Курсов проект

Дисциплина: Проектиране и интегриране на софтуерни системи

Фаза 3: Реализация на системата

Версия 1.0

|  |  |  |
| --- | --- | --- |
| **Фак. №** | **Име на студент** | **Секция от документа** |
| **62121** | Даниел Димитров |  |
| **62151** | Виктор Христов |  |
| **62167** | Иван Чучулски |  |

Януари, 2021

Съдържание

[1 Въведение 3](#_Toc374967428)

[1.1 Цел 3](#_Toc374967429)

[1.2 Резюме 3](#_Toc374967430)

[1.3 Дефиниции и акроними 3](#_Toc374967431)

[2 Използвани технологии 3](#_Toc374967432)

[3 Реализация на базата от данни 4](#_Toc374967433)

[4 Реализация на бизнес логиката 5](#_Toc374967434)

[5 Реализация на потребителския интерфейс 5](#_Toc374967435)

[6 Внедряване на системата 5](#_Toc374967436)

[7 Разпределение на дейностите по реализацията 5](#_Toc374967437)

[8 Приложения 5](#_Toc374967438)

# Въведение

## Цел

Настоящият документ има за цел да представи използваните технологии за реализирането на системата, да представи техните основни характеристики, както и да обоснове избора за използването им.

## Резюме

*Създайте резюме на документа и опишете структурата му.*

## Дефиниции и акроними

*Дефинирайте всички термини, понятия и акроними, използвани в документа.*

# Използвани технологии

За реализиране на проекта са използвани следните технологии :

* Java SE 15
  + Java RMI
* JavaFX 14
* XAMPP версия 7.4.2.
  + Apache 2.4.41
  + MariaDB 10.4.11
  + phpMyAdmin 5.0.1
* Google Books public REST API
* Gson
* Gradle 6.7

Приложението се състои от сървърно и клиентско приложение. Те са написани на Java Standard Edition версия 15. Изборът на Java за технология за реализиране беше мотивиран от факта, че езикът е един от използваните обектно-ориентирани езици за общо ползване, който има поддръжка на повечето съвременни платформи и има наличие на множество инструменти, които подпомагат разработката.

За комуникацията между сървърното и клиентското приложение използваме протокола Java RMI. RMI беше избран като подход за реализирането на клиент-сървър комуникацията, тъй като има вградена поддръжка за конкурентно свързване и комуникиране с множество клиенти по сигурен начин. Факта, че това свойство може да се използва наготово е огромно предимство пред варианта да се използват сокети и ръчна синхронизация на клиентите.

Двете приложения имат графичен потребителски интерфейс, реализиран чрез софтуерната платформа JavaFX. Изборът на JavaFX пред вградената библиотека Swing беше провокиран от по-голямата гъвкавост на JavaFX, богатият избор от готови графични компоненти, както и наличието на инструменти, които улесняват разработката като SceneBuilder. Най-голямото предимство на JavaFX е и факта, че в същността си налага Model-view controller(MVC) архитектурата, която е отразена в структурата на класовете – view-компонентът се представя чрез отделни .fxml файлове, които имат подобна структура на HTML документ, а в обикновен .java файл се реализира контролерът.

Сървърът използва релационна база от данни MariaDB, в който пази информация на потребителските профили и техните колекции от книги. При разработката беше използван софтуера XAMPP, който улеснява конфигурирането и настройката на базата посредством phpMyAdmin инструмента.

За набавяне на информацията относно книгите се използва REST API-то на Google Books. API-то предоставя множество възможности за търсене на книги по различни критерии, има богата селекция от книги, като за всяка книга има подробна информация за автор, издател, година на публикуване и жанр. Върнатите резултати са в JSON формат, като за обработването им се използва библиотеката на Google – Gson.

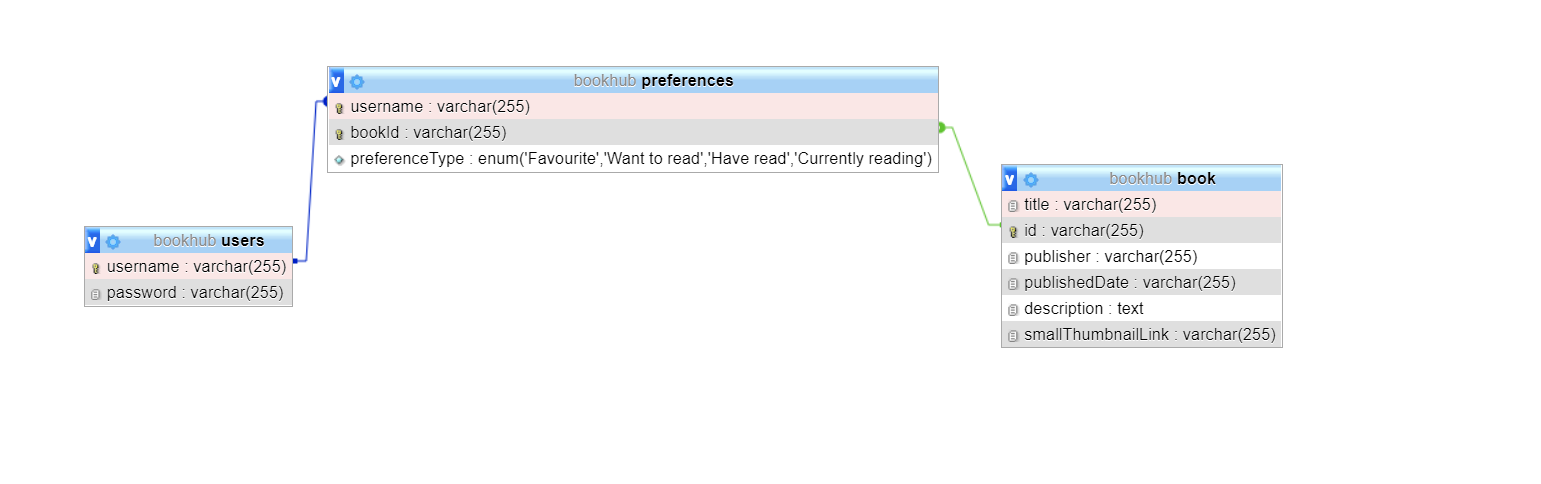
При разработката е използван Gradle - инструмент за автоматизиране на процеса по набавянето на необходимите външни компоненти, както и компилиране и изпълняване на приложенията. Gradle има възможност за дефиниране на конкретни стъпки на процеса на компилиране, откъде точно да се набавят компонентите и каква версия да се използва, както и възможност за създаване на вложени проекти, като именно тази характеристика беше особено удобна за това приложение.

При RMI клиентът използва интерфейс на отправяне на заявки към сървъра, а сървъра имплементира функционалностите в този интерфейс. Поради това структурата на проекта всъщност се състои от три проекта – интерфейс на сървъра, сървърно приложение и клиентско приложение, като клиентът и сървърът зависят от интерфейса на сървъра.

С помощта на Gradle скриптове беше описана тази зависимост и разработката на системата се случване в трите проекта. Това водеше до предимството клиентът да има зависимост (спрямо сървъра) само от неговия интерфейс и абстракциите, които се използват в този интерфейс, които също се намират в проекта bookhubAPI.

# 3. Реализация на базата от данни

Базата данни, която се използва в системата е релационна. В нея има три таблици, които са описани на следната диаграма.



Таблицата “user” представя данните на обикновения потребител – неготово потребителско име и парола. Първичният ключът на таблицата е потребителското име, тъй като то трябва да е уникално за всеки потребител.

Таблицата “book” представлява данните за книгите, като заглавието на книгата, нейният автор, издател и т.н. Първичният ключ на таблицата е id - уникален идентификатор, който е част от детайлите за книгата, върнати от API-то на Google Books.

Таблицата “preference” осъществява функционалността потребител да добавя книга към своя колекция. Първичният ключ на таблицата се състои от потребителското име и id-то на книгата. В таблицата има изброим тип, който представлява вида на колекцията – „любими“, „желая да прочета“, „прочетени“, „в момента се четат*“.*

# Реализация на бизнес логиката

* Модул **bookhubAPI**
  + пакет **api.enums**
    - **BookStatus** – изброим тип, които моделира категориите на книгите.
    - **SearchCategory** - изброим тип, които моделира филтрите, които могат да се прилагат от потребителя при търсене на книга.
  + пакет **api.interfaces**
    - **Book** – интерфейс, който съдържа информация за дадена книга. Използва се от сървъра и клиента.
    - **ServerObjectInterface** – интерфейс, който се използва за комуникация между клиентите и сървърът.
* Модул **bookhubClient**
  + пакет **client**
    - **Client** – класа отговаря за пускането на GUI на клиента.
    - **ClientController** – класа отговаря за имплементирането на бизнес логиката на клиента.
* Модул **bookhubServer**
  + пакет **database**
    - **DatabaseConnector** – класа отговаря за комуникацията между сървъра и базата данни.
    - **User** – класа моделира потребител и се използва в сървъра за показване на всички регистрирани потребители.
  + пакет **dto**
    - **Items**, **BookTransfer** – Data Transfer Objects(**DTO**) обекти, необходими за моделиране на данните, които сървърът получава от **RESTful** Google Books API и десериализира JSON резултата, който се връща от услугите в Java класове.
  + пакет **google.books**
    - **GoogleBooksAPI** – този клас отговаря за консумирането на **JSON** резултата, който сървъра получава от Google Books API и го връща в подходящият формат за обработка на сървъра.
  + пакет **implementations**
    - **BookImpl** – клас, имплементация на интерфейса **Book**.
    - **ServerObjectInterfaceImpl** – клас, имплементация на интерфейса **ServerObjectInterface**.
  + пакет **server**
    - **Server** – класът отговаря за стартирането на сървъра и **GUI** интерфейсът му.
    - **ServerController** *–* класа отговаря за конфигурирането на сървъра и имплементацията на бизнес логиката на сървъра .

# Реализация на потребителския интерфейс

Потребителския интерфейс се разделя на 3 екрана със следните функционалности:

### Екран за регистрация и логин

* Регистрация на потребител
* Влизане в потребителски профил

### Екран за търсене на книги чрез Google Books API и добавяне към профил

* Търсене на книги по зададено име, автор или издателство
* Селектиране на книги и изобразяване на тяхното описание и изображение
* Добавяне на селектирана книга в колекцията на потребителя със зададен статус

### Екран за разглеждане и редактиране на добавени книги

* Разглеждане на книгите в колекцията на потребителя
* Селектиране на книги и изобразяване на тяхното описание и изображение
* Филтриране на книги по име на книга или избран статус
* Сменяне на статус на избрана книга
* Премахване на книга от колекцията на потребителя

## Екранни изображения

### Екран за регистрация и логин

### Екран за търсене на книги чрез Google Books API и добавяне към профил

### *Екран за разглеждане и редактиране на добавени книг*

# Внедряване на системата*.*

Необходимият софтуер за внедряването на системата е

* Oracle JDK (Java Development Kit) версия 15.
* софтуерният пакет XAMPP, в който се съдържа базата данни MariaDB.
* Gradle версия 6.7, чрез който ще се набавят необходимите допълнителни библиотеки за компилиране на компонентите, както и самият Gradle ще бъде използван за изпълнение на сървърното и клиентското приложение.

Като конфигурационни настройки на базата данни е нужно потребителят „root“ да няма парола, тъй като при свързването с базата данни, конекторът използва „root“ като потребителско име и парола „“ - празен низ.

Използването на Java и Gradle позволяват изпълнението на програмите да са независими от използваната операционна система, защото ще използваме Gradle Wrapper, който е платформено независим скрипт за изпълнение на Gradle команди.

След инсталирането и конфигурирането на необходимия софтуер ще покажем стъпките за Windows:

1. стартиране на базата от данни от контролния панел на XAMPP
2. отворяне на терминал (cmd или powershell) в проекта на bookhubAPI
   1. изпълнява се командата

“.\gradle wrapper”

която стартира daemon за Gradle Wrapper-а

* 1. изпълнява се командата

“.\gradlew build”

която компилира проекта

1. отворяне на терминал в проекта на bookhubServer
   1. изпълнява се командата

“.\gradle wrapper”

която стартира daemon за Gradle Wrapper-а

* 1. изпълнява се командата

“.\gradlew run”

която компилира и стартира проекта. Тази команда трябва да се изпълни точно веднъж, тъй като трябва да има само една инстанция на сървъра.

1. отворяне на терминал в проекта на bookhubClient
   1. изпълнява се командата

“.\gradle wrapper”

която стартира daemon за Gradle Wrapper-а

* 1. изпълнява се командата

“.\gradlew run”

която компилира и стартира проекта. Множество нови клиенти могат да се получат при всяко изпълнение на тази команда.

# Разпределение на дейностите по реализацията

Всички участници участваха в специфицирането на изискванията и разработването на софтуерната архитектура. Даниел и Виктор разработваха основните функционалности на RMI сървъра и клиента, както и реализираха .fxml файловете и , които са

Иван разработи дизайна на базата данни, написа скриптовете за създаване и разрушаване на необходимите таблици. Иван създаде на класовете за комуникация с REST API-то на Google Books и също така написа скриптовете, които се използват от Gradle.