Институт Транспорта и связи

Факультет компьютерных наук

Курсовой проект

по теме УППРАВЛЕНИЕ БИБЛИОТЕКОЙ
по дисциплине ТЕХНОЛОГИИ КОНСТРУИРОВАНИЯ ПО

Студентов группы номер 4003 BD
Гулиева Дмитрия
Чернышева Олега

Проверил _____

Дата_____

Подпись_____

Рига 2004

СОДЕРЖАНИЕ

Цель работы

Целью данной работы является разработка программного объектно-ориентированного обеспечения для библиотеки в соответствии с требованиями, предъявляемыми к унифицированному процессу разработки программного обеспечения.

Предметная область

В рамках данной работы подразумевается разработка программного обеспечения, устанавливаемого на персональный компьютер управляющего библиотекой. В библиотеке осуществляется регистрация всех читателей в базе данных, в которой может храниться следующая информация о читателе:

- фамилия, имя читателя;
- телефон читателя;
- адрес проживания читателя;
- персональный код читателя;

информация о книге:

- номер книги;
- название;
- имя и фамилия автора;

Разрабатываемая система должна обеспечить добавление, удаление, редактирование и поиск записей в базе данных.

ЭТАП «НАЧАЛО»

Определим набор требований, предъявляемых к разрабатываемому продукту, после первой встречи с заказчиком.

Программный продукт представляет собой СУБД предназначенную для регистрации книг читателей библиотеки. Данная система должна выполнять следующие основные функции:

1. Работа с информацией о книге;
2. Возможность работы с информацией о читателе.

Идентификация актёров и вариантов использования:

В соответствии с поставленными требованиями, определим актера, фиксирующего роли внешних объектов, взаимодействующего с системой.

Описание актёров:

Актёр **Librarian** – описывает пользователя работающего с базой данной. Для актера Librarian определим соответствующие USE-CASE'ы, путем рассмотрения актера и его взаимодействия с системой. На рис.1 представлена начальная Use Case диаграмма, построенная на основании требований заказчика.

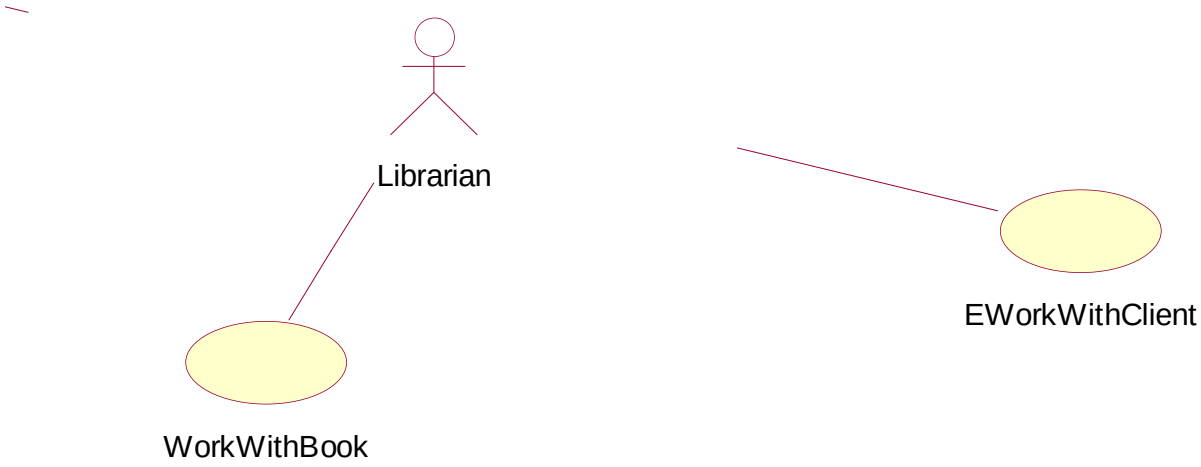


Рис. 1 Начальная Use Case диаграмма (20% от полного представления)

Описание Use Case'ов:

Актёр Librarian инициирует следующие USE-CASE'ы:

- **WorkWithBook** - представляет собой функциональную возможность системы для работы с книгами.
- **WorkWithClient** – предоставляет собой функциональную возможность системы для работы с читателями.

ЭТАП «РАЗВИТИЕ»

На данном этапе разрабатываются сценарии работы программной системы для каждого из Use-case'ов, строятся диаграммы последовательности для каждого из сценариев, на основании объектов, входящих в диаграммы последовательности, определяются их абстракции – классы и осуществляется планирование итераций этапа Конструирование.

На этом этапе, после разговора с заказчиком возникла необходимость дополнительных функциональных возможностей системы. В итоге был определён дополнительный набор требований, предъявляемых к разрабатываемому продукту:
Разрабатываемая СУБД должна при работе с информацией о читателе обеспечить следующие функциональные возможности:

- добавление сведений о читателе;
- удаление сведений о читателе;
- редактирование сведений о читателе;
- поиск сведений о читателе;
- выдача и возврат книги.

А при работе с информацией о книге:

- добавление сведений о книгах;
- удаление сведений о книгах;
- редактирование сведений о книгах;
- поиск сведений о книгах.

В соответствии с уточненными требованиями, опишем дополнительные функциональные возможности системы:

Описание Use Case'ов:

- **AddClient** – функция добавления в базу данных нового читателя.
- **SearchClient** – функция поиска записи о клиенте в базе данных по заданному критерию:
  - Фамилия
  - Имя
  - Код
  - Телефон
- **EditClient** – функция редактирования записей в базе данных о клиентах.
- **RegBookClient** – функция управления информацией о книге, дата выдачи, дата возврата, кем была выдана книга.
- **DleteClient**- функция удаления записи в базе данных о клиенте.

- **AddBook** – функция добавления в базу данных новой книги.
- **Search book** – функция поиска записи в базе данных по заданному критерию:
    - Названию
    - Имени
    - Фамилии автора
    - Номеру
    - Кто взял
- **EditBook** – функция редактирования записей в базе данных.

В итоге на этапе развитие конечная Use Case диаграмма примет вид (рис.2):

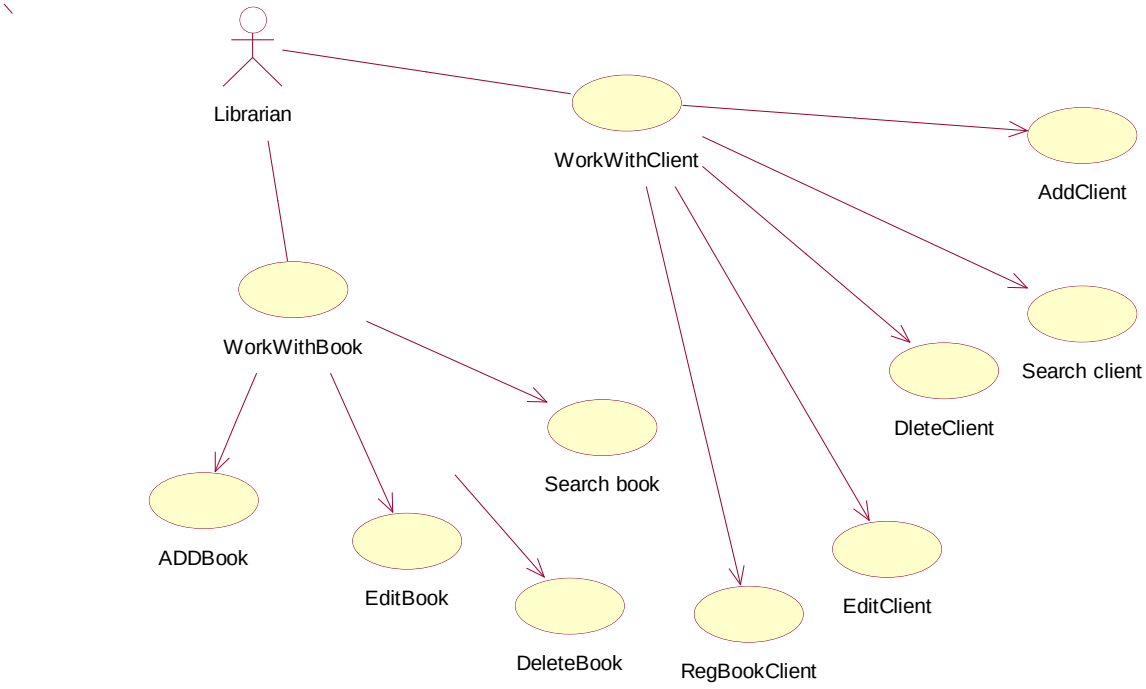


рис.2 Конечная Use Case диаграмма (80% от полного представления)

Определение сценариев:

- Сценарий для Use-Case'а WorkWithbook :

UseCase WorkWithbook стоновится доступным пользователю после нажатия на кнопку Add\Delete book на главной форме.

- Сценарий для Use-Case'а WorkWitClient:

UseCase WorkWithbook стоновится доступным пользователю после нажатия на кнопку WorkWitClient на главной форме.

- Сценарий для Use-Case’a Search client:

При нажатии на кнопку Search, главное окно становится недоступным, пользователь попадает в окно режима поиска.

- Сценарий для Use-Case’a EditClient:

При нажатии на кнопку Change на главном окне, главное окно становится недоступным, пользователь попадает в окно редактирования. Пользователь выбирает поля для редактирования, в которых необходимо произвести изменения, нажимает на кнопку Aplply, затем окно редактирования закрывается, главное окно становится доступным.

- Сценарий для Use-Case’ADDClient:

При нажатии на кнопку ADD на главном окне, главное окно становится недоступным, пользователь попадает в окно добавления. Пользователь заполняет поля для ввода, и нажимает на кнопку ADD, затем окно редактирования закрывается, главное окно становится доступным.

- Сценарий для Use-Case’DleteClient:

При нажатии на кнопку Delete, выдаётся сообщение о подтверждении об удалении записи о клиенте, в случае подтверждения, происходит удаление, в противном случае закрывается диалоговое окно.

- Сценарий для Use-Case’a BookRegClient:

При нажатии на кнопку RegBook на главном окне, главное окно становится недоступным, пользователь попадает в окно редактирования. Пользователь выбирает поля для редактирования, в которых необходимо произвести изменения, например дата, когда книга взята, то, что она отдана или изменить код книги и нажимает на кнопку Take или Give Back, затем окно редактирования закрывается, главное окно становится доступным.

- Сценарий для Use-Case’a EditBook:

При нажатии на кнопку Change на главном окне, главное окно становится недоступным, пользователь попадает в окно редактирования. Пользователь выбирает поля для редактирования, в которых необходимо произвести изменения, нажимает на

кнопку Aplply, затем окно редактирования закрывается, главное окно становится доступным.

- Сценарий для Use-Case'AddBook:

При нажатии на кнопку ADD на главном окне, главное окно становится недоступным, пользователь попадает в окно добавления. Пользователь заполняет поля для ввода, и нажимает на кнопку ADD, затем окно редактирования закрывается, главное окно становится доступным.

- Сценарий для Use-Case'a Search book:

При нажатии на кнопку Search на главном окне, главное окно становится недоступным, пользователь попадает в окно режиме поиска.

- Сценарий для Use-Case'Dlete book:

При нажатии на кнопку Delete, выдаётся сообщение о подтверждении об удалении записи о книге, в случае подтверждения, происходит удаление, в противном случае закрывается диалоговое окно.

Разработка диаграмм последовательности

Разработка диаграммы последовательности ведется отдельно для каждого Use-case'a.

Диаграмма последовательности для Use-Case'a WorkWithbook выглядит следующим образом (рис.3):

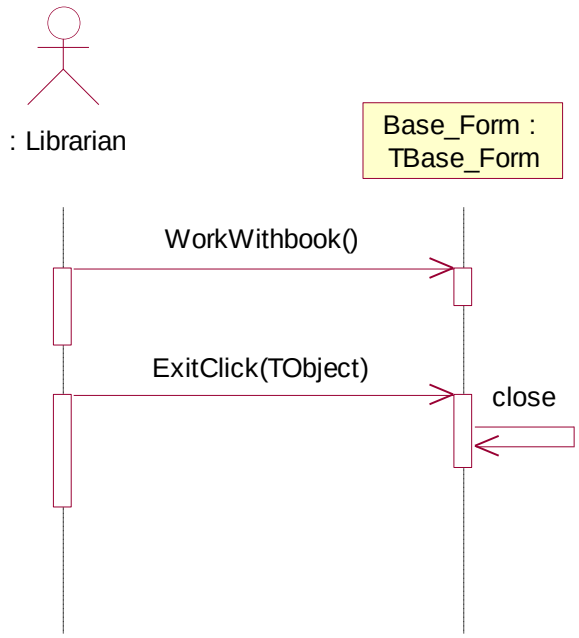


рис.3

Диаграмма последовательности для Use-Case’а WorkWitClient выглядит следующим образом (рис.4):

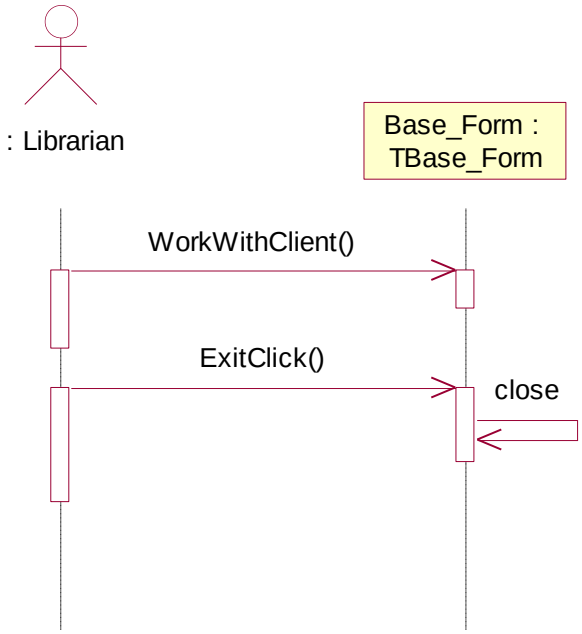


рис.4

Диаграмма последовательности для Use-Case’AddBook выглядит следующим образом (рис.5):

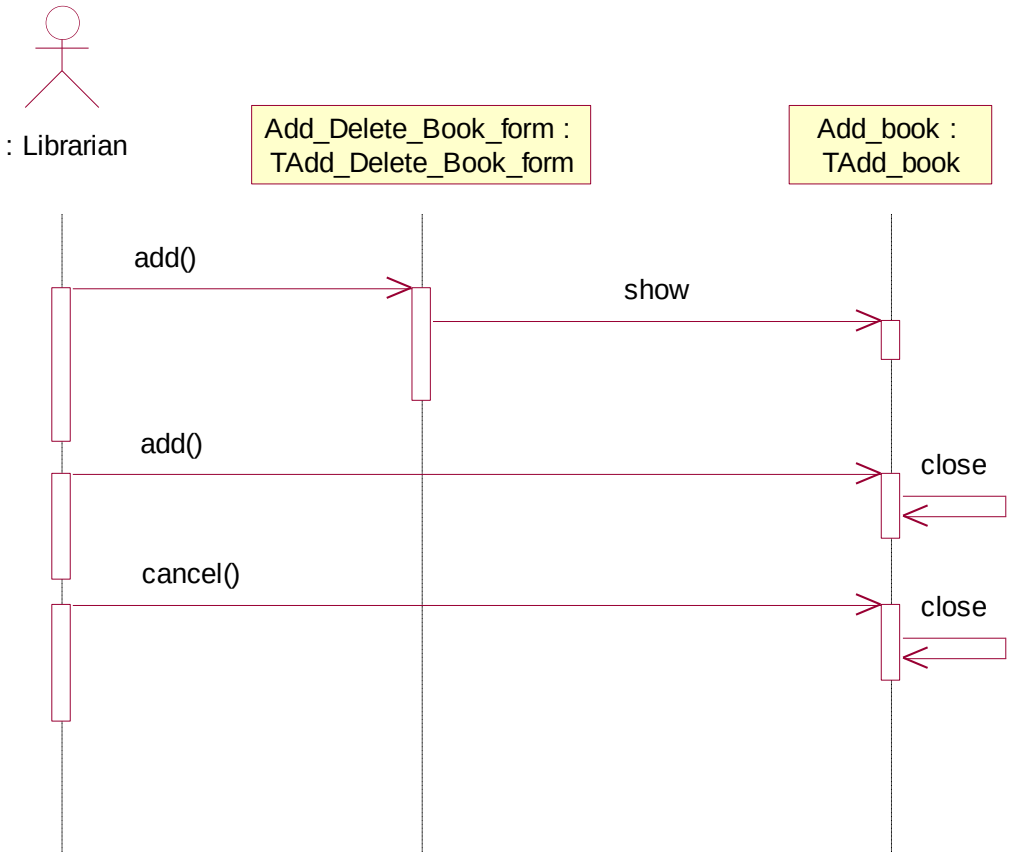


рис.5

Диаграмма последовательности для Use-Case’Dlete book выглядит следующим образом (рис.6):

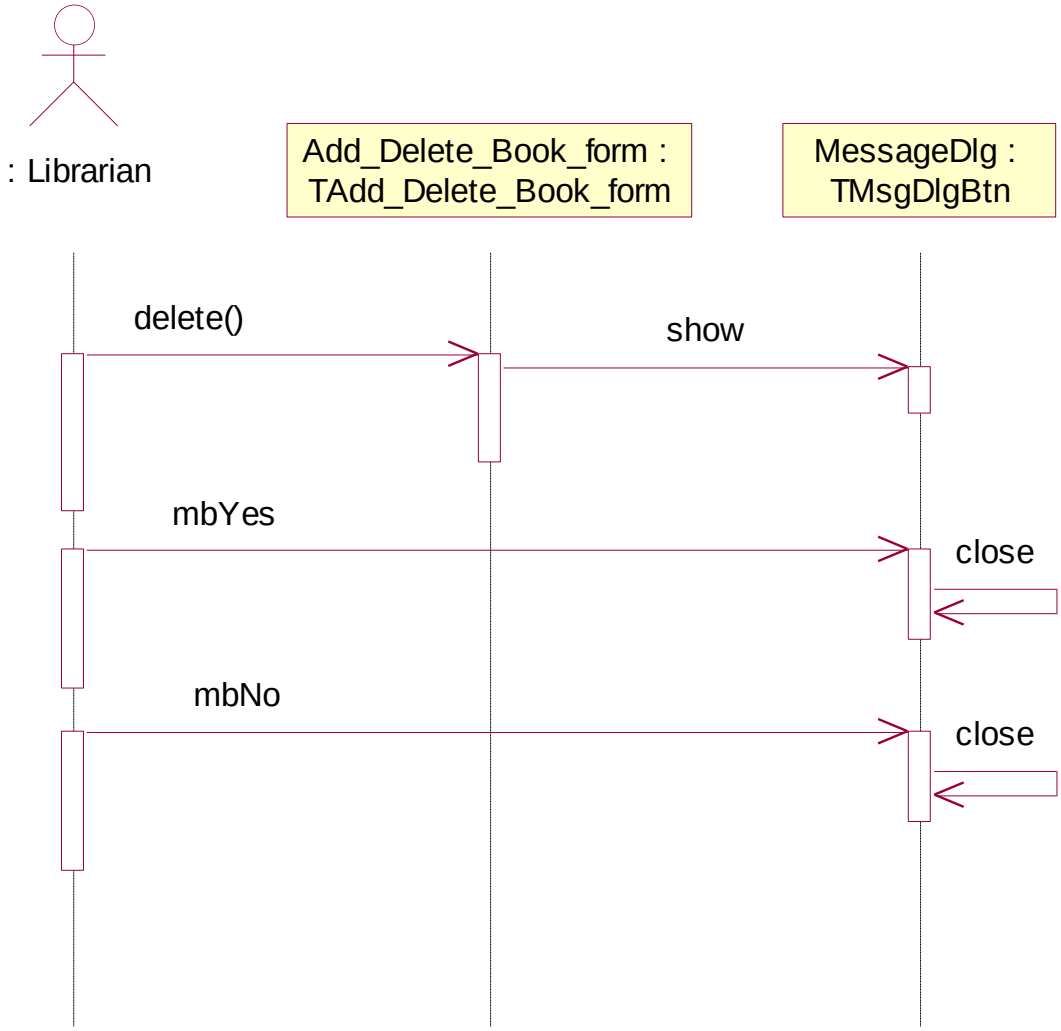


рис.6

Диаграмма последовательности для Use-Case’a EditBook выглядит следующим образом (рис.7):

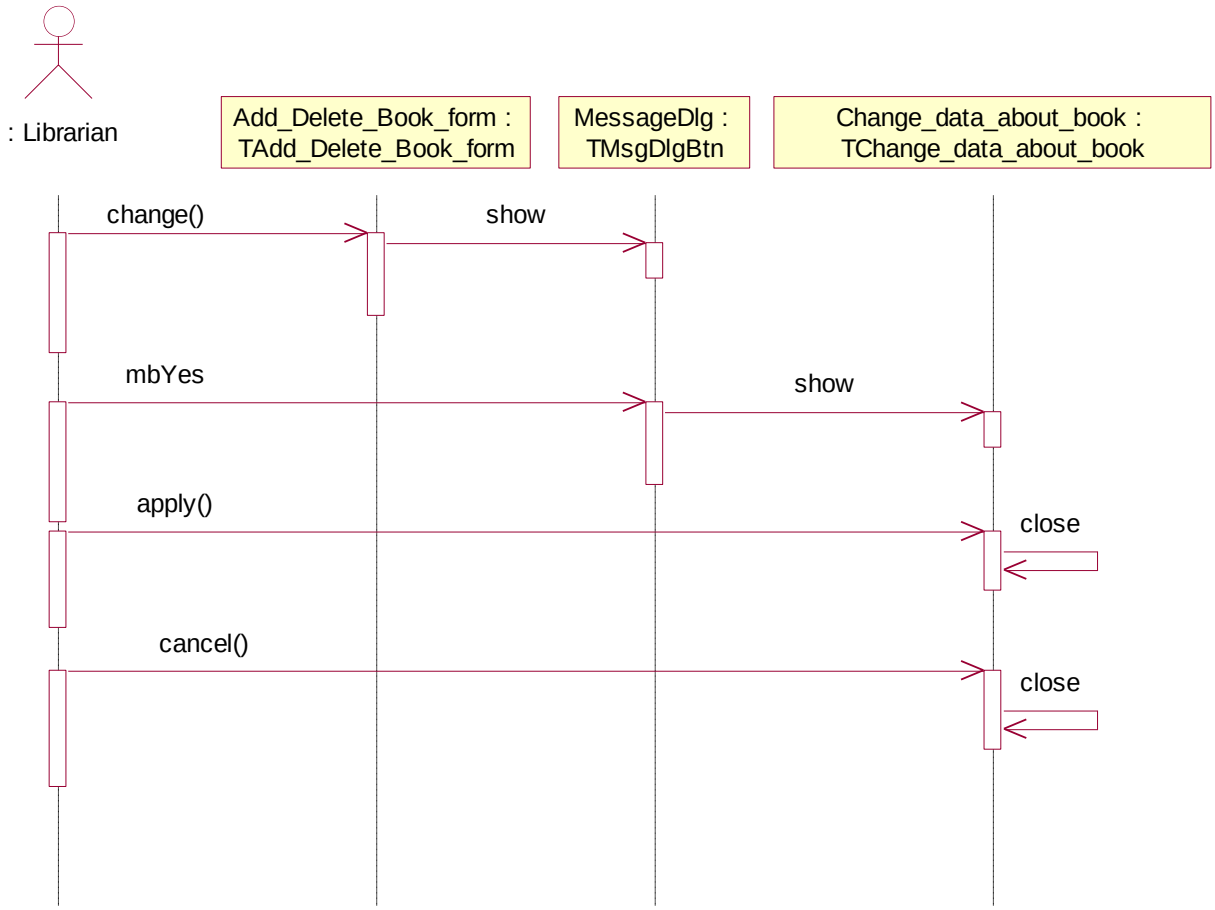


рис.7

Диаграмма последовательности для Case'a Search book выглядит следующим образом (рис.8):

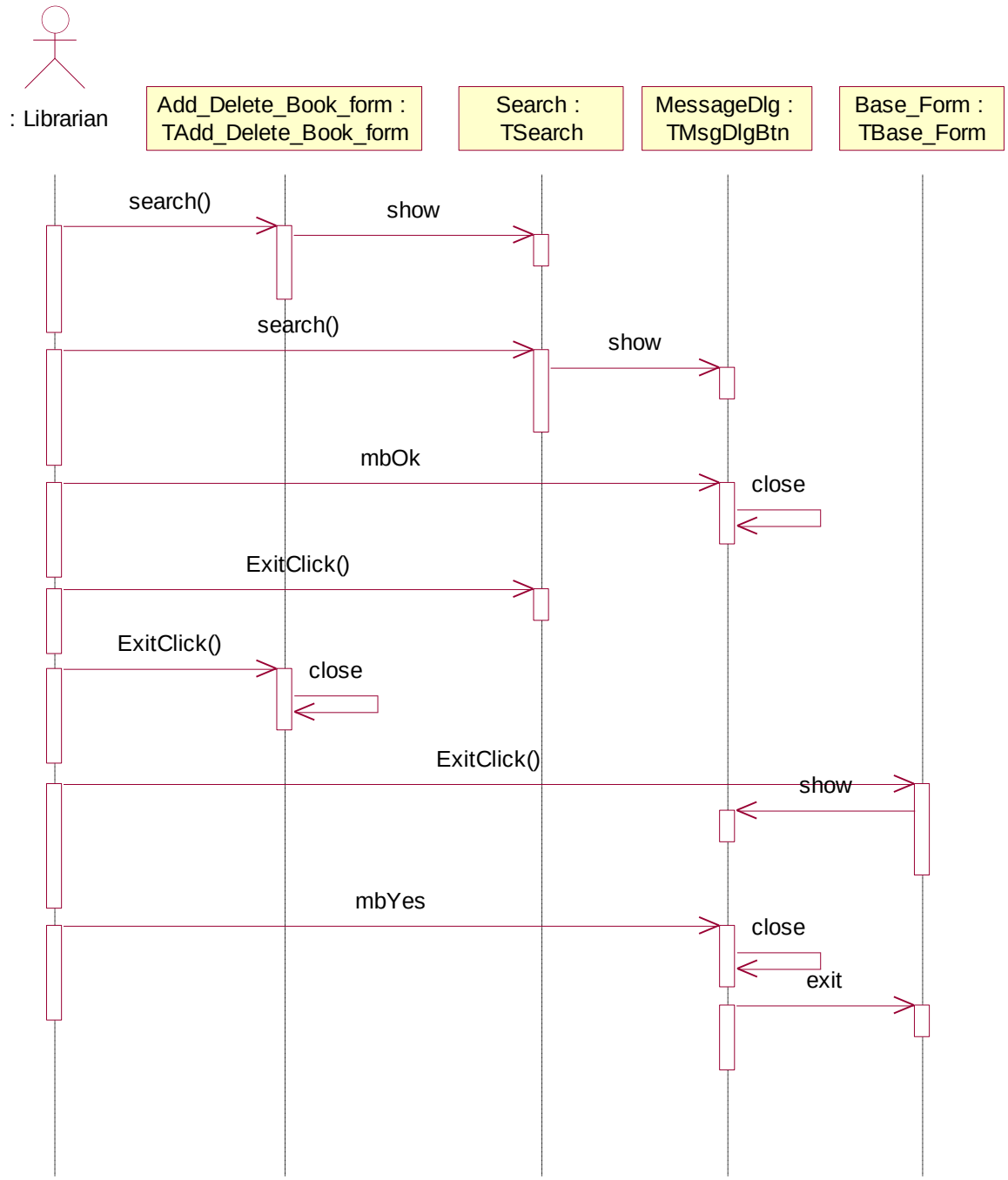


рис.8

Диаграмма последовательности для Use-Case’ADDClient выглядит следующим образом (рис.9):

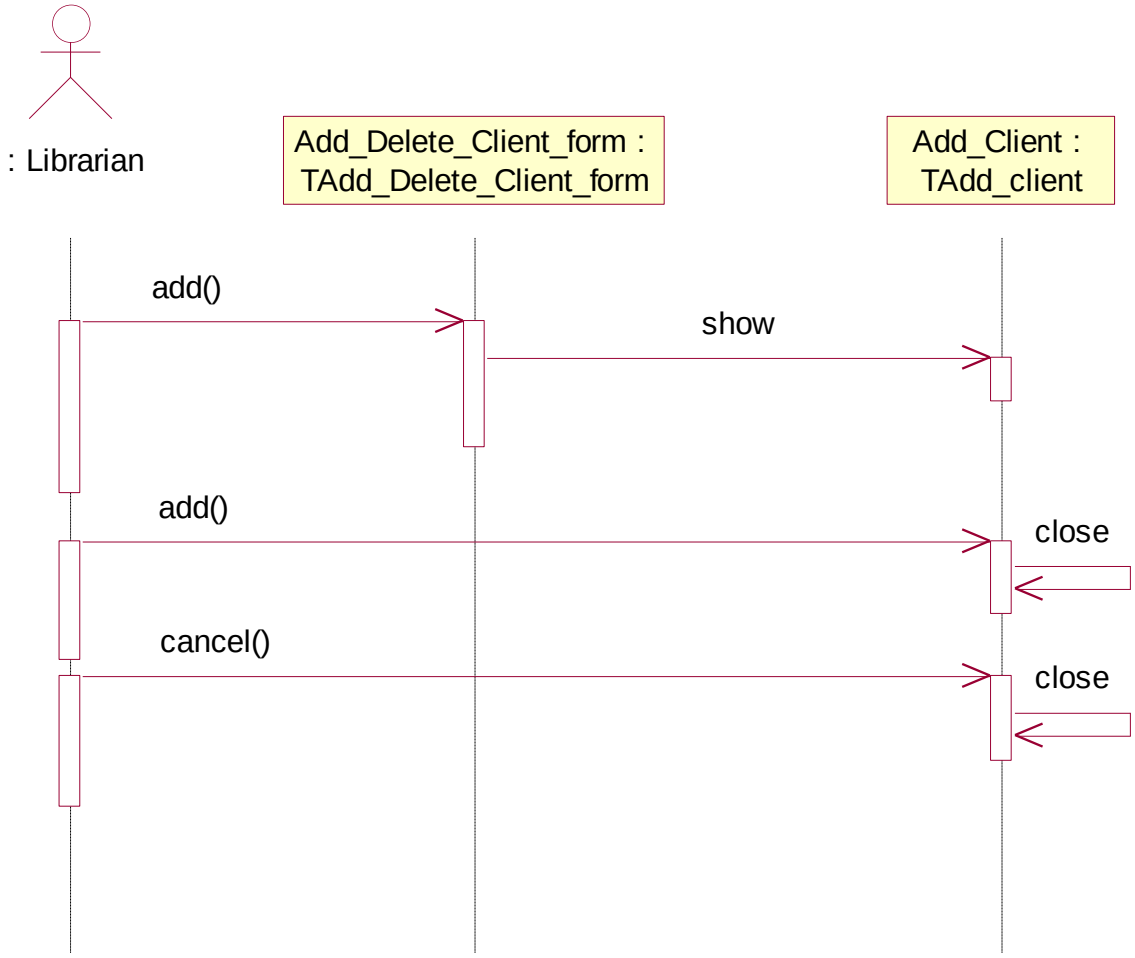


рис.9

Диаграмма последовательности для Use-Case’a EditClient выглядит следующим образом (рис.9):

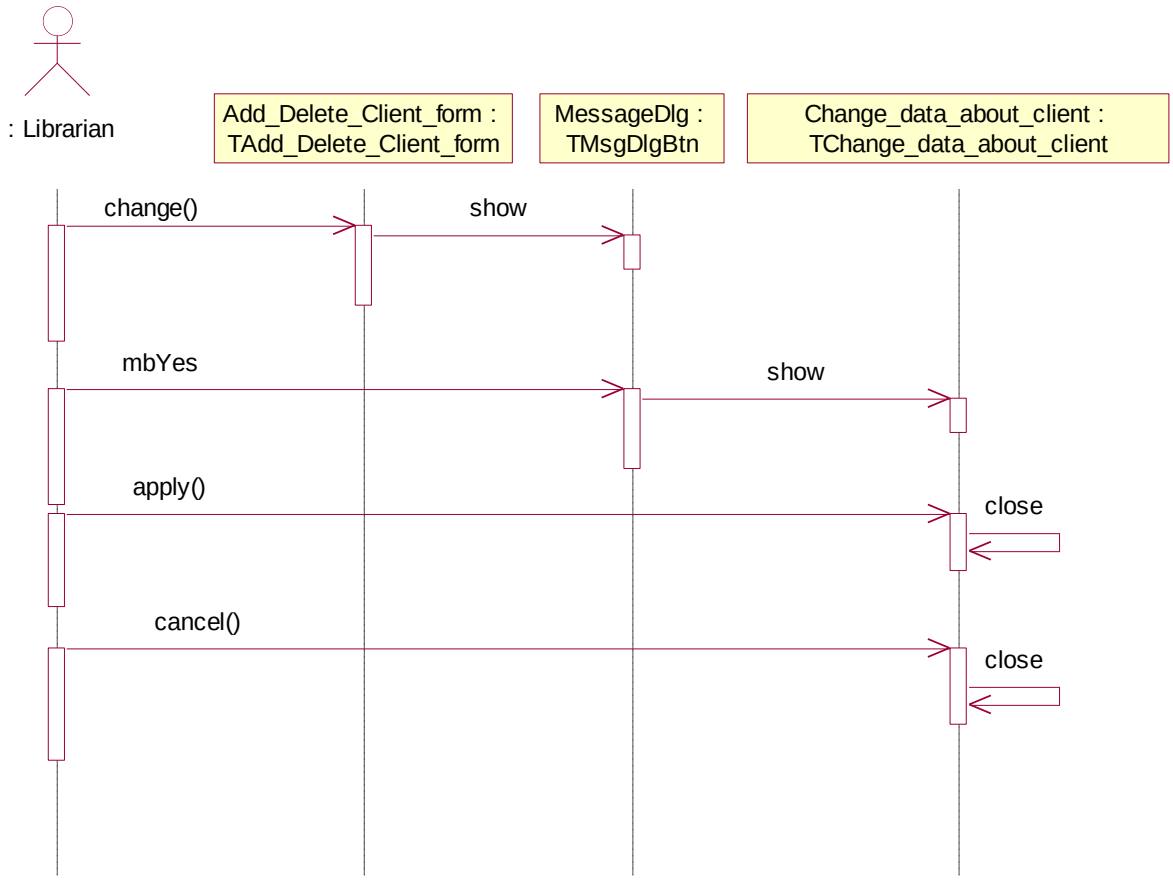


рис.9

Диаграмма последовательности для Use-Case’а DleteClient выглядит следующим образом (рис.10):

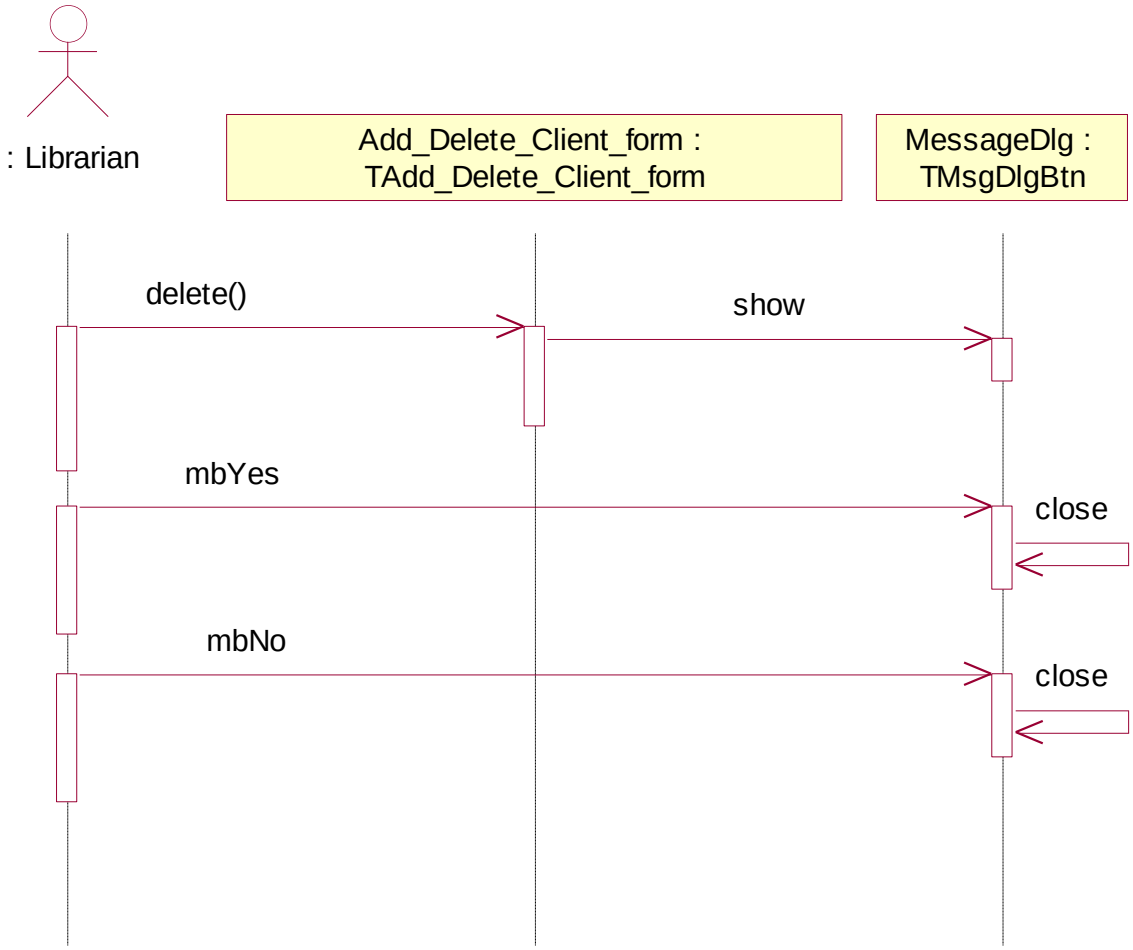


рис.10

Диаграмма последовательности для Use-Case’a Search client выглядит следующим образом (рис.11):

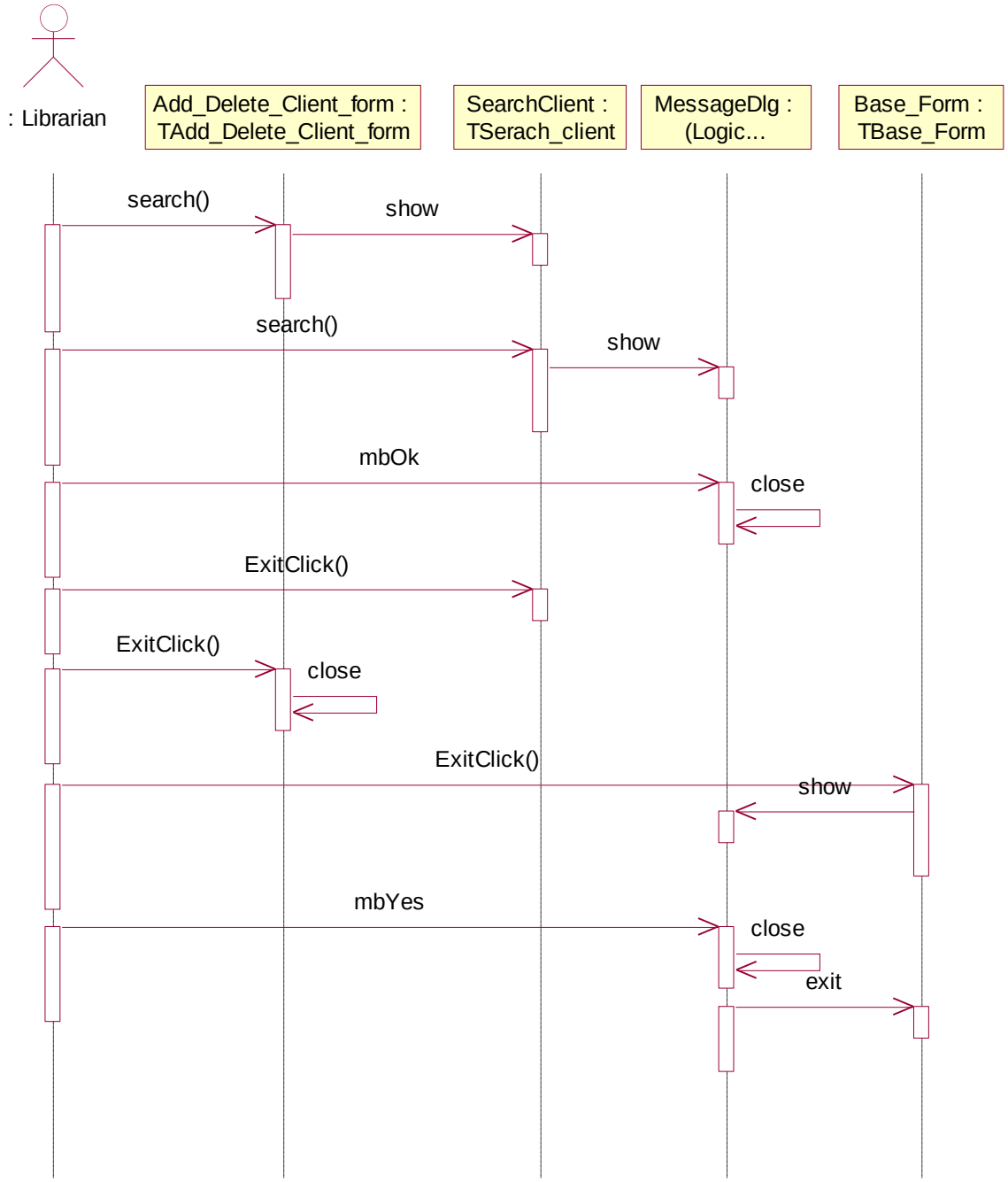


рис.11

Диаграмма последовательности для Use-Case’a BookRegClient выглядит следующим образом (рис.12):

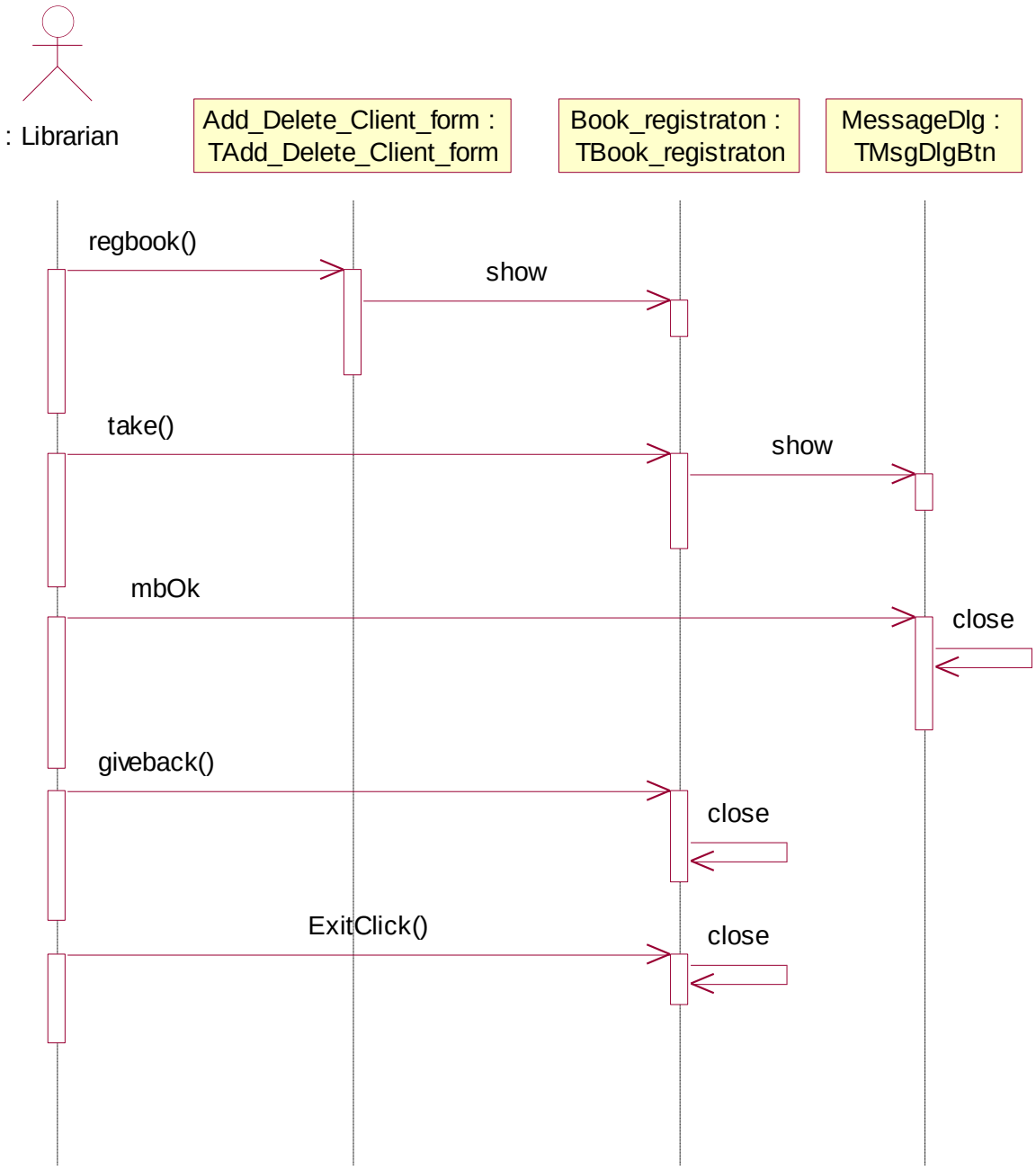


рис.12

Примечание: все представленные диаграммы последовательностей являются конечными.

Диаграмма классов

На основании разработанных диаграмм последовательностей возможно выделить следующие классы:

- **TBase_Form-**объектом класса является главная форма.
- **TAdd_Client -** объектом класса является форма, предназначенная для добавления информации о клиентах.
- **TAdd_book -** объектом класса является форма, предназначенная для добавления информации о книгах.
- **TAdd_Delete_Book_form -** объектом класса является форма, предназначенная для удаления записей о клиентах из БД.
- **TAdd_Delete_Client_form-** объектом класса является форма, предназначенная для удаления записей о клиентах из БД.
- **TChange_data_about_book -** объектом класса является форма, предназначенная для изменения записей о клиентах из БД.
- **TChange_data_about_client-** объектом класса является форма, предназначенная для изменения записей о клиентах из БД.
- **Tsearch- -** объектом класса является форма, предназначенная для поиска записей о книгах в БД.
- **TSearchClient -** объектом класса является форма, предназначенная для поиска записей о клиентах в БД.
- **TBook_registraton -** объектом класса является форма, предназначенная для выдачи/возврата книги клиентом.

Все создаваемые классы являются наследниками стандартного класса TForm языка Delphi.

Так же важно отметить, что возникла необходимость в создании следующих классов, имеющих стереотип запись:

- **Book**-класс содержащий такие атрибуты, как название, имя, фамилия автора книги и т.д.
- **User**-класс содержащий такие атрибуты, как имя, фамилия клиента и т.д.
- **Vzjat** - класс содержащий такие атрибуты, как идентификатор книги и выдача.

Планирование итераций конструирования

На данной стадии выполнения работ производится разбиение процесса конструирования на отдельные итерации с целью обеспечения возможности управления риском в процессе разработки, а также для определения очередности выполнения работ с точки зрения архитектурного построения программной системы. Обычно планирование итераций производится путем разбиения общего объема работ по Use-case'ам и, если требуется, далее по сценариям, входящим в отдельные Use-case'ы.

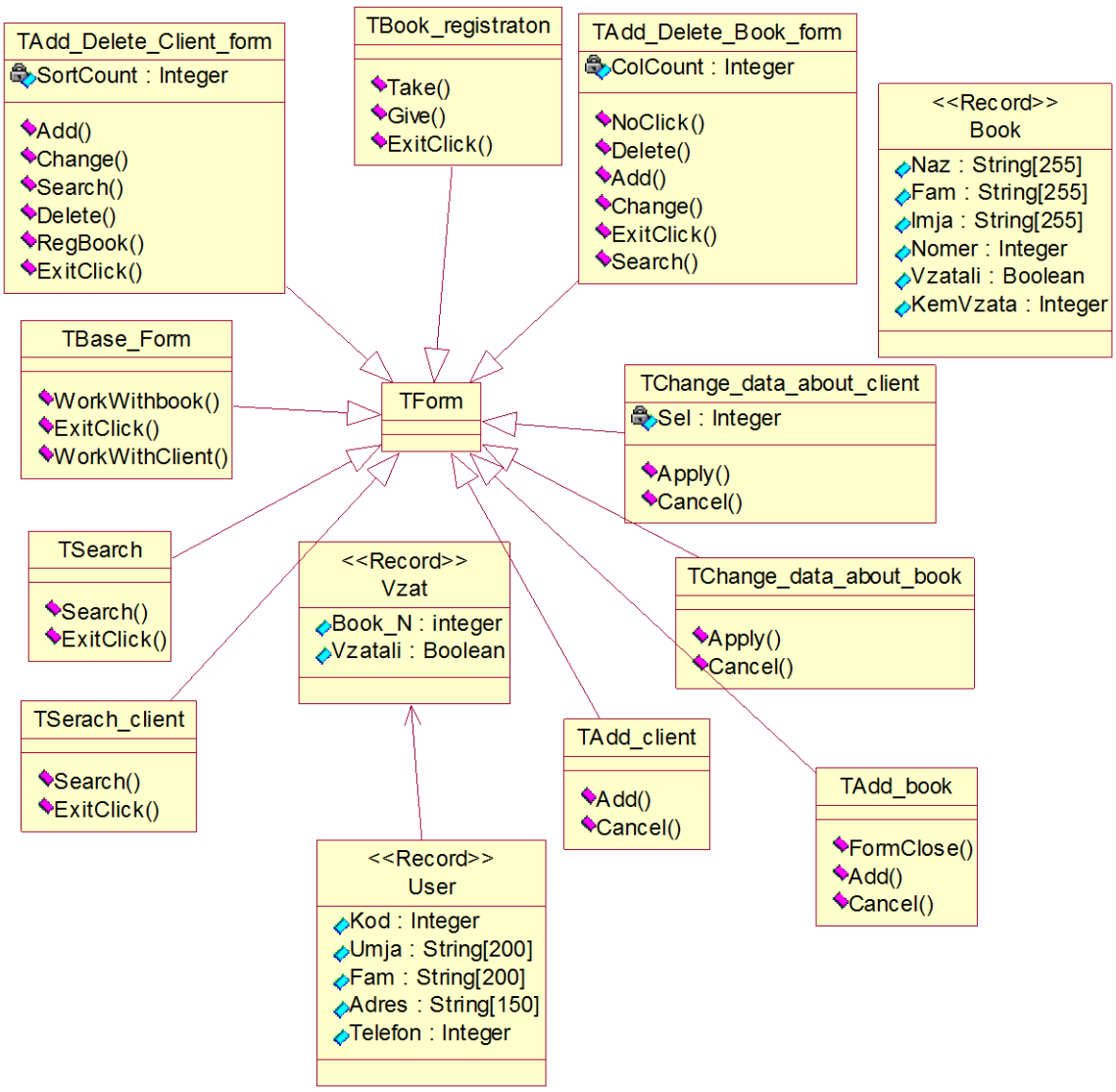


Рис. 12. Предварительная диаграмма классов

Приведем предварительную таблицу значений метрик, характеризующих качество разрабатываемой программной системы:

Метрика	TSerach_client	TSearch	TAdd_client	TAdd_book	TChange_data_about_book	TChange_data_about_client	TAdd_Delete_Book_form	TBook_registraton	TAdd_Delete_Client_form	TBase_Form	Book	Vzat	User	Среднее значение
WMC	2	2	2	3	2	2	6	3	6	3	0	0	0	3.1
NOC	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Метрики, вычисляемые для системы														
DIT	2													
NC	13													
NOM	31													

Составим начальный план итераций с учётом риска реализации use case’ов:

Итерация №1

1. Работа с книгой
2. Работа с читателем

Итерация №2

1. Добавление информации о книге
2. Удаление информации о книге
3. Поиск информации о книге
4. Редактирование информации о книге

Итерация №3

1. Добавление информации о книге
2. Удаление информации о книге
3. Поиск информации о книге
4. Редактирование информации о книге
5. Приём/выдача книги пользователю

ЭТАП КОНСТРУИРОВАНИЕ

Итерация 1:

Для реализации сценариев

- Use Case'a WorkWithBook
- Use Case'a WorkWithClient

необходимо создать программный код для следующих методов классов представленных на диаграмме ниже (Рис.13):

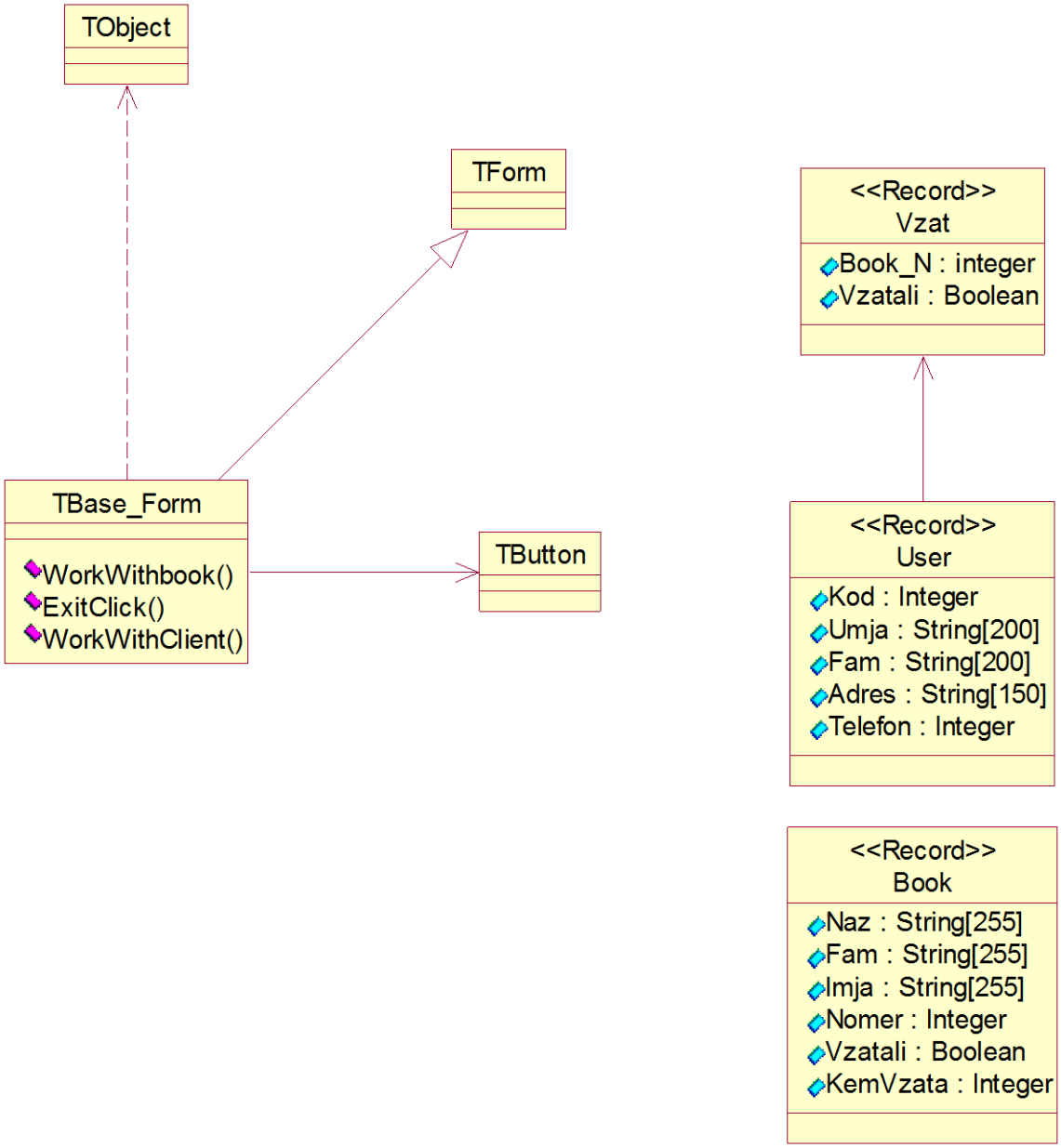


Рис.13 Диаграмма классов на первой итерации

Примечание: классы Tform, Tobject, Tbutton –не создавались, это классы языка Delphi.

Оценка качества логической структуры модели с помощью метрик Чидамбера – Кемерера

Описание процесса подсчёта метрик на первой итерации:

Нахождение WMC: Пусть, начальный вес каждого метода равен единице. Тогда, вес взвешенных методов будет равен их количеству.

Нахождение NOC: Ни один класс не имеет дочерних форм.

Нахождение количества сцеплений между классами объектов (CBO метрика):

CBO- количество сотрудничеств, предусмотренных для класса (количество классов с которыми он соединен). В нашем случае для класс TBase_Form значение будет равно трём, так как класс используют методы или экземплярные переменные стандартных классов языка Delphi.

Расчёт метрик RFC, OSavg, Npavg

Класс TBase_Form

Метод	Вызываемые методы(RFC)	Кратность	Значение OSavg	Значение NPavg
WorkWithbook	-	0	0	1
WorkWithClient	-	0	0	1
ExitClick	Close	1	1	1
колич/ среднее	1		0,3	1

Расчёт метрики LCOM

Класс TBase_Form

Число пар методов – 0
СВЯЗАНЫ = 0
НЕСВЯЗАНЫ = 0
LCOM = 0

Класс Book

Число пар методов – 0
СВЯЗАНЫ = 0
НЕСВЯЗАНЫ = 0
LCOM = 0

Класс User

Число пар методов – 0
СВЯЗАНЫ = 0
НЕСВЯЗАНЫ = 0
LCOM = 0

Класс Vzat

Число пар методов – 0
СВЯЗАНЫ = 0
НЕСВЯЗАНЫ = 0
LCOM = 0

Расчёт метрики NOO – количество операций, переопределяемых подклассом. Переопределения отсутствуют.

Расчёт метрики NOA – количество операций, добавленных подклассом. Подклассы специализируются добавлением приватных операций и свойств. Приватных операций и свойства отсутствуют.

Расчёт метрики SI – индекс специализации. Специализация достигается добавлением, удалением или переопределением операций.

$$SI = (NOO * \text{уровень})/M_{общ} \text{ , где}$$

Уровень - номер уровня в иерархии, на котором находится подкласс.

$M_{общ}$ - общее количество методов класса.

Для класса TBase_Form

$SI = (0 * 1)/3=0$

Оценка качества на первой итерации представлена в таблице ниже:

Метрика	TBase_Form	User	Book	Vzat	Среднее значение
WMC	3	0	0	0	0,75
NOC	0	0	0	0	0
CBO	3	0	0	0	0,75
RFC	1	0	0	0	0,25
LCOM	0	0	0	0	0
CS (a/o)	0/3	5/0	6/0	2/0	3.25/0,75
NOO	0	0	0	0	0
NOA	0	0	0	0	0
SI	0	0	0	0	0
OS_{avg}	0,3	0	0	0	0,075
NP_{avg}	1	0	0	0	0,25
DIT	2				
NC	4				
NOM	3				
$LOC_{\sum}$	164				

Итерация 2:

Для реализации сценариев

- Use Case’a AddBook
- Use Case’a DeleteBook
- Use Case’a SearchBook
- Use Case’a ChangeBook

необходимо создать программный код для следующих методов классов представленных на диаграмме ниже (Рис.14):

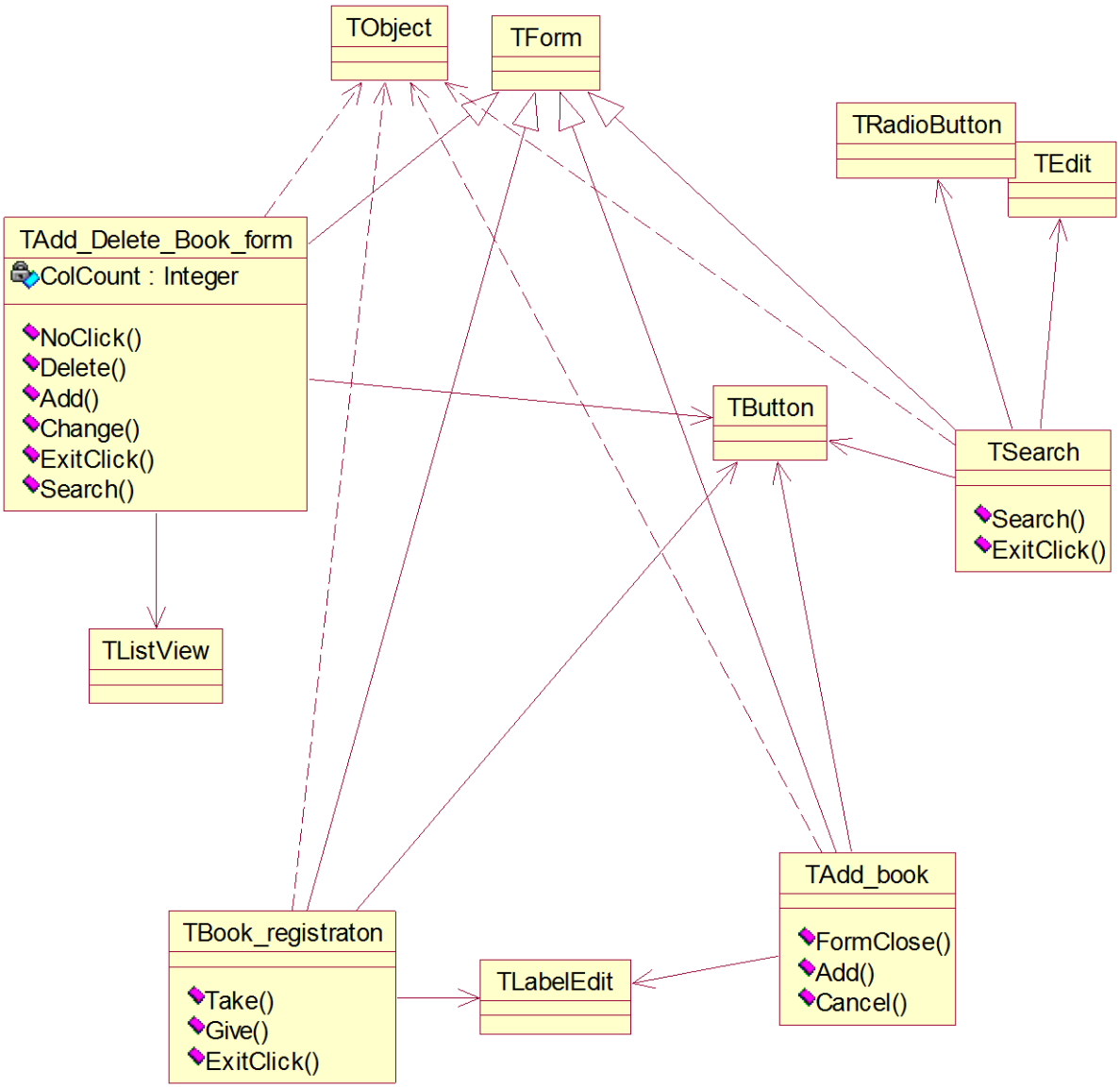


Рис.14 Диаграмма классов на второй итерации

Для оценки качества проведенной разработки выполним подсчет значения метрик для реализованных классов и системы в целом:

Метрика	TAdd_Delete_Book_form	TBook_registraton	TAdd_book	TSearch	Среднее значение
WMC	6	3	3	2	3,5
NOC	0	0	0	0	0
CBO	4	4	4	5	4,25
RFC	7	3	6	10	6,5
LCOM	0	0	0	0	3,5
CS (a/o)	1/6	0/3	0/3	0/2	0,25/3,5
NOO	0	0	0	0	0
NOA	0	0	0	0	0
SI	0	0	0	0	0
OS_{avg}	1,75	2	2	2,6	2,0875
NP_{avg}	1	1	1	1	1
DIT	2				
NC	4				
NOM	14				
LOC_{Σ}	477				

Итерация 3:

Для реализации сценариев

- Use Case’a AddClient
- Use Case’a DeleteClient
- Use Case’a SearchClient
- Use Case’a ChangeClient
- Use Case’a RegBookClient

Необходимо создать программный код для следующих методов классов представленных на диаграмме ниже (Рис.15):

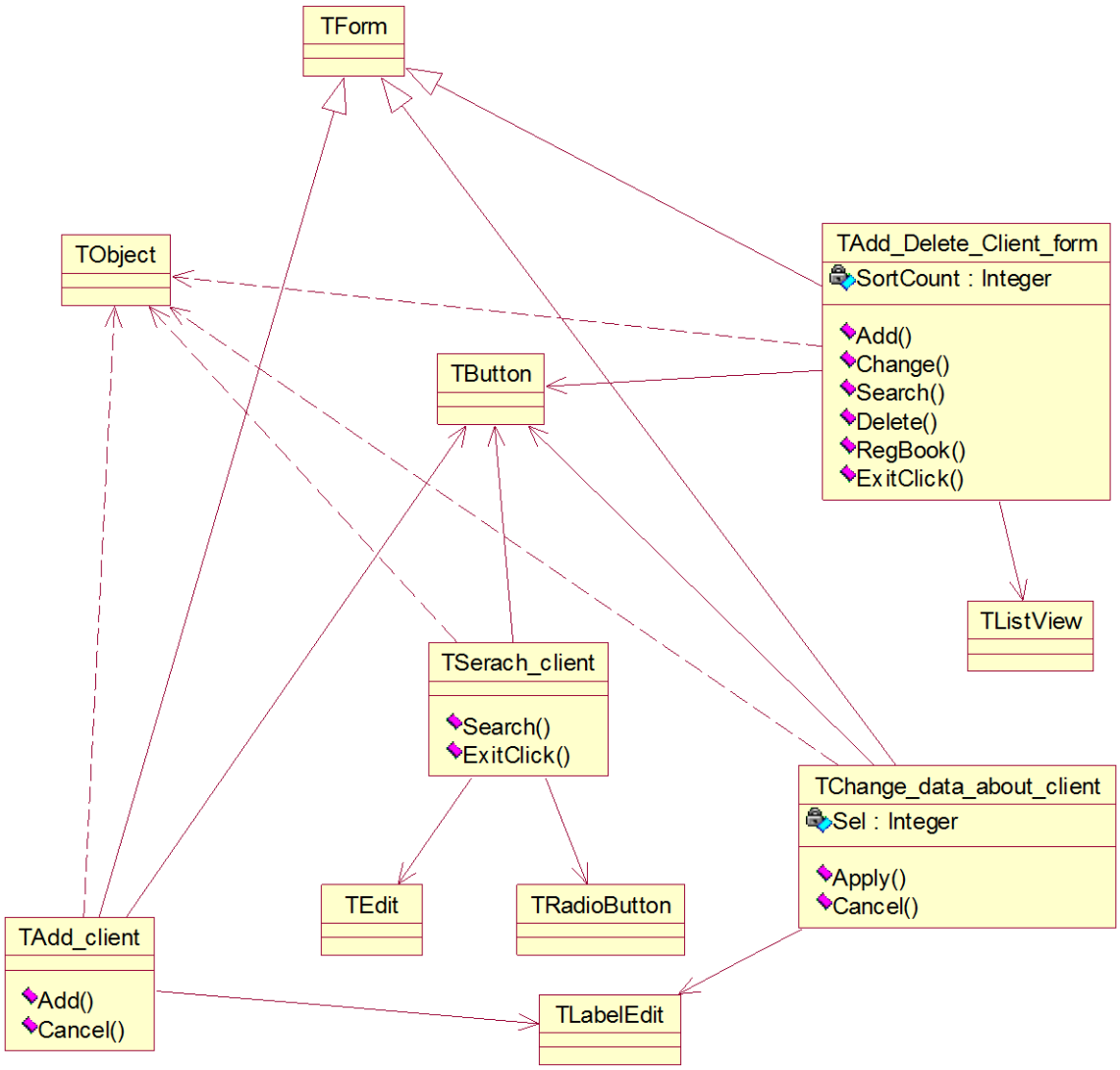


Рис.15 Диаграмма классов на третьей итерации

Для оценки качества проведенной разработки выполним подсчет значения метрик для реализованных классов и системы в целом:

Метрика	TAdd_Delete_Client_form	TChange_data_about_client	TSerach_client	TAdd_client	Среднее значение
WMC	6	2	2	2	3
NOC	0	0	0	0	0
CBO	4	4	4	4	4
RFC	8	9	5	5	6,75
LCOM	0	0	0	0	0
CS (a/o)	1/6	1/2	6/2	2/2	2,5/3
NOO	0	0	0	0	0
NOA	0	0	0	0	0
SI	0	0	0	0	0
OS_{avg}	5,6	8	5	5,6	6,05
NP_{avg}	1	1	1	1	1
DIT	2				
NC	4				
NOM	12				
$LOC_{\sum}$	753				

Далее сведем в таблицу полученные значения метрик на разных итерациях и сравним полученные результаты.

Значения метрик на разных итерациях

Метрика	Итерация 1	Итерация 2	Итерация 3
WMC	0,75	3,5	3
NOC	0	0	0
CBO	0,75	4,25	4
RFC	0,25	6,5	6,75
LCOM	0	3,5	0
CS	3.25/0,75	0,25/3,5	2,5/3
NOO	0	0	0
NOA	0	0	0
SI	0	0	0
OSavg	0,075	2,0875	6,05
Npavg	1	1	1
DIT	2	2	2
NC	4	4	4
NOM	3	14	12
LOCsum	164	477	753

Сравнивая полученные значения метрик, можно заметить, что некоторые их значения увеличились, а это в свою очередь свидетельствует о постепенном возрастании сложности продукта, что в принципе вполне естественно.

Примечание: метрики для классов языка Delphi не подсчитывались, так как мы их не создавали.

Оценка качества проектирования

С ростом CBO многократность использования класса, вероятно, уменьшается. Высокое значение CBO усложняет модификацию и тестирование. CBO для каждого класса должно иметь разумно низкое значение.

Метрика RFC если в ответ на сообщение может быть вызвано большое количество методов, то усложняется тестирование и отладка класса.

Метрика LCOM показывает насколько методы не связаны друг с другом через свойства (переменные).

CS не превышает рекомендуемого значения CS $\leq$ 20 методов.

Отсутствие NOO указывают на то, что проблем с проектированием нет , что разработчик не нарушает абстракцию суперкласса. Это не ослабляет иерархию классов, не усложняет тестирование и модификацию программного обеспечения. Рекомендуемое значение $\leq$3 методов не превышено.

Отсутствие NOA указывают на то, что подкласс не удаляется от абстракции суперкласса. Для рекомендуемых значений CS = 20 и DIT = 6 рекомендуемое значение NOA $\leq$ 4 методов (для класса-листа). Рекомендуемое значение не превышено.

Отсутствие SI указывают на то, что нет вероятности того, что в иерархии классов есть классы, нарушающие абстракцию суперкласса. Рекомендуемое значение SI $\leq$ 0.15 не превышено.

Чем больше параметров у операции, тем сложнее сотрудничество между объектами. Поэтому значение NP_{avg} должно быть как можно меньшим. Рекомендуемое значение NP_{avg} = 0.7. немного превышено, но не значительно.

Ниже представлена общая диаграмма классов для всего проекта (Рис.16):

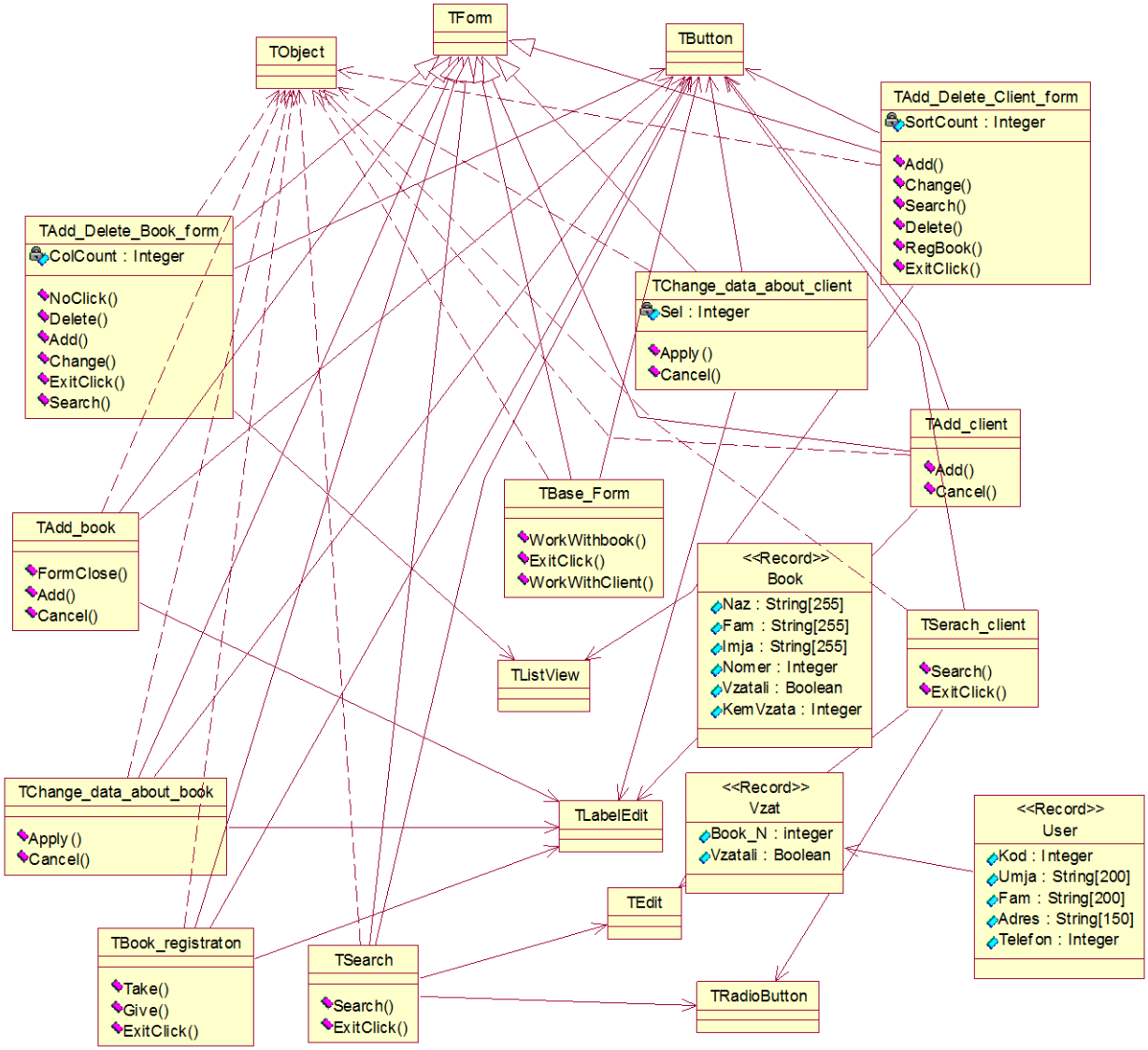


Рис.16 Общая диаграмма классов для всего проекта

Компонентная диаграмма
На рисунке представлена компонентная диаграмма (Рис.17).

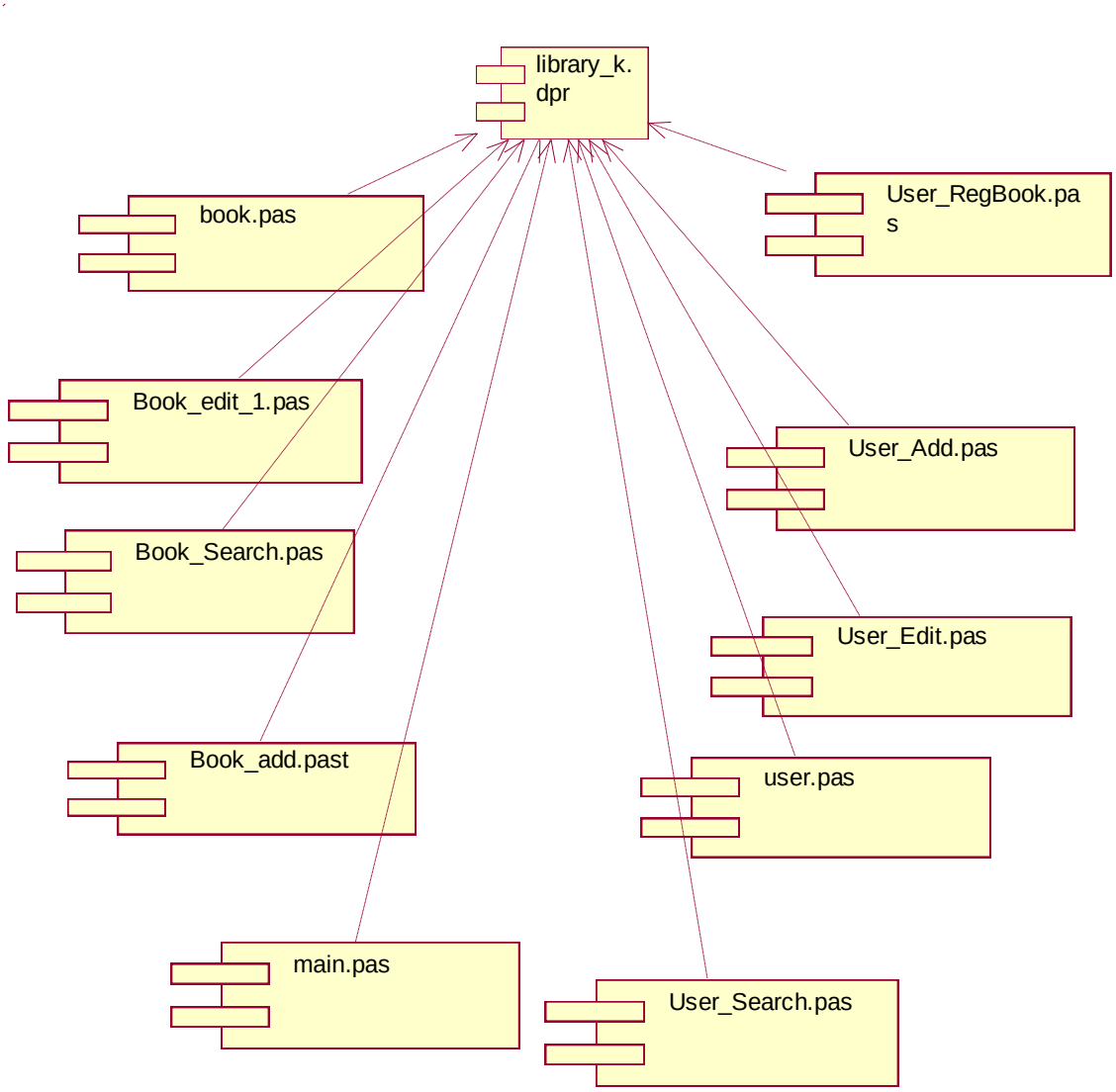
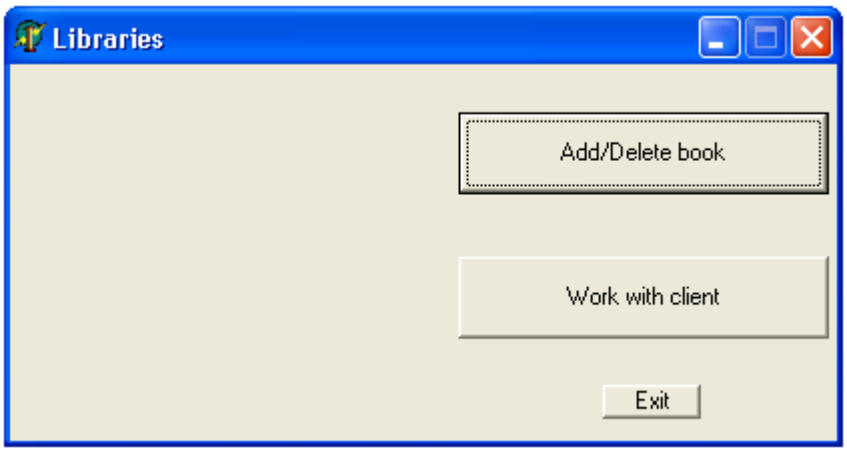
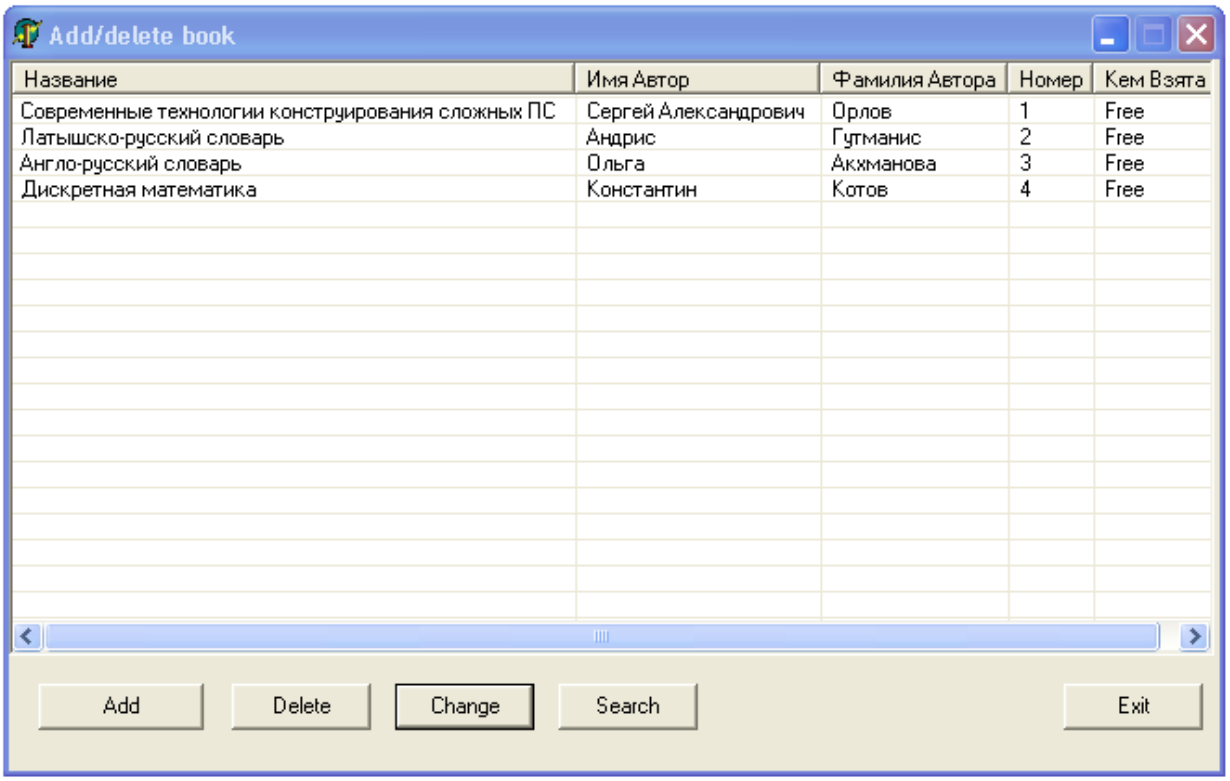


Рис.17 Компонентная диаграмма

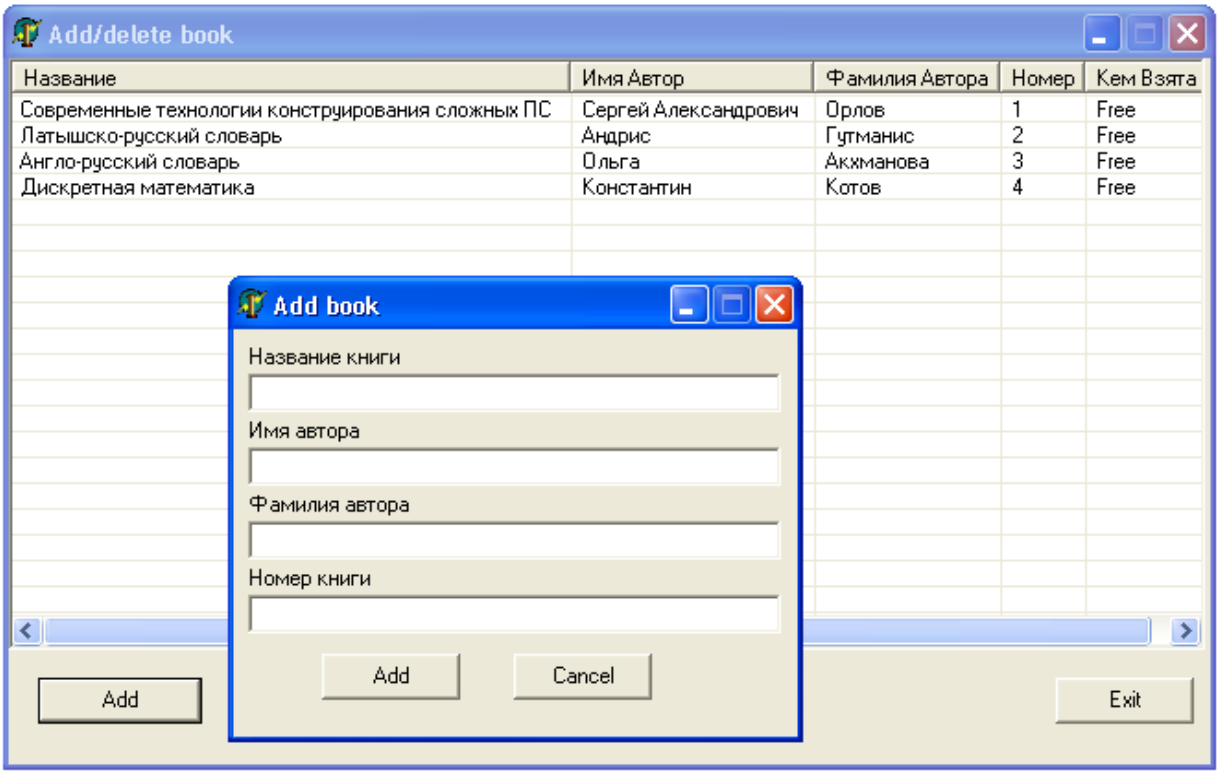
Результат этапа кодирование



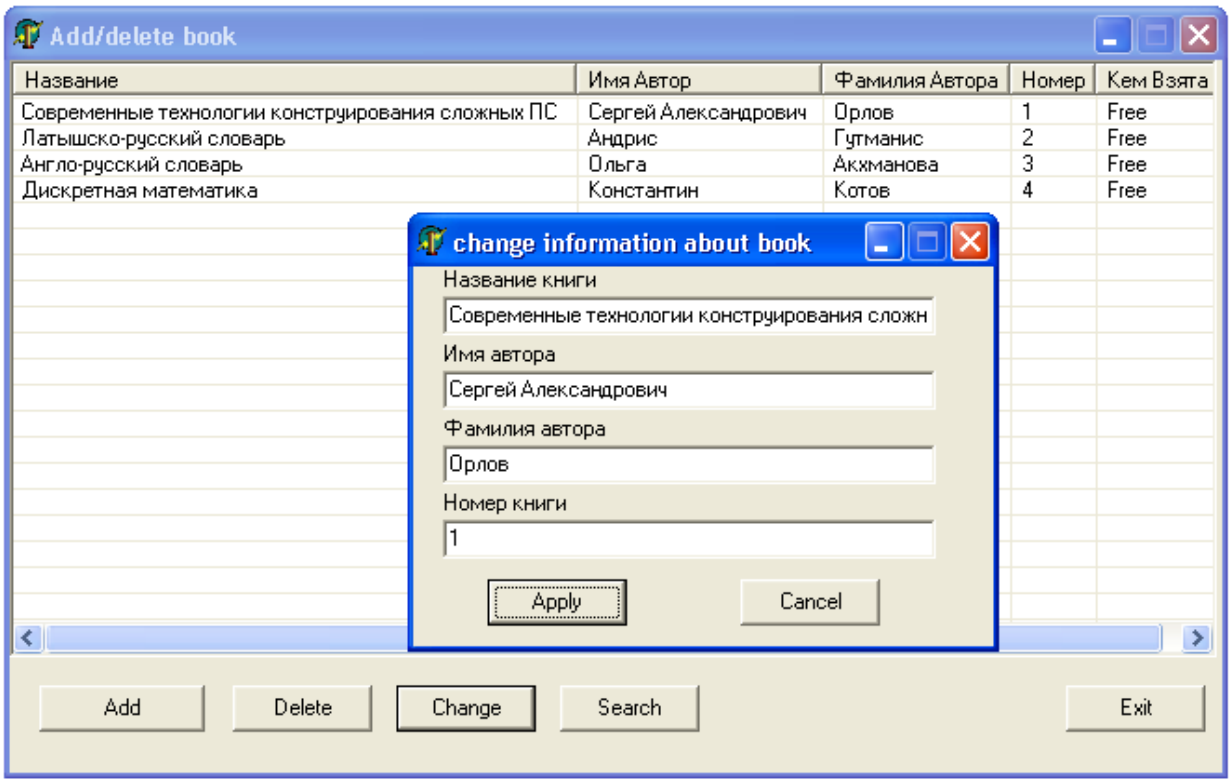
Главное окно программы



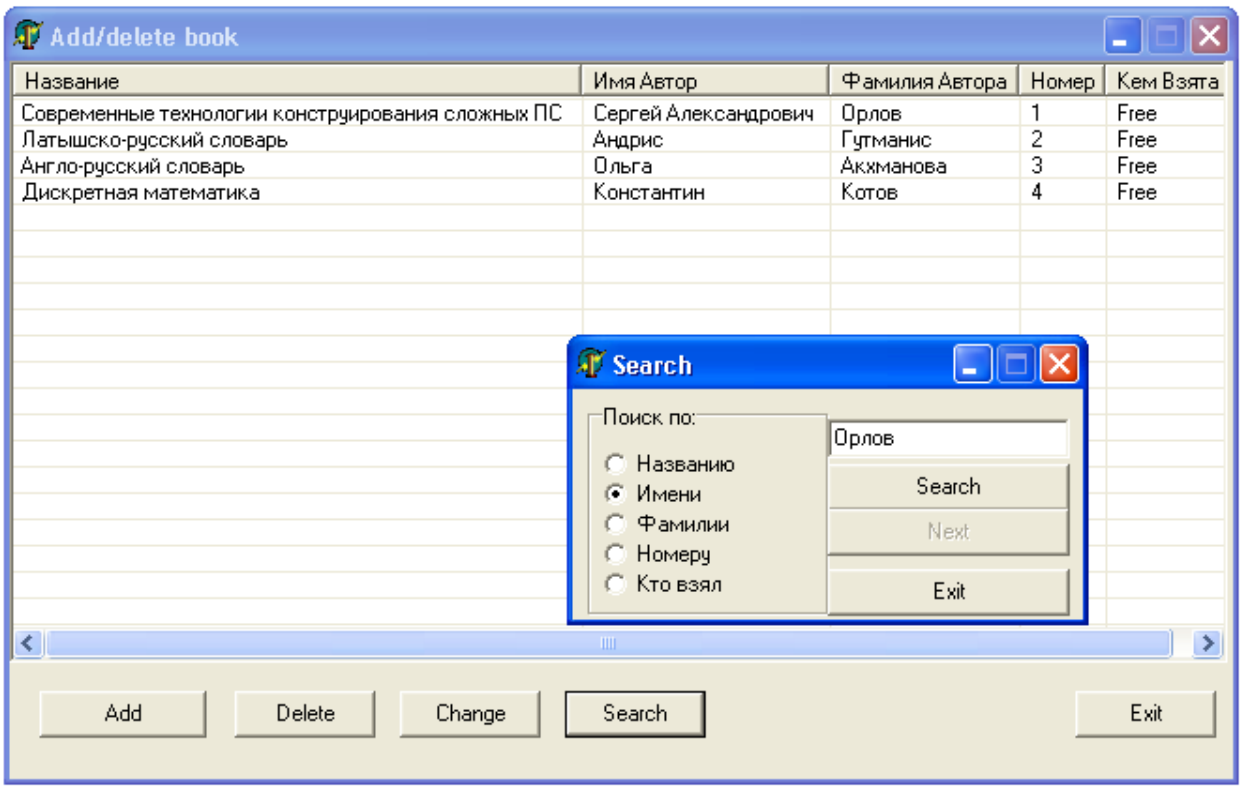
Окно работы с книгами



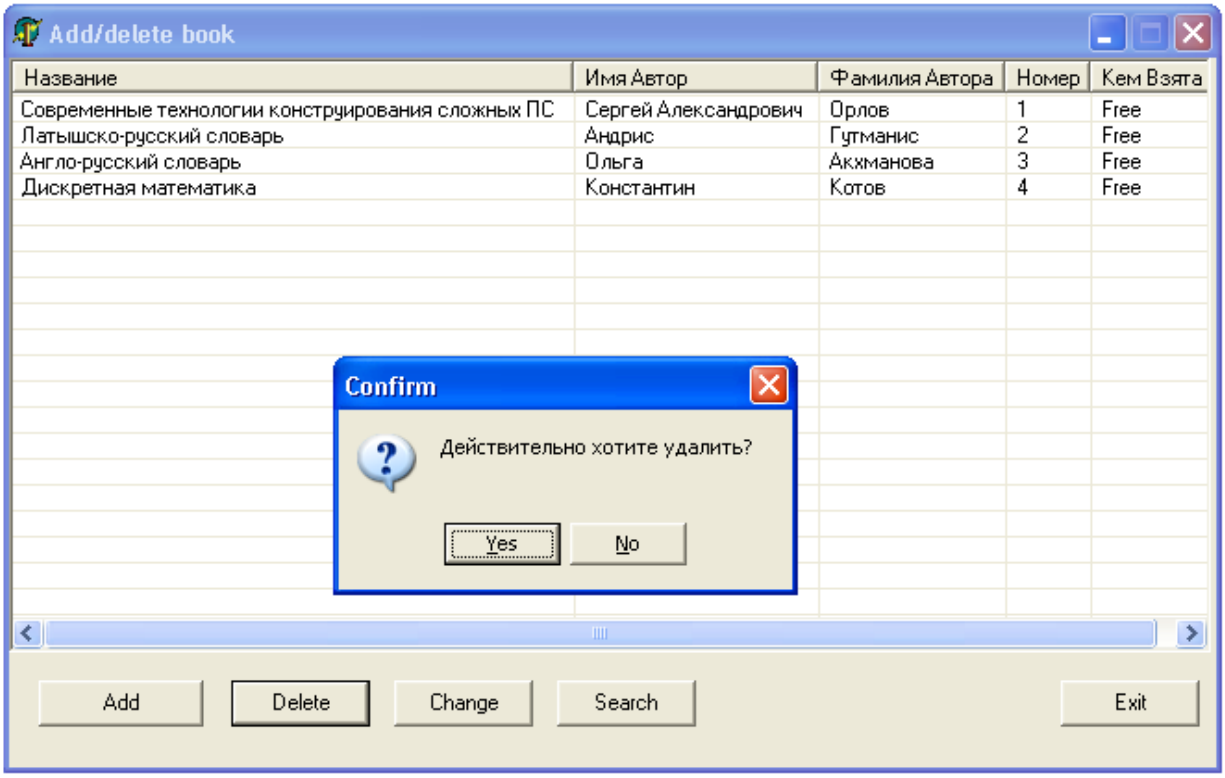
Окно добавления книги в БД



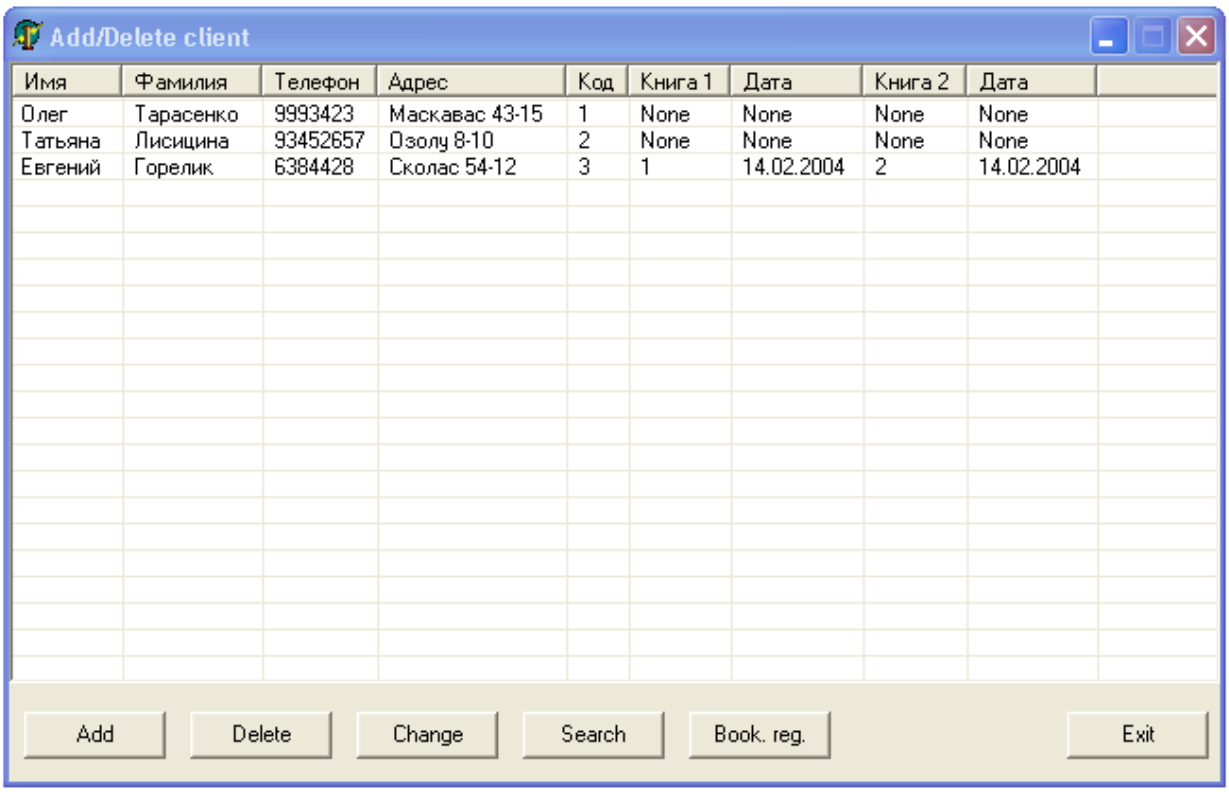
Окно изменения информации о книге в БД



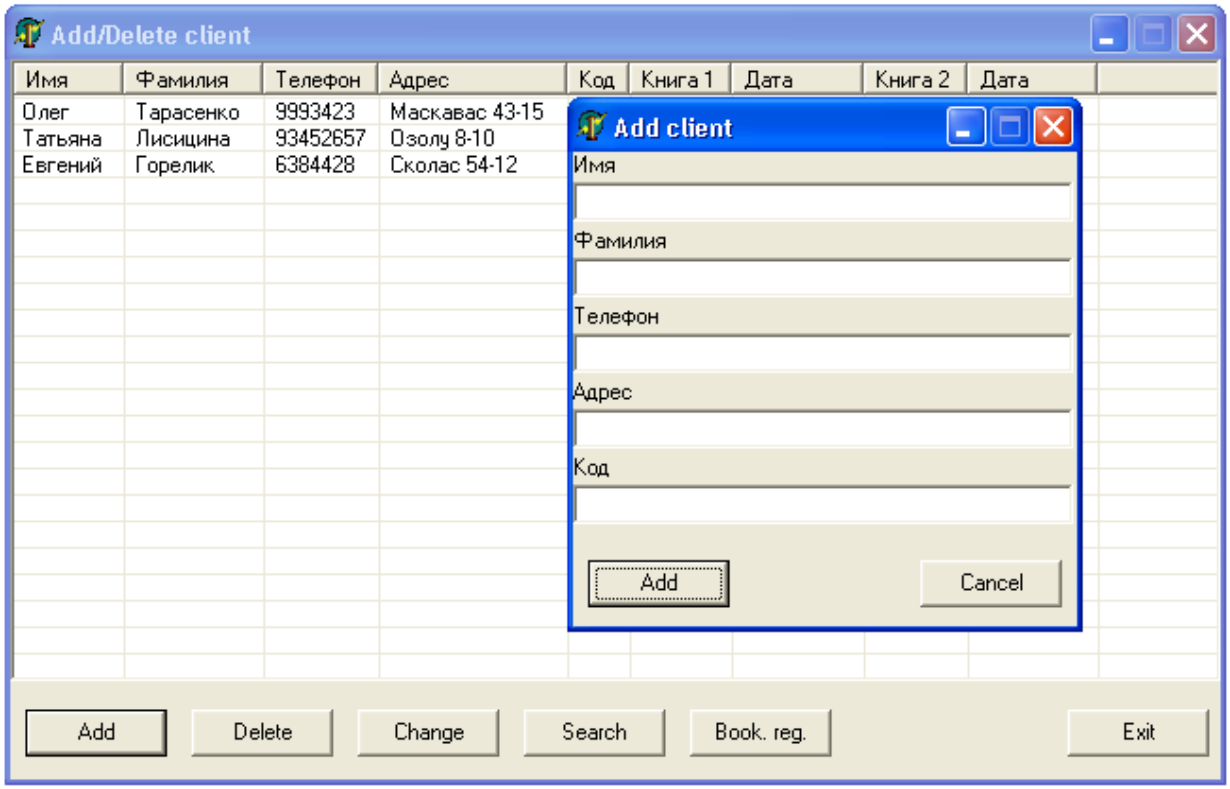
Окно поиска книги в БД



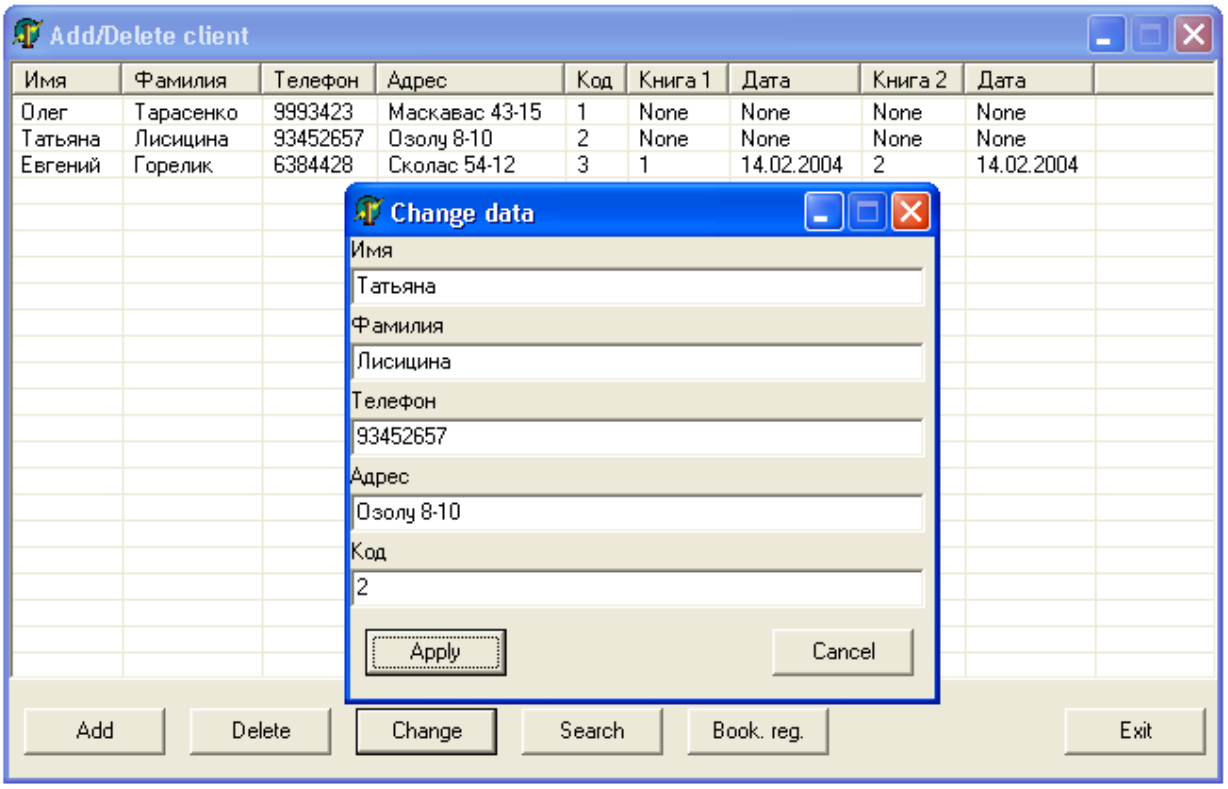
Окно удаления книги из БД



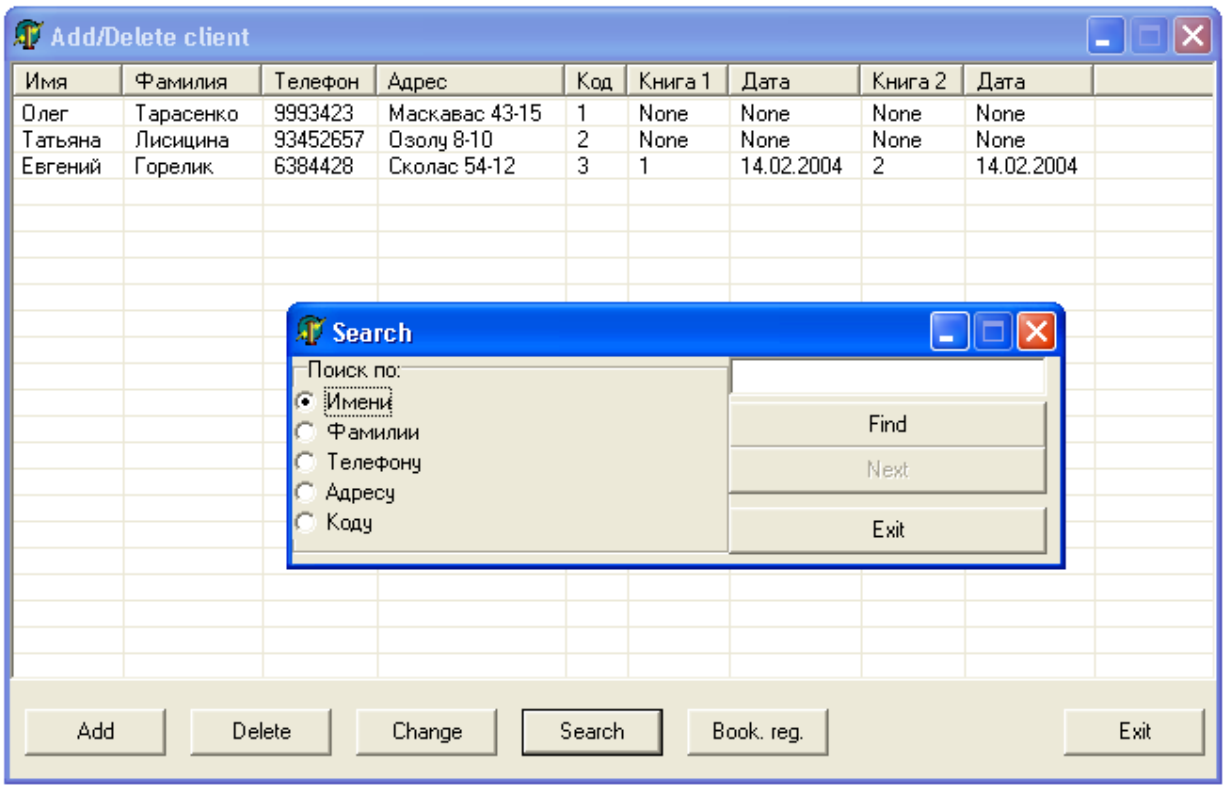
Окно работы с читателями



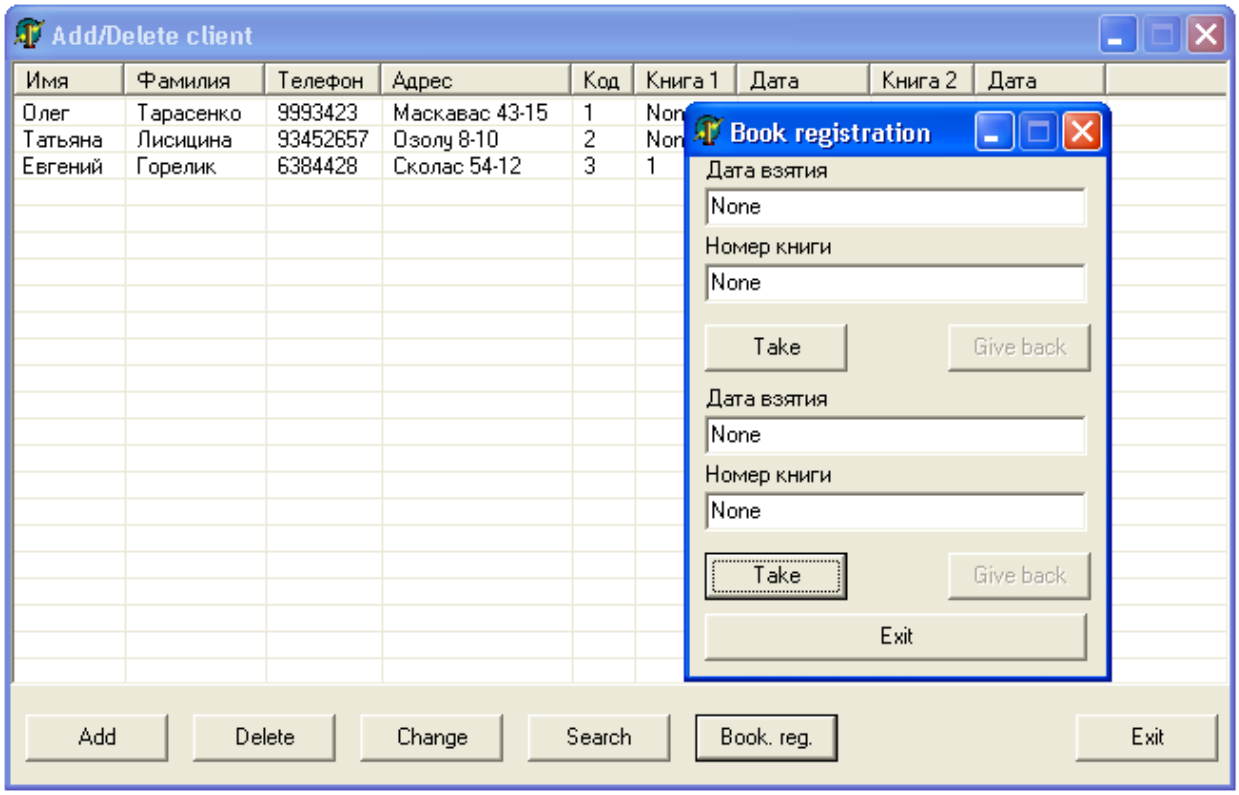
Окно добавления информации о читателе в БД



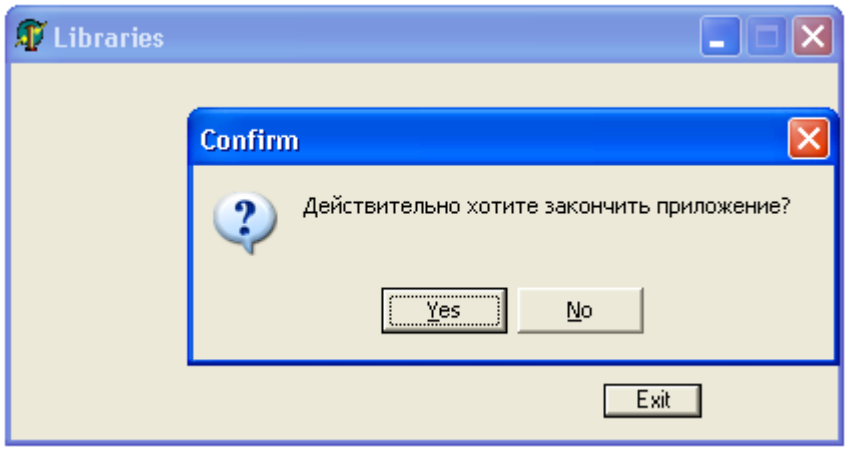
Окно редактирования информации о читателе в БД



Окно поиска информации о читателе в БД



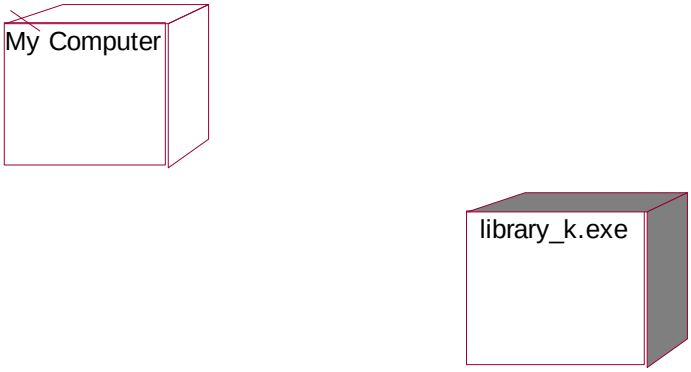
Окно выдачи/возврата книг читателю в БД



Окно завершения работы с БД

Диаграмма размещёния.

В конечном итоге, программа представляет собой единственный запускаемый файл, рассчитанный на работу в локальном режиме на стандартном однопроцессорном компьютере. Диаграмма размещения приведена на рисунке 13.



Выводы

В ходе написания курсового проекта были применены на практике и более подробно изучены технологии визуального конструирования сложных программных систем. Был получен дополнительный опыт разработки визуальной модели системы, её оценивание качества с помощью набора метрик Чидамбера и Кемерера. Ознакомились с программной средой Rational Rose. Можно сделать вывод о том, что визуальное проектирование сложных ОО программ, на первых этапах разработки, существенно облегчает труд разработчика, предоставляет возможность зрительного восприятия и начального анализа будущего проекта.

В результате проделанной работы была получена объектно-ориентированная программная система с заданными характеристиками и отвечающая поставленным требованиям.

В ходе планирования итераций конструирования удалось найти решение, позволяющее получать после каждой завершенной итерации не только законченные с точки зрения программной реализации классы, конструирование которых планировалось на данной итерации, но и работоспособную (в рамках разрабатываемого Use-case’а) систему, что позволяет не только значительно уменьшить риск разработки посредством обеспечения возможности отладки кода на каждой итерации, но и снизить накладные расходы, связанные с достаточно трудоемким процессом пересчета метрик, в случае если разработка методов класса ведется в ходе нескольких итераций.

Список используемой литературы

1. Орлов С.А., Современные технологии конструирования сложных программных систем. – Рига: ИТС, 1999.-289 с.:ил.
2. Орлов С.А., Введение в визуальное моделирование. – Рига: ИТС, 2001. – 55с.: ил.
3. Орлов С.А. Конспект лекций по технологиям конструирования программных систем.
4. Дарахвелидзе П.Г., Марков Е.П., Программирование в Delphi 4. – СПб.: БХВ – Санкт-Петербург, 1999.- 864с., ил.