МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Рубцовский индустриальный институт (филиал)
федерального государственного бюджетного образовательного учреждения
высшего образования
«Алтайский государственный технический университет им. И.И. Ползунова»
(РИИ АлтГТУ)

Л.А. ПОПОВА

РАЗРАБОТКА КУРСОВОГО ПРОЕКТА ПО ПРОГРАММИРОВАНИЮ ПРИЛОЖЕНИЙ

Методические указания по подготовке и оформлению курсового проекта
по программированию приложений для студентов второго курса
очной и заочной форм обучения направления
«Информатика и вычислительная техника»

Рубцовск 2021

ББК 32.973.2

Попова Л.А. Разработка курсового проекта по программированию приложений: Методические указания по подготовке и оформлению курсового проекта по программированию приложений для студентов второго курса очной и заочной форм обучения направления «Информатика и вычислительная техника» / Л.А. Попова. – Рубцовск: РИИ, 2021. – 51 с. [ЭР].

Методические указания разработаны на основе ФГОС ВО направления подготовки «Информатика и вычислительная техника», утвержденного приказом Минобрнауки РФ №5 от 12.01.2016, рабочей программы дисциплины «Программирование приложений», образовательного стандарта высшего образования АлтГТУ «Курсовой проект (курсовая работа)». Указания содержат общие положения, нормативные ссылки, требования к программе, составленной на алгоритмическом языке C#, к структуре и содержанию пояснительной записки курсового проекта.

Указания предназначены для студентов второго курса, обучающихся по направлению подготовки «Информатика и вычислительная техника».

Рассмотрены и одобрены на
заседании каф. ПМ РИИ
Протокол № 9 от 18.03.2021 г.

Содержание

ВВЕДЕНИЕ 4
1 НОРМАТИВНЫЕ ССЫЛКИ 4
2 СОДЕРЖАНИЕ И ПОРЯДОК ВЫПОЛНЕНИЯ КУРСОВОГО ПРОЕКТА .. 5
2.1 Общие положения 5
2.2 Структура пояснительной записки 6
2.3 Требования к оформлению текста пояснительной записки 10
3 ТРЕБОВАНИЯ К ПРОГРАММНОМУ КОДУ 13
4 ПОРЯДОК РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА 13
4.1 Разработка модели данных и создание БД Library (библиотека) 14
4.2 Разработка интерфейса программы в соответствии с технологией Windows Forms и Entity Framework 25
4.3 Разработка интерфейса программы в соответствии с технологией WPF и Entity Framework 36
5 СПИСОК ТЕМ КУРСОВОГО ПРОЕКТА 47
6 ВОПРОСЫ К ЗАЩИТЕ КУРСОВОГО ПРОЕКТА ПО ДИСЦИПЛИНЕ 48
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 49
Приложение А. Форма титульного листа пояснительной записки 50
Приложение Б. Форма бланка задания на курсовой проект 51

ВВЕДЕНИЕ

Выполнение курсовых проектов является составной частью основной образовательной программы (ООП) высшего образования (ВО) и заключительным этапом изучения студентами соответствующей учебной дисциплины согласно ФГОС ВО направления подготовки. Курсовой проект – один из видов самостоятельной учебной работы студентов, представляющей собой решение учебной или реальной профессиональной задачи и предусматривающей учебные занятия в виде самостоятельной работы, консультаций и защиты выполненного проекта [1].

Данные методические указания предназначены для студентов направления подготовки «Информатика и вычислительная техника» и определяют состав, порядок выполнения курсового проекта по дисциплине «Программирование приложений» и оформления пояснительной записки, а также содержат список тем.

Целью курсового проекта является систематизация, закрепление и расширение знаний по учебной дисциплине «Программирование приложений» в процессе решения прикладных задач.

Задачами курсового проекта являются:

- формирование умений и навыков самостоятельной работы студентов;
- овладение современными методами поиска, обработки и использования информации
- закрепление навыков составления программы на алгоритмическом языке C# и ее отладки в среде программирования;
- приобретение навыков оформления программной документации в соответствии с ЕСПД (единой системой программной документации);
- формирование у студентов общекультурных, общепрофессиональных и профессиональных компетенций.

Качество курсового проекта определяется полнотой постановки задачи, алгоритма и руководства пользователя, эффективностью программы, составленной в соответствии с правилами объектно-ориентированного программирования.

1 НОРМАТИВНЫЕ ССЫЛКИ

Курсовой проект должен быть выполнен и оформлен в соответствии с нормативными документами и рекомендациями профилирующей кафедры и вуза в целом.

– ГОСТ Р 1.5–2012 Стандарты национальные Российской федерации. Правила построения, изложения, оформления и обозначения

– ГОСТ 2.102–2013 ЕСКД Виды и комплектность конструкторских документов

– ГОСТ 2.004–88 ЕСКД Общие требования к выполнению конструкторских и технологических документов на печатающих и графических устройствах вывода ЭВМ

– ГОСТ 2.104–2006 ЕСКД Основные надписи
– ГОСТ 2.106–95 ЕСКД Общие требования к текстовым документам
– ГОСТ 2.106–96 ЕСКД Текстовые документы
– ГОСТ 3.1105–2011 ЕСТД Формы и правила оформления документов общего назначения
– ГОСТ 7.1–2003 СИБИД Библиографическая запись. Библиографическое описание. Общие требования и правила составления
– ГОСТ 19.401–78 ЕСПД Текст программы. Требования к содержанию и оформлению
– ГОСТ 19.402–78 ЕСПД Описание программы
– ГОСТ 19.502–78 ЕСПД Описание применения. Требования к содержанию и оформлению
– ГОСТ 19.701–90 ЕСПД Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения
– ГОСТ 19.105–78 ЕСПД Общие требования к программным документам
– ГОСТ 19.404–79 ЕСПД Пояснительная записка. Требования к содержанию и оформлению
– СТО АлтГТУ 12 570–2013. Система качества. Образовательный стандарт высшего профессионального образования АлтГТУ. Общие требования к текстовым, графическим и программным документам

2 СОДЕРЖАНИЕ И ПОРЯДОК ВЫПОЛНЕНИЯ КУРСОВОГО ПРОЕКТА

2.1 Общие положения

Курсовой проект состоит из текстового документа (пояснительная записка) и программы на языке C#.

Курсовой проект студенты выполняют согласно заданию, выданному руководителем. Выполнение отдельных этапов (разделов проекта) и представление ее к защите должно соответствовать срокам, установленным в задании.

Курсовой проект представляют на проверку руководителю поэтапно или полностью выполненной. Руководитель проверяет проект на полноту и решает вопрос о допуске к защите.

Оценка курсового проекта может быть дана по итогам проверки пояснительной записки и собеседования со студентом. Студентам, успешно защитившим курсовой проект, в ведомости выставляются отметки «отлично», «хорошо», «удовлетворительно», а также соответствующие оценки по 100-балльной шкале в диапазонах 75–100, 50–74 или 25–49 баллов с учетом полноты и качества выполненного проекта, результатов защиты, дополнительных факторов (систематического характера работы или, наоборот, отставания от установленных сроков).

2.2 Структура пояснительной записки

Пояснительная записка курсового проекта должна выполняться на листах формата А4 и содержать следующие структурные элементы:

- титульный лист;
- задание;
- содержание;
- постановку задачи;
- алгоритм решения задачи;
- руководство пользователя;
- список использованных источников;
- приложения.

2.2.1 Титульный лист должен содержать сведения согласно СТО АлтГТУ 12 570–2013. Форма титульного листа приведена в приложении А.

2.2.2 В задании должны быть указаны: учебная дисциплина, по которой выполняется проект; Ф.И.О. студента, его группа; тема проекта; разделы разработки и сроки их выполнения; срок представления проекта к защите; Ф.И.О. руководителя, его должность и дата выдачи задания. Задание помещается после титульного листа и включается в общую нумерацию листов пояснительной записки. Форма бланка задания приведена в приложении Б.

2.2.3 Содержание состоит из последовательно перечисленных наименований разделов, подразделов и приложений (заголовков первого и второго уровней) с указанием страниц, на которых они находятся. Содержание включают в общую нумерацию листов пояснительной записки и размещают после листа задания.

Слово «Содержание» записывается в виде заголовка симметрично тексту с прописной буквы. Наименования, включенные в содержание, также начинаются с прописной буквы, остальные буквы – строчные.

2.2.4 Раздел «Постановка задачи» содержит подразделы:

- Характеристика задачи;
- Выходная информация;
- Входная информация.

«Характеристика задачи «*Название программы*» содержит:

- назначение задачи;
- перечень объектов, при управлении которыми решается задача;
- периодичность и продолжительность решения задачи;
- условия, при которых прекращается решение задачи;
- связь данной задачи с другими;
- распределение действий между персоналом и техническими средствами при различных ситуациях решения задачи.

«Выходная информация» содержит перечень выходных сообщений (таблица 1) и описание структурных единиц информации (таблица 2).

Таблица 1

Перечень выходных сообщений

Идентификатор	Форма представления	Наименование	Периодичность	Получатель	Число строк
D0?? F0?? V0??	Документ Файл Видеограмма				

Таблица 2

Описание структурных единиц информации

Наименование выходного сообщения		Идентификатор	
Наименование структурных единиц	Обозначение	Длина в знаках	Диапазон

Также в данном подразделе должен быть представлен внешний вид выходных документов и видеограмм.

«Входная информация» содержит перечень входных сообщений и описание структурных единиц, которые оформляются аналогично выходной информации (таблицы 3-4).

Таблица 3

Перечень входных сообщений

Идентификатор	Форма представления	Наименование	Периодичность	Поставщик	Число строк
F0?? V0??	Файл Видеокадр				

Таблица 4

Описание структурных единиц информации

Наименование входного сообщения		Идентификатор	
Наименование структурных единиц	Обозначение	Длина в знаках	Диапазон

В подразделах «Выходная информация» и «Входная информация» должен быть представлен внешний вид документов, видеограмм и видеокадров.

2.2.5 Раздел «Алгоритм решения задачи» содержит подразделы:

- Назначение и характеристика;
- Используемая информация;
- Результаты решения;
- Описание данных;
- Алгоритм решения.

В подразделе «Назначение и характеристика» приводится назначение алгоритма, сведения о процессе управления, для которого он предназначен, ограничения на использование алгоритма.

В подразделе «Используемая информация» указывается перечень и описание файлов, сформированных из входных сообщений или получаемых в результате решения других задач, или видеокадров. Для каждого файла и видеокадра указывается обозначение, наименование и максимальное число записей или строк.

В подразделе «Результаты решения» содержится перечень и описание файлов, формируемых в результате решения задачи, документов или видеограмм, а также файлов, сохраняемых для последующих решений этой задачи.

В подразделе «Описание данных» приводится описание пользовательских типов данных, процедур, функций и переменных.

В подразделе «Алгоритм решения» описываются действия, определяющие процесс перехода от входной информации к выходной, а также способы формирования результатов решения и получения выходных документов и видеограмм. Способ описания алгоритма может быть графическим (в виде блок-схемы), текстовым или смешанным.

При использовании блок-схем рекомендуется следовать ГОСТ 19.701–90 ЕСПД.

При текстовом описании алгоритма решения задачи каждый пункт алгоритма должен отвечать на вопрос «Что сделать?», быть кратким и содержательным одновременно. Ниже приведен пример описания алгоритма таким способом.

1 Связать файловую переменную f с файлом на диске “students.dat”.
2 Вывести на терминал пункты меню.
3 Ввести номер пункта меню.
4 Анализировать введенное значение.
5 Если введено значение 1
  5.1 Открыть файл f на запись.
  5.2 Ввести в переменную n количество записей.
  5.3 Вывести на терминал видеокадр V005 Данные о студенте.
  5.4 Повторять n раз
    5.4.1 Ввести в переменную Student данные: номер группы, Ф.И.О. студента, размер стипендии.
    5.4.2 Вывести данные в файл f из переменной Student.
  5.5 Закрыть файл f.
  5.6 Перейти на п. 2.
  …

2.2.6 Раздел «Руководство пользователя» содержит подразделы:

- Введение;
- Назначение и условия применения;
- Подготовка к работе;

– Описание операций;
– Аварийные ситуации.

В подразделе «Введение» указывается область применения, описание возможностей задачи, уровень подготовки пользователя, перечень необходимых эксплуатационных документов для пользователя.

В подразделе «Назначение и условия применения» указываются функции, которые выполняет программа, аппаратная и программная конфигурации, необходимые для установки и правильной работы программы.

В подразделе «Подготовка к работе» указывается состав дистрибутивного носителя (архивного файла), порядок загрузки данных и программ и проверки их работоспособности.

В подразделе «Описание операций» содержится описание пунктов и подпунктов меню программы, всех входных и выходных сообщений, действий пользователя.

В подразделе «Аварийные ситуации» указываются причины, по которым работа программы может быть невозможна, а также описываются действия пользователя при отказе технических средств, при обнаружении несанкционированного вмешательства в данные, действия по восстановлению данных при отказе носителей. Даются рекомендации пользователям.

2.2.7 Список использованных источников должен включать все источники, указанные в алфавитном порядке, использованные при выполнении курсового проекта. В тексте пояснительной записки должны быть сделаны ссылки на каждый источник.

2.2.8 В приложение выносятся материалы, дополняющие текст пояснительной записки. Приложения оформляют как продолжение записки на последующих ее листах, после списка использованной литературы.

Приложения обозначают прописными буквами русского алфавита, начиная с буквы А, которая следует после слова «Приложение». Например, «Приложение А».

Приложения могут быть обязательными или информационными (рекомендуемого или справочного характера).

Каждое приложение следует начинать с новой страницы с указанием наверху посередине страницы слова «Приложение» и его обозначение, а под ним в скобках пишут слово «обязательное», а для информационного – «рекомендуемое» или «справочное». Допускается размещение на одной странице двух и более последовательно расположенных приложений, если их можно полностью разместить на этой странице.

Каждое приложение должно иметь заголовок к размещаемому тексту. Заголовок записывают симметрично относительно текста с прописной буквы отдельной строкой.

Приложения должны иметь общую с основной частью пояснительной записки сквозную нумерацию страниц. В тексте пояснительной записки должны быть даны ссылки на все приложения. Все приложения должны быть перечислены в содержании документа с указанием их номеров и заголовков.

Программные документы должны оформляться в соответствии с требованиями стандартов ЕСПД, указанных в нормативных ссылках.

В приложения должны входить:

- Листинг программы;
- Распечатка файла данных;
- Контрольный пример;
- Результаты работы программы.

«Листинг программы» содержит распечатку текста программы.

В «Распечатке файла данных» приводится содержание файла с исходными данными (распечатка текстового файла или данные, введенные в двоичный файл).

«Контрольный пример» содержит результаты ручного расчета задачи и выводы о том, какие данные должны быть получены при автоматизированном решении задачи.

В «Результатах работы программы» описываются действия для проверки правильности работы программы, а также проверяется соответствие полученных данных контрольному примеру.

2.3 Требования к оформлению текста пояснительной записки

Текст пояснительной записки курсового проекта, включая приложения, оформляется печатным способом на листах формата А4 (210×297 мм), с применением программных, печатающих и графических средств ЭВМ, согласно ГОСТ 2.004, ГОСТ2.106 и СТО АлтГТУ 12 570–2013.

2.3.1 Основной текст работы, начиная с содержания, оформляются в соответствии со следующими параметрами:

- поля: левое – 30 мм, правое – 10 мм, сверху и снизу – 20 мм;
- гарнитура шрифта – Times New Roman;
- размер шрифта – 14,
- межстрочный интервал – полуторный;
- отступ красной строки – 1,25 см;
- выравнивание – по ширине;
- отступы до и после абзаца – 0.

Для оформления заголовков каждого структурного элемента (названий разделов) следует использовать следующие параметры:

- гарнитура шрифта – Times New Roman;
- размер шрифта – 14;
- отступ красной строки – 1,25 см;
- межстрочный интервал – полуторный;
- выравнивание – по ширине;
- отступы после абзаца – 12 пт.

Каждый структурный элемент должен начинаться с новой страницы.

2.3.2 Иллюстрации (рисунки, схемы и т.п.) следует располагать непосредственно после текста, в котором они упоминаются, или на следующей странице. Иллюстрации в тексте пояснительной записки следует нумеровать арабскими цифрами сквозной нумерацией, например, «Рисунок 1».

Иллюстрации каждого приложения обозначают отдельной нумерацией арабскими цифрами с добавлением перед цифрой обозначения приложения. Например, «Рисунок А.1».

Также допускается нумеровать иллюстрации в пределах раздела. В этом случае номер иллюстрации состоит из номера раздела и порядкового номера иллюстрации, разделенных точкой. Например, «Рисунок 1.1».

Иллюстрации, при необходимости, могут иметь тематический заголовок или пояснительные данные. Каждый рисунок в работе должен иметь обозначение. При оформлении рисунков по всей пояснительной записке следует придерживаться единого стиля оформления.

В качестве примера приведено оформление рисунка 1.

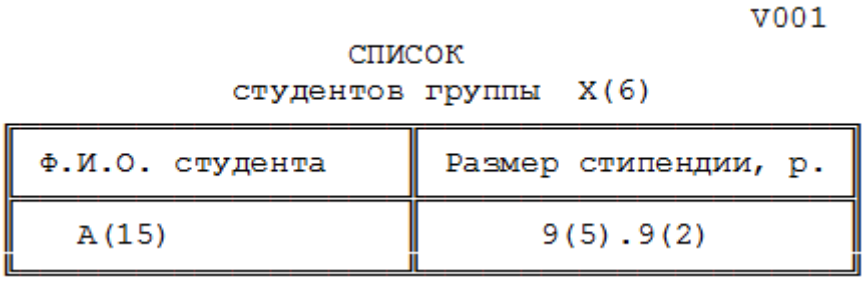


Рисунок 1 – Вид видеограммы V001

В случае использования пояснительных данных к рисунку, обозначение рисунка с кратким пояснением (названием) оформляется внизу рисунка, по центру страницы, без абзацного отступа, без точки в конце фразы.

2.3.3 В пояснительной записке таблица является методом унифицированного текста, и такой текст обладает большой информационной емкостью, наглядностью, позволяет строго классифицировать, кодировать информацию, легко сравнивать и суммировать аналогичные данные.

Таблицу помещают под текстом, в котором впервые дана на нее ссылка. Все таблицы нумеруют арабскими цифрами в пределах всего текста. Над правым верхним углом таблицы помещают слово «Таблица» с указанием ее порядкового номера. Например, Таблица 1. Указание номера таблицы выполняется без значка № перед цифрой и без точки после нее. Если у таблицы есть тематический заголовок, то его располагают посередине страницы под указанием ее номера и пишут с прописной буквы без точки в конце.

Таблицы каждого приложения обозначают отдельной нумерацией арабскими цифрами с добавлением перед цифрой обозначения приложения, например, «Таблица А.1», если она приведена в приложении А.

На все таблицы в тексте должны быть приведены ссылки, при этом следует писать слово «таблица». Например, «В соответствии с таблицей 1…» или «…в таблице 2».

В качестве примера приведено оформление таблицы 5.

Таблица 5

Наименование выходного сообщения		Идентификатор	
Список студентов группы		V001	
Наименование структурных единиц	Обозначение	Длина в знаках	Диапазон
Номер группы	number	X(6)	«А»–«Я»; «а»–«я»; «0»–«9» ; «-»
Ф.И.О. студента	fio	A(15)	«А»–«Я»; «а»–«я»; «.»; «-»
Размер стипендии	grants	9(5).9(2)	0.00–30000.00

Заголовки столбцов и строк таблицы следует писать с прописной буквы, а подзаголовки – со строчной буквы, если они составляют одно предложение с заголовком, или с прописной буквы, если они имеют самостоятельное значение. В конце заголовков и подзаголовков таблиц точки не ставят. Заголовки и подзаголовки указывают в единственном числе (см. таблицу 6).

Таблица 6

Наименование величины	Единица измерения	Сокращенное обозначение		Размер единицы в СИ
		русское	латинское	
Масса	Тонна	т	t	$1 \cdot 10^{3}$ кг
Объем	Литр	л	L	$1 \cdot 10^{-3}$ м^3

Таблицы сверху, слева, справа и снизу ограничивают линиями. Лучше использовать сплошную одинарную тонкую линию. Текст в таблице оформляется по тем же параметрам, что и в пояснительной записке. В случае большого объема табличных данных допускается оформлять шрифтом меньшего размера, чем основной текст (например, 12 или 11 пт), с одинарным межстрочным интервалом.

Если строки таблицы выходят за формат страницы, ее делят на части, помещая одну часть под другой. При этом нумеруют арабскими цифрами столбцы в первой части таблицы.

Во второй и последующих частях таблицы вместо названий столбцов указывают их номера. Ниже показан пример оформления таблицы, имеющей разрыв (таблица 7).

Таблица 7

Перечень входных сообщений

Идентификатор	Форма представления	Наименование	Периодичность	Поставщик	Число строк
1	2	3	4	5	6
F002	Файл	Файл исходных данных	по запросу	Секретарь	999
V005	Видеокадр	Данные о студенте	по запросу	Секретарь	8

Продолжение таблицы 7

1	2	3	4	5	6
V006	Видеокадр	Критерий для поиска по номеру группы	по запросу	Методист	4
V007	Видеокадр	Критерий для поиска по фамилии	по запросу	Методист	4

Обозначение единицы физической или денежной величины, общей для всех данных в строке (или столбце), следует указывать после ее наименования.

Заменять кавычками повторяющиеся в таблице цифры, математические знаки, знаки процента и номера, обозначение марок материалов и типоразмеров изделий, обозначения нормативных документов не допускается.

При отсутствии отдельных данных в ячейках таблицы следует ставить тире.

3 ТРЕБОВАНИЯ К ПРОГРАММНОМУ КОДУ

Программные документы должны включать:

- текст программы, оформленный по ГОСТ 19.401;
- описание программы, выполненное по ГОСТ 19.402.

Текст программы должен быть написан на алгоритмическом языке C# и реализован в среде программирования Visual Studio.

Основные конструкции, операторы и правила создания программ на языке C# описаны в литературных источниках [2-9].

В программе, входящей в состав курсового проекта, необходимо в соответствии с полученным заданием организовать выполнение действий со свободным выбором пунктов меню, в которых предусмотреть следующие возможности:

- создание или дополнение данных в файл (в зависимости от типа файла и исходных данных);
- вывод содержимого исходного файла на экран;
- вывод в текстовый файл или на экран монитора данных (или списка) в зависимости от введенных критериев в запросе;
- выход из программы.

Кроме того, программа должна быть снабжена подсказками, необходимыми для работы пользователя, и справкой, а текст программы должен содержать комментарии, описывающие назначение пользовательских типов данных, переменных, процедур, функций или отдельных частей программы.

Ниже приведены темы курсового проекта по вариантам и задания для их выполнения.

4 ПОРЯДОК РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

Структура программы должна быть разработана в соответствии с технологией ORM Entity Framework Code First. Результатом разработки должно

быть оконное приложение, интерфейс которого соответствует одной из двух технологий: Windows Forms или WPF.

Ниже приведен пример разработки программы для предметной области «Библиотека», в которой автоматизируется процесс регистрации читателей, ввод данных о книгах и их выдаче читателям.

4.1 Разработка модели данных и создание БД Library (библиотека)

4.1.1 Теоретическое описание

Количество и назначение таблиц в базе данных, поля таблиц зависят от постановки задачи (технического задания), от типа приложения и категории пользователей, которые будут с ним работать.

База данных будет состоять из трех таблиц, содержащих данные о читателях, книгах в библиотечном фонде и выданных читателям.

Пояснения по назначению таблиц, их атрибутов (полей или свойств класса) представлены в таблицах 1-3.

Таблица 8

Readers	Читатель	Тип поля в классе	Тип поля и его свойства в БД
Id	Номер билета	Int32	int, PK, not null
SecondName	Фамилия	String	Nvarchar(30), not null
FirstName	Имя	String	Nvarchar(30), not null
MiddleName	Отчество	String	Nvarchar(30), null
DateOfBirth	Дата рождения	DateTime	Datetime, not null
Telephone	Телефон	String	Nvarchar(20), null
Email	Электронная почта	String	Nvarchar(max), null
Street	Улица	String	Nvarchar(30), null
House	Дом	String	Nvarchar(10), null
Flat	Квартира	Int32	Int, null
City	Город	String	Nvarchar(30), null
ZipCode	Почтовый Индекс	Int32	Int, null

Таблица 9

Books	Каталог книг	Тип поля в классе	Тип поля и его свойства в БД
Id	Код книги	Int32	int, PK, not null
Title	Название	String	Nvarchar(30), not null
Author	Автор	String	Nvarchar(50), not null

Таблица 10

ReaderCards	Карточка читателя	Тип поля в классе	Тип поля и его свойства в БД
Id	Код записи	Int32	int, PK, not null
ReaderId	Номер читателя	Int32	Int, not null
BookId	Код книги	Int32	Int, not null
StartDate	Дата выдачи	DateTime	Datetime, not null

Примечание: not null – запретить пустые ячейки; null – разрешить пустые ячейки для поля.

Должна получиться схема (диаграмма) базы данных, представленная на рисунке :

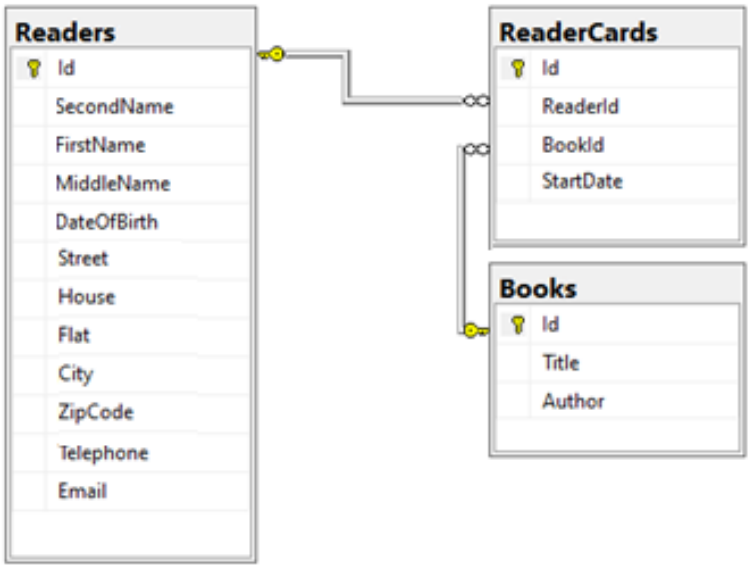
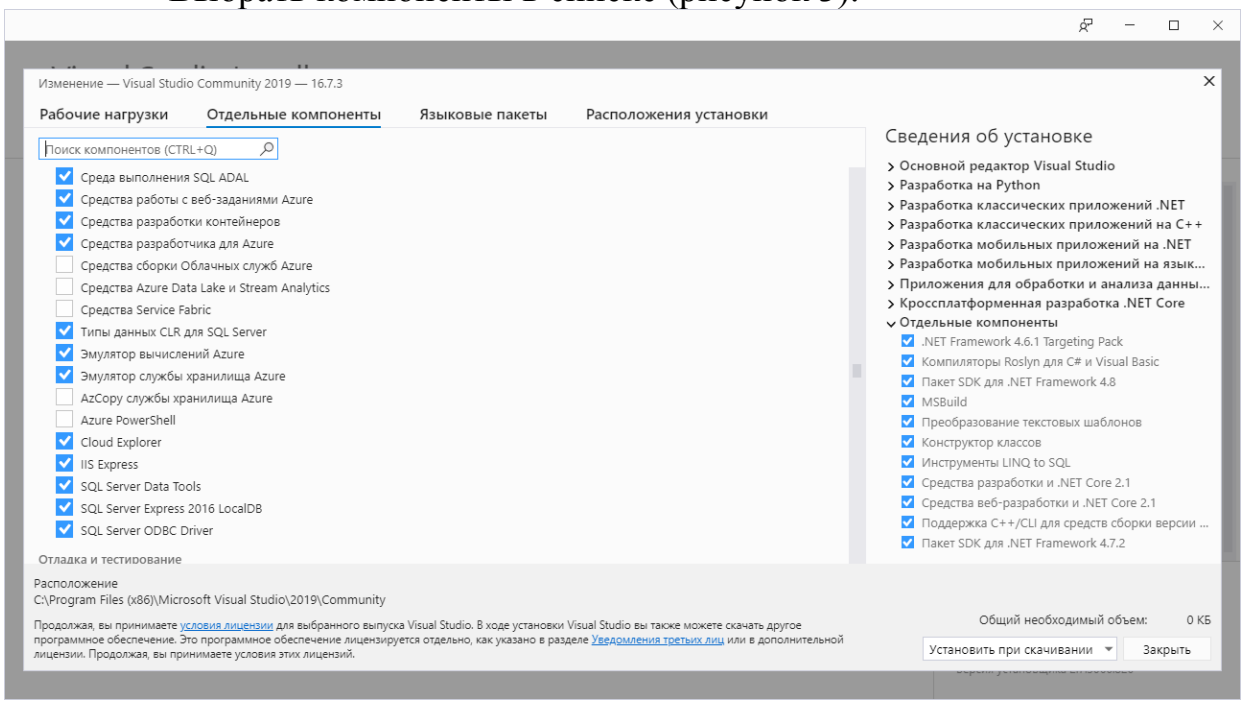


Рисунок 2 – Схема базы данных

4.1.2 Задания

1. Запустить **Visual Studio**.
2. Добавить к приложению компоненты (если не установлены) для работы с СУБД (**SQL Server Express LocalDB**), **Диспетчер пакетов Nuget** и **Конструктор классов**. Для этого:
   – Вызвать установщик (**Visual Studio Installer** из меню "Пуск") или выполнить команду **Средства – Получить средства и компоненты** … – вкладка **Отдельные компоненты**;
   – Выбрать компоненты в списке (рисунок 3).



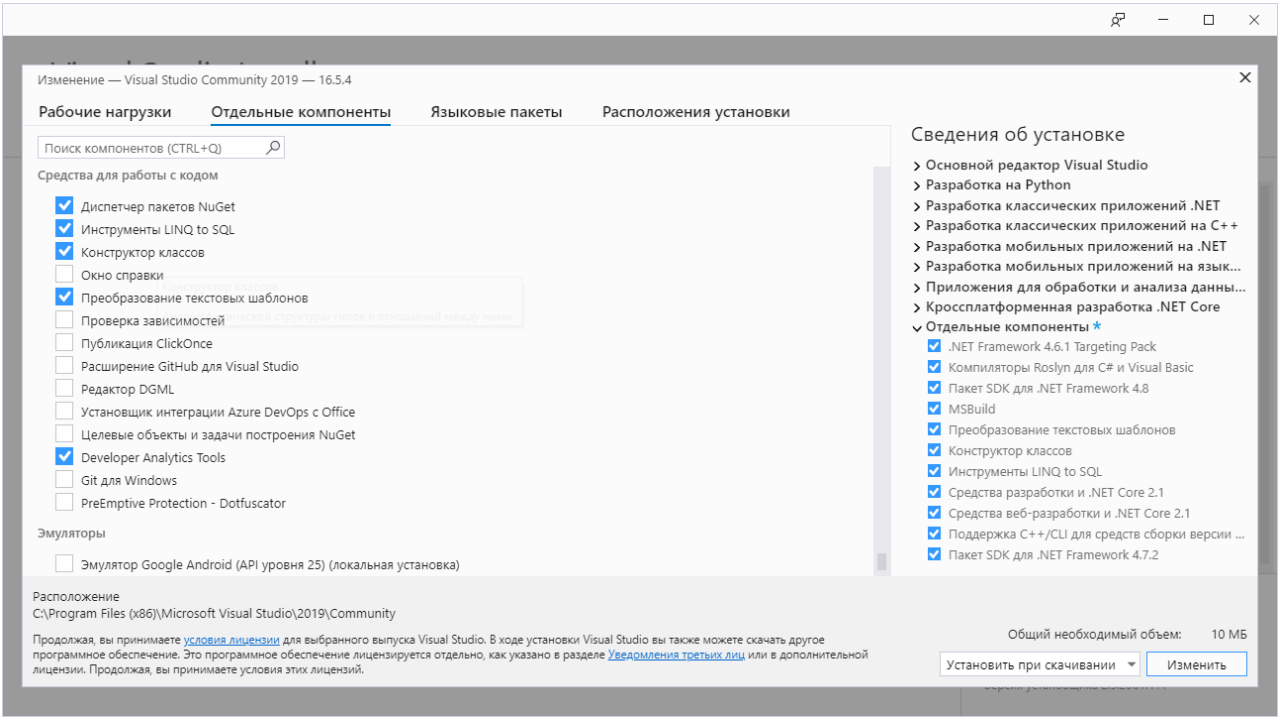


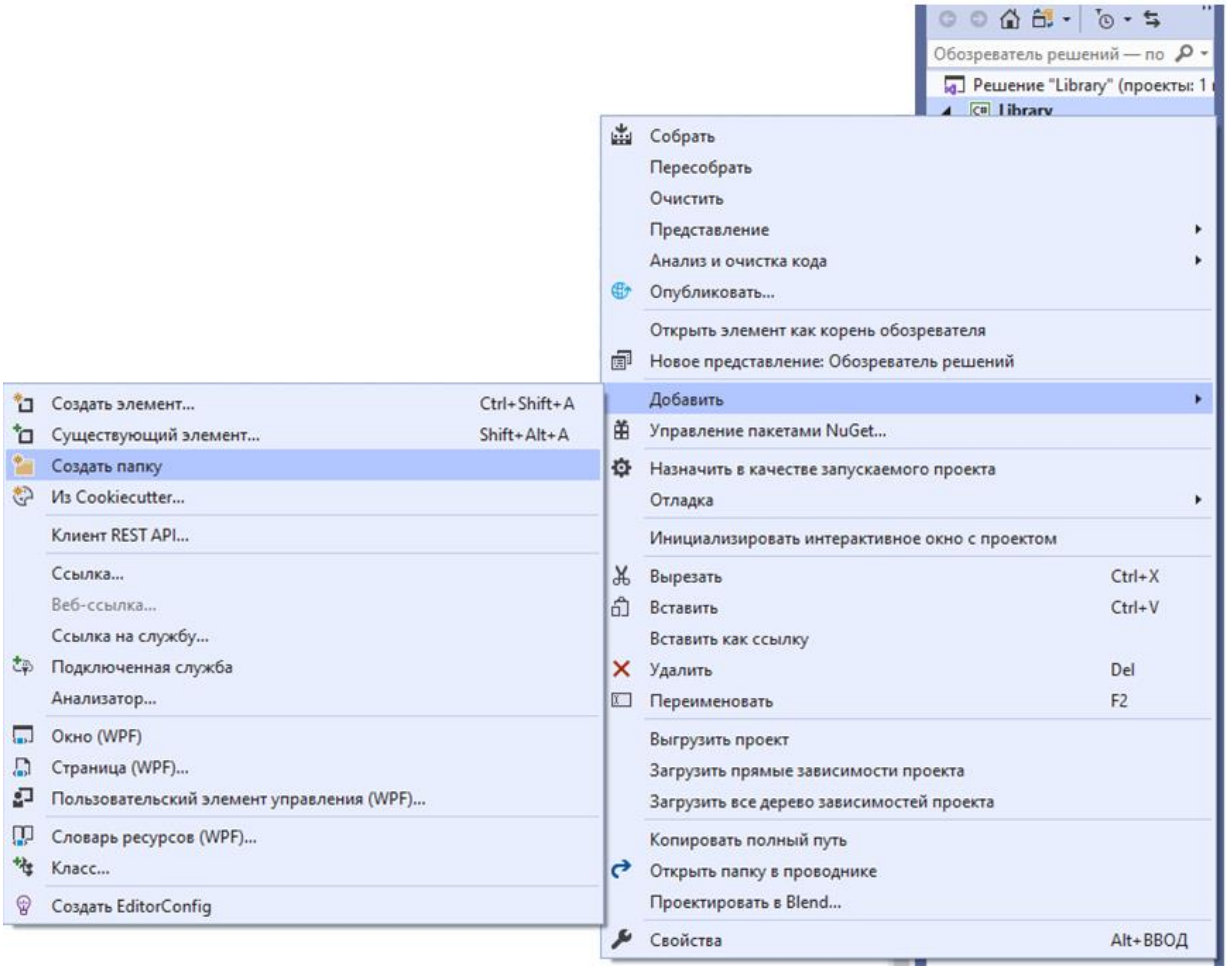
Рисунок 3 – Диалоговые окна для добавления компонентов к среде разработки

4.1.3 Полезные ссылки:

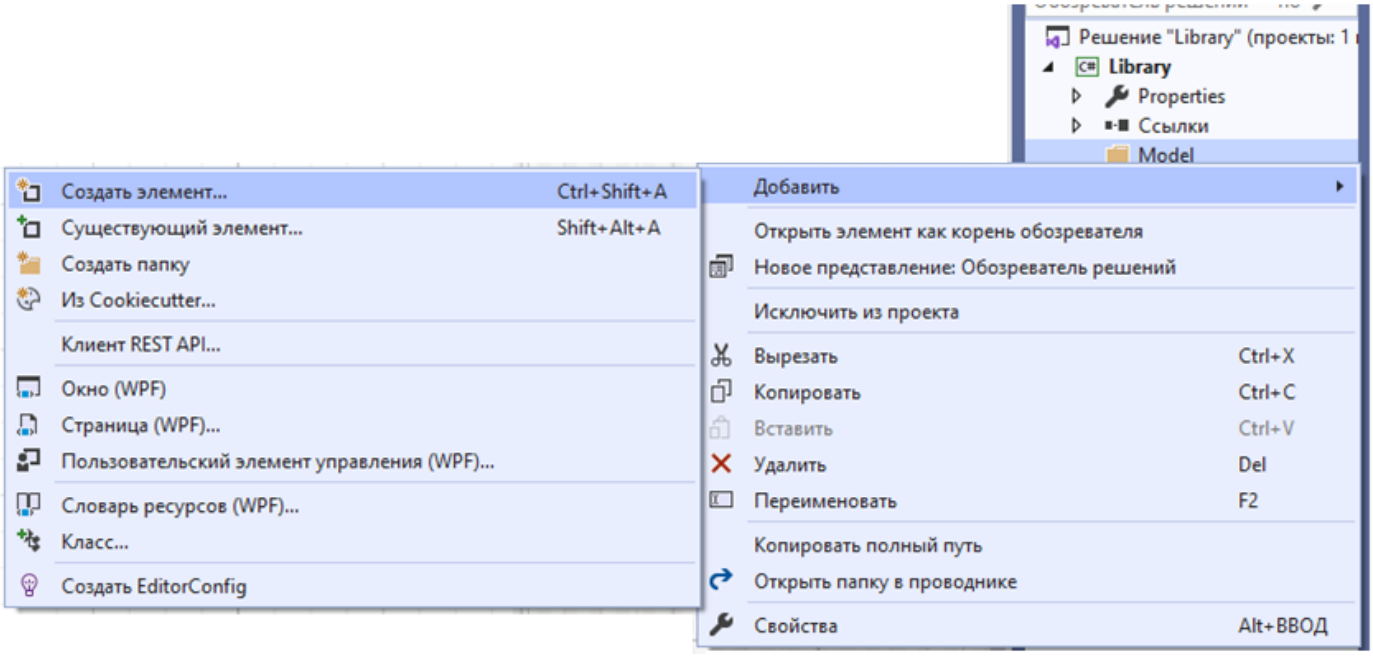
1. Entity Framework – https://docs.microsoft.com/ru-ru/ef/ef6/ https://docs.microsoft.com/ru-ru/ef/ef6/modeling/code-first/workflows/new-database
2. Модель БД и миграции https://www.youtube.com/watch?v=Od1X5o7za9k
3. Скачать и установить Microsoft SQL Server Management Studio - 18.4: https://docs.microsoft.com/ru-RU/sql/ssms/download-sql-server-management-studio-ssms?view=aps-pdw-2016

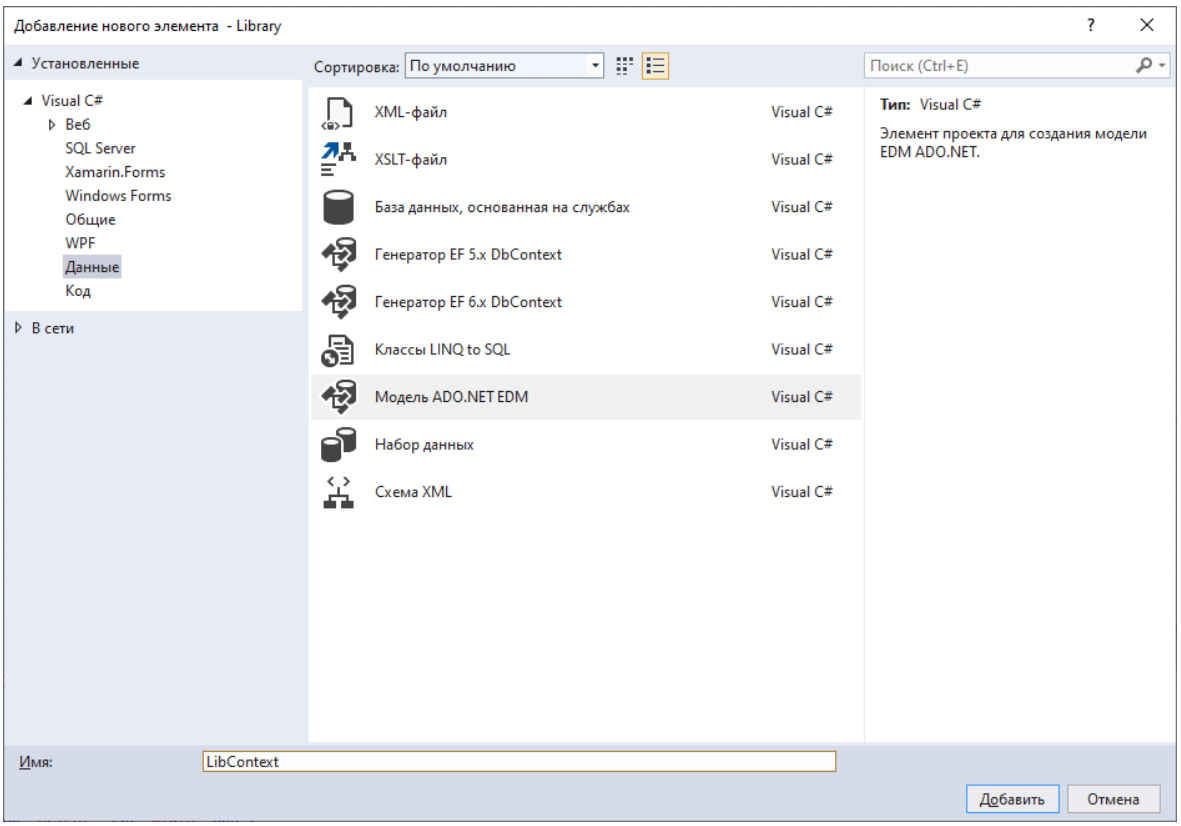
4.1.4 Добавление в проект классов для создания модели данных и соответствующей ей базы данных

1. Создать проект **Windows Forms** (.NET Framework) или **WPF** (.NET Framework) в соответствии с выбранным типом интерфейса.
2. Ввести имя решения и проекта **Library**.
3. Разделить логику приложения и модель БД.
4. Переименовать **Form1** в **MainForm** (с помощью контекстного меню).
5. Для проекта **Library** вызвать контекстное меню для **Добавить – Создать папку**, ввести имя **Model**.

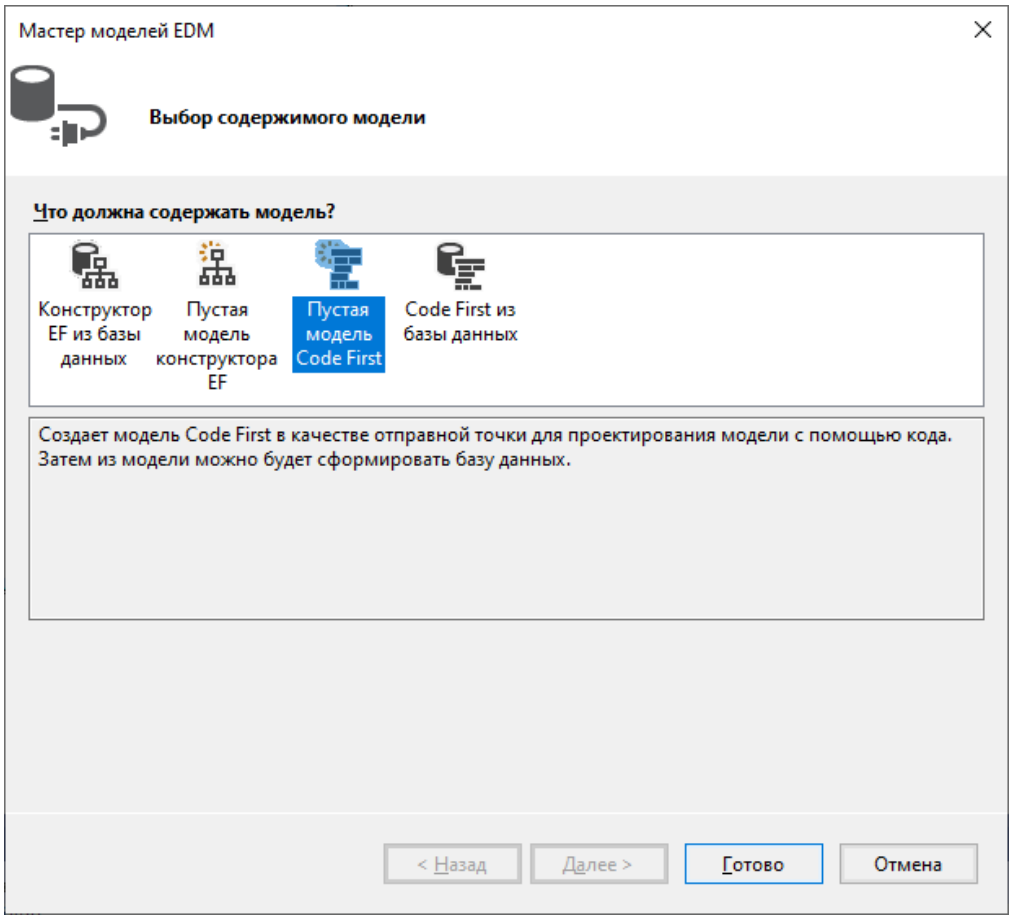


6. Вызвать контекстное меню для папки **Model** – **Добавить** – **Создать элемент**… – **Данные** – **Модель ADO.NET EDM** – **Добавить** – **Пустая модель Code First**.





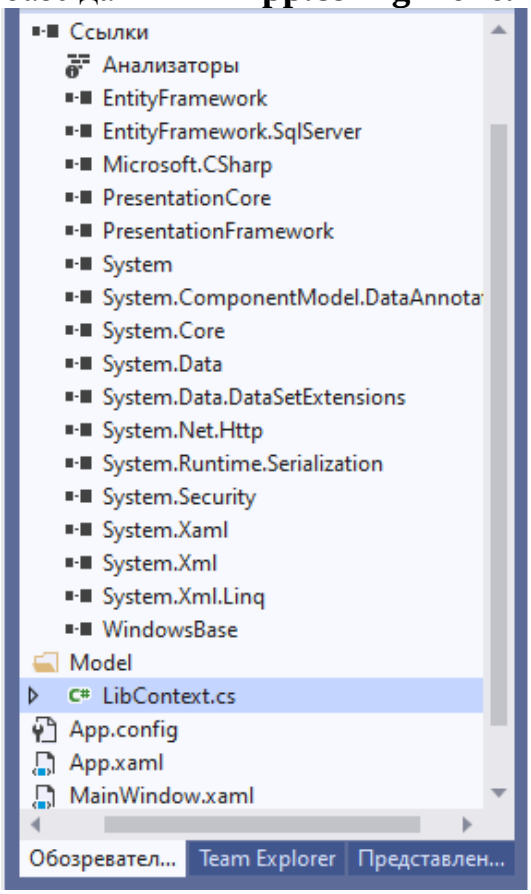
7. Ввести имя для контекста (связи модели и базы данных) **LibContext**. Нажать кнопку **Добавить**.



✓ Содержимое файла контекста, созданного автоматически.

```
namespace Library.Model
{
    using System;
    using System.Data.Entity;
    using System.Linq;

    public class LibContext : DbContext
    {
        // Контекст настроен для использования строки подключения "LibContext" из файла конфигурации
        // приложения (App.config или Web.config). По умолчанию эта строка подключения указывает на базу данных
        // "Library.Model.LibContext" в экземпляре LocalDb.
        //
        // Если требуется выбрать другую базу данных или поставщик базы данных, измените строку подключения "LibContext"
        // в файле конфигурации приложения.
        public LibContext()
            : base("name=LibContext")
        {
        }

        // Добавьте DbSet для каждого типа сущности, который требуется включить в модель. Дополнительные сведения
        // о настройке и использовании модели Code First см. в статье http://go.microsoft.com/fwlink/?LinkId=390109.

        // public virtual DbSet<MyEntity> MyEntities { get; set; }
    }

    //public class MyEntity
    //{
    //    public int Id { get; set; }
    //    public string Name { get; set; }
    //}
}
```

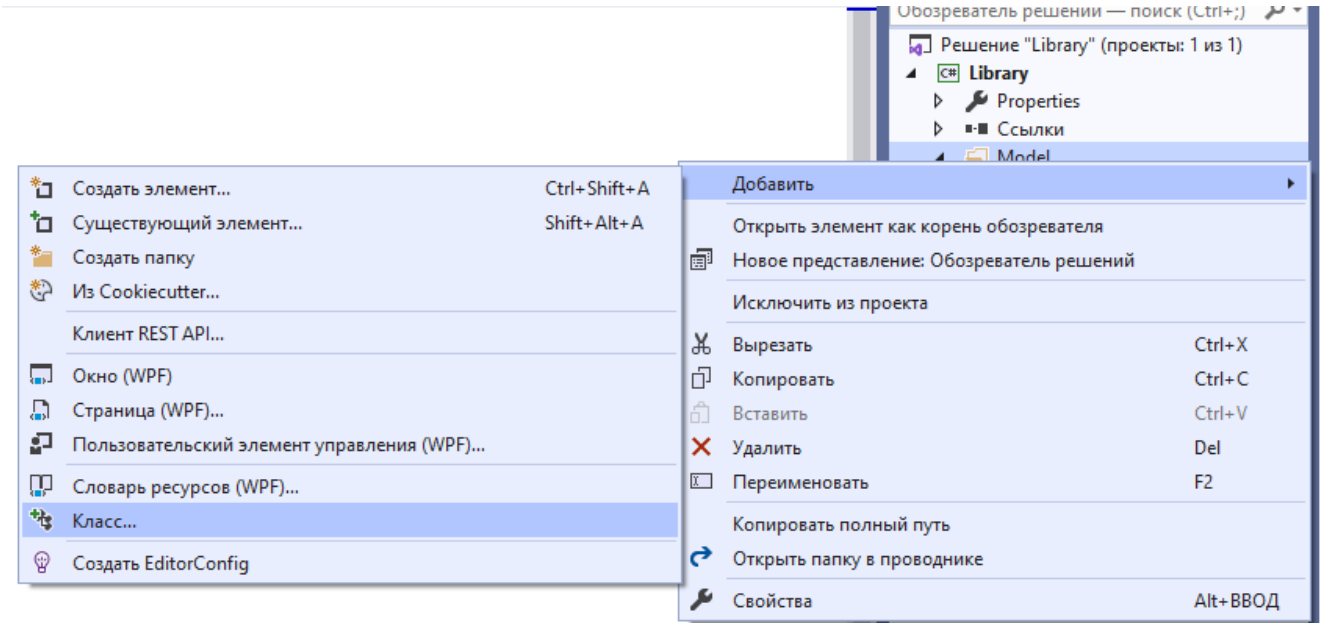
✓ Автоматически будут добавлены: класс контекста, строка подключения к базе данных в **App.config** и библиотека **EntityFramework**.



8. В файле **App.config** изменить имя БД, задать `catalog=Library;`

```
<connectionStrings>
  <add name="LibContext" connectionString="data source=(LocalDb)\MSSQLLocalDB;initial catalog=Library.Model.LibContext;
</connectionStrings>
```

9. Создать классы будущей БД.



10. Добавить в каждый класс свойства, на основе которых будут формироваться таблицы базы данных, и задать их характеристики. Обратите внимание, что доступ к классу должен быть **public**.

Reader.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Library.Model
{
    public class Reader
    {
        public int Id { get; set; }

        [Required]  // Запрет null-значений.
        [MaxLength(30)]  // Максимальная длина.
        public string SecondName { get; set; }

        [Required]
        [MaxLength(30)]
        public string FirstName { get; set; }

        [MaxLength(30)]
        public string MiddleName { get; set; }

        [Required]
        public DateTime DateOfBirth { get; set; }

        [MaxLength(20)]
        public string Telephone { get; set; }

        public string Email { get; set; }

        [MaxLength(30)]
        public string Street { get; set; }

        [MaxLength(10)]
        public string House { get; set; }

        // Знак ? обозначает, что null-значений разрешены.
        public int? Flat { get; set; }

        [MaxLength(30)]
        public string City { get; set; }

        // Разрешить null-значений.
        public Nullable<int> ZipCode { get; set; }

    }
}

```

Book.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Library.Model
{
    public class Book
    {
        public int Id { get; set; }

        [Required]
        [MinLength(5)]
        [MaxLength(30)]
        public string Title { get; set; }

        [Required]
        [StringLength(50, MinimumLength = 5)]
        public string Author { get; set; }

    }
}
```

ReaderCard.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Library.Model
{
    public class ReaderCard
    {
        public int Id { get; set; }

        public int ReaderId { get; set; } // Внешний ключ.

        public int BookId { get; set; } // Внешний ключ.

        [Required]
        public DateTime StartDate { get; private set; }

        public virtual Reader Reader { get; set; } // Навигационное свойство.

        public virtual Book Book { get; set; } // Навигационное свойство.

        // virtual используется для "ленивой загрузки" - lazy loading.
        // При таком способе подгрузки при первом обращении к объекту,
        // если связанные данные не нужны, то они не подгружаются.
        // Однако при первом же обращении к навигационному свойству
        // эти данные автоматически подгружаются из бд.
    }
}
```

11. Добавить в классы **Reader** и **BookCatalog** свойство для связи с классом **ReaderCard**:

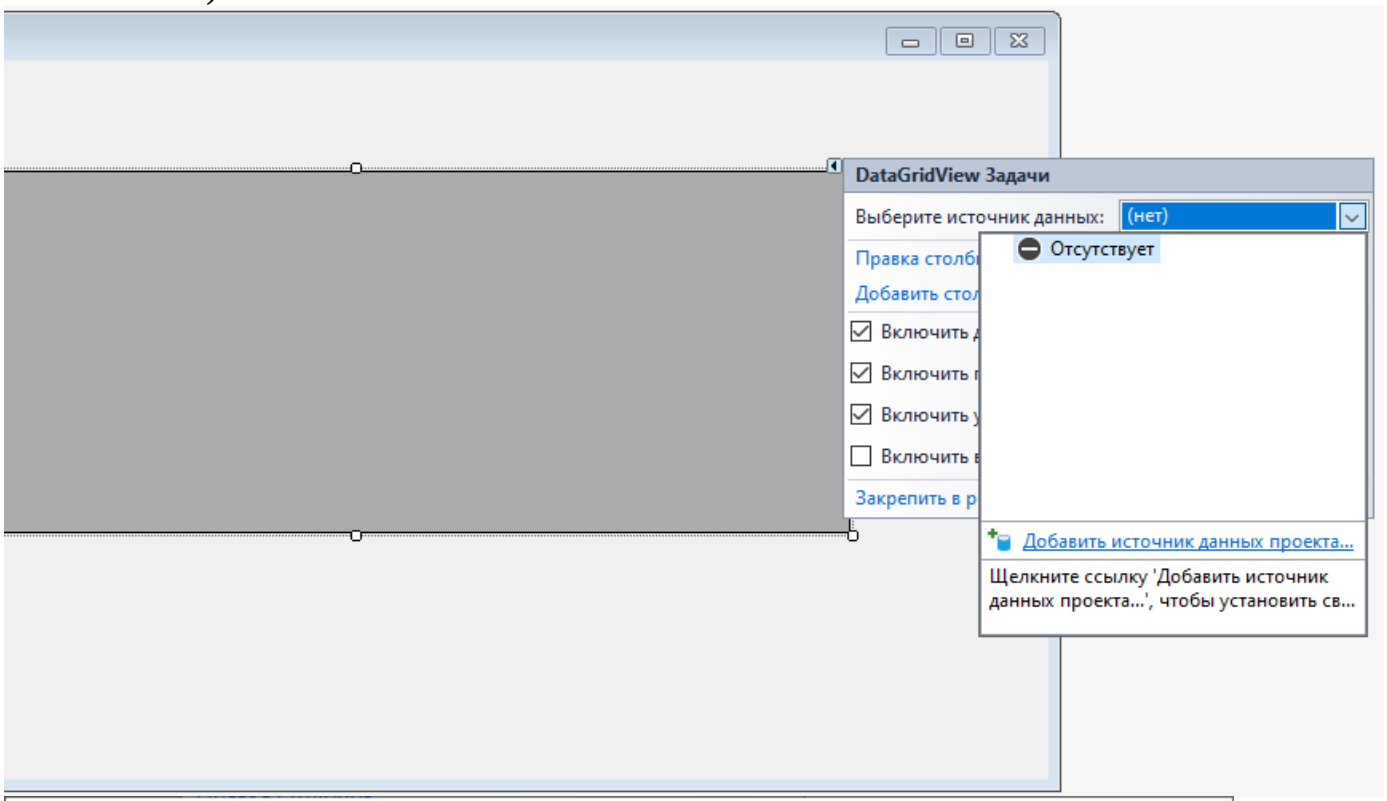
```
public virtual ICollection<ReaderCard> ReaderCards { get; set; }
```

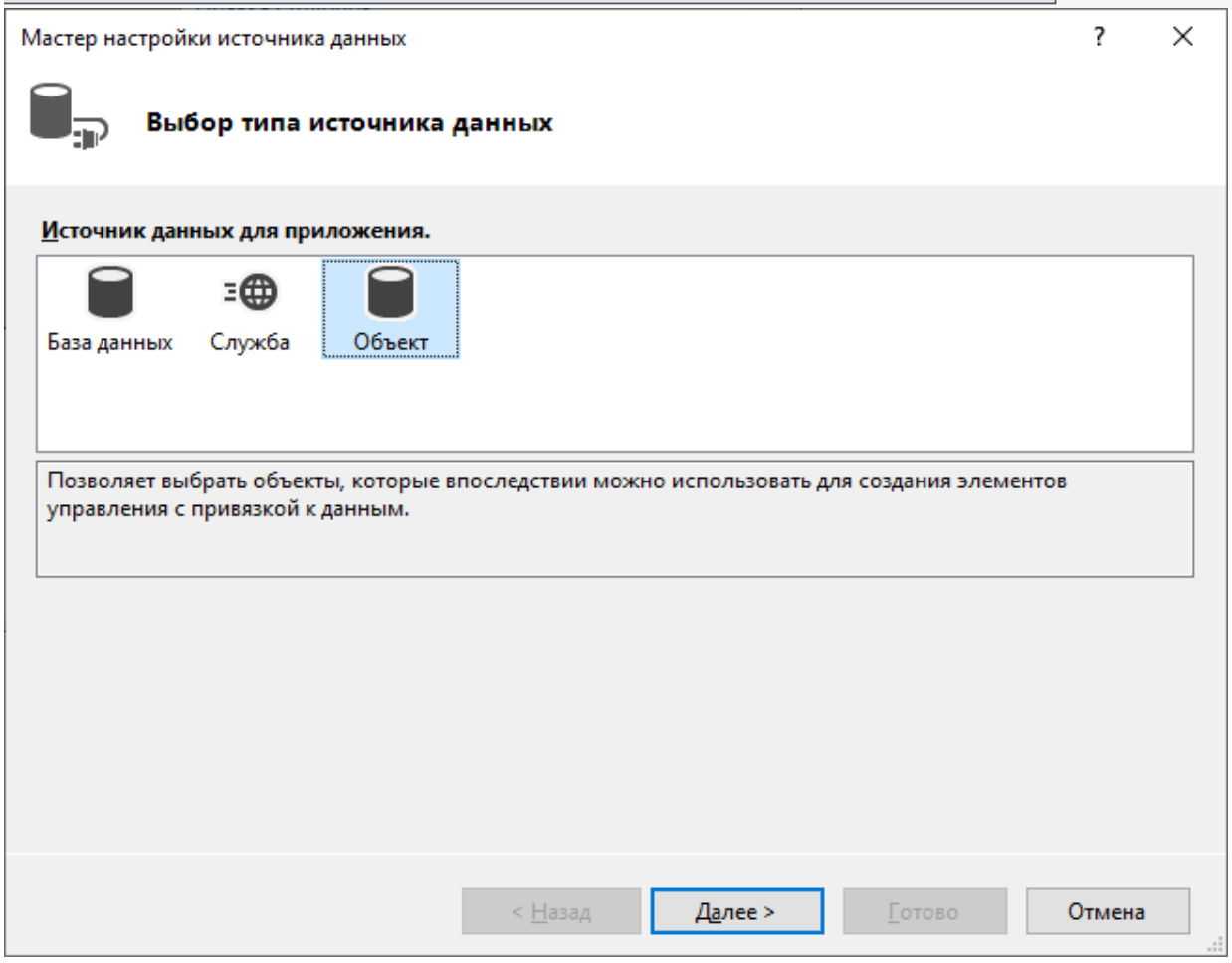
12. Добавить свойства (для связи с классами проекта) в класс контекста **LibContext**. Он должен иметь следующее содержимое:

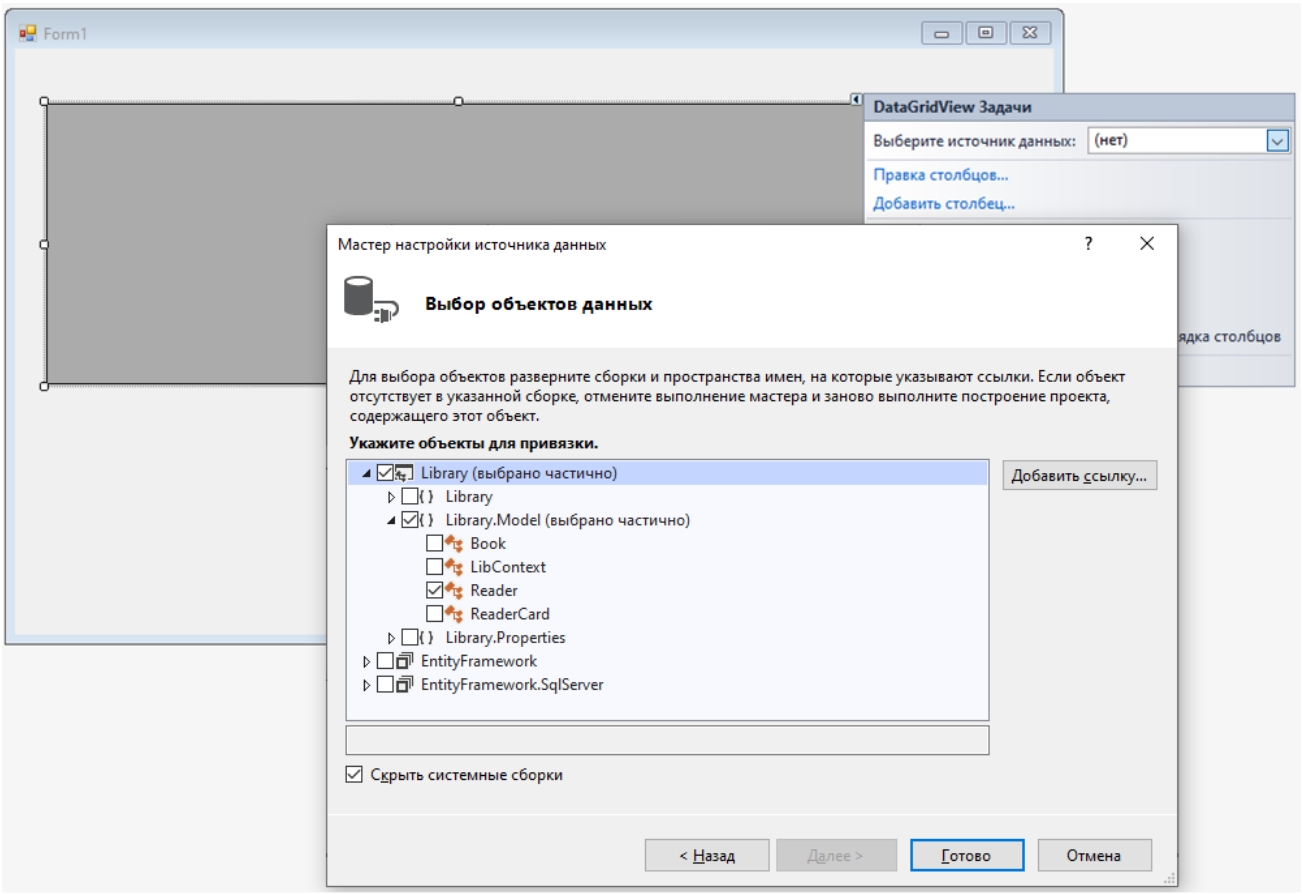
```
public class LibContext : DbContext
{
    public LibContext()
        : base("name=LibDBConnection")
    {
    }

    public DbSet<Reader> Readers { get; set; }

    public DbSet<Book> Books { get; set; }

    public DbSet<ReaderCard> ReaderCards { get; set; }

}
```

Обратите внимание на то, что класс контекста определяет таблицы БД.

13. Запустить программу на выполнение (она должна запуститься без ошибок).
14. Добавить на форму таблицу и связать ее с объектом класса (например, Reader).







4.2 Разработка интерфейса программы в соответствии с технологией Windows Forms и Entity Framework

4.2.1 Разработка программного интерфейса Library (библиотека)

1. Дописать команды для загрузки данных из таблиц БД в класс **MainForm**:

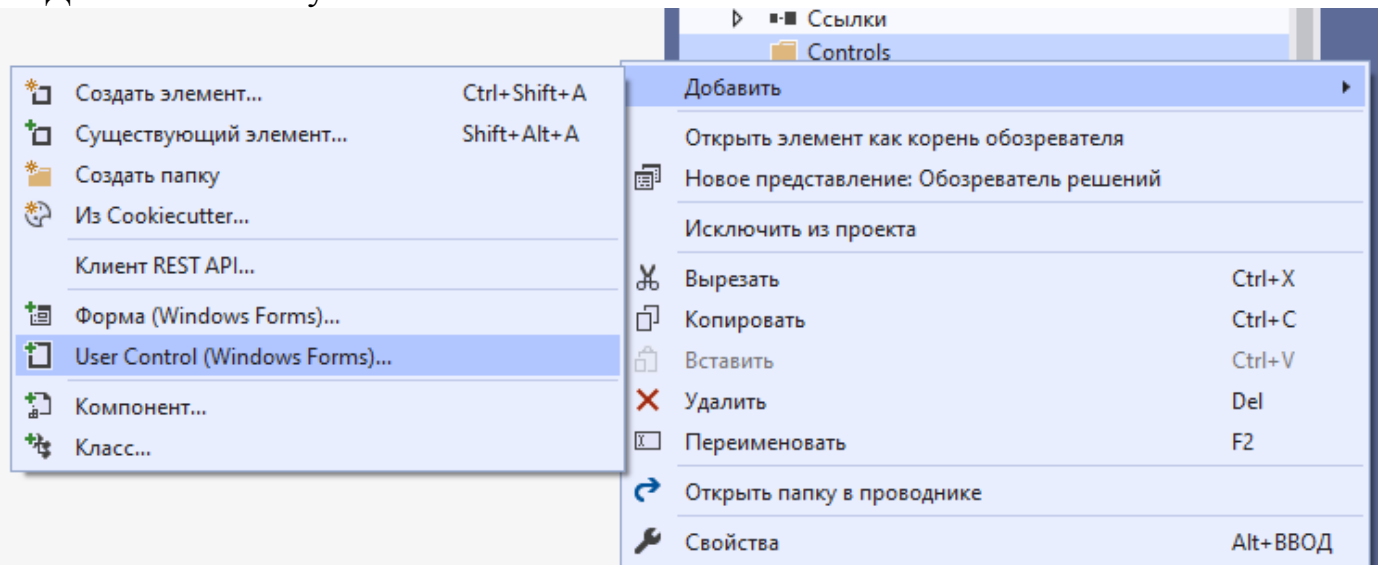
```
using System.Windows.Forms;

namespace Library
{
    public partial class MainForm : Form
    {
        LibContext db;
        public MainForm()
        {
            InitializeComponent();
            db = new LibContext();
            db.Readers.Load();
            dataGridView1.DataSource = db.Readers.Local.ToBindingList();
        }
    }
}
```

2. Снова запустить программу на выполнение.
   После этого будет создана база данных **Library**. Найти ее можно в **Обозревателе объектов SQL Server** или в программе **Microsoft SQL Server Management Studio**.

3. В класс **Reader** добавить два вычисляемых свойства (изменений в БД не будет):

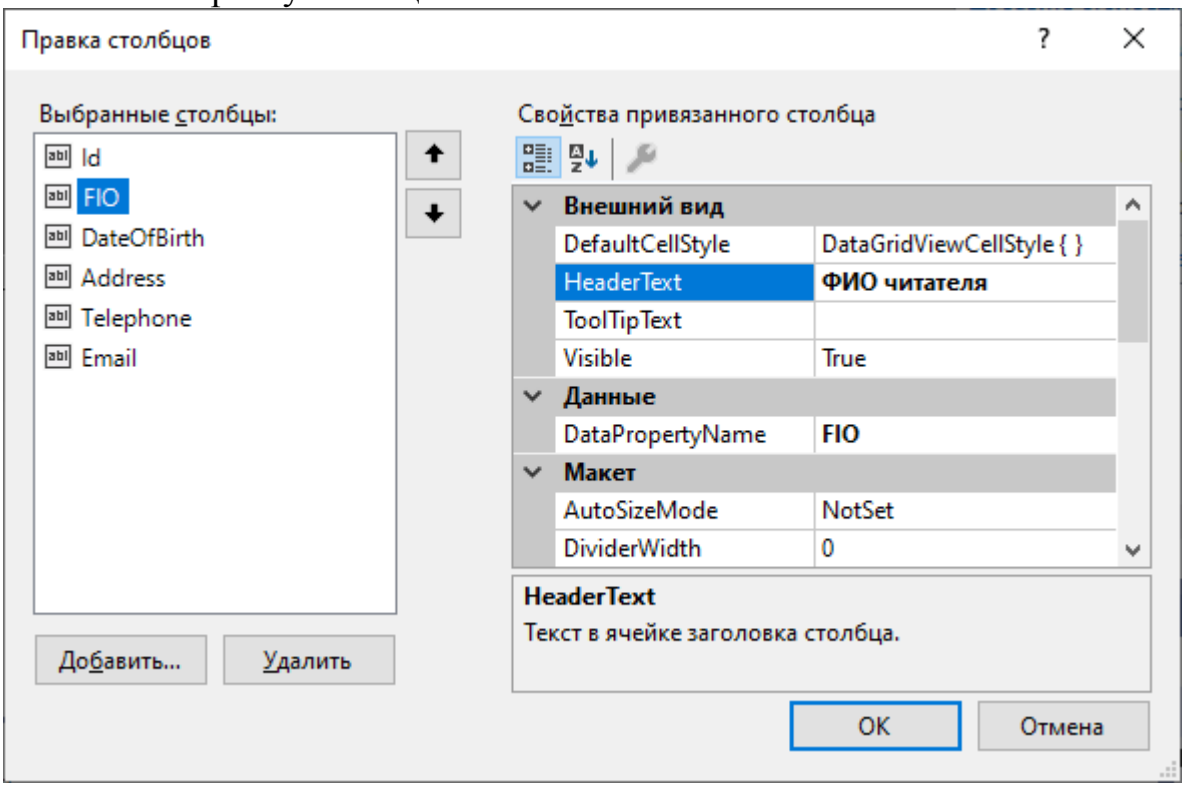
```
public string FIO => SecondName + " " + FirstName + " " + MiddleName;

public string Address => Street + ", " + House + (Flat != null ? "-" + Flat : "");
```

4. Запустить программу на выполнение (для фиксации изменений в проекте).
5. Закрыть исполняемый файл.
6. Перейти к конструктору **MainForm**.
7. Изменить заголовок формы.
8. Удалить таблицу с данными о читателях (созданную при выполнении предыдущей практической работы).
9. Добавить главное меню (**MenuStrip**), содержащее пункты и подпункты: **Справочники** (**Читатели**, **Книги**), **Карточки читателей**, **Отчеты**, **Справка** и **Выход**.
10. Добавить на форму контейнер **Panel**, в котором будут располагаться другие объекты (таблицы, кнопки и др.) в зависимости от выбранной команды.
11. Создать папку **Controls** в проекте **Library**.
    Примечание: в этой папке будут храниться объекты, которые можно «внедрять» на формы. Один и тот же UserControl может быть расположен на разных формах. Используются для того, чтобы не загромождать код для формы, разграничить функционал и упростить восприятие кода программы.
12. Добавить в папку **Controls** объект **UserControl**:

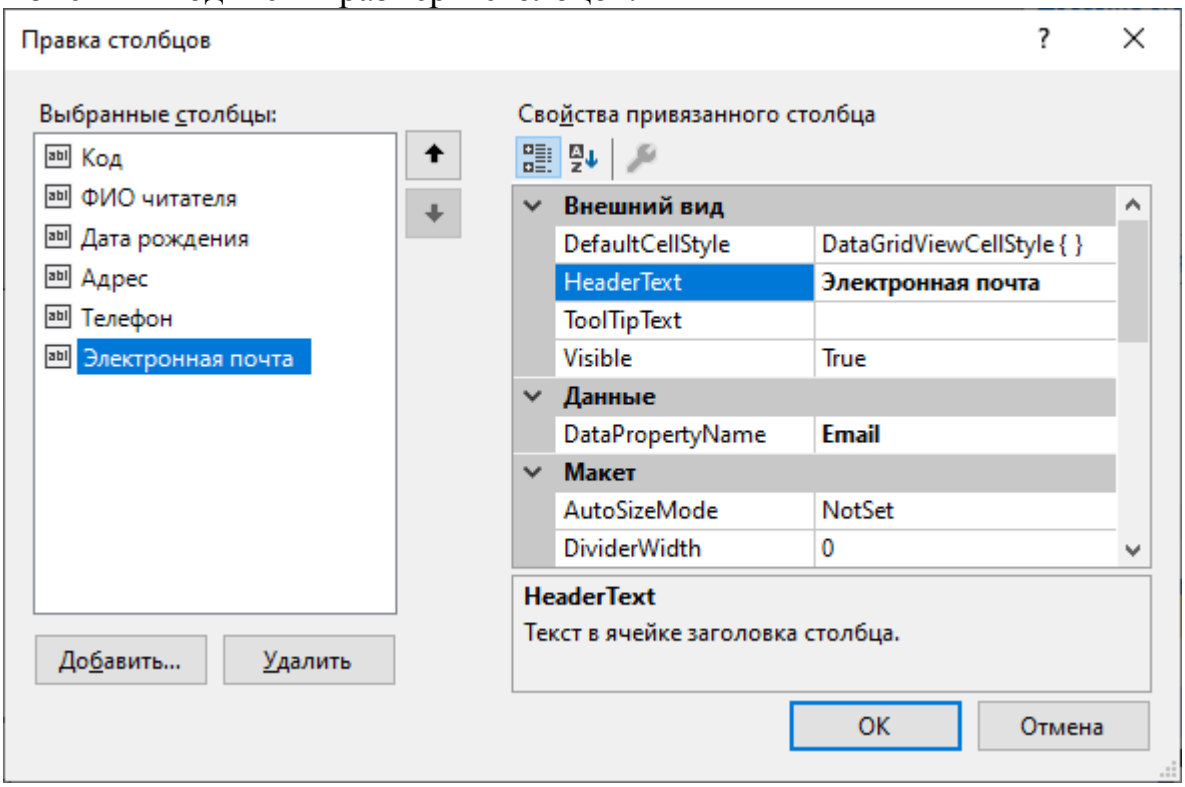


13. Ввести имя **ReaderControl.cs**.
14. Изменить размеры объекта UserControl и другие свойства, например, цвет, шрифт (по желанию).
15. Добавить на **ReaderControl** заголовок таблицы (**Label**) **Список читателей** и таблицу (**dataGridView1**).

16. Добавить для таблицы **источник данных** – **Добавить источник данных проекта**… – **Объект – Library** – **Library.Model** – **Reader**.
17. Выполнить правку столбцов:



18. Изменить подписи и размеры столбцов:



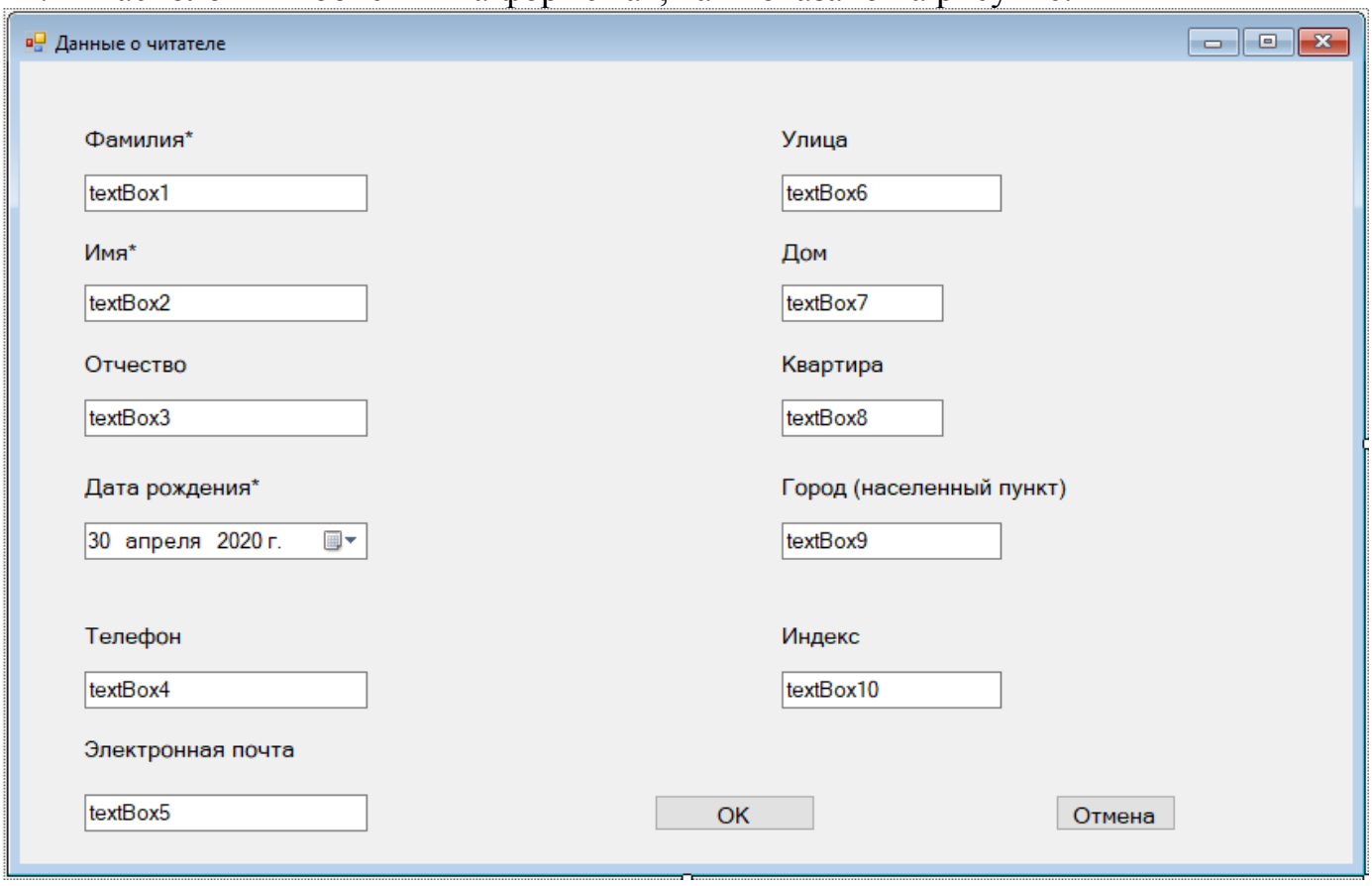
Примечание: размеры столбцов следует подобрать после запуска программы на выполнение и ввода данных.

19. Изменить свойства таблицы: для **AllowUserToAddRows** значение **False**, а для **SelectionMode** значение **FullRowSelect**, чтобы можно было выделять всю строку.

20. Вырезать код для работы с таблицей читателей из **MainForm**.cs и вставить его в **ReaderControl**.cs, добавить необходимые ссылки (using …).
21. Для пункта меню **Читатели** создать метод для обработки события (**Click**): переименовать (вызвать команду из контекстного меню) метод в соответствии с принятыми «правилами именования», записать код, добавив необходимые ссылки.

```
private void ReaderClick(object sender, EventArgs e)
{
    var uc = new ReaderControl();
    if (panel1.Controls.Count > 0)
        panel1.Controls.RemoveAt(0);
    panel1.Controls.Add(uc);
}
```

22. Добавить в проект (в папку **Forms**) форму для редактирования данных о читателе (**ReaderEdit**).
23. Изменить ее заголовок.
24. Расположить объекты на форме так, как показано на рисунке:



Примечание: 1) В поля (textBox) текст вводить не надо! 2) Звездочками обозначены поля, обязательные для заполнения (не допускающие значения null).

Для кнопок задать следующие значения свойств:

button1 **Text** = "**OK**"; **DialogResult** = **OK**;

button2 **Text** = "**Отмена**"; **DialogResult** = **Cancel**.

Для всех полей, календаря и button1 изменить свойство:
Modifiers = **Protected Internal**.

25. Добавить на **ReaderControl** три кнопки: **Добавить**, **Изменить**, **Удалить**.
26. Ввести программный код для кнопок:

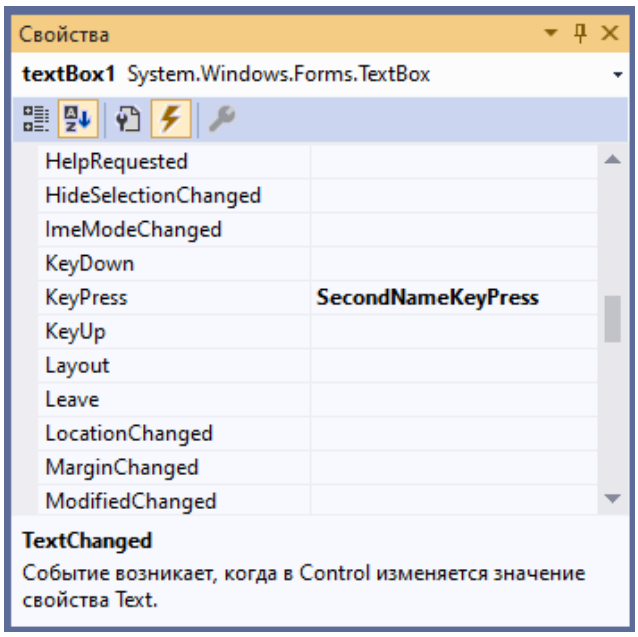
```
private void AddClick(object sender, EventArgs e) // кнопка "Добавить"
{
    ReaderEditForm readerForm = new ReaderEditForm();
    readerForm.button1.Text = "Добавить";
    Reader reader = new Reader();
    DialogResult result = readerForm.ShowDialog(this);
    SaveData(readerForm, result, reader, "добавлены");
}

private void EditClick(object sender, EventArgs e)  // кнопка "Изменить"
{
    if (dataGridView1.SelectedRows.Count > 0)
    {
        int index = dataGridView1.SelectedRows[0].Index;
        bool converted = Int32.TryParse(dataGridView1[0, index].Value.ToString(), out int id);
        if (converted == false) return;

        Reader reader = db.Readers.Find(id);
        ReaderEditForm readerForm = new ReaderEditForm();
        readerForm.button1.Text = "Изменить";
        readerForm.textBox1.Text = reader.SecondName;
        readerForm.textBox2.Text = reader.FirstName;
        readerForm.textBox3.Text = reader.MiddleName;
        readerForm.dateTimePicker1.Value = reader.DateOfBirth;
        readerForm.textBox4.Text = reader.Telephone;
        readerForm.textBox5.Text = reader.Email;
        readerForm.textBox6.Text = reader.Street;
        readerForm.textBox7.Text = reader.House;
        readerForm.textBox8.Text = reader.Flat.ToString();
        readerForm.textBox9.Text = reader.City;
        readerForm.textBox10.Text = reader.ZipCode.ToString();

        DialogResult result = readerForm.ShowDialog(this);
        SaveData(readerForm, result, reader, "изменены");
    }
}

private void DeleteClick(object sender, EventArgs e) // кнопка "Удалить"
{
    if (dataGridView1.SelectedRows.Count > 0)
    {
        int index = dataGridView1.SelectedRows[0].Index;
        bool converted = Int32.TryParse(dataGridView1[0, index].Value.ToString(), out int id);
        if (converted == false)
            return;

        Reader reader = db.Readers.Find(id);
        db.Readers.Remove(reader);
        db.SaveChanges();

        MessageBox.Show("Объект удален");
    }
}
```

```
private void SaveData(ReaderEditForm readerForm, DialogResult result, Reader reader, string str)
{
    if (result == DialogResult.OK)
    {
        try
        {
            reader.SecondName = readerForm.textBox1.Text;
            reader.FirstName = readerForm.textBox2.Text;
            reader.MiddleName = readerForm.textBox3.Text;
            reader.DateOfBirth = readerForm.dateTimePicker1.Value;
            reader.Telephone = readerForm.textBox4.Text;
            reader.Email = readerForm.textBox5.Text;
            reader.Street = readerForm.textBox6.Text;
            reader.House = readerForm.textBox7.Text;
            if (readerForm.textBox8.Text == "") reader.Flat = null;
            else reader.Flat = int.Parse(readerForm.textBox8.Text);
            reader.City = readerForm.textBox9.Text;
            if (readerForm.textBox10.Text == "") reader.Flat = null;
            else reader.ZipCode = int.Parse(readerForm.textBox10.Text);
            if (str == "добавлены") db.Readers.Add(reader);
            db.SaveChanges();

            MessageBox.Show("Данные о читателе " + str);
        }
        catch (Exception exept)
        {
            MessageBox.Show("Данные не " + str + ".\n " + exept.Message);
        }
    }
}
```

27. Создать методы-обработчики событий для полей ввода на форме **ReaderEdit**. Для этого:
    a. перейти к коду формы и ввести методы для обработки вводимых символов и строки:

```
private static void AllowLetters(KeyPressEventArgs e, char charCode)
{
    // Разрешить вводить только буквы
    // и символ backspace (код равен 8).
    if (!char.IsLetter(charCode) && charCode != 8)
        e.Handled = true;
}

private static void AllowNumbers(KeyPressEventArgs e, char charCode)
{
    // Разрешить вводить только цифры и символ backspace.
    if (!char.IsDigit(charCode) && charCode != 8)
        e.Handled = true;
}

private string StringConversion(string s) =>
            s.Substring(0, 1).ToUpper() + s.Substring(1).ToLower();
```

    b. создать событие **KeyPress** для поля **textBox1** и переименовать его в **SecondNameKeyPress**:



c. ввести код метода для обработки события:

```
private void SecondNameKeyPress(object sender, KeyPressEventArgs e)
{
    char charCode = e.KeyChar;
    AllowLetters(e, charCode);
    // 13 - код символа Enter используется
    // для перехода к следующему текстовому полю.
    if (charCode == 13)
    {
        string s = textBox1.Text;
        textBox1.Text = StringConversion(s);
        textBox2.Focus();
    }
}
```

d. создать аналогичные события **KeyPress** для всех остальных полей ввода.

Примечания: 1) Таблица с заголовком, расположенные на **ReaderControl**, являются выходной информацией (видеограммой) приложения. 2) Форма **ReaderEdit** представляет собой входную информацию (видеокадр). 3) Их структура должна быть описана в соответствующих пунктах постановки задачи (документации программы).

28. Аналогичным способом создать программные средства для работы с данными о книгах.

4.2.2 Отбор данных и сортировка в проекте Library (библиотека)

1. Добавить в класс **Reader**:

a. ***конструктор*** (чтобы не было сбоев при обращении к списку выданных книг):

```
public Reader()
{
    ReaderCards = new List<ReaderCard>();
}
```

b. ***перегрузку метода*** для преобразования объекта в строку данных:

```
public override string ToString()
{
    return FIO;
}
```

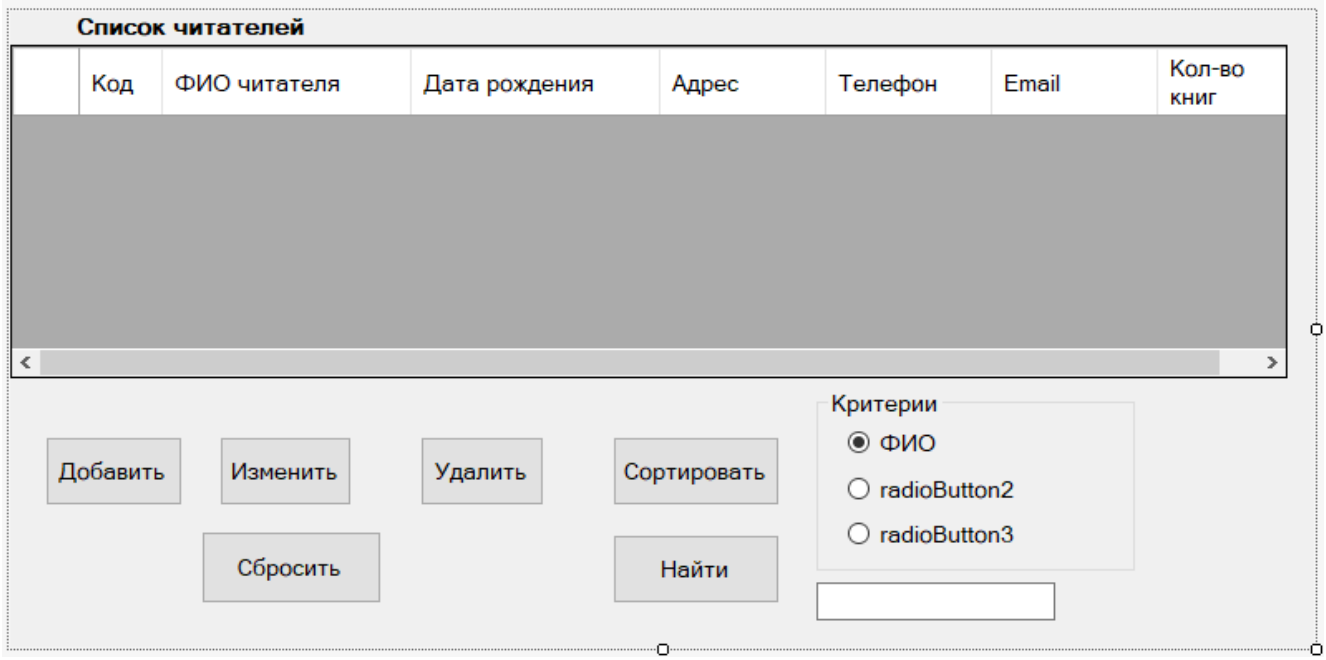
2. Добавить в классе **Book**:

```
public string Name => Author + " " + Title;

public Book()
{
    ReaderCards = new List<ReaderCard>();
}

public override string ToString()
{
    return Name;
}
```

3. Добавить в классе **ReaderCard** конструктор:

```
public ReaderCard()
{
    StartDate = DateTime.Today;
}
```

4. Добавить на **ReaderControl.cs** [Конструктор] элементы в соответствии с рисунком:

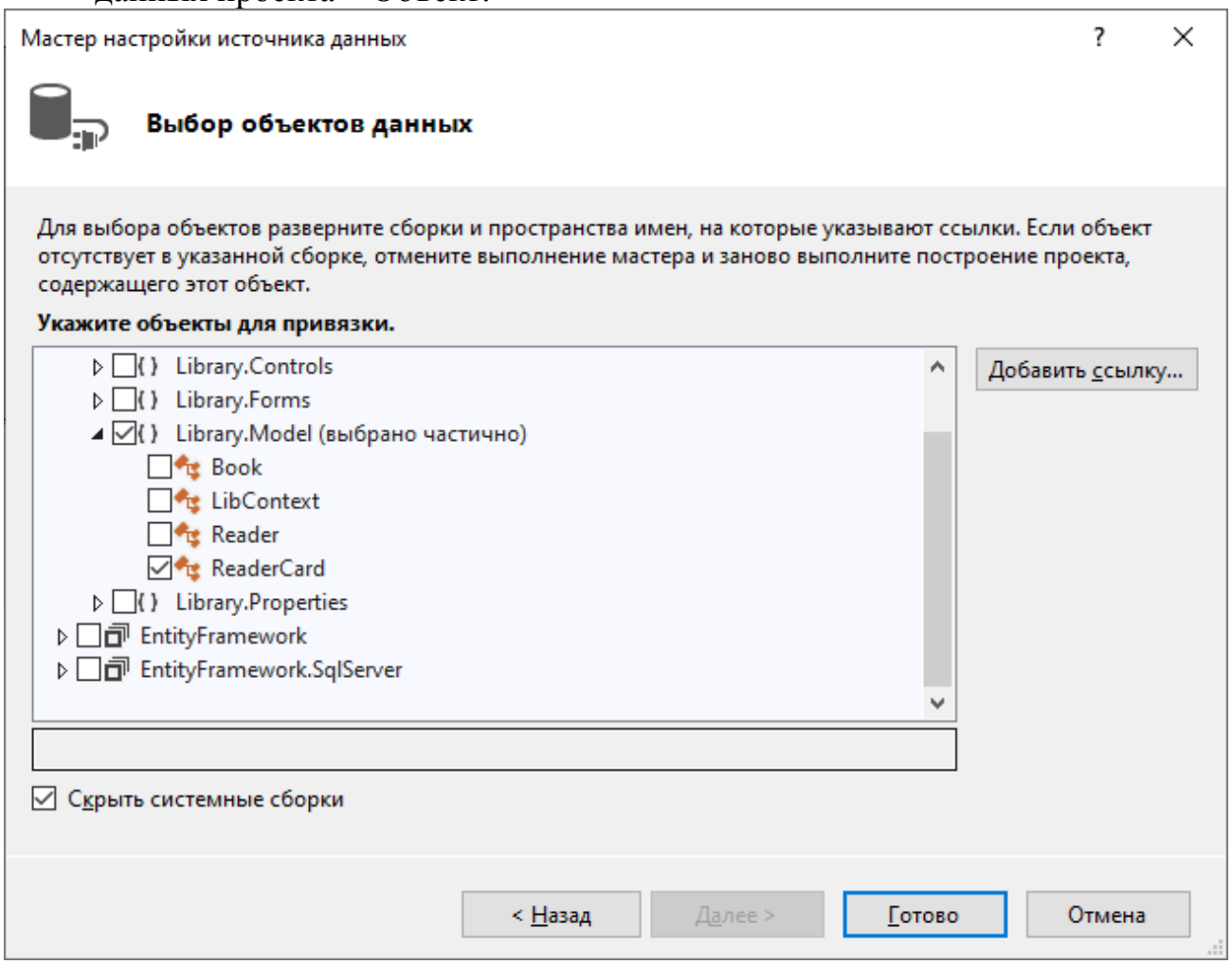


5. Добавить методы для обработки событий нажатия на кнопки:

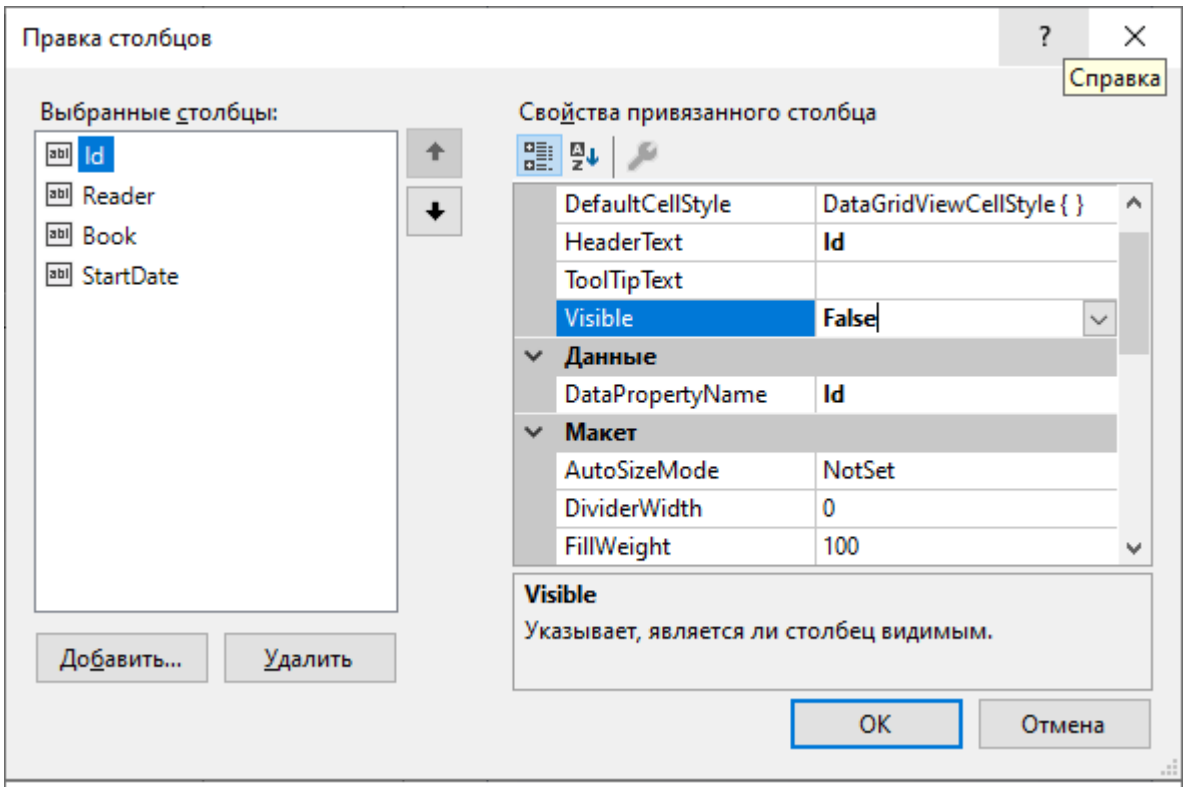
```
// Сортировка данных по ФИО читателя.
private void SortClick(object sender, EventArgs e)
{
    dataGridView1.DataSource = db.Readers.Local.OrderBy(p => p.FIO).ToList();
}

// Отбор данных по части ФИО читателя.
// Вывод в таблицу в отсортированном виде.
private void SearchClick(object sender, EventArgs e)
```

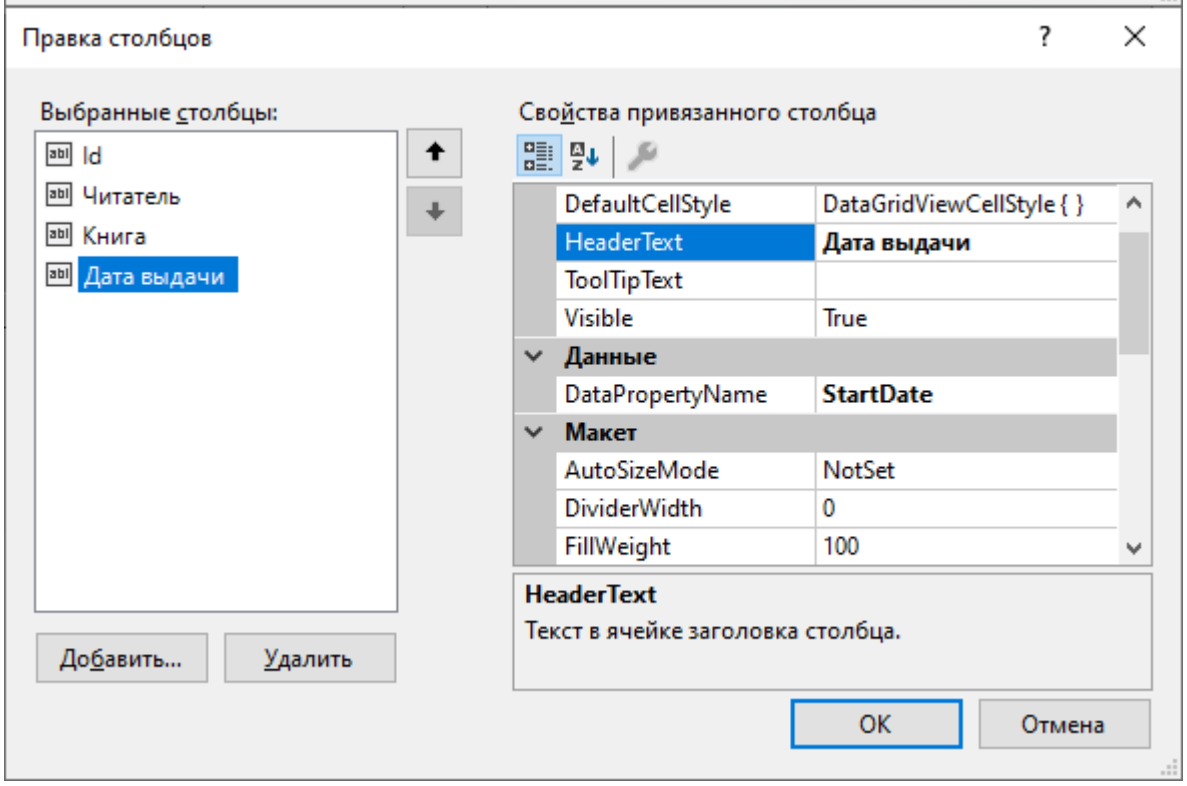
```
{
    dataGridView1.DataSource = db.Readers.Local.Where(p =>
p.FIO.Contains(textBox1.Text)).OrderBy(p => p.FIO).ToList();
}

// Восстановление данных в таблице.
private void ResetClick(object sender, EventArgs e)
{
    textBox1.Text = "";
    dataGridView1.DataSource = db.Readers.Local.ToBindingList();
}
```

6. Задать еще ***два критерия*** для отбора данных. Для этого:
   a. в режиме конструктора изменить значения **radioButton2** и **radioButton3**;
   b. изменить методы **SortClick** и **SearchClick**.
7. В режиме конструктора добавить таблицу, добавить для нее источник данных проекта – Объект:



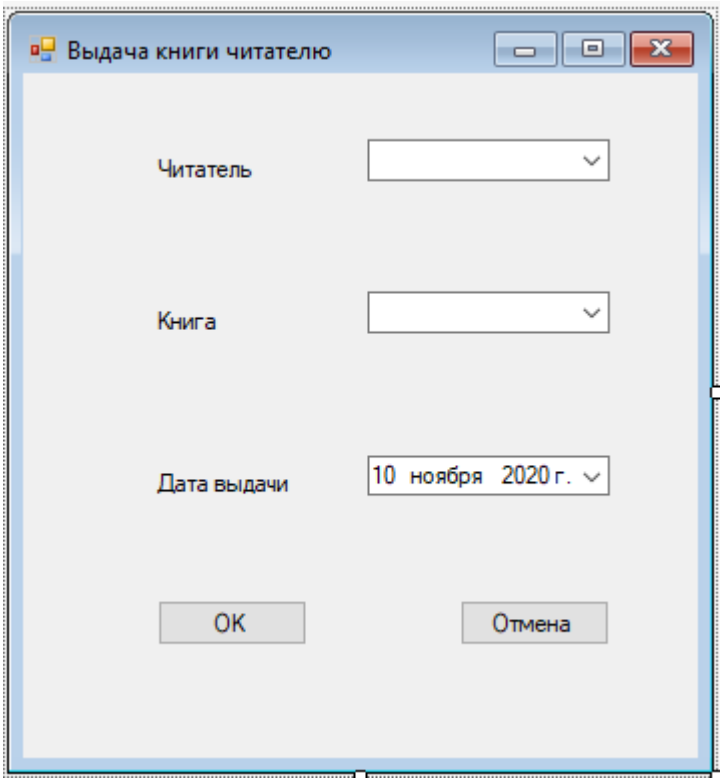
8. Добавить кнопки для работы с данными и настроить столбцы таблицы:





Примечание: столбец **Id** используется только для внутренней организации структуры данных, поэтому данные из него в таблицу не должны выводиться (Visible – False).

9. Изменить свойства таблицы для выбора целой строки при работе с данными (по аналогии с таблицами, созданными в предыдущей работе).
10. Создать форму (видеокадр) для ввода данных о выданной читателю книге:



11. Изменить для элементов **comboBox** свойство **Modifiers** на значение **Protected Internal**. Для календаря запретить возможность изменения (он должен показывать только текущую дату).
12. Для кнопок задать соответствующие значения свойства **DialogResult**.
13. Ввести программный код для **ReaderCardControl**.

```
public partial class ReaderCardControl : UserControl
{
    LibContext db;
    public ReaderCardControl()
    {
        InitializeComponent();
        db = new LibContext();
        db.ReaderCards.Load();
        dataGridView1.DataSource = db.ReaderCards.Local.ToBindingList();
    }
```

Например, код для обработки выдачи книги будет следующим:

```
// Выдать книгу.
private void AddClick(object sender, EventArgs e)
{
    ReaderCardForm readerCardForm = new ReaderCardForm();
    List<Reader> readers = db.Readers.ToList();
    readerCardForm.comboBox1.DataSource = readers;
    readerCardForm.comboBox1.ValueMember = "Id";
    readerCardForm.comboBox1.DisplayMember = "FIO";

    List<Book> books = db.Books.ToList();
    readerCardForm.comboBox2.DataSource = books;
    readerCardForm.comboBox2.ValueMember = "Id";
    readerCardForm.comboBox2.DisplayMember = "Name";

    DialogResult result = readerCardForm.ShowDialog(this);
    if (result == DialogResult.OK)
    {
        try
        {
            ReaderCard readerCard = new ReaderCard();
            readerCard.Reader = (Reader)readerCardForm.comboBox1.SelectedItem;
            readerCard.Book = (Book)readerCardForm.comboBox2.SelectedItem;

            db.ReaderCards.Add(readerCard);
            db.SaveChanges();
            MessageBox.Show("Данные добавлены ");
        }
        catch (Exception exept)
        {
            MessageBox.Show("Данные не добавлены." + exept.Message);
        }
    }
}
```

14. Добавить на **ReaderCardControl.cs[Конструктор]** элементы для отбора (задать разные критерии) и сортировки данных о выданных читателям книгах. Предусмотреть возможность отбора данных по нескольким критериям. Ввести программный код.

4.3 Разработка интерфейса программы в соответствии с технологией WPF и Entity Framework

4.3.1 Разработка программного интерфейса Library (библиотека)

1. В класс **Reader** добавить два вычисляемых свойства (изменений в БД не будет):

```
public string FIO => SecondName + " " + FirstName + " " + MiddleName;

public string Address => Street + ", " + House + (Flat != null ? "-" + Flat : "");
```

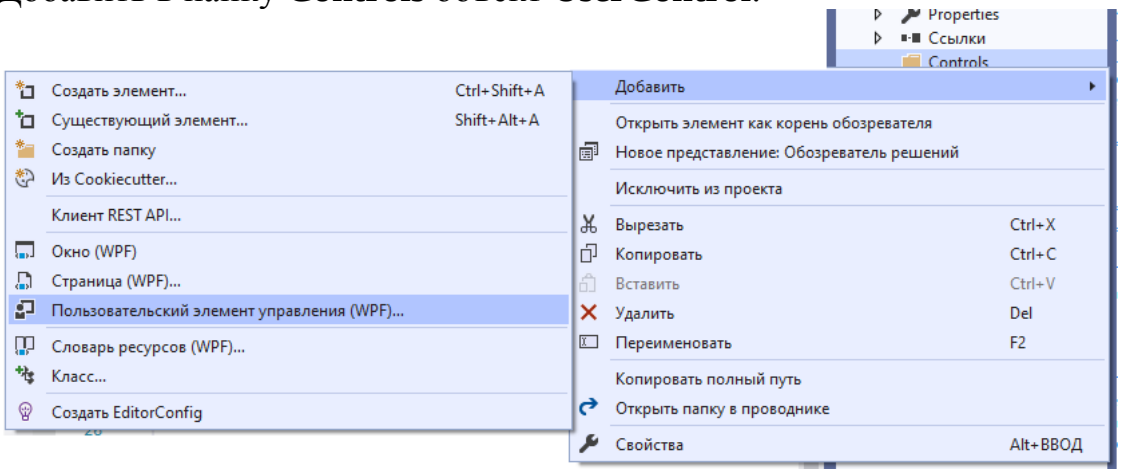
2. Выполнить разметку формы:

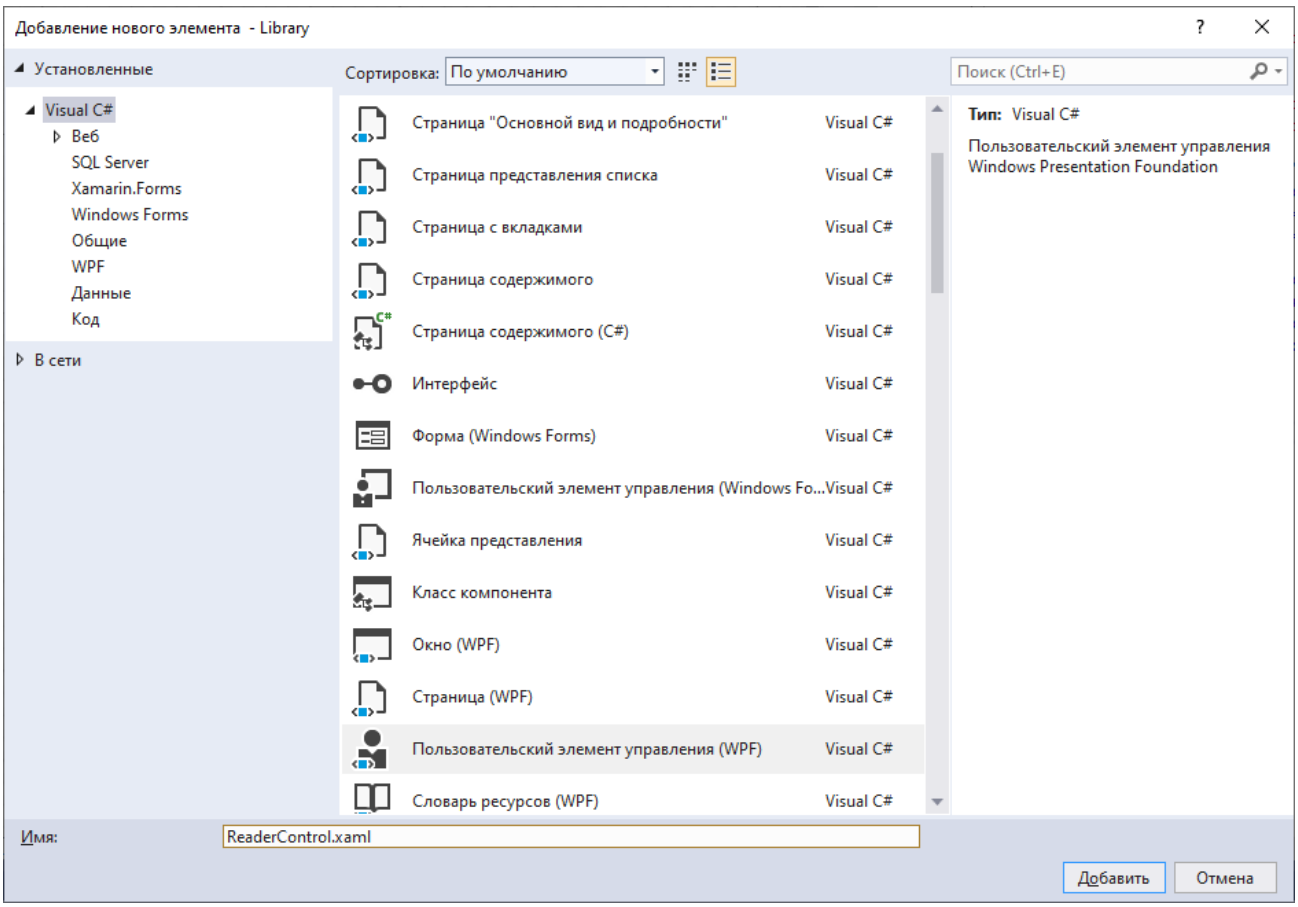
```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="1*"/>
        <RowDefinition Height="9*"/>
    </Grid.RowDefinitions>
</Grid>
```

3. Добавить на первую строку главное меню (**MenuStrip**), содержащее пункты и подпункты: **Справочники** (**Читатели**, **Книги**), **Карточки читателей**, **Отчеты**, **Справка** и **Выход**.

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="1*"/>
        <RowDefinition Height="9*"/>
    </Grid.RowDefinitions>
    <Menu VerticalAlignment="Top" Grid.ColumnSpan="2">
        <MenuItem Header="Справочники">
            <MenuItem Header="Читатели"></MenuItem>
            <MenuItem Header="Книги"></MenuItem>
        </MenuItem>
        <MenuItem Header="Карточки читателей"></MenuItem>
        <MenuItem Header="Отчеты"></MenuItem>
        <MenuItem Header="Справка"></MenuItem>
        <MenuItem Header="Выход"></MenuItem>
    </Menu>
</Grid>
```

4. Создать папку **Controls** в проекте **Library**.
   Примечание: в этой папке будут храниться пользовательские элементы управления, которые можно «внедрять» на формы.
5. Добавить в папку **Controls** объект **UserControl**:





6. Задать размер формы Height="500" Width="800", а пользовательского элемента Height="450" Width="800".
7. Сделать для ReaderControl.xaml разметку и добавить на него таблицы и три кнопки:

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="1*" />
        <ColumnDefinition Width="10*"/>
        <ColumnDefinition Width="1*"/>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="1*"/>
        <RowDefinition Height="4*"/>
        <RowDefinition Height="0.8*"/>
        <RowDefinition Height="0.5*"/>
    </Grid.RowDefinitions>

    <DataGrid x:Name="bookGrid" AutoGenerateColumns="False"  Grid.Column="1"
              HorizontalAlignment="Left" Height="Auto" Margin="0"
              Grid.Row="1" VerticalAlignment="Top" Width="Auto">
        <DataGrid.Columns>
            <DataGridTextColumn Binding="{Binding FIO}" Header="ФИО читателя" Width="2*"/>
            <DataGridTextColumn Binding="{Binding DateOfBirth}" Header="Дата рождения" Width="1.3*"/>
            <DataGridTextColumn Binding="{Binding Address}" Header="Адрес" Width="1.7*"/>
            <DataGridTextColumn Binding="{Binding Telephone}" Header="Телефон" Width="1.5*"/>
            <DataGridTextColumn Binding="{Binding Email}" Header="Электронная почта" Width="1.5*"/>
        </DataGrid.Columns>
    </DataGrid>
    <Label Content="Список читателей" Grid.Column="1" HorizontalAlignment="Center"
           VerticalAlignment="Center" FontSize="16"/>
    <Grid Grid.Row="2" Grid.Column="1">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="0.5*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="0.5*"/>
            <ColumnDefinition Width="1*"/>
        </Grid.ColumnDefinitions>
        <Button x:Name="addButton" Content="Добавить" Margin="5" Grid.Column="0"/>
        <Button x:Name="changeButton" Content="Изменить" Margin="5" Grid.Column="2"/>
        <Button x:Name="deleteButton" Content="Удалить" Margin="5" Grid.Column="4"/>
    </Grid>
</Grid>
```

8. Задать свойства для таблицы:
   Свойства – Строки – SelectionUnit = FullRow;
   Разное – IsReadOnly = True.
9. Для пунктов меню на главной форме добавить обработчики событий:
   В файле дизайна формы (MainWimdow.xaml):

```
<MenuItem Header="Читатели" Click="ReaderClick"></MenuItem>
<MenuItem Header="Книги" Click="BookClick"></MenuItem>
```

   В кодовом файле:

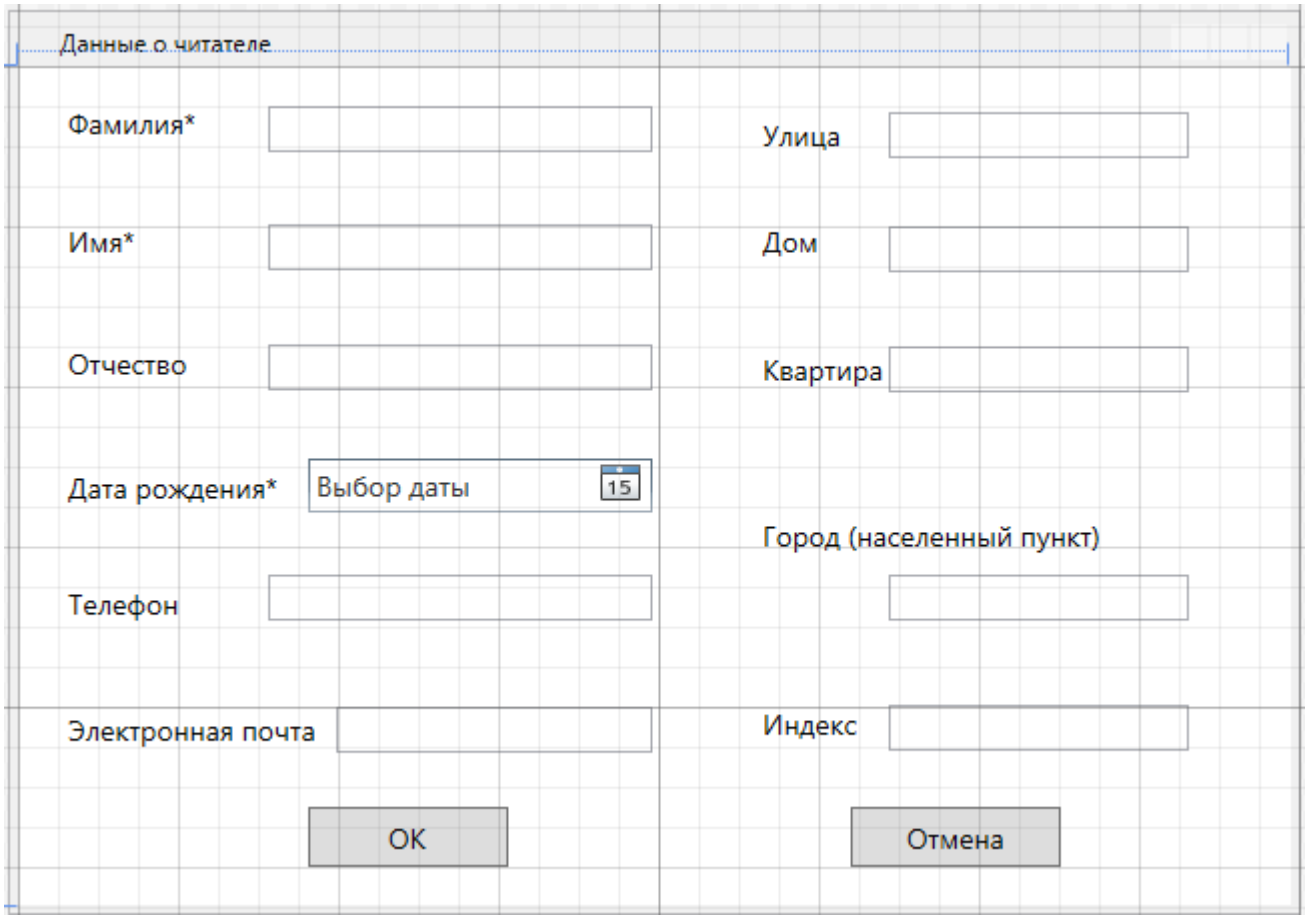
```
private void ReaderClick(object sender, RoutedEventArgs e)
{
    if (MainGrid.Children.Count > 1)
        MainGrid.Children.RemoveAt(1);
    ReaderControl readerControl = new ReaderControl();
    MainGrid.Children.Add(readerControl);
    Grid.SetRow(readerControl, 1);
}
```

10. Запустить программу на выполнение.

Будет создана база данных **Library** или будет выведено ее содержимое. Найти базу данных можно в **Обозревателе объектов SQL Server** или в программе **Microsoft SQL Server Management Studio**.

11. Создать новое окно **ReaderWindow.xaml** для редактирования данных о читателях. Расположить в нем 10 объектов TextBox, один календарь DatePicker и подписи к ним (11 объектов TextBlock или Label); две кнопки:

```
<Button Content="OK"  x:Name="button1" IsDefault="True" Click="OkClick" />
<Button Content="Отмена"  x:Name="button2" IsCancel="True" />
```



12. Ввести код обработчика события:

```
private void OkClick(object sender, RoutedEventArgs e)
{
    this.DialogResult = true;
}
```

13. Ввести код в классе ReaderControl:

```
/// <summary>
/// Логика взаимодействия для ReaderControl.xaml
/// </summary>
public partial class ReaderControl : UserControl
{
    LibContext db;
    public ReaderControl()
    {
        InitializeComponent();
        db = new LibContext();
        db.Readers.Load();
        readerGrid.ItemsSource = db.Readers.Local.ToBindingList();
    }
```

```
private void AddButtonClick(object sender, RoutedEventArgs e)
{
    ReaderWindow readerWin = new ReaderWindow();
    readerWin.button1.Content = "Добавить";
    Reader reader = new Reader();
    var result = readerWin.ShowDialog();
    SaveData(readerWin, result, reader, "добавлены");
}

private void ChangeButtonClick(object sender, RoutedEventArgs e)
{
    if (readerGrid.SelectedItems.Count > 0)
    {
        var reader = readerGrid.SelectedItems[0] as Reader;
        ReaderWindow readerWin = new ReaderWindow();
        readerWin.button1.Content = "Изменить";
        readerWin.textBox1.Text = reader.SecondName;
        readerWin.textBox2.Text = reader.FirstName;
        readerWin.textBox3.Text = reader.MiddleName;
        readerWin.datePicker1.SelectedDate = reader.DateOfBirth;
        readerWin.textBox4.Text = reader.Telephone;
        readerWin.textBox5.Text = reader.Email;
        readerWin.textBox6.Text = reader.Street;
        readerWin.textBox7.Text = reader.House;
        readerWin.textBox8.Text = reader.Flat.ToString();
        readerWin.textBox9.Text = reader.City;
        readerWin.textBox10.Text = reader.ZipCode.ToString();
        var result = readerWin.ShowDialog();

        SaveData(readerWin, result, reader, "изменены");
    }
}
```

```
private void SaveData(ReaderWindow readerWin, bool? result, Reader reader, string str)
{
    if (result == true)
    {
        try
        {
            reader.SecondName = readerWin.textBox1.Text;
            reader.FirstName = readerWin.textBox2.Text;
            reader.MiddleName = readerWin.textBox3.Text;
            reader.DateOfBirth = readerWin.datePicker1.DisplayDate;
            reader.Telephone = readerWin.textBox4.Text;
            reader.Email = readerWin.textBox5.Text;
            reader.Street = readerWin.textBox6.Text;
            reader.House = readerWin.textBox7.Text;
            if (readerWin.textBox8.Text == "") reader.Flat = null;
            else reader.Flat = int.Parse(readerWin.textBox8.Text);
            reader.City = readerWin.textBox9.Text;
            if (readerWin.textBox10.Text == "") reader.Flat = null;
            else reader.ZipCode = int.Parse(readerWin.textBox10.Text);
            if (str == "добавлены") db.Readers.Add(reader);
            db.SaveChanges();

            MessageBox.Show("Данные о читателе " + str);
        }
        catch (Exception exept)
        {
            MessageBox.Show("Данные не " + str + ".\n " + exept.Message);
        }
    }
}

private void DeleteButtonClick(object sender, RoutedEventArgs e)
{
    if (readerGrid.SelectedItems.Count > 0)
    {
        var reader = readerGrid.SelectedItems[0] as Reader;
        if (reader != null)
        {
            db.Readers.Remove(reader);
            db.SaveChanges();
        }
        MessageBox.Show("Объект удален");
    }
}
```

4.3.2 Отбор данных и сортировка в проекте Library (библиотека)

1. Добавить в класс **Reader**:
   a. ***конструктор*** (чтобы не было сбоев при обращении к списку выданных книг):

```
public Reader()
{
    ReaderCards = new List<ReaderCard>();
}
```

   b. ***перегрузку метода*** для преобразования объекта в строку данных:

```
public override string ToString()
{
```

```
        return FIO;
    }
```

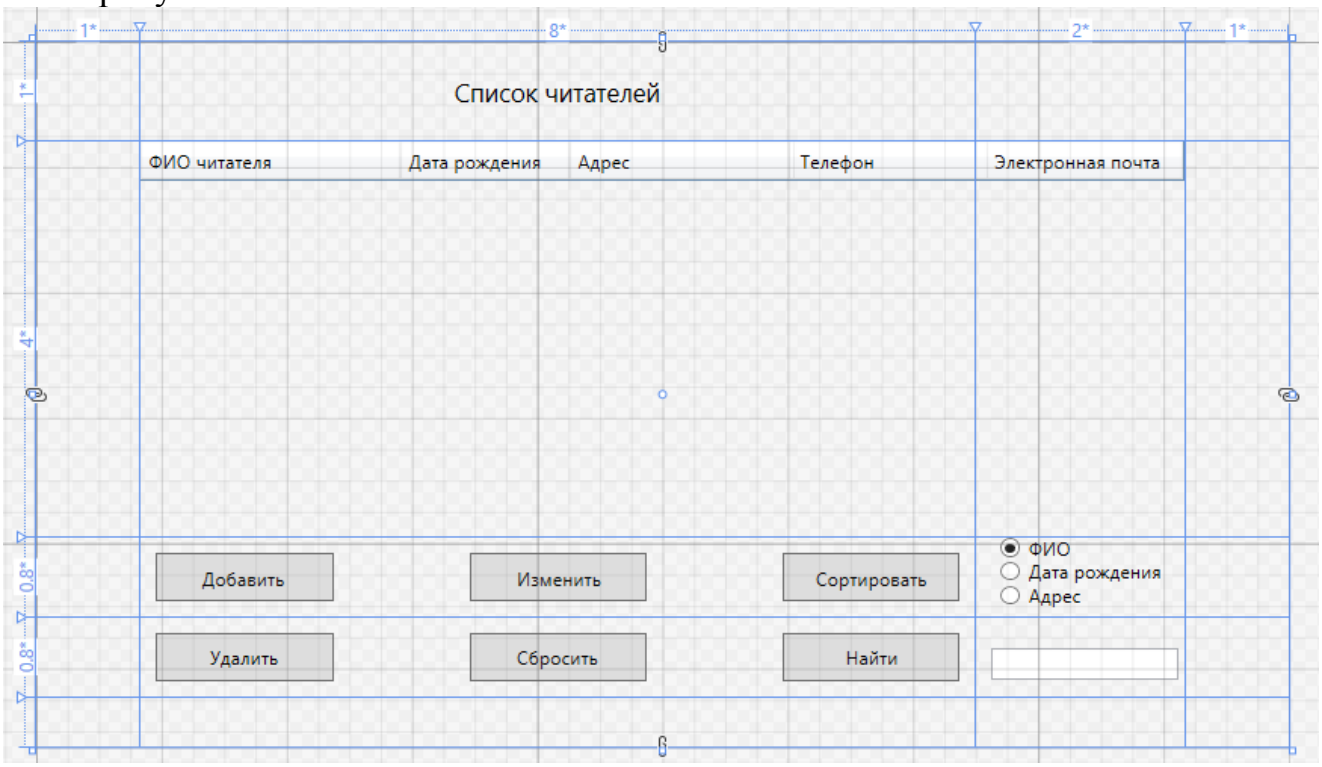
2. Добавить в классе **Book**:

```
    public string Name => Author + " " + Title;

    public Book()
    {
        ReaderCards = new List<ReaderCard>();
    }

    public override string ToString()
    {
        return Name;
    }
```

3. Добавить в классе **ReaderCard** конструктор:

```
    public ReaderCard()
    {
        StartDate = DateTime.Today;
    }
```

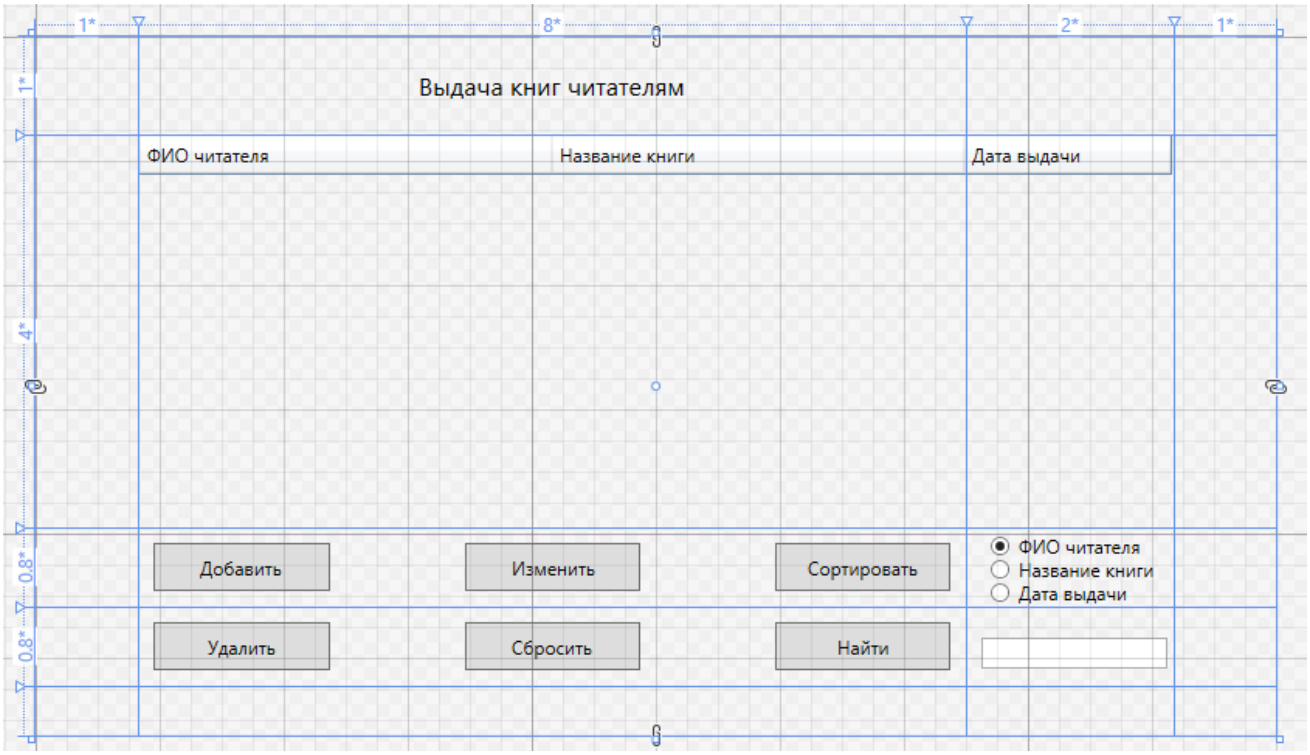
4. Добавить на **ReaderControl.cs** [Конструктор] элементы в соответствии с рисунком:



5. Добавить код для кнопок **Сортировать**, **Сбросить** и **Найти**:

```
private void SortButtonClick(object sender, RoutedEventArgs e)
{
    readerGrid.ItemsSource = db.Readers.Local.OrderBy(p => p.FIO).ToList();
}

private void ResetButton1Click(object sender, RoutedEventArgs e)
{
    searchTextBox.Text = "";
    readerGrid.ItemsSource = db.Readers.Local.ToBindingList();
}

private void SearchButtonClick(object sender, RoutedEventArgs e)
{
    readerGrid.ItemsSource = db.Readers.Local.Where(p =>
    p.FIO.Contains(searchTextBox.Text)).OrderBy(p => p.FIO).ToList();
}
```

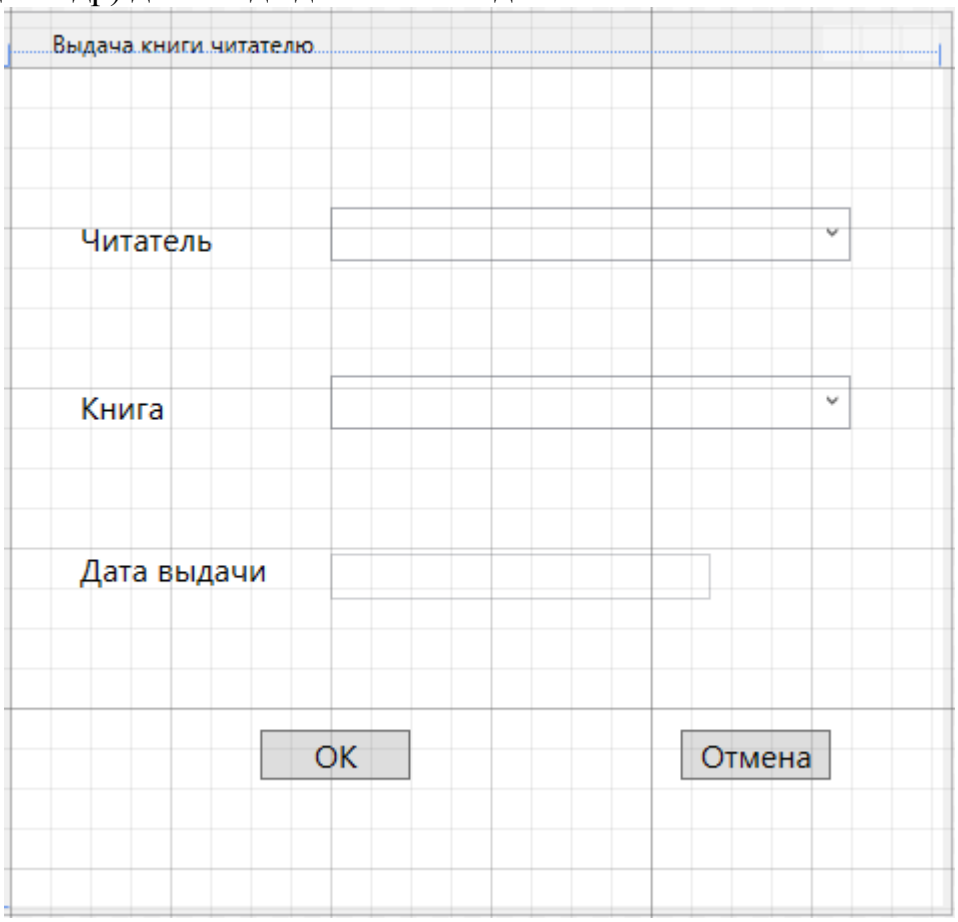
6. Дописать в код команды для определения критерия сортировки и поиска и вывода результатов в таблицу.
7. Создать пользовательский элемент **ReaderCardControl.cs**.



8. Задать свойства для таблицы:
   Свойства – Строки – SelectionUnit = FullRow;
   Разное – IsReadOnly = True.
   Код таблицы:

```
<DataGrid x:Name="readerCardGrid" AutoGenerateColumns="False"
          Grid.Column="1" Grid.ColumnSpan="2"
          HorizontalAlignment="Left" Height="Auto" Margin="0"
          Grid.Row="1" VerticalAlignment="Top" FontSize="12"
          SelectionMode="Single" IsReadOnly="True">
    <DataGrid.Columns>
        <DataGridTextColumn Binding="{Binding Reader}" Header="ФИО читателя" Width="2*"/>
        <DataGridTextColumn Binding="{Binding Book}" Header="Название книги" Width="2*"/>
        <DataGridTextColumn Binding="{Binding StartDate, StringFormat=dd.MM.yyyy}"
                            Header="Дата выдачи" Width="1*"/>
    </DataGrid.Columns>
</DataGrid>
```

9. Добавить в проект новое диалоговое окно **ReaderCardWindow.xaml** (видеокадр) для ввода данных о выданной читателю книге:



10. Для кнопок задать соответствующие значения свойства **DialogResult** (см. код для кнопок окна ReaderWindow.xaml).
11. Ввести программный код для **ReaderCardControl**.

```
/// <summary>
/// Логика взаимодействия для ReaderCardControl.xaml
/// </summary>
public partial class ReaderCardControl : UserControl
{
    LibContext db;
    public ReaderCardControl()
    {
        InitializeComponent();
        db = new LibContext();
        db.ReaderCards.Load();
        readerCardGrid.ItemsSource = db.ReaderCards.Local.ToBindingList();
    }
```

Например, код для обработки выдачи книги будет следующим:

```
private void AddButtonClick(object sender, RoutedEventArgs e)
{
    ReaderCardWindow readerCardWindow = new ReaderCardWindow();
    List<Reader> readers = db.Readers.ToList();
    readerCardWindow.comboBox1.ItemsSource = readers;
    List<Book> books = db.Books.ToList();
    readerCardWindow.comboBox2.ItemsSource = books;
    readerCardWindow.dateToday.Text = DateTime.Today.ToString("D");
    var result = readerCardWindow.ShowDialog();
    if (result == true)
    {
        try
        {
            int index = readerCardWindow.comboBox1.SelectedIndex;
            ReaderCard readerCard = new ReaderCard();
            readerCard.ReaderId = readers[index].Id;
            index = readerCardWindow.comboBox2.SelectedIndex;
            readerCard.BookId = books[index].Id;
            db.ReaderCards.Add(readerCard);
            db.SaveChanges();
            MessageBox.Show("Данные о читателе добавлены");
        }
        catch (Exception exept)
        {
            MessageBox.Show("Данные не добавлены" + ".\n " + exept.Message);
        }
    }
}
```

Изменение данных возможно только в день выдачи. Код этого обработчика события будет следующим:

```
private void ChangeButtonClick(object sender, RoutedEventArgs e)
{
    if (readerCardGrid.SelectedItems.Count > 0)
    {
        var readerCard = readerCardGrid.SelectedItems[0] as ReaderCard;
        if (readerCard.StartDate.ToString("D") == DateTime.Today.ToString("D"))
        {
            ReaderCardWindow readerCardWindow = new ReaderCardWindow();
            List<Reader> readers = db.Readers.ToList();
            readerCardWindow.comboBox1.ItemsSource = readers;
            List<Book> books = db.Books.ToList();
            readerCardWindow.comboBox2.ItemsSource = books;
            readerCardWindow.dateToday.Text = readerCard.StartDate.ToString("D");
            readerCardWindow.comboBox1.Text = readerCard.Reader.ToString();
            readerCardWindow.comboBox2.Text = readerCard.Book.ToString();
```

```
            var result = readerCardWindow.ShowDialog();
            if (result == true)
            {
                try
                {
                    int index = readerCardWindow.comboBox1.SelectedIndex;
                    readerCard.ReaderId = readers[index].Id;
                    index = readerCardWindow.comboBox2.SelectedIndex;
                    readerCard.BookId = books[index].Id;
                    db.SaveChanges();
                    readerCardGrid.Items.Refresh();
                    MessageBox.Show("Данные о читателе изменены");
                }
                catch (Exception exept)
                {
                    MessageBox.Show("Данные не изменены" + ".\n " + exept.Message);
                }
            }
            readerCardGrid.ItemsSource = db.ReaderCards.Local.ToBindingList();
        }
        else MessageBox.Show("Изменение данных не возможно");
    }
}
```

12. Добавить код для обработчиков событий при нажатии на все кнопки на **ReaderCardControl.cs**.

5 СПИСОК ТЕМ КУРСОВОГО ПРОЕКТА

1. Разработка программы "Кинотеатры"
2. Разработка программы "Спортивные секции"
3. Разработка программы "Резервирование авиабилетов"
4. Разработка программы "Коммунальные услуги"
5. Разработка программы "Шахматные турниры"
6. Разработка программы "Фотоуслуги"
7. Разработка программы "Видеостудия"
8. Разработка программы "Регистратура поликлиники"
9. Разработка программы "Прокат товаров"
10. Разработка программы "Телефонные переговоры"
11. Разработка программы "Телефонный справочник"
12. Разработка программы "Продуктовый склад"
13. Разработка программы "Аттестация студентов"
14. Разработка программы "Банковские расчеты"
15. Разработка программы " Конференции"
16. Разработка программы "Интернет-провайдер"
17. Разработка программы "Продажа железнодорожных билетов"
18. Разработка программы "Внеучебная деятельность студентов"
19. Разработка программы "Книжный магазин"
20. Разработка программы "Лыжная база"

6 ВОПРОСЫ К ЗАЩИТЕ КУРСОВОГО ПРОЕКТА ПО ДИСЦИПЛИНЕ

1. Сформулируйте цель и задачи, поставленные в работе, на основе которых выполнялось разработка и написание программы. (ПК-5.1)
2. Какие преимущества языка программирования были использованы при написании программы? (ПК-5.1)
3. Какие технологии были использованы при разработке структуры проекта? (ПК-5.1)
4. Охарактеризуйте, какую структуру должен иметь проект, чтобы соответствовать требованиям ООП. (ПК-5.1)
5. Опишите преимущества технологии ORM, которые используются для работы с БД в коде программы. (ПК-5.1)
6. Перечислите классы, использованные при построении модели БД в программе. (ПК-5.1)
7. Что должно быть указано в строке подключения кодового файла App.cofig для работы с БД? (ПК-5.1)
8. Опишите назначение класса контекста в программе. (ПК-5.1)
9. Охарактеризуйте методы отбора и сортировки данных, которые были использованы при написании программы. (ПК-5.1)
10. Опишите порядок формирования отчетов в формате .docx при написании программы. (ПК-5.1)
11. Опишите порядок проведения отладки программы. (ПК-5.1)
12. Какие требования к разработке интерфейса программы были применены в работе? (ПК-3.1)
13. Какие компоненты интерфейса оконного приложения Вы использовали? (ПК-3.1)
14. Опишите элементы интерфейса, которые необходимо размещать на видеограммах. (ПК-3.1)
15. Опишите элементы интерфейса, которые необходимо размещать в отчетах. (ПК-3.1)
16. Опишите элементы интерфейса, которые были использованы в диалоговых окнах. (ПК-3.1)
17. Опишите обработчики событий для проверки вводимых данных, которые использовали в работе. (ПК-3.1)
18. Опишите способ разметки и расположения компонентов на форме, который был использован в работе. (ПК-3.1)
19. Какие элементы интерфейса на форме позволяют управлять выходными данных? (ПК-3.1)
20. Охарактеризуйте компоненты формы, предназначенные для вывода данных из разных объектов в спроектированную часть интерфейса программы. (ПК-3.1)

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. СТО АлтГТУ 12 400–2015 Образовательный стандарт высшего образования АлтГТУ Курсовой проект (курсовая работа). Общие требования к содержанию, организации выполнения и защите

2. Абрамян, А.В. Разработка пользовательского интерфейса на основе технологии Windows Presentation Foundation: учебник по курсу «Основы разработки пользовательского интерфейса» для студентов направления 02.03.02 «Фундаментальная информатика и информационные технологии» (бакалавриат) / А.В. Абрамян, М.Э. Абрамян ; Южный федеральный университет. – Ростов-на-Дону ; Таганрог : Южный федеральный университет, 2018. – 302 с. : ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=499453 (дата обращения: 17.12.2020). – Библиогр.: с. 294. – ISBN 978-5-9275-2375-7. – Текст : электронный.

3. Рояк, М.Э. Программирование под Windows графических интерфейсов пользователя : учебное пособие : [16+] / М.Э. Рояк, И.М. Ступаков ; Новосибирский государственный технический университет. – Новосибирск : Новосибирский государственный технический университет, 2018. – 72 с. : ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=575018 (дата обращения: 24.01.2021). – Библиогр. в кн. – ISBN 978-5-7782-3754-4. – Текст : электронный.

4. Абрамян, М. Э. Технология LINQ на примерах. Практикум с использованием электронного задачника Programming Taskbook for LINQ : учебное пособие / М. Э. Абрамян. — Москва : ДМК Пресс, 2014. — 326 с. — ISBN 978-5-94074-981-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/66478 (дата обращения: 09.01.2021). — Режим доступа: для авториз. пользователей.

5. Подбельский, В. В. Язык декларативного программирования XAML / В. В. Подбельский. — Москва : ДМК Пресс, 2018. — 336 с. — ISBN 978-5-97060-573-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/111428 (дата обращения: 24.01.2021). — Режим доступа: для авториз. пользователей.

6. Документация по C# [режим доступа] https://docs.microsoft.com/ru-ru/dotnet/csharp/

7. Программирование на C# и .NET [режим доступа] https://metanit.com/sharp/

ПРИЛОЖЕНИЯ

Приложение А
(обязательное)
Форма титульного листа пояснительной записки

Министерство науки и высшего образования Российской Федерации
Рубцовский индустриальный институт (филиал)
ФГБОУ ВО «Алтайский государственный технический университет
им. И.И. Ползунова»

Технический факультет
Кафедра "Прикладная математика"

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

курсовой проекта по _____
дисциплина

_____
обозначение документа

_____

_____
тема проекта

Проект выполнил
студент группы _____
группа подпись дата инициалы, фамилия

Руководитель
проекта _____
должность подпись дата инициалы, фамилия

Проект защищен с оценкой _____

Рубцовск 20__

Приложение Б
(обязательное)
Форма бланка задания на курсовой проект

Рубцовский индустриальный институт (филиал)
ФГБОУ ВО «Алтайский государственный технический университет
им. И.И. Ползунова»
Кафедра "Прикладная математика"

ЗАДАНИЕ № ___

на курсовой проект по дисциплине _____

студенту группы _____

номер группы, Фамилия Имя Отчество

направления подготовки 09.03.01 «Информатика и вычислительная техника»

Тема _____

1. Исходные данные для проекта

_____

2. Содержание пояснительной записки

_____

3. Перечень графического материала

_____

4. Разделы курсового проекта и сроки их выполнения

Наименование разделов проекта и их содержание	Трудоем-кость, %	Срок выполнения	Руково-дитель
Составление задания на курсовой проект	3%		
Изучение ГОСТов, методических указаний и литературных источников по программированию	12%	1 контр.точка	
Постановка задачи	15%	2 контр.точка	
Разработка алгоритма решения задачи	10%		
Разработка контрольного примера	10%	3 контр.точка	
Написание и отладка программы	30%		
Составление справки к программе и руководства пользователя	10%	4 контр.точка	
Оформление результатов работы и пояснительной записки	10%	защита к/р	

Срок представления проекта к защите «___» _____ 20__ г.

Руководитель
проекта _____

должность подпись дата инициалы, фамилия