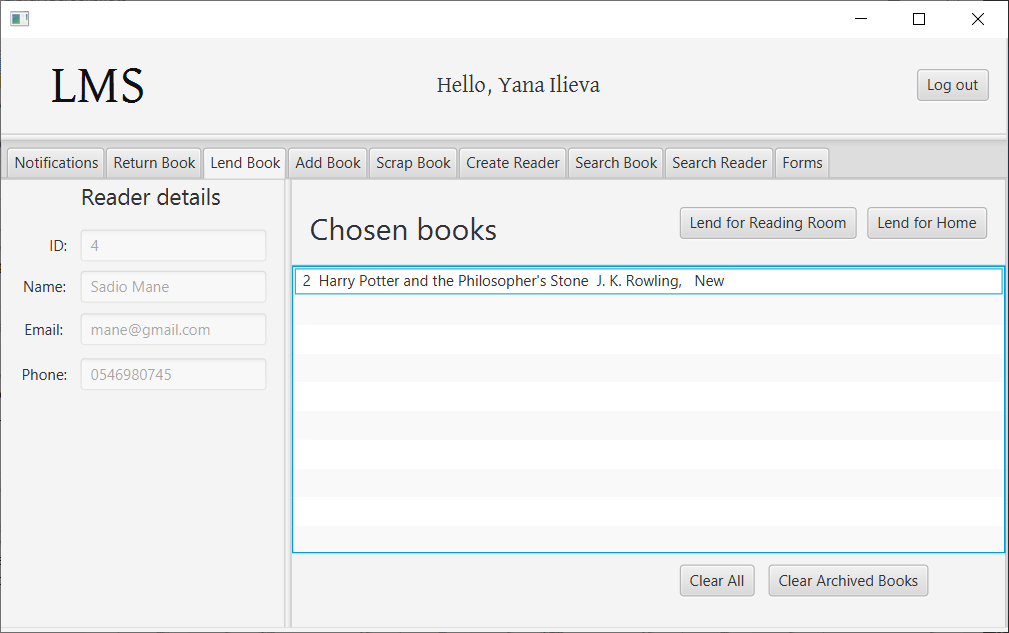
Въвеждане на данни за нова книга и извеждане на съобщение при въведени невалидни такива

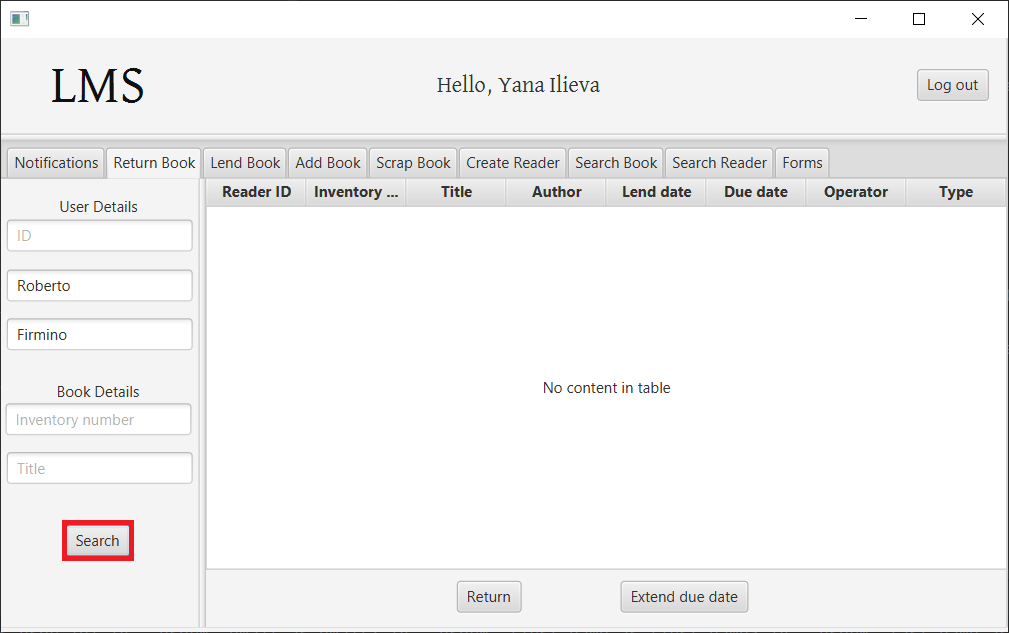


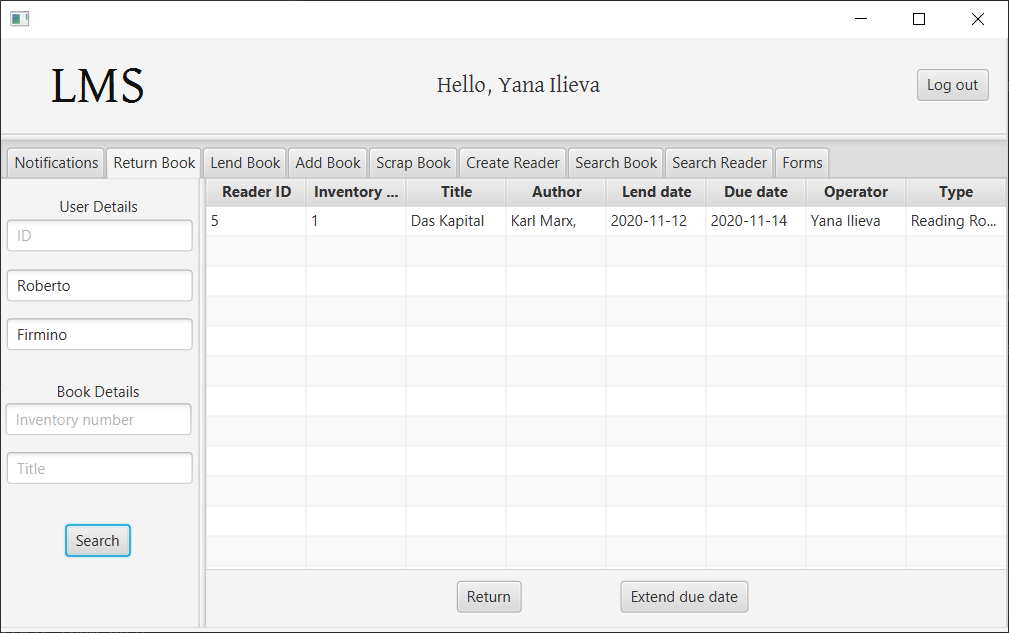
Форма за създаване на читателски профил



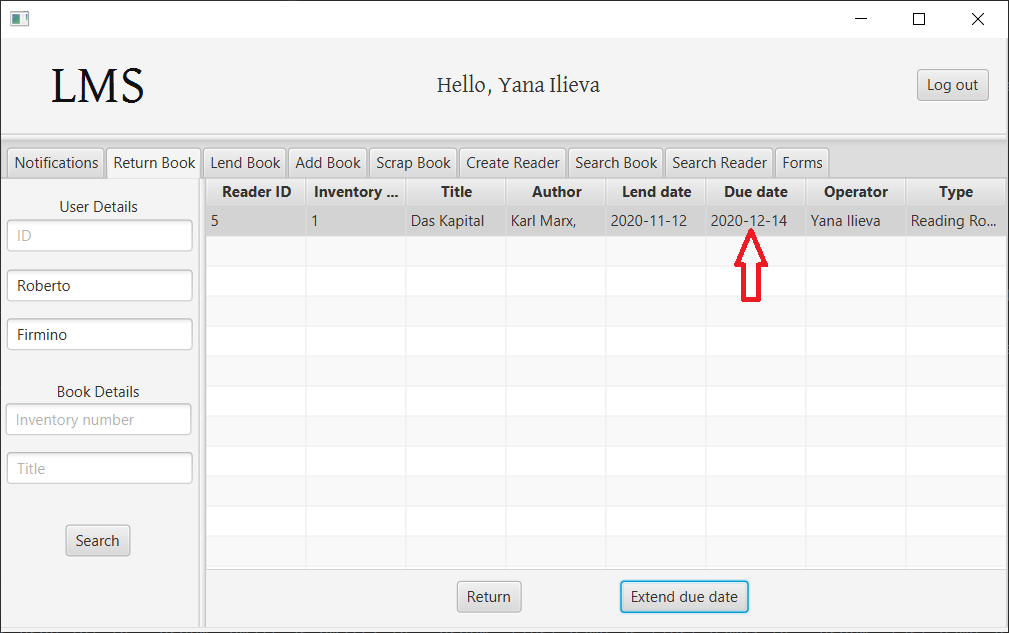
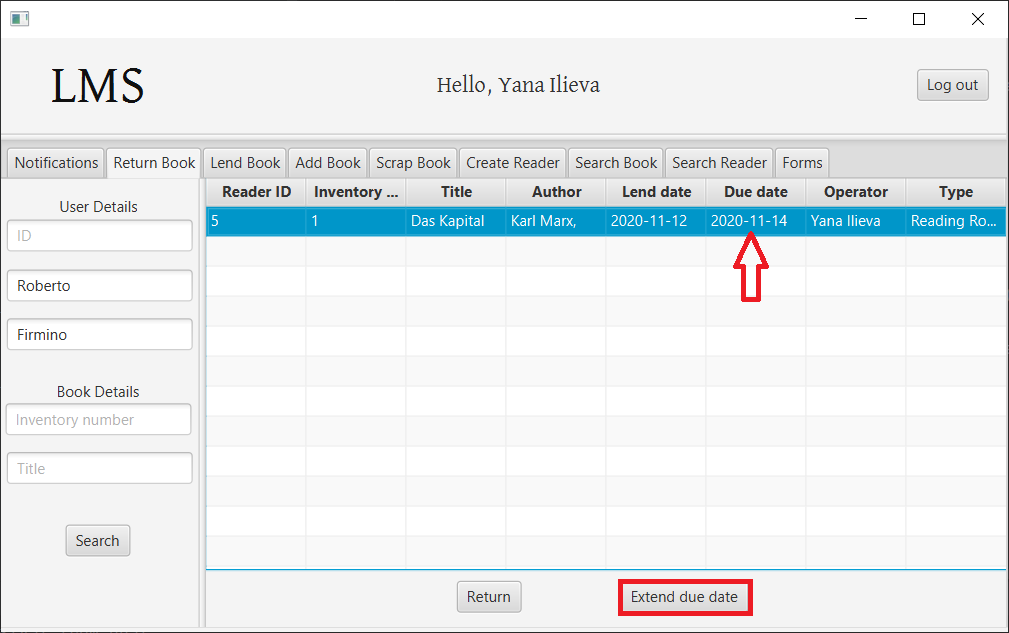
Форма за отдаване на книги на читател



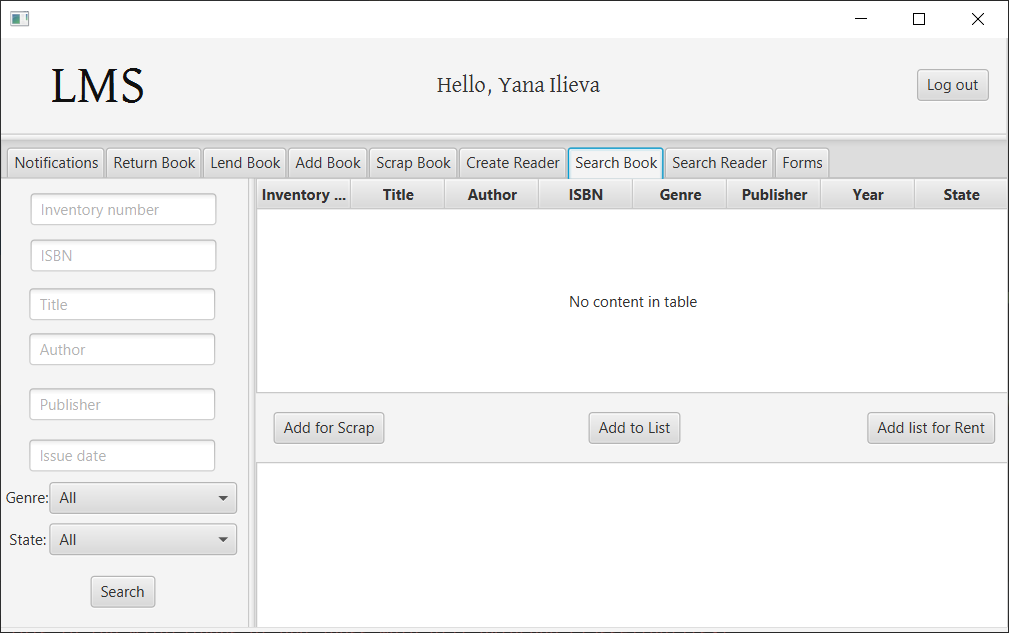
Прозорец за търсене на заети книги по читател



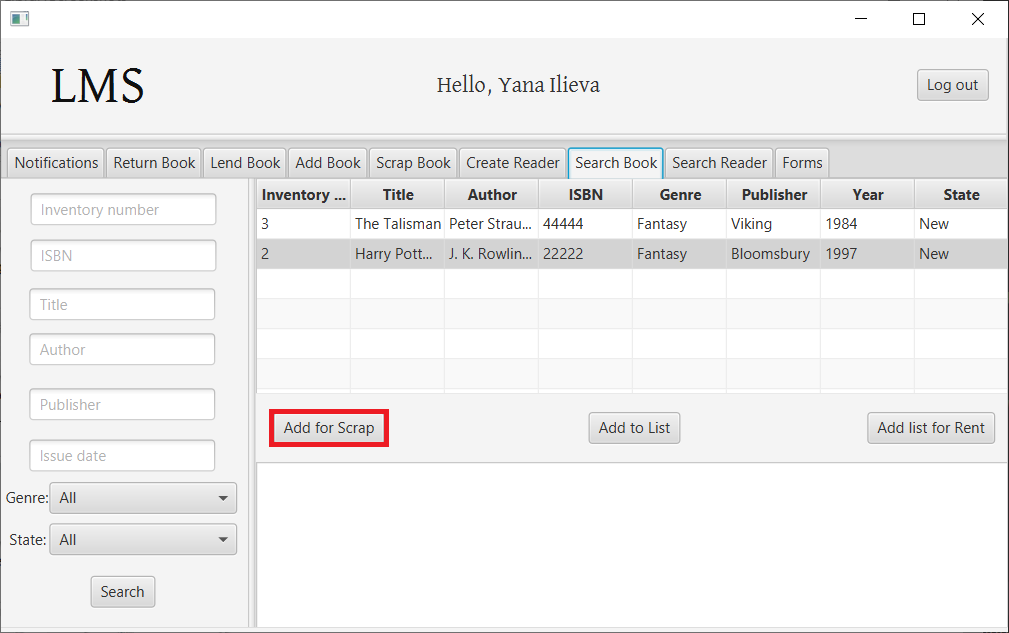
Функционалност за удължаване на срока за връщане на книга

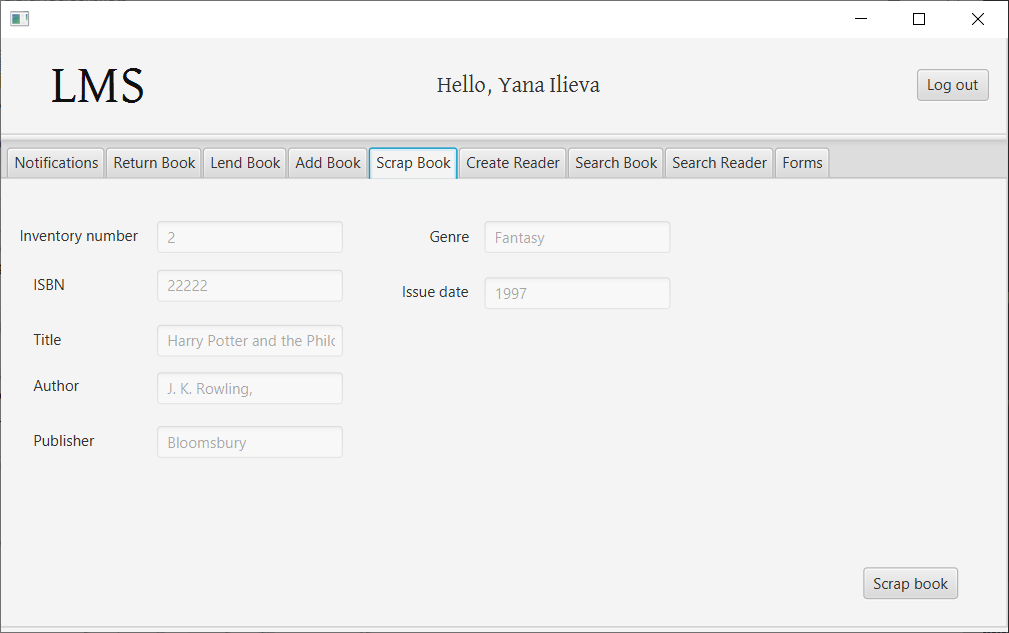


Прозорец за търсене на книги

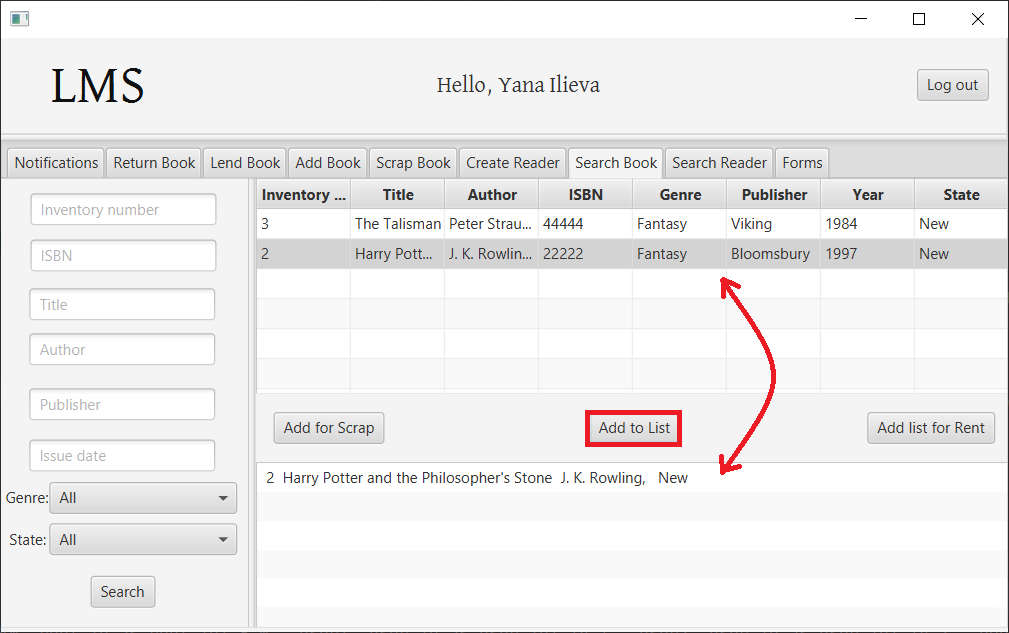


Търсене на книга и добавяне за бракуване

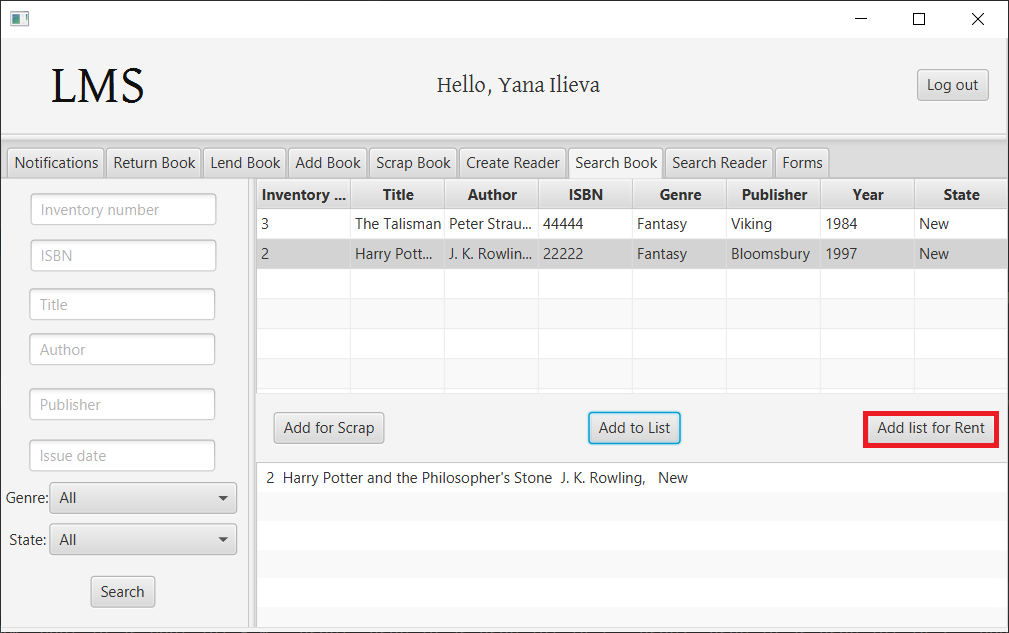


Прехвърляне на данните в прозорец за бракуване на книга

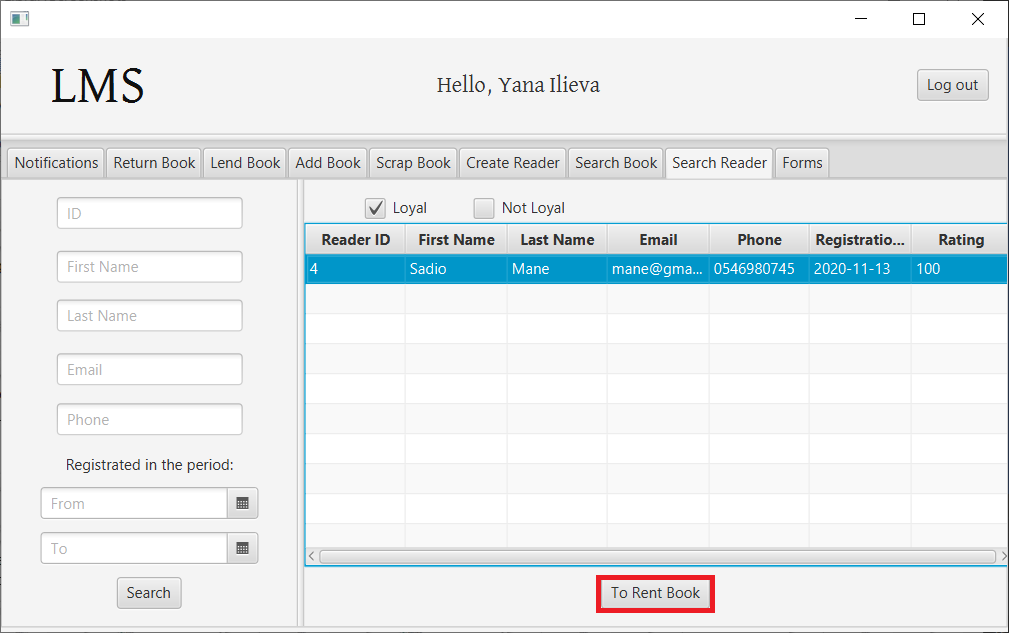
Селектиране на книги за отдаване



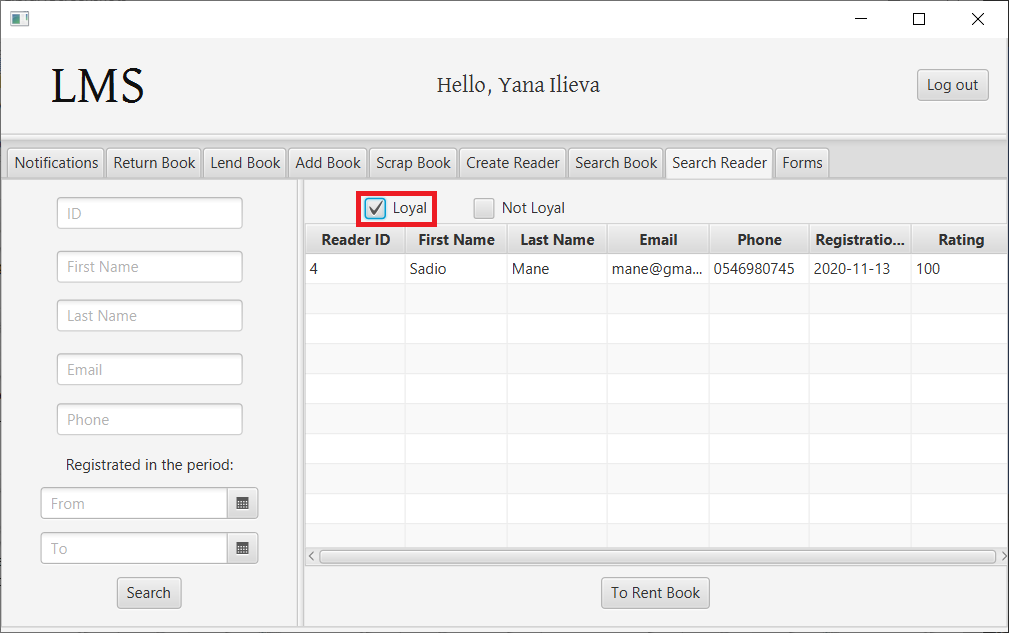
Добавяне на книги към списък за отдаване



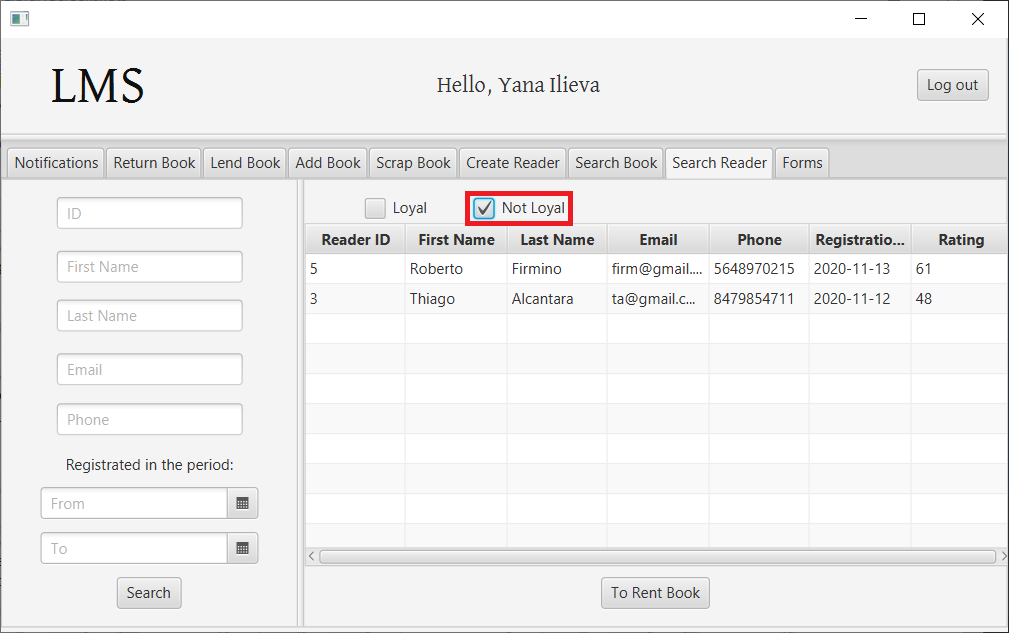
Търсене на читател, на когото да се отдадат книги



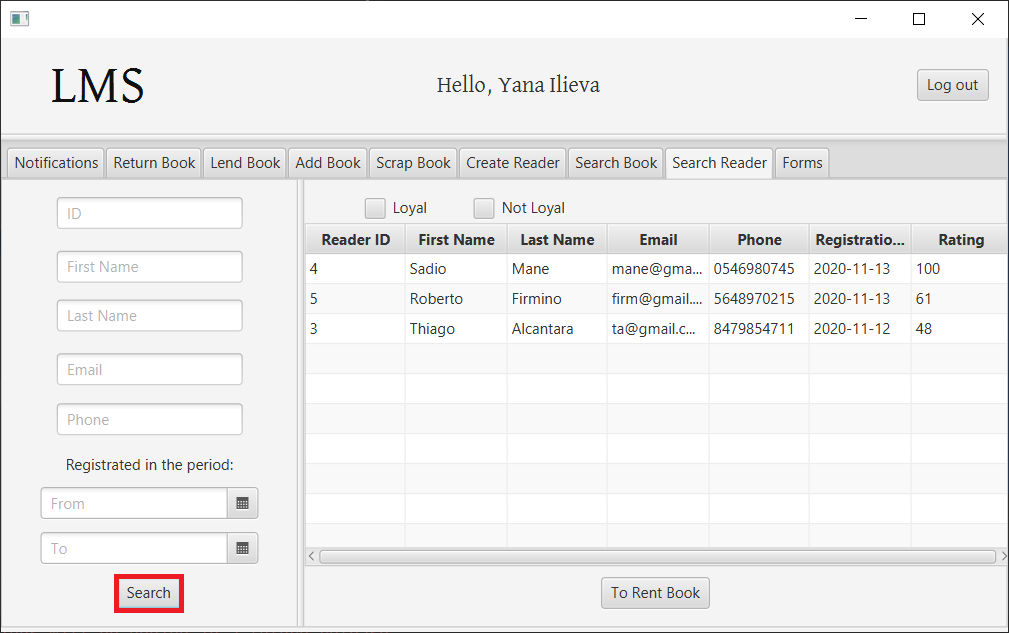
Показване на лоялни читатели, такива с рейтинг над 85



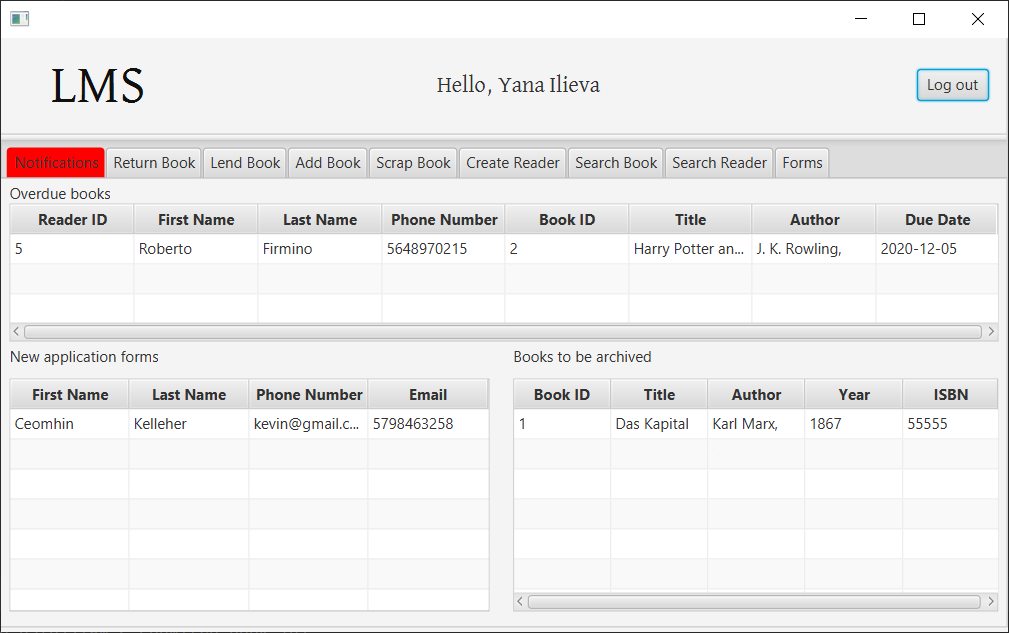
Показване на нелоялни читатели, такива с рейтинг под 85



Прозорец за търсене на читатели



Показване на нотификации



Показване на всички формуляри в базата данни

