Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Компьютерные системы и сети (КСиС)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

СЕТЕВОЕ ПРОГРАММНОЕ СРЕДСТВО   
«ЭЛЕКТРОННАЯ БИБЛИОТЕКА»

БГУИР КР I-40 01 01 628 ПЗ

Студент: гр.351006 Шульга Е.С.

Руководитель: Трус В.В.

Минск 2015

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc421464915)

[1 ПОСТАНОВКА ЗАДАЧИ 6](#_Toc421464916)

[2 обзор аналогов 7](#_Toc421464917)

[2.1 MyHomeLib 7](#_Toc421464918)

[2.2 MyRuLib 8](#_Toc421464919)

[3 Описание форматов данных 9](#_Toc421464920)

[3.1 Хранение файлов книг 9](#_Toc421464921)

[3.2 Метаданные коллекций книг 11](#_Toc421464922)

[3.3 Список жанров 14](#_Toc421464923)

[4 проектирование программного средства 15](#_Toc421464924)

[4.1 Проектирование архитектуры 15](#_Toc421464925)

[4.2 Проектирование серверного приложения 15](#_Toc421464926)

[4.3 Проектирование клиентского приложения 16](#_Toc421464927)

[5 разработка программного средства 18](#_Toc421464928)

[5.1 Разработка структур данных 18](#_Toc421464929)

[5.2 Разработка серверного программного средства 27](#_Toc421464930)

[5.3 Разработка клиентского программного средства 29](#_Toc421464931)

[6 Руководство по использованию 31](#_Toc421464932)

[Заключение 32](#_Toc421464933)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 33](#_Toc421464934)

[ПРИЛОЖЕНИЕ А (обязательное) Исходные коды методов 34](#_Toc421464935)

# ВВЕДЕНИЕ

Долгое время единственным источником распространения и сохранения информации являлась обычная печатная книга. Однако в связи с бурным развитием технологий, в настоящее время возникло такое явление как информационный взрыв[[1]](#footnote-1)). Нарастающее число публикаций поставило перед библиотеками вопрос хранения столь большого массива информации. Появление электронной публикации позволяет не печатать информацию в виде бумажных книг, а хранить её на гораздо более ёмких электронных носителях в электронных библиотеках [1].

В свою очередь развитие Интернета способствует широкому распространению информации. Электронные библиотеки имеют возможность предоставления доступа к книгам дистанционно.

Также электронные книги решают и другие задачи, такие как упрощение получения книги читателем, уменьшение стоимости издания и распространения книги и т.д. [2]

Согласно некоторым источникам, первая в истории электронная книга была набрана на компьютере в 1971 году Майклом Хартом. В дальнейшем коллекция электронных материалов продолжала пополняться. Была создана первая общедоступная электронная библиотека, в настоящее время известная как проект «Гутенберг» [3].

Одной из крупнейших электронных библиотек русскоязычного сегмента Интернета является Либрусек [4]. Принцип его работы основан на википодобной архитектуре, то есть наполнением сайта, исправлением ошибок занимаются пользователи. Это обеспечивает высокую степень актуальности и обновляемости библиотеки.

В настоящее время пользователи библиотеки начали заниматься независимыми проектами, связанными с Либрусеком. Например, появились обновляемые копии базы книг, программы для работы с ними вне сети Интернет [4]. Однако, такие программы для своей работы требуют наличия у пользователя всей базы книг, что приводит к нерациональному использованию дискового пространства.

В данной курсовой работе предлагается решение данной проблемы путем размещения базы книг на одном компьютере в сети и организации доступа к этой базе с других компьютеров; предлагается реализация сетевой электронной библиотеки, которая может быть использована в локальных сетях как зеркало сайта Либрусек.

# ПОСТАНОВКА ЗАДАЧИ

Входными данными для работы являются упакованные файлы книг, список метаданных книг в специальном формате и список жанров книг. Необходимо изучить форматы хранения этих данных и организовать простой и понятный сетевой интерфейс доступа к ним.

В рамках курсового проекта необходимо спроектировать и реализовать программное средство, включающее в себя клиентское и серверное приложение.

Серверное приложение должно быть реализовано как веб-сервис, осуществляющий импорт входных данных, организующий их хранение и предоставляющий программный интерфейс для удаленного доступа к этим данным без права их модификации: поиск записей книг по шаблону, получение списка найденных книг, получение файла книги. Также необходимо обеспечить возможность получения и вывода информации о запросах.

Реализовать графический интерфейс клиентского приложения, предоставляющее возможность формирования шаблона для поиска, отображения полученного списка книг, возможность выбора книги из списка и ее скачивания. Также необходимо реализовать функцию автообнаружения сервера, если клиент и сервер находятся в одной сети.

Обе части приложения должны иметь возможность простого развертывания на компьютерах пользователей. Необходимо реализовать сохранение вводимых пользователем параметров приложений с возможностью использования их при последующих запусках. Должна быть реализована возможность сохранения книг в различных форматах.

Для закрепления знаний, полученных на учебных занятиях в рамках курсов КСиС, ООТПиСП, ВебТех и ТРПО, языком программирования был выбран язык C#, для проектирования клиент-серверной архитектуры была выбрана технология WCF, для разработки графического интерфейса была выбрана технология WPF.

# обзор аналогов

В связи с тем, что большинство людей предпочитает не иметь большие коллекции книг на диске, программ для организации домашних электронных библиотек существует не так много. Рассмотрим наиболее распространенные.

## MyHomeLib

**2.1.1** Данное программное средство является наиболее используемым при организации домашней библиотеки в странах СНГ. Данное приложение может использоваться и для каталогизации собственной коллекции книг пользователя, и как клиент для работы с копией библиотеки Либрусек (и других библиотек, использующих такие же форматы служебных файлов).

**2.1.2** Главная функциональная возможность данного приложения – поиск по автору, названию, жанру и другим параметрам книг в коллекции. Пример окна поиска приведен на рисунке 2.1.

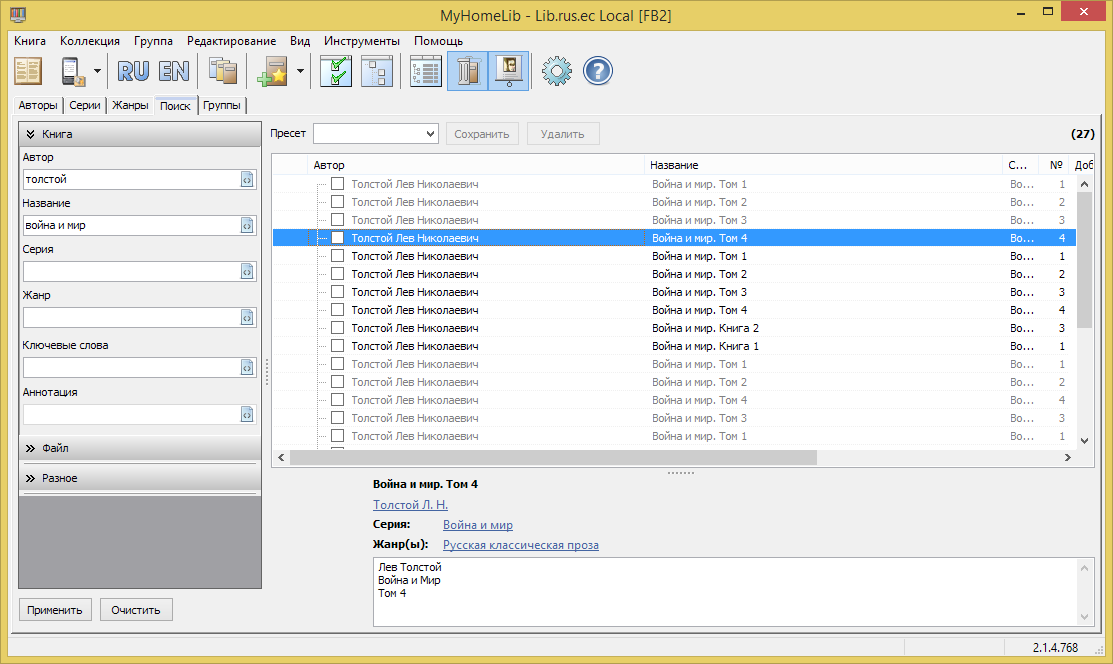


Рисунок 2.1 – Поиск книг в программном средстве MyHomeLib

**2.1.3** После нахождения требуемой книги программное средство осуществляет его экспорт в заранее определенную папку на диске. Возможность чтения книг появляется после экспорта с помощью сторонних приложений.

**2.1.4** Основными недостатками данного программного средства является отсутствие возможности работы по сети и медленное выполнение следующих операций: запуск программы, поиск книг, отображение списков.

## MyRuLib

**2.2.1** Данное программное средство является кроссплатформенным аналогом MyHomeLib. Так же может использоваться для организации и управления домашней библиотекой и для доступа к копии библиотеки Либрусек.

**2.2.2** MyRuLib так же предоставляет возможность поиска по следующим критериям: автор, название. Пример работы программы (поиска книг) представлен на рисунке 2.2.

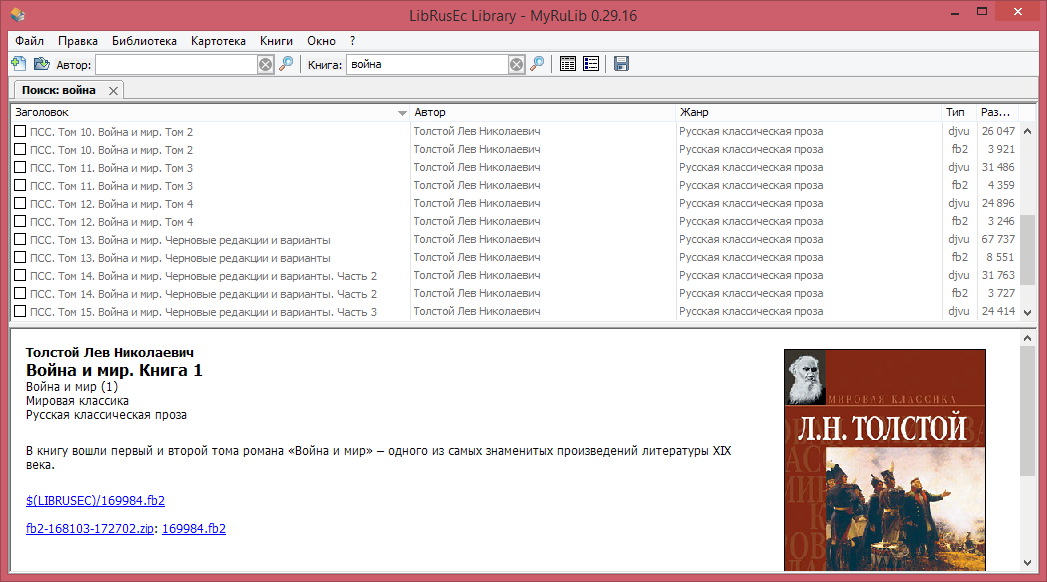


Рисунок 2.2 – Поиск книг в программном средстве MyRuLib

**2.2.3** Данному программному средству присущи следующие недостатки:

* Долгое время импорта книг. Это связано с тем, что MyRuLib не использует файлы метаданных, предоставляемые сетевыми электронными библиотеками.
* Невозможность поиска одновременно и по автору, и по названию (отсутствие возможности задания шаблона для поиска).
* Отсутствие возможности выбора папки для экспорта и неудобно реализованная возможность сохранения книг на диск. Вместо этого при выборе книг предлагается сразу запустить приложение для чтения (такие приложения могут быть не установлены у пользователя).

# Описание форматов данных

В настоящее время люди, занимающиеся распространением электронных книг, организовали унифицированную форму распространения книг из нескольких электронных библиотек. Кроме Либрусека, в таком же формате распространяются копии веб-сайта Флибуста [6] и некотороые другие. Кроме того, и для Либрусека, и для Флибусты существует несколько вариантов распространения: *fb2* копии, содержащие только книги формата .fb2, и *usr* копии, которые могут содержать книги различных форматов: doc, rtf, txt, pdf, epub и т.д.

Перед началом проектирования и разработки необходимо описать структуру входных данных.

## Хранение файлов книг

**3.1.1** В связи с большим размером коллекций целесообразно оптимизировать способы их хранения. Принято использовать упаковку книг в zip архивы (пример папки хранилища книг можно увидеть на рисунке 2.1).

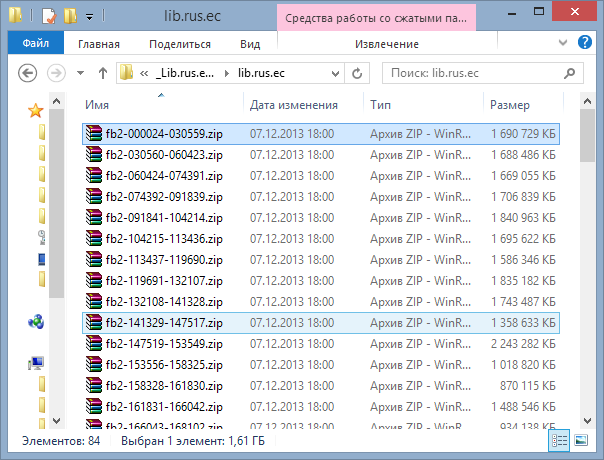


Рисунок 2.1 – Пример папки-хранилища книг

**3.1.2** Можно увидеть следующее правило формирования имен архивов:

<Тип-архива>-<ID-первой-книги>-<ID-последней-книги>.zip

Причем <Тип-архива> может быть *fb2* или *usr* (для коллекций разных форматов). Две другие части имени используются для упрощения управлением коллекцией и для уникальной идентификации каждого архива. Кроме того, как будет сказано ниже, имена архивов и имена файлов метаданных, их описывающих, должны совпадать.

**3.1.3** На рисунке 2.2 можно увидеть содержание одного из архивов.

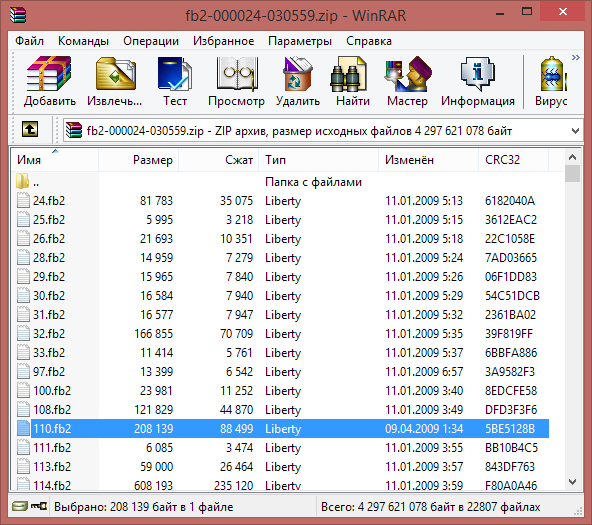


Рисунок 2.2 – Пример содержания архива из хранилища книг

Файлы, представленные на рисунке 2, непосредственно являются книгами. Можно просто распаковывать архивы и выбирать необходимые книги. Однако это неудобно, так как имена файлов книг ничего не говорят об авторе, или названии, или содержании книги – вместо этого используется числовой ключ (при создании пользовательских коллекций имена могут быть произвольными, однако это повлияет на файлы метаданных).

Для коллекций библиотек Либрусека и Флибусты данные ключи используются для уникальной идентификации каждой книги (утверждается что они будут уникальными).

Для организации доступа к книгам из этих архивов и разрабатывается данная курсовая работа.

## Метаданные коллекций книг

**3.2.1** Для описания коллекций книг был разработан специальный формат файлов. В настоящее время его используют для создания коллекций копий веб-сайтов электронных библиотек. Также его целесообразно использовать при каталогизации обширных собственных библиотек. Файлы этого формата имеют расширение inpx.

По своей структуре файлы данного формата представляют собой zip-архив, содержащий файлы с расширением inp. В качестве примера на рисунке 2.3 приведена структура inpx файла библиотеки Либрусек.

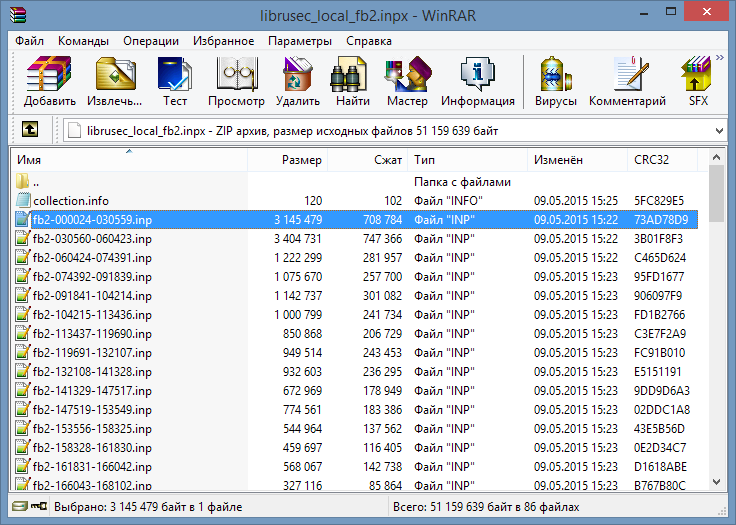


Рисунок 2.3 – Пример содержания inpx файла

В качестве первого файла можно увидеть файл collection.info. Он является необязательным и содержит комментарий, относящийся к данной коллекции.

**3.2.2** Как было сказано в пункте 2.1, имена .inp файлов должны совпадать с именами архивов, к которым они относятся.

Данные файлы являются текстовыми в формате UTF-8. Они построчно содержат информацию обо всех книгах, содержащихся в коллекции. Пример такого файла приведен на рисунке 2.4.



Рисунок 2.4 – Пример inp файла

Кроме того, утверждается, что количество строк такого файла и число книг в соответствующем архиве совпадает, то есть каждая книга имеет свою запись в файле метаданных (безошибочность достигается тем, что создание inpx файла производится автоматически, а не вручную).

**3.2.3** Каждая строка содержит определенным образом расположенные метаданные. Некоторые из этих полей необязательны, то есть они могут содержать пустую строку (тем не менее они должны присутствовать), некоторые поля могут содержать несколько значений. Поля разделяются символом с кодом 04 (EOT – End of Transmission). Названия и описания полей приведены в таблице 2.1.

**3.2.4** В формате .inp файла реализована возможность наличия у книги нескольких (не менее одного) авторов. Признаком конца имени автора является символ с кодом 58 (':'), разделителем фамилии, имени и отчества служит символ с кодом 44 (','):

<Фамилия1>,<Имя1>,<Отчество1>:<Фамилия2>,<Имя2>,<Отчество2>:

Поля фамилии, имени и отчества не являются обязательными и могут быть пропущены. Наличие разделителей полей обязательно.

Таблица 2.1 – Поля записи книги в файле метаданных

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Условное обозначение | Название | Примечание |
| 0 | AUTHOR | Полное имя автора | См. 2.2.4 |
| 1 | GENRE | Жанр | См. 2.2.5 |
| 2 | TITLE | Название |  |
| 3 | SERIES | Название серии | Может быть не заполнено |
| 4 | SERNO | Номер в серии | Может быть не заполнено |
| 5 | FILE | Имя файла в архиве | Для Либрусека совпадает с полем LIBID |
| 6 | SIZE | Размер файла | Размер файла книги в байтах |
| 7 | LIBID | ID книги | Совпадает с FILE; уникально идентифицирует каждую книгу |
| 8 | DEL | Индикатор удаления файла | См. 2.2.6 |
| 9 | EXT | Расширение файла | Для fb2 коллекций – всегда fb2 |
| 10 | DATE | Дата добавления | Дата добавления в базу Либрусека; может быть не заполнено |
| 11 | LANG | Язык книги | Может быть не заполнено |
| 12 | LIBRATE | Внешний рейтинг | Рейтинг книги на сайте Либрусек; обычно не актуально и не заполнено |
| 13 | KEYWORDS | Ключевые слова | Как правило не заполнены |

**3.2.5** Поле жанров, как и поле авторов, может содержать несколько значений; признаком окончания жанра также служит символ с кодом 58 (':'). Однако, это поле принимает одно из предопределенных значений, причем эти значения инициализируются из специального файла жанров. Список жанров формируется библиотекой Либрусек. Подробнее про инициализацию списка жанров см. пункт 2.3.

**3.2.6** Индикатор удаления файла является служебной информацией. Для большинства книг он установлен в 0 (то есть книга не удалена). Если же индикатор установлен в 1, то это может означать, что с книгой что-то произошло: файл был добавлен, а затем удален модераторами, или имеется другая версия книги с существенно лучшим качеством. Тем не менее, если книга уже попала в базу книг, то она не удаляется, а помечается этим флагом.

## Список жанров

**3.3.1** В большие сетевые библиотеки пользователи постоянно добавляют новые книги, и необходимо обеспечить их категоризацию по жанрам. Может так получиться, что новая книга не принадлежит ни одному из заранее определенных жанров, может понадобится расширение списка.

Список жанров предоставляется электронной библиотекой, поэтому при создании копии какой-либо библиотеки рекомендуется использовать актуальный список жанров именно этой библиотеки.

Рассмотрим формат файла списка жанров библиотеки Либрусек.

**3.3.2** Файлы списка жанров имеют расширение .glst. Файлы этого формата являются текстовыми; в них построчно размещена информация о каждом жанре. Формат каждой строки:

<Номер-группы-жанра>.<Номер-жанра>.<Номер-поджанра> <Имя-жанра>;<Описание-жанра>

**3.3.3** Существует возможность упорядочивания по трёхуровневой иерархической структуре: группа жанра, жанр, поджанр. Фактически при этом признак, который устанавливается каждой книге, – это поджанр. Тогда группировка по жанрам и группам жанров является условной, облегчающую присвоение записям книг данного признака.

Номер группы жанра, номер жанра, номер поджанра разделены между собой символом с кодом 46 ('.'); после номера поджанра следует символ с кодом 32 (' ').

**3.3.4** Поле имени жанра содержит сокращенное название жанра. Обычно оно записывается с использованием букв латинского алфавита и знаков подчеркивания, например: detective, det\_espionage, vers\_libre и т.д. Данное поле служит для упрощения идентификации каждого жанра.

**3.3.5** За полем имени жанра и до конца строки следует поля описания жанра. В этом поле принято размещать название поджанра в удобном для чтения пользователем виде. Описание поджанров на сайте Либрусек принято заполнять на русском языке, например: Советская классическая проза, Киносценарии, Европейская старинная литература и т.д.

**3.3.6** Формат файлов предусматривает также размещение комментариев. Комментарием является текст, ограниченный слева символом с кодом 35 ('#') и справа концом строки. Комментарии должны распологаться на отдельных строках. Комментарии могут быть полезны при группировке поджанров по жанрам и группам. Пример комментария:

#---------- 0.2 Детективы и Триллеры -----------------------

# проектирование программного средства

## Проектирование архитектуры

**4.1.1** Программное средство будет состоять из двух частей: серверное и клиентское приложения. Они должны быть независимы, но должен существовать способ их взаимодействия.

**4.4.2** Независимость приложений заключается в следующем:

* независимость развертывания, то есть клиент и сервер могут быть развернуты не обязательно на одном компьютере, причем должна обеспечиваться возможность их взаимодействия;
* независимость настройки: и клиент, и сервер обладают присущим только им списком параметров, причем должна существовать возможность изменения этого списка параметров независимо от другой части программного средства;
* независимость старта: предполагается, что развернут один сервер, и с ним взаимодействует большое число клиентов; должна обеспечиваться параллельная обработка запросов от большого числа клиентов.

**4.4.3** Необходимо разработать программное средство с учетом разделения функций, выполняемых клиентом и сервером.

Основная функция, выполняемая – это импорт входных данных. Еще одна немаловажная функция сервера – извлечение и передача клиенту по запросу файла книги.

Клиент же должен предоставлять возможность обнаружения сервера, установления с ним соединения, формирования запроса для поиска, отображение результатов запроса, получение и сохранение файла книги. Преобразование файла книги в один из выбранных форматов должно происходить на клиенте.

**4.4.4** Еще один параметр, по которому проводится разделение – это данные. Практически все входные данные доступны только серверному приложению. Клиент же получает данные по мере необходимости или по запросу пользователя.

## Проектирование серверного приложения

**4.2.1** Для начала, в соответствие с форматами, рассмотренными в разделе 3, необходимо обеспечить импорт входных данных. Импорт целесообразно производить в следующей последовательности: инициализация списка жанров, инициализация записей книг с помощью файла метаданных, инициализация хранилища, откуда будет производиться извлечение книг. Важным является то, что инициализация жанров должна обязательно производиться до инициализации метаданными, так как в запись книги обязательно должна входить информация о ее жанре.

**4.2.2** Инициализация жанров должна происходить путем построчного считывания файла жанров с игнорированием строк-комментариев. Подходящие строки должны разделяться на поля в соответствие с форматом.

**4.2.3** Затем должна производиться инициализация метаданных. Оно должно производиться следующим образом: для начала из inpx файла, представляющий собой по сути архив, считываются поочередно записи файлов. Затем для каждого из этих файлов производится построчное чтение информации о записях книг. После этого необходимо обеспечить разделение данной информации в соответствие с форматом и заполнение соответствующих полей записей книг. Необходимо предусмотреть возможность наличия у одной книги нескольких авторов и жанров, а также наличие пустых полей. Также необходимо обеспечить сохранение информации о том, из какой записи архива производится считывание данных, так как эта информация соответствует тому архиву из хранилища, в котором располагается книга.

**4.2.4** После завершения инициализации необходимо обеспечить возможность взаимодействия клиентов с сервером. Целесообразно реализовать данную возможность с помощью каких-либо технологий RPC (Remote procedure call – удаленный вызов процедур).

**4.2.5** Необходимо реализовать интерфейс для ввода пользователем параметров инициализации. Поскольку сервер не предполагает активного прямого взаимодействия с пользователем после его запуска, то может быть реализован текстовый интерфейс консольного приложения. При введенных неверных данных необходимо запросить повторный ввод. Также для удобства пользователя следует реализовать возможность сохранения после закрытия приложения и повторного использования введенных параметров.

## Проектирование клиентского приложения

**4.3.1** Основная функция клиентского приложения – предоставление возможности поиска и получения книг. Графический интерфейс должен соответствовать данной функции. Необходимо предусмотреть просмотр пользователем основной информации о книге перед ее скачиванием. Макет главного окна клиентского приложения представлен на рисунке 4.1.

Макет спроектирован таким образом, чтобы клиенту было удобно с первого раза разобраться в способе использования данного программного средства: предполагается, что взгляд пользователя следут справа налево и сверху вниз.

Для начала пользователь осуществляет заполнение шаблона для поиска, например: заполняет текстовые поля названия книги и имени автора, выбирает требуемый жанр. Данные действия будут производиться в области формирования шаблона для поиска. Затем результат обработки сервером запроса будет отображаться в область вывода списка книг. Ниже области списка присутствует область отображения краткой информации о книге. В этой области появляется информация о текущей выбранной книге. После этого предполагается нажатие кнопки для скачивания книги.

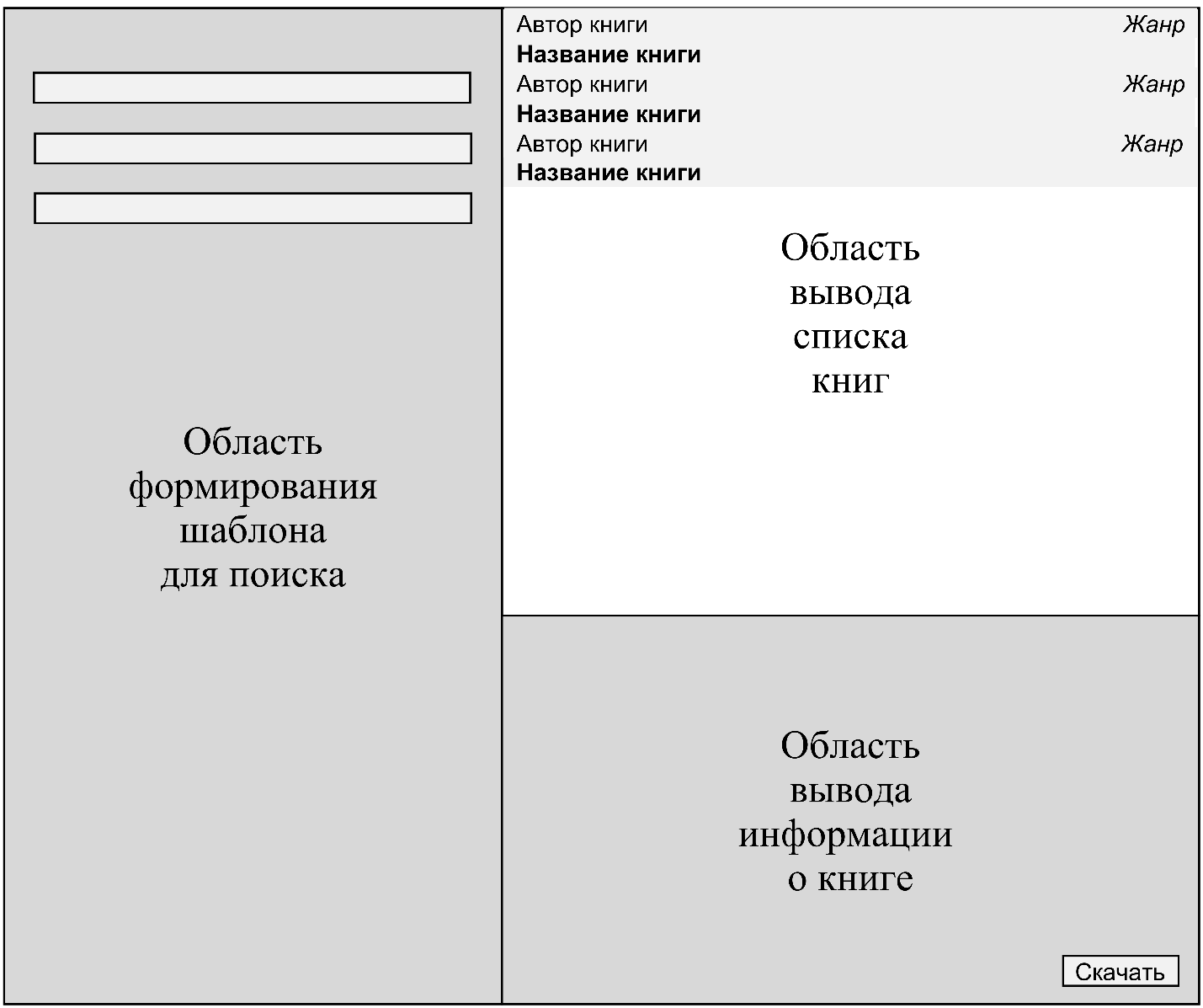
****

Рисунок 4.1 – Макет главного окна клиентского приложения

**4.3.2** Необходимо предусмотреть возможность ввода пользователем сервера для подключения. Однако также должна существовать возможность автообнаружения доступных серверов.

Пользователь должен обладать возможностью выбора папки, в которую будут помещаться скачиваемые с сервера книги.

Для того, чтобы пользователь не вводил каждый раз при запуске одни и те же параметры, необходимо предусмотреть возможность сохранения настроек.

**4.3.3** Книги, предоставляемые библиотекой Либрусек, а, следовательно, находящиеся на сервере, представлены в формате fb2. Однако у пользователя может быть электронное устройство или программа для чтения, которые могут не поддерживать данный формат.

Для решения этой проблемы необходимо обеспечить возможность автоматической конвертации скачиваемой книги в нужный формат. Выбор формата так же должен осуществляться пользователем и сохраняться после перезапусков приложения.

# разработка программного средства

Перед началом реализации возможностей, заявленных на этапе проектирования программного средства, необходимо разработать структуры данных, предназначенные для хранения в оперативной памяти данных, полученных в результате импорта при инициализации серверного приложения. Исходя из принципа ограничения сложности рационально организовать программу в виде некоторого числа классов, каждый из которых будет выполнять определенную задачу.

## Разработка структур данных

**5.1.1** Для инициализации и хранения списка жанров был разработан класс GenresList. Схема алгоритма инициализации представлена на рисунке 5.1.

Для хранения каждой записи жанра был реализован класс GenresListEntity. Структура свойств данного класса представлена в таблице 5.1. Описание данных свойств приведено в пункте 3.3.

Таблица 5.1 – Структура класса GenresListEntity

|  |  |  |
| --- | --- | --- |
| Тип элемента | Название элемента | Назначение элемента |
| int | genreGroupNumber | Номер группы жанра |
| int | genreNumber | Номер жанра |
| int | subgenreNumber | Номер поджанра |
| string | name | Имя жанра |
| string | description | Описание жанра |

Инициализация экземпляра данного класса производится в блоках 6-8 схемы, представленной на рисунке 5.1.

В качестве хранилища жанров в классе GenresList используется массив типа List<GenresListEntity>.

Для доступа к списку жанров были разработаны специальные методы, которые приведены в таблице 5.2.

Следует отметить, что данный список после его инициализации явялется единственным и унифицированным хранилищем жанров для программного средства. Поэтому было решено для уникальной идентификации жанров использовать индекс жанра в массиве жанров в классе GenresList. Данное решение значительно сократило место, необходимое для хранения записей книг, а также ускорило обработку данных, в том числе, поиск по жанру.

Сортировка списка жанров, производимая в блоке 11 на схеме на рисунке 5.1, необходима для того, чтобы, после того, как список жанров будет передан клиенту, он отобразился бы в упорядоченном виде, что значительно упростит поиск жанров.

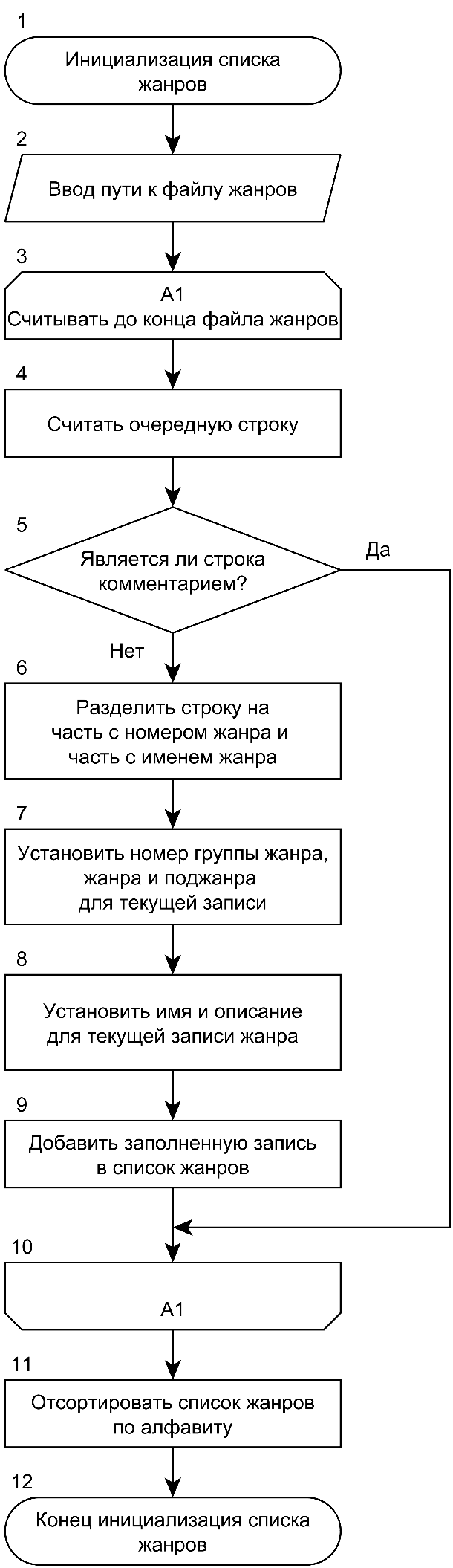


Рисунок 5.1 – Инициализация списка жанров

Таблица 5.2 – Методы класса GenresList для доступа к жанрам

|  |  |  |
| --- | --- | --- |
| Тип возвращаемого значения | Имя метода | Описание |
| string | getGenreName | По индексу возвращает имя жанра |
| string | getGenreDescription | По индексу возвращает описание жанра |
| int | getGenreID | По имени жанра возвращает его индекс |
| List<GenresListEntity> | getAvailableGenres | Возвращает список жанров. Используется при инициализации клиентского приложения |

**5.1.2** Для инкапсуляции методов, предназначенных для извлечения файлов книг, был реализован класс статический BookExtractor. Он включает в себя два доступных извне метода: initialize и extract.

Перед началом работы с этим классом его необходимо проинициализировать. Фактически, инициализация заключается в указании папки на диске, в которой расположены упакованные файлы книг.

После инициализации для получения потока, содержащего файл необходимой книги, был реализован метод extract. Следует отметить, что ему необходимо передать запись книги (экземпляр класса BookEntity, см. пункт 5.1.3), который, при иных входных данных, нужно получить заранее (например, из класса MetadataDB, см. пункт 5.1.7). Он работает следующим образом: получает имя архива, в котором находится файл данной записи книги, открывает данный архив из хранилища, получает запись данного архива, открывает эту запись, возвращает поток, связанный с файлом, находящимся в архиве. Теперь можно передать данный поток куда необходимо и после передачи записать данный поток на диск.

**5.1.3** Для хранения записей метаданных книг был реализован класс BookEntity. Он инкапсулирует поля, предоставляемые форматом файла метаданных, методы их инициализации и методы получения информации о записях книг.

На рисунках 5.2 и 5.3 приведен алгоритм инициализации записи книги.

В качестве реализации возможности наличия у одной книги нескольких авторов и жанров были разработаны классы Authors и Genres. Их инициализация производится в блоках 4 и 5. Реализация данных классов рассмотрена в пунктах 5.1.4 и 5.1.5.

Поле archiveName соответствует имени inp файла из inpx архива. Его инициализация проводится на другом этапе: на этапе импорта метаданных из inpx файла.

Поля, агрегированные в данном классе, соответствуют полям данных inpx файла, пояснение которым дается в пункте 3.2.4 и представлены в таблице 5.3.

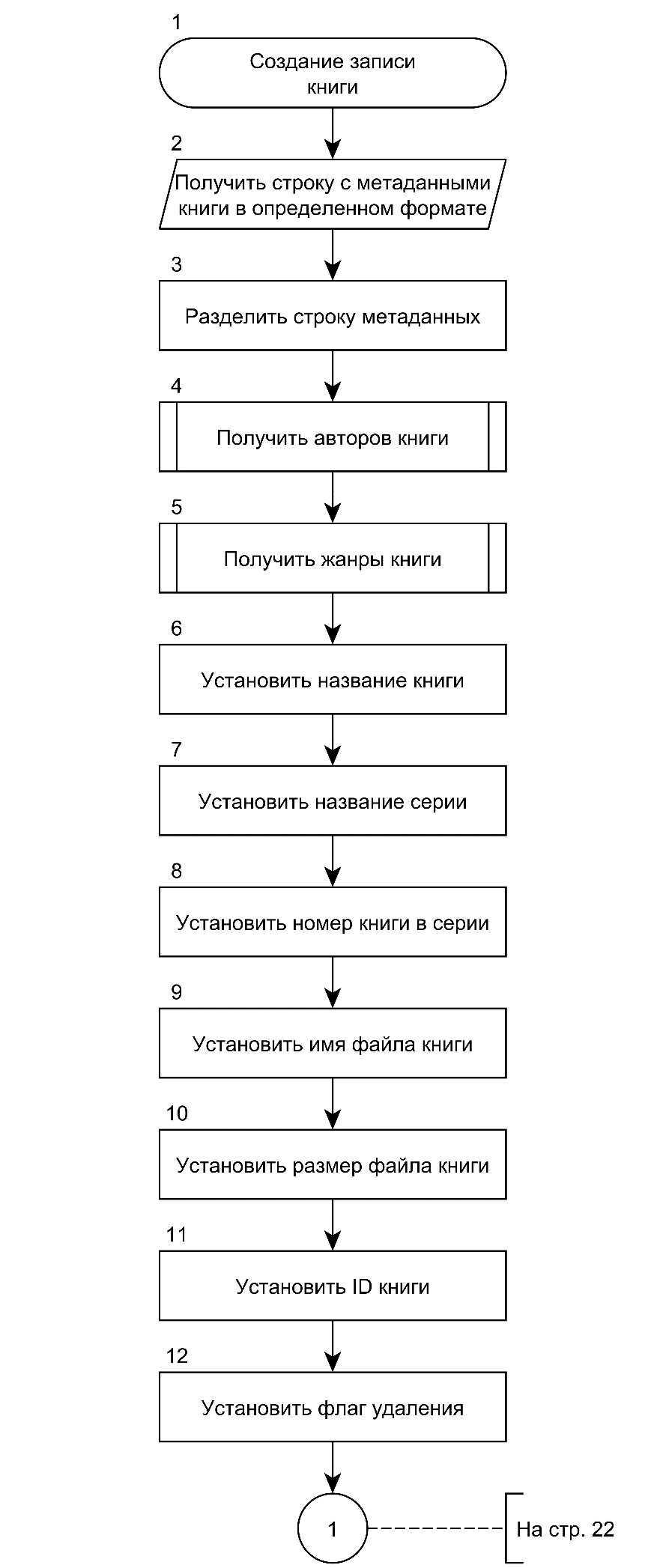


Рисунок 5.2 – Инициализация записи книги

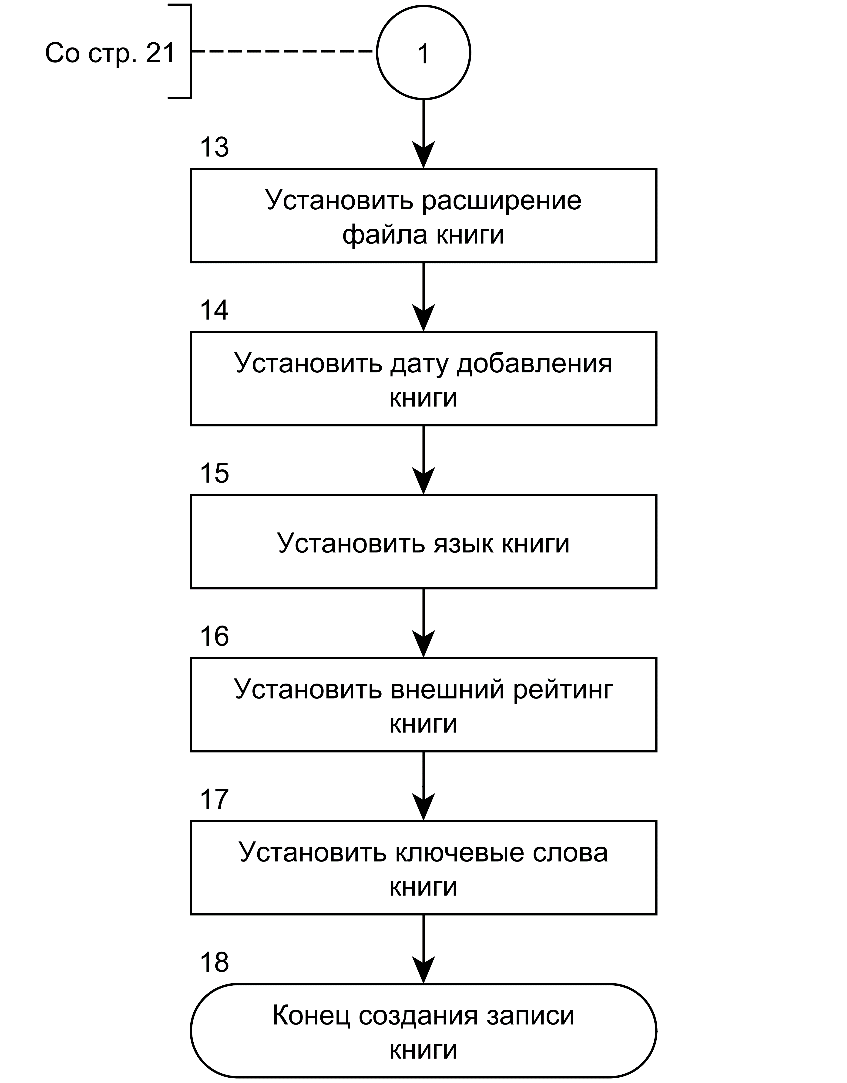


Рисунок 5.3 – Окончание инициализации записи книги

Таблица 5.3 – Свойства, агрегированные в классе BookEntity

|  |  |  |
| --- | --- | --- |
| Тип данных | Имя свойства | Соответствующее имя поля |
| Authors | authors | AUTHOR |
| Genres | genres | GENRE |
| string | title | TITLE |
| string | seriesTitle | SERIES |
| int | numberInSeries | SERNO |
| string | filename | FILE |
| int | fileSize | SIZE |
| int | bookID | LIBID |
| bool | isDeleted | DEL |
| string | extension | EXT |
| string | dateAdded | DATE |
| string | language | LANG |
| string | bookRate | LIBRATE |
| string | keywords | KEYWORDS |
| string | archiveName | - |

**5.1.4** Для реализации возможности наличия у одной книги нескольких авторов были реализованы классы Author, агрегирующий поля имени, фамилии, отчества, а также (для упрощения поиска автора по строке, которая может содержать как имя, так и фамилию или отчество или их часть) полного имени, и Authors, агрегирующий список типа List<Author>.

Алгоритм получения списка авторов книг представлен на рисунке 5.4.

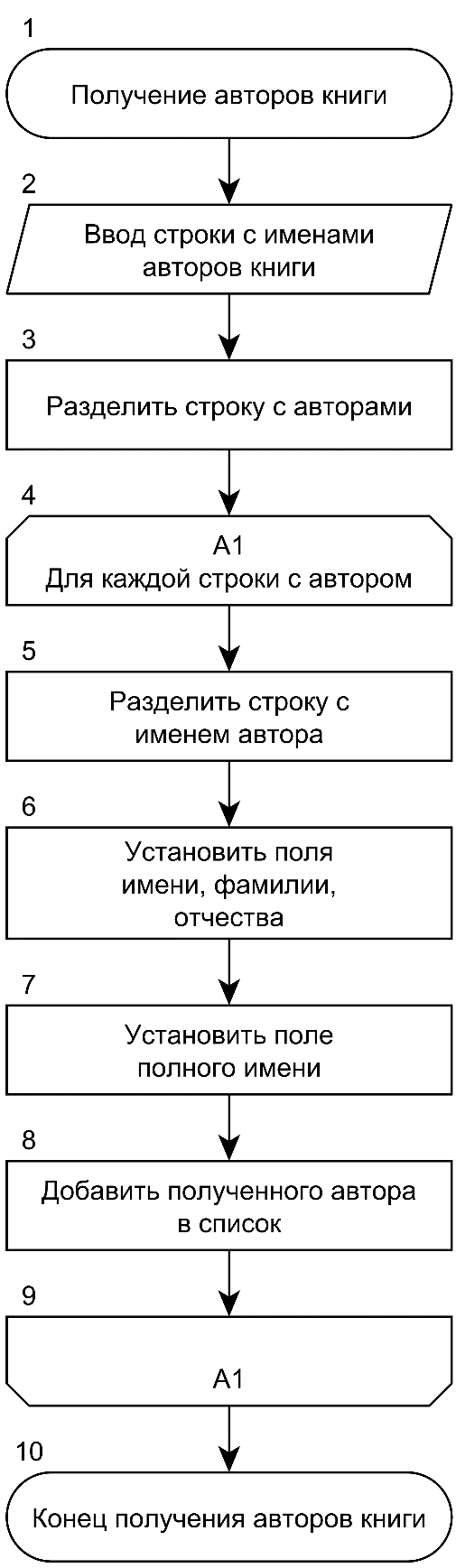


Рисунок 5.4 – Получение авторов книги

**5.1.5** Для обеспечения наличия у одной книги нескольких жанров был разработан класс Genres. Он агрегирует список типа List<int>. В данном списке содержатся номера жанров, а для получения имен и описаний жанров следует обратиться к классу GenresList. Как было сказано в пункте 5.1.1, данное решение позволило оптимизировать расход памяти и ускорить время поиска жанров.

Алгоритм получения жанров представлен на рисунке 5.5.

Предполагается, что все жанры, представленные в файле метаданных известны заранее и представлены в особом списке жанров. Тем не менее, нельзя исключать наличие ошибок, полученных при заполнении характеристик книг. Поэтому, при обнаружении жанра, имя которого не содержится в списке жанров, такой жанр игнорируется.

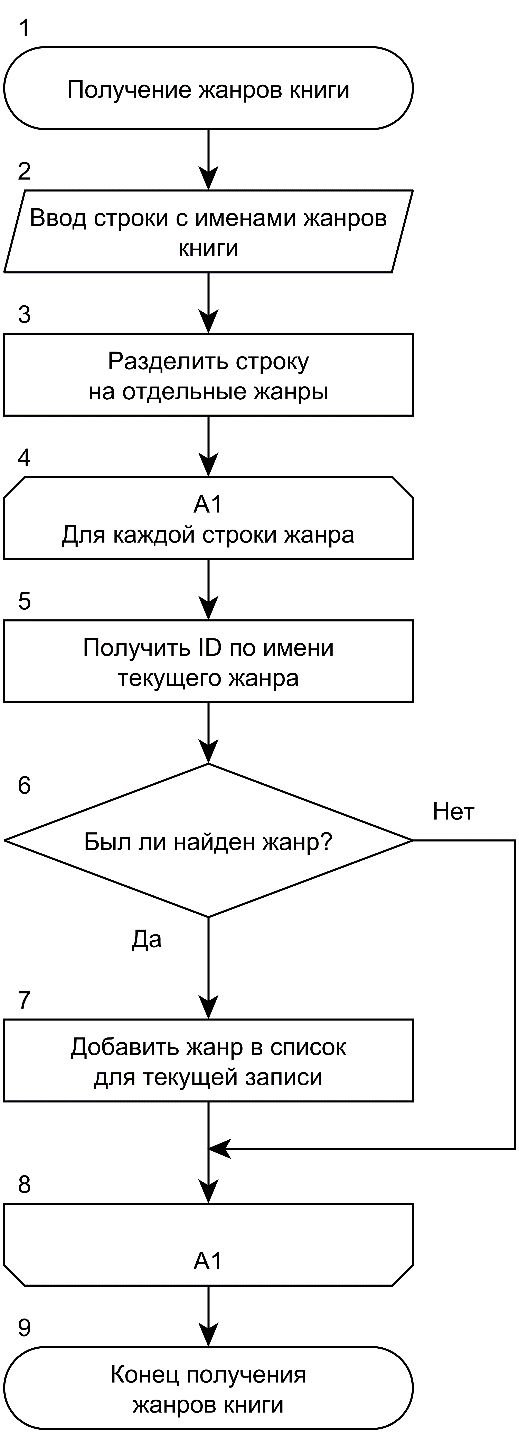


Рисунок 5.5 – Получение жанров книги

**5.1.6** Предположим, дан список записей книг типа List<BookEntity>, и необходимо извлечь какие-либо данные из этого списка. Для этих целей был разработан статический класс MetadataQuery. Методы, агрегированные в данном классе, представлены в таблице 5.4.

Таблица 5.4 – Методы класса MetadataQuery

|  |  |  |
| --- | --- | --- |
| Тип возвращаемого значений | Название | Описание |
| List<BookEntity> | selectBooksByAuthor | Возвращает все книги, принадлежащие всем авторам, в полном имени которых содержится строка для поиска |
| List<BookEntity> | selectBooksByTitle | Возвращает книги, в названии которых содержится строка для поиска |
| List<BookEntity> | selectBooksByGenre | Возвращает все книги, принадлежащие жанру с определенным ID |
| List<BookEntity> | selectBooksByGenres | Возвращает книги, лежащие на пересечении искомых жанров |
| BookEntity | selectBookByID | Возвращает книгу по ее ID |
| List<BookEntity> | selectBooksByTemplate | Производит отсеивание книг списка в соответствие с полями шаблона |

Основной и самый универсальный метод, который будет использоваться при поиске – selectBooksByTemplate. Данный метод производит последовательное отсеивание книг по критериям: имя автора, название, жанр.

**5.1.7** В качестве декоратора для класса MetadataQuery был реализован класс MetadataList. Он содержит поле типа List<BookEntity>, которое является основным хранилищем метаинформации книг в программном средстве. В данном классе определены методы обращения к метаинформации, реализованные как более общие в классе MetadataQuery.

Перед тем, чтобы использовать методы данного класса, его необходимо проинициализировать. Для этого необходимо вызвать метод import класса InpxImport, который возвращает список типа List<BookEntity>, полученный в результате импорта inpx файла. Реализацию данного класса приведена в пункте 5.1.8.

**5.1.8** Импорт метаданных инкапсулирован в статическом классе InpxImport. Алгоритм импорта представлен на рисунке 5.6.

Данный алгоритм на каждой итерации считывает из файла метаданных строку и создает запись книги. Стоит отметить, что поле имени архива устанавливается в данном алгоритме, а не при создании записи книги.

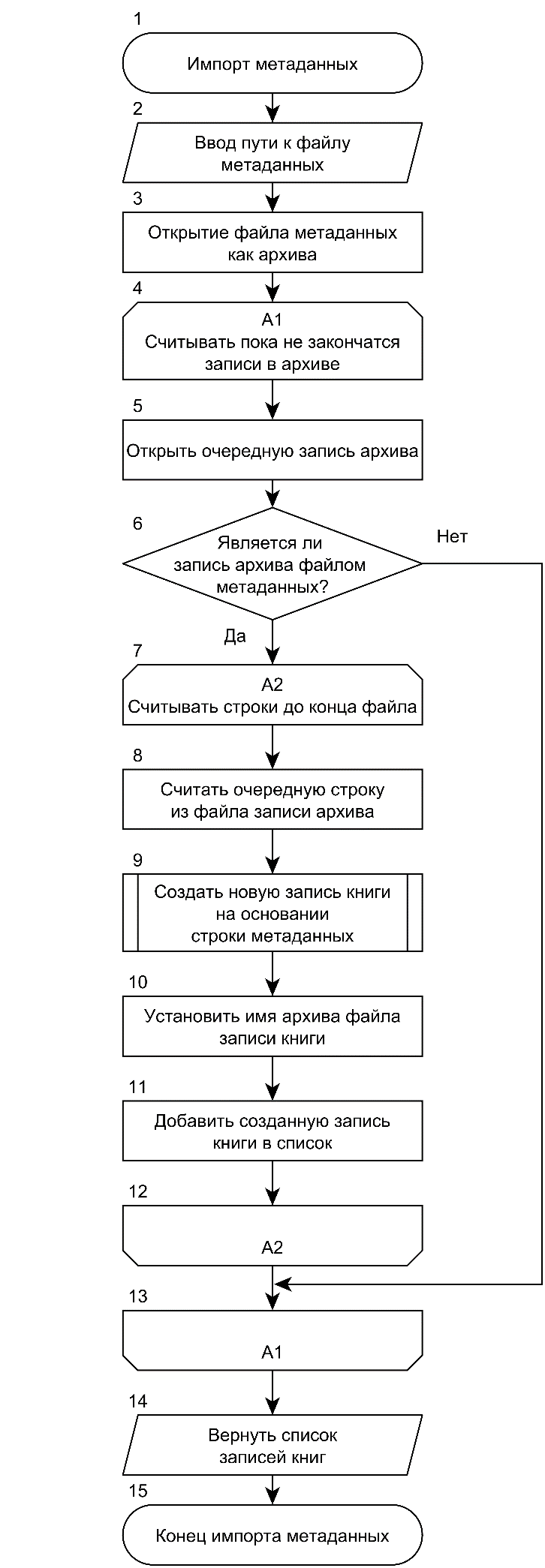


Рисунок 5.6 – Импорт метаданных

## Разработка серверного программного средства

**5.2.1** Сервер не требует активного взаимодействия с пользователем во время своей работы. При запуске пользователь осуществляет ввод необходимы данных, а затем сервис автоматически принимает и обрабатывает запросы от клиентов.

В связи с этим целесообразно реализовать сервер в виде консольного приложения, на котором обеспечивается хостинг (данный вид хостинг называется автохостингом) и работа веб-сервиса.

**5.2.2** Для начала необходимо совершить инициализацию сервера. Алгоритм инициализации представлен на графическом материале ГУИР.351006-001 ПД.

Инициализация состоит из трёх частей: инициализация списка жанров (см. пункт 5.1.1), инициализация метаданных (см. пункт 5.1.8), инициализация хранилища книг (см. пункт 5.1.2).

**5.2.3** В связи с тем, что в пункте 4.2.4 было запланировано применения технологий RPC, при реализации серверного программного средства было решено применить технологию WCF (Windows Communication Foundation). Данная технология предоставляет все возможности для построения сервис-ориентированных программных средств.

Создание веб-сервиса с использованием WCF базируется на задании трех базовых частей: address (адреса сервиса), binding (привязки сервиса), contract (контракт данных для взаимодействия).

**5.2.4** Для реализации сервиса была выбрана привязка NetTcpBinding. Она обеспечивает взаимодействие программных средств, написанных с ипользованием технологий .Net и WCF по бинарному протоколу передачи данных TCP. Данная привязка наиболее проста в использовании и обеспечивает максимальную скорость обработки (в отличие, например, от привязок, основанных на HTTP протоколе, так как в них передача данных осуществляется в виде текста, а значит необходимы преобразования данных при передаче и приёме).

**5.2.5** Один компьютер может обладать несколькими сетевыми картами, и, следовательно, находиться в нескольких сетях и обладать несколькими IP-адресами. Для того, чтобы сервис был доступен из всех подключенных к нему сетей, необходимо в качестве адреса при создании конечной точки указать имя хоста компьютера.

Действия по установки привязки и адреса производятся в блоке 16 на схеме алгоритма, представленной на графическом материале ГУИР.351006-001 ПД.

**5.2.6** Для обеспечения взаимодействия сервиса и его клиентов необходимо также реализовать контракт взаимодействия, который представляет собой некий интерфейс. Данный интерфейс необходимо сделать доступным и сервису, и клиентам. Однако, данный интерфейс необходимо реализовать только на сервисе. Клиенты же получают доступ к реализации с помощью технологии WCF – происходит удаленный вызов процедур.

**5.2.7** В качестве контракта был реализован интерфейс IService. Методы данного интерфейса представлены в таблице 5.5.

Данный интерфейс реализован в классе Service. Экземпляры данного класса создаются при подключении клиентов к сервису. По сути, данный класс является заместителем, перенаправляющим запросы от клиентов к классу MetadataList. Также, для предоставления пользователю возможности просмотра подключений и запросов от клиентов, данный класс обеспечивает отображение подобной статистической информации.

Таблица 5.5 – Интерфейс контракта взаимодействия IService

|  |  |  |
| --- | --- | --- |
| Возвращаемое значение | Имя метода | Описание |
| List<BookEntity> | selectBooksByAuthor | Получение списка книг на основании имени или любой части имени автора |
| List<BookEntity> | selectBooksByTitle | Получение списка книг на основании названия или части названия книги |
| List<BookEntity> | selectBooksByGenre | Получение списка книг, относящихся к требуемому жанру |
| List<BookEntity> | selectBooksByGenres | Получение списка книг, принадлежащих одновременно всем искомым жанрам |
| BookEntity | selectBookByID | Получение записи книги на основании ее ID |
| List<BookEntity> | selectBooksByTemplate | Получение списка книг, соответствующих шаблону |
| Stream | extractBook | Извлечение файла книги по ее записи |
| Stream | extractBookByID | Извлечение файла книги по ее ID |
| List<BookEntity> | getAvailableGenres | Получение списка доступных жанров |

**5.2.8** Инициализация сервера входными данными производится при каждом его запуске. Пользователю, осуществляющему администрирование сервера, необходимо каждый раз заново вводить пути к файлам входных данных. Однако, для удобства пользователя, в серверном программном средстве предусмотрена возможность сохранения следующих настроек: пути файла жанров, файла метаданных, хранилища книг. Данные настройки сохраняются в папке пользователя и автоматически используются при запусках сервера.

Также была реализована возможность сброса всех настроек для их повторной установки. Стоит отметить, что программное средство запросит повторного ввода инициализирующих параметров в случае их отсутствия или произошедшей ошибке при инициализации.

## Разработка клиентского программного средства

**5.3.1** Основное назначение клиента – предоставление удобного интерфейса для получения информации о книгах и самих книг. Поэтому главная задача при его реализации – создание графического интерфейса согласно макету, представленному на рисунке 4.1 в пункте 4.3.1. Реализованный интерфейс является масштабируемым и легко приспосабливается под практически любые размеры экранов.

**5.3.2** Для подключения клиента к серверу необходимо, чтобы пользователь указал адрес сервера, на котором размещен сервис электронной библиотеки.

Еще один параметр, который нужно указать при запуске приложения – это путь папки, в которую будут помещаться скачиваемые с сервера электронные книги.

**5.3.3** Графический интерфейс предоставляет возможность поиска по жанру. Однако, для этого на клиенте и сервере необходимо наличие одинакового списка жанров. Поэтому при запуске клиента посылается запрос серверу на получение доступного списка жанров. Затем этот список используется для выбора жанров.

Данный запрос выполняет также функцию проверки правильности адреса сервера, указанного клиентом, и проверки доступности сервера: если получить список жанров не удастся, то будет предложено ввести другой адрес сервера.

**5.3.4** Предположим, что сервер и клиент запущены в одной локальной сети. Пользователю необходимо узнать адрес сервера, чтобы использовать программное средство. Однако это может быть довольно сложной задачей.

Для решения данной проблемы была реализована возможность автообнаружения сервиса клиентом, предоставляемая технологией WCF. Суть ее в следующем: сервис постоянно широковещательно рассылает UDP пакеты с адресом и контрактом сервиса. На клиенте запускается специальный слушатель, который получает пакеты такого типа. Если клиент обнаруживает, что в сети действует подходящий сервис, то, значит, что сервер найден, можно получить его адрес и установить соединение.

**5.3.5** Как уже было сказано выше, основной применяемый запрос для поиска на клиенте – это вызов метода selectBooksByTemplate. Для его вызова необходимо передать экземпляр класса BookEntity. Причем, большинство полей может быть не заполнено. Этот случай означает, что по данному полю подходят все книги и отсеивание не производится.

**5.3.6** Для получения файла книги используется метод extractBook, которому необходимо передать экземпляр класса BookEntity. Данный метод эффективнее, чем метод extractBookById, так как позволяет эффективнее использовать ресурсы сервера, что связано с тем, что нет необходимости на сервере вызывать методы получения записи книги по ее ID.

# Руководство по использованию

# Заключение

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

|  |  |
| --- | --- |
| [1] Википедия, Сжатие данных [Электронный ресурс]. - Электронные данные. - Режим доступа: http://ru.wikipedia.org/wiki/Сжатие%20данных.. | |
| [2] A Universal Algorithm for Sequential Data Compression / J. Ziv, A. Lempel // IEEE Transactions of Information Theory. - 1977. - Т.IT-23, № 3, pp. 337-343. | |
| [3] Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео: учебно-справочное издание / Д. Ватолин [и др.] - М: ДИАЛОГ-МИФИ, 2003. - 384 с. | |
| [4] Data Compression. The Complete Reference. Fourth Edition. / D. Salomon - London: Springer-Verlag, 2007. | |
| [5] Википедия, Кодирование длин серий, [Электронный ресурс]. - Электронные данные. - Режим доступа: http://ru.wikipedia.org/wiki/RLE. |
| [6] A corpus for the evaluation of lossless compression algorithms / A. Ross, Bell T. - Christchurch: Department of Computer Science, University of Canterbury, | |
| [7] Text Compression / T. C. Bell, J. G. Cleary, I. H. Witten. - Englewood Cliffs: Prentince Hall, 1990. | |
| [8] MSDN, BackgroundWorker - класс [Электронный ресурс]. - Электронные данные. - Режим доступа: http://msdn.microsoft.com/ru-ru/library/system.componentmodel.backgroundworker(v=vs.110).aspx. | |
| [9] Large Corpus [Электронный ресурс]. - Электронные данные. - Режим доступа: http://corpus.canterbury.ac.nz/resources/large.zip. | |
| [10] ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. - Введ. 01.01.1992, М: Изд-во стандартов, 1991. |
| [11] Доманов, А.Т. Стандарт предприятия. Дипломные проекты (работы). Общие требования / А. Т. Доманов, Н. И. Сорока - Минск: БГУИР, 2010. |

# ПРИЛОЖЕНИЕ А (обязательное) Исходные коды методов

1. 1) Информационный взрыв – постоянное увеличение скорости и объёмов публикаций (объёма информации) в масштабах планеты [5]. [↑](#footnote-ref-1)