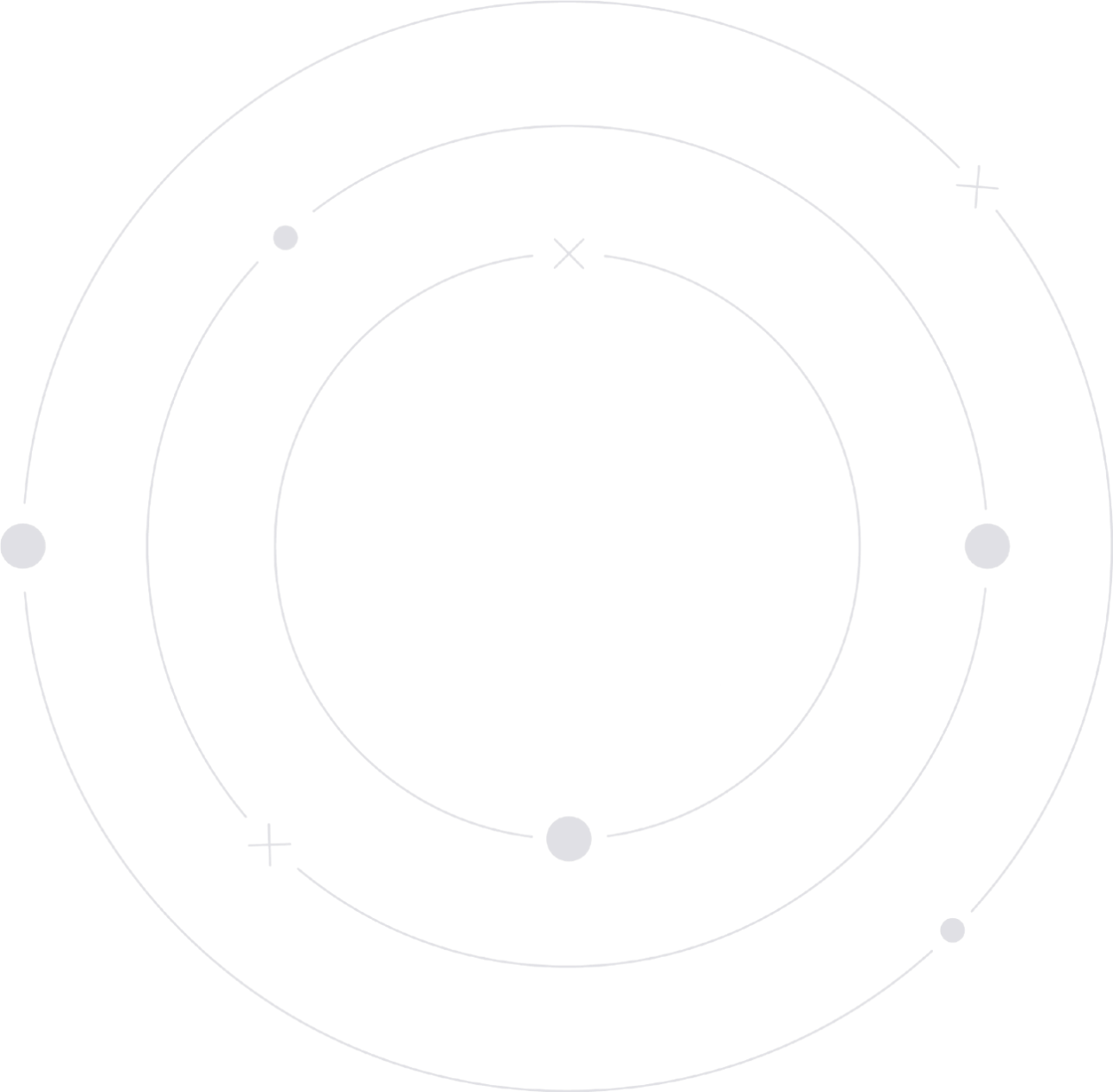


Пример быстрой разработки приложения   
на платформе Турбо Х

Часть 2



**Аннотация**

Назначение этого документа – предоставить читателю базовые сведения о возможностях платформы Турбо Х.

Изложение материала основано на простом примере – «Домашняя бухгалтерия». В этом проекте мы обеспечим возможность пользователю фиксировать свои текущие расходы и получать соответствующие сводные данные.

Предполагается, что:

* читатель владеет базовой терминологией и навыками программирования;
* читателю доступна техническая документация платформы;
* платформа Турбо X и MSSQL-Server установлены на компьютере читателя.

Сведения, содержащиеся в настоящем документе, отражают текущую позицию ООО «КОНСИСТ» по обсуждаемым вопросам на момент публикации. Так как ООО «КОНСИСТ» должно реагировать на изменяющиеся условия рынка, этот документ не может считаться обязательством компании и может претерпеть изменения. Это означает, что ООО «КОНСИСТ» не может гарантировать точность любой информации, представленной в этом документе, после даты его публикации. Настоящий документ не является описанием товаров и услуг, предлагаемых сейчас для приобретения. Этот документ должен использоваться только в информационных целях.

ООО «КОНСИСТ» НЕ ПРЕДОСТАВЛЯЕТ НИКАКИХ ГАРАНТИЙ, ЯВНЫХ, ПОДРАЗУМЕВАЕМЫХ ИЛИ ПРЕДУСМОТРЕННЫХ ЗАКОНОМ, ОТНОСИТЕЛЬНО СВЕДЕНИЙ, СОДЕРЖАЩИХСЯ В ЭТОМ ДОКУМЕНТЕ.

© ООО «КОНСИСТ», 2021 г. Все права защищены.

Оглавление

[1. Урок 2. Создание проекта, ИБ и структуры таблиц 4](#_Toc73111373)

[1.1. Создание проекта 4](#_Toc73111374)

[1.2. Создание структуры данных. Справочники. 6](#_Toc73111375)

[1.3. Создание структуры данных. Документ 11](#_Toc73111376)

[1.4. Компиляция 12](#_Toc73111377)

[1.5. Создание информационной базы 12](#_Toc73111378)

[1.6. Первый запуск 14](#_Toc73111379)

[2. Урок 3. Реализация справочников 16](#_Toc73111380)

[2.1. Структура классов и реализация простых справочников 16](#_Toc73111381)

[2.2. Использование редактора ресурсов для форм 24](#_Toc73111382)

[2.3. Реализация справочника «Валюты» 27](#_Toc73111383)

[2.3.1. Описание записи справочника 27](#_Toc73111384)

[2.3.2. Описание формы справочника валют 31](#_Toc73111385)

[2.3.3. Описание списочного представления 35](#_Toc73111386)

[2.4. Командное меню 37](#_Toc73111387)

[3. Урок 4. Реализация документа 42](#_Toc73111388)

[3.1. Описание записи документа 42](#_Toc73111389)

[3.2. Описание формы документа 45](#_Toc73111390)

[3.3. Описание списочного представления документа 49](#_Toc73111391)

[4. Урок 5. Структура учета и отчет 50](#_Toc73111392)

[4.1. Проводки 50](#_Toc73111393)

[4.2. Область учета 51](#_Toc73111394)

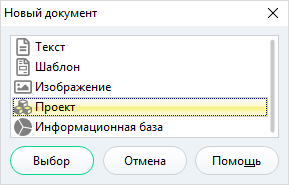
[4.3. Проводки 52](#_Toc73111395)

[4.4. Журнал операций 52](#_Toc73111396)

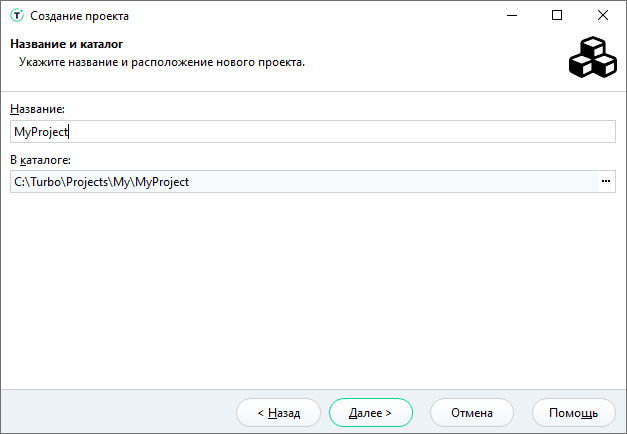
[4.5. Отчет 53](#_Toc73111397)

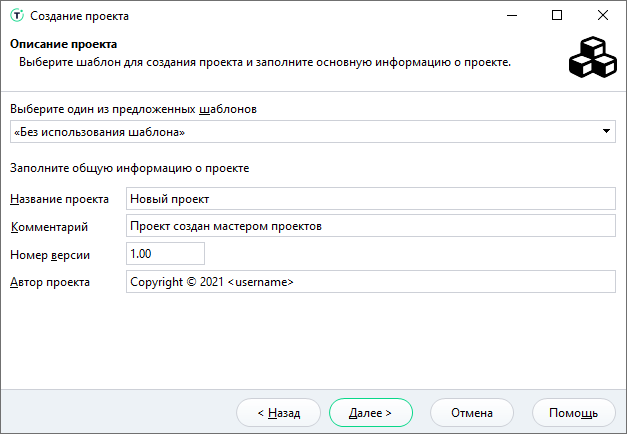
1. Урок 2. Создание проекта, ИБ и структуры таблиц
   1. Создание проекта

Для создания нового проекта используется пункт главного меню «Файл», подпункт «Новый» и режим «Проект»:



В мастере создания проекта выбираем сервер «Мой компьютер», заходим под именем «Администратор», выбираем режим «Создать новый», задаем имя и расположение создаваемого проекта.



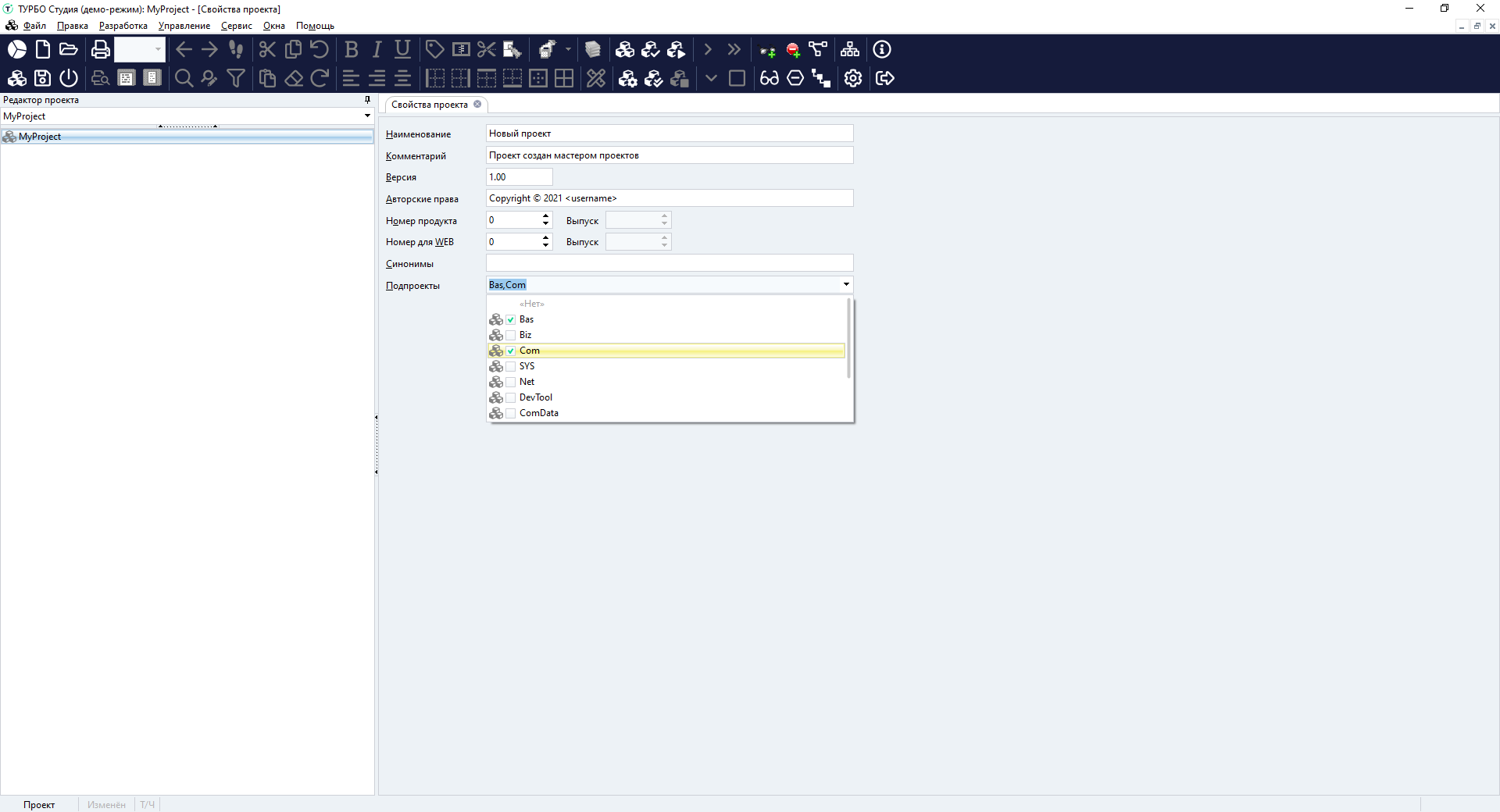


Шаг создания информационной базы пропускаем и соглашаемся с предложением «Открыть проект».

В окне «Редактор проекта» отобразится пустое дерево проекта. В окне настройки свойств проекта (вызывается двойным щелчком по имени проекта) выбираем нужные нам подпроекты Com и Bas и сохраняем изменения кнопкой «Применить».

В подпроекте Com реализованы базовые компоненты платформы, такие как экранные формы, записи (данные), элементы построения интерфейса (окна или формы, поля ввода, списки, меню) и т.п.

В подпроекте Bas реализован базовый функционал для бизнес-приложений: справочники, администрирование пользователей и настройки, календари и т.д.)



* 1. Создание структуры данных. Справочники.

Представим очень простую систему с одним документом и несколькими справочниками следующего вида:

**Строка**   
*Doc struct Position*

Получатель *Recipient*

Сумма *Sum*

Валюта *Cur*

Вид расхода *Expense*

Комментарий *Comment*

**Получатель** *Recipient*

Название *Name*

**Вид расхода** *Expense*

Название Name

**Валюта** *Cur*

Название

**Документ** *Doc*

Дата *Date*

Плательщик *Person*

Общая сумма *SumTotal*

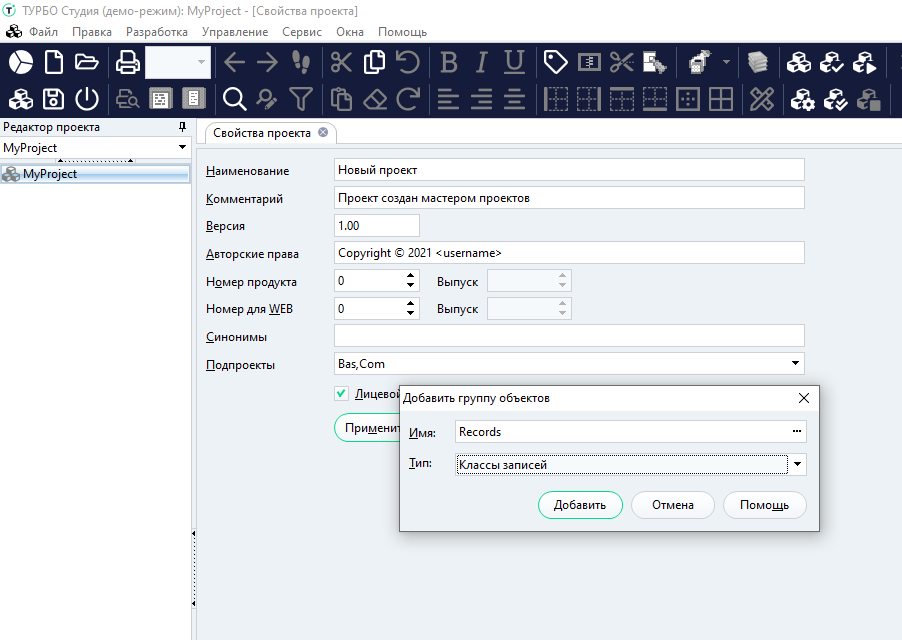
Валюта *Cur*

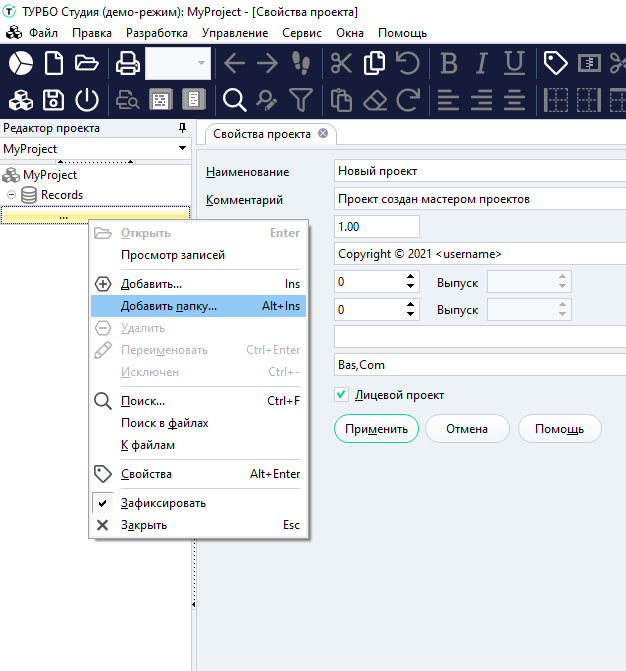
**Плательщик** *Person*

Название *Name*

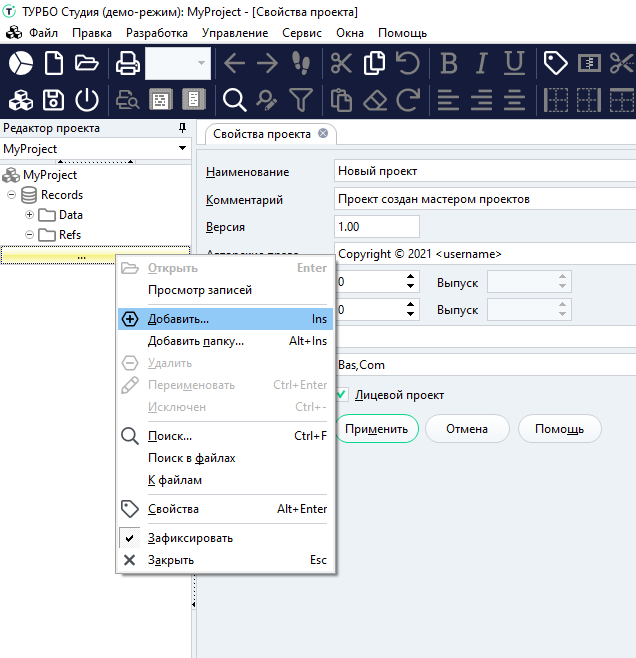
Структура дерева проекта создается в окне «Редактор проекта». Элементы добавляются правой кнопкой мыши -> «Добавить» или нажатием клавиши Ins.

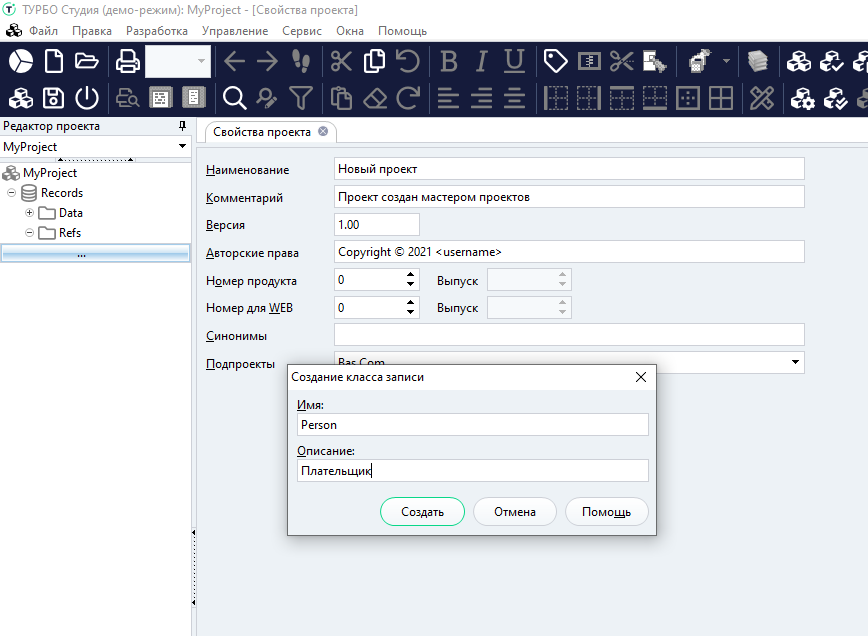
В нашем проекте описание записей расположим в группе объектов Records типа «Классы записей», в которой создадим папку Refs (для справочников) и папку Data (для документа).





В папке Ref создадим описание записи «Плательщик» («Добавить»):





Результат создания:

|  |
| --- |
| record Person title "Плательщик";    end; |

Воспользуемся уже реализованным в проекте Com базовым типом, который соответствует нашей задаче (укажем в описании имя предка):

|  |
| --- |
| record Person inherited Com.Base.CodeName title "Плательщик";    end; |

Соответственно, запись справочника будет содержать два требуемых поля – краткое обозначение (код или code) и полное наименование (имя или name).

Отметим, что код будет использоваться как уникальный идентификатор плательщика.

Описания справочников «Получатель» и «Вид расходов» создаются аналогично.

Справочник Refs\Cur (Валюта) помимо унаследованных будет иметь дополнительные поля:

|  |
| --- |
| record Cur inherited Com.Base.CodeName title "Валюта";  field BaseUnit :Cur;  field ToMultiply :Logical;  field Rate :Numeric period Date;  field Accuracy :Integer;  field Altername :String;  end; |

Соответственно, можно будет реализовать пересчет номинала валюты относительно базовой по изменяющемуся во времени курсу , задать точность отображения номинала и дополнительное (альтернативое) обозначение.

Синтаксис и семантика языка описания данных приведены в документации и справке.

Отметим, что поставка платформы содержит развернутые базовые справочники для большинства сущностей, которые используются в создаваемых бизнес-приложениях.

* 1. Создание структуры данных. Документ

Предположим, что в одной записи о расходах (например, счете или кассовом чеке, в общем, каком-то Документе) может быть несколько расходных позиций (строчек чека или счета), поэтому структуру записи о расходах мы представим так:

**Общая часть (заголовок документа)**

|  |  |
| --- | --- |
| **Дата** | **Дата** |
| **Плательщик** | **Ссылка на запись справочника** |
| **Сумма** | **Сумма** |
| **Валюта** | **Ссылка на запись справочника** |

**Строчная часть (позиции документа)**

|  |  |
| --- | --- |
| **Получатель1** | **Ссылка на запись справочника** |
| **Вид расхода 1** | **Ссылка на запись справочника** |
| **Сумма 1** | **Сумма** |
| **Валюта 1** | **Ссылка на запись справочника** |
| **Комментарий 1** | **Строка** |
| **…** | **…** |
| **Получатель N** | **Ссылка на запись справочника** |
| **Вид расхода N** | **Ссылка на запись справочника** |
| **Сумма N** | **Сумма** |
| **Валюта N** | **Ссылка на запись справочника** |
| **Комментарий N** | **Строка** |

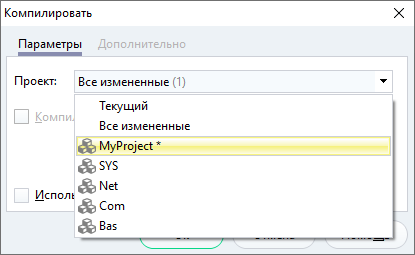
Турбо позволяет описывать подобные документы в рамках одной структуры данных. Описание такой структуры на языке платформы будет выглядеть так: Data\Doc:

|  |
| --- |
| hierarchical document Doc inherited Com.Base.CodeName title "Счёт";  field Date :date title "Дата";  field Person :Refs.Person title "Плательщик";  field Cont :Logical title "Проведено";  field Cur :Refs.Cur title "Валюта";  field SumTotal :Numeric title "Сумма чека";  struct Positions array Integer title "Строки чека";  field Recipient :Refs.Recipient title "Получатель";  field Sum :Numeric title "Сумма";  field Cur :Refs.Cur title "Валюта";  field Expense :Refs.Expense title "Вид расходов";  field Comment :string title "Примечание";  end;  end; |

На этом этапе структуру базы данных для наших документов можно считать созданной. В дальнейшем средствами платформы будет автоматически создана физическая база данных в соответствующей СУБД.

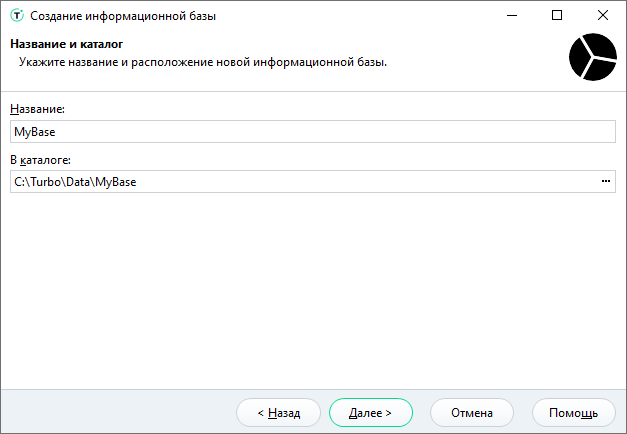
* 1. Компиляция

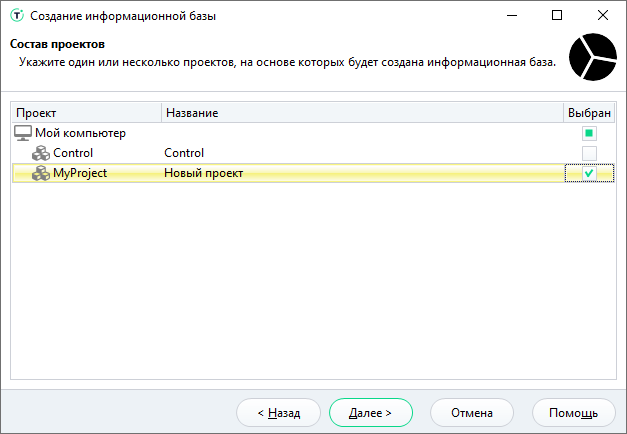
Для создании информационной базы проект должен быть откомпилирован. Чтобы откомпилировать проект необходимо выбрать пункт «Компилировать» во вкладке «Разработка» или использовать сочетание клавиш Ctrl+F9.



* 1. Создание информационной базы

Перед первым запуском необходимо создать информационную базу, используя соответствующий пункт меню: «Сервис» -> «Администрирование» -> «Мой компьютер» -> «Информационные базы» -> контекстное меню -> пункт «Создать/добавить базу» -> «Создать новую», затем задать название и расположение ИБ и отметить соответствующий проект:

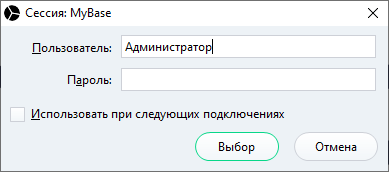




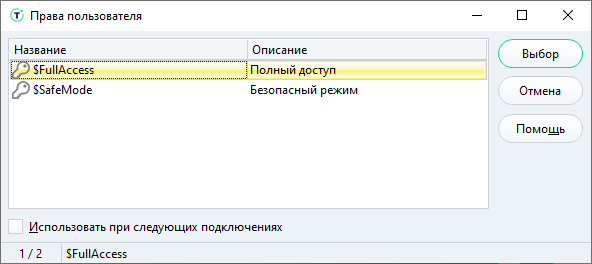
Последующие настройки оставляем по умолчанию.

* 1. Первый запуск

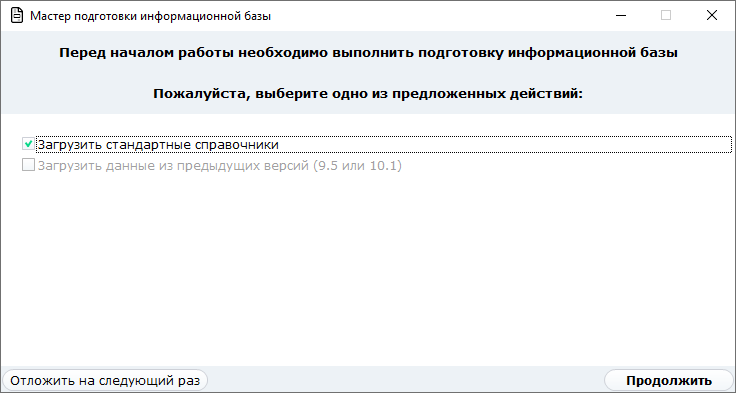
Запустим приложение в режиме отладки клавишей F9. Выбираем из списка ранее созданную информационную базу, указываем пользователя базы данных, обладающего соответствующими привилегиями (Администратор)



и используем право полного доступа.

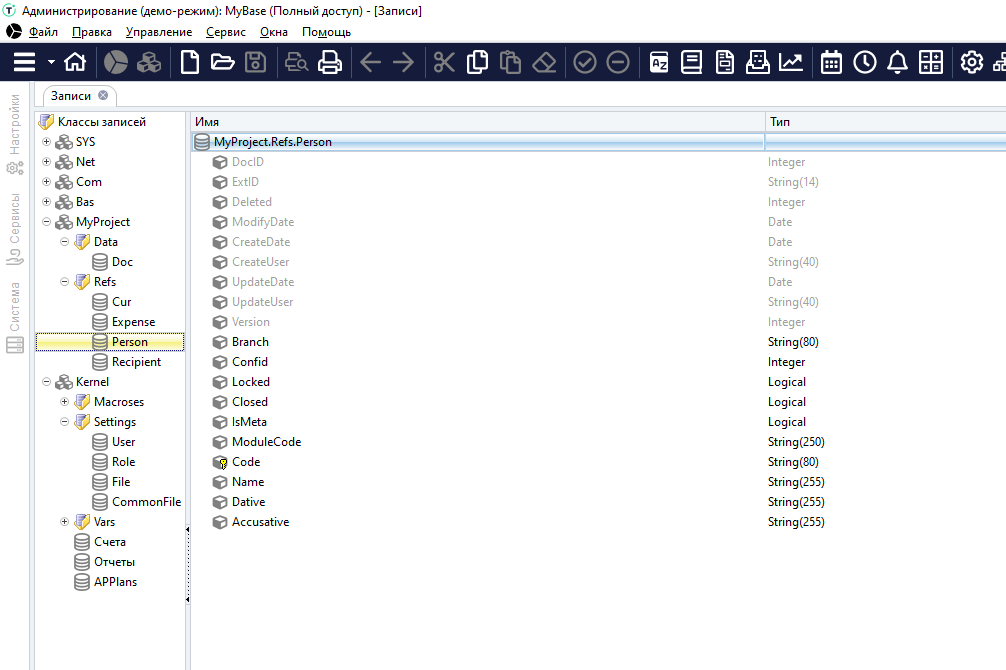


Открывается мастер подготовки ИБ. Загружаем стандартные справочники,



дожидаемся окончания подготовки и перезапускаем программу, чтобы изменения вступили в силу.

В главном меню выбираем пункт «Сервис» -> «Просмотр записей». В открывшемся окне можно просмотреть таблицы данных, как созданные в нашем проекте, так и унаследованные от базовых. Можем видеть, что для справочника Person помимо полей «Code» и «Name» унаследованы другие поля базового класса

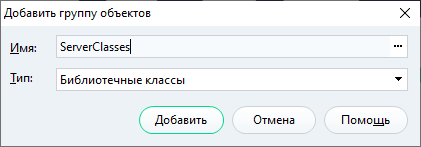


1. Урок 3. Реализация справочников
   1. Структура классов и реализация простых справочников

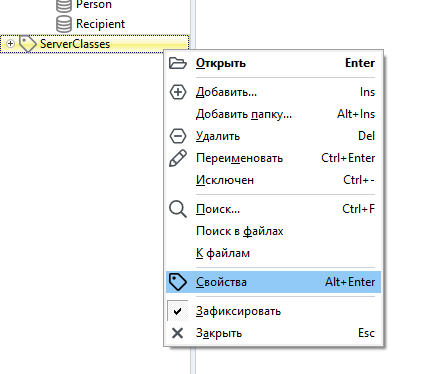
Для работы со справочными данными понадобится реализация программного и пользовательского интерфейса, обеспечивающего доступ к данным. Как правило, для каждой записи реализуются три класса: класс-обертка (Rec), клаcc-редактор (Frm) и класс-таблица (Tab).

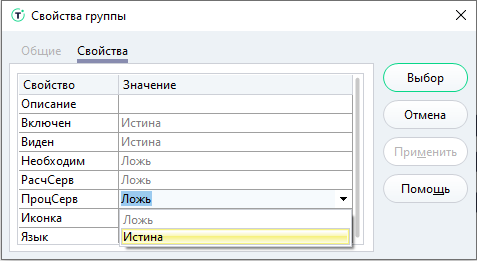
Также используются термины «Запись», «Форма» или «Бланк» и «Картотека». Картотека – списочное отображение совокупности записей, форма – инструмент отображения и редактирования конкретной записи из картотеки.

Соответствующая группа классов с типом «Библиотечные классы» называется ServerClasses (добавляется на уровне проекта):



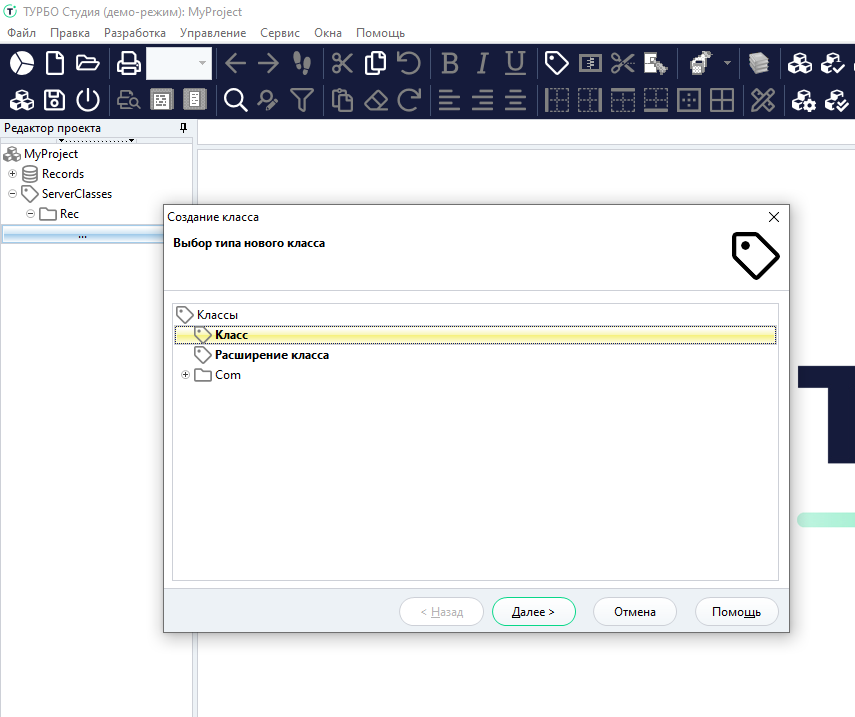
Для классов, размещаемых в этой папке, установим свойство, отвечающее за исполнение процедур в серверной части Turbo (поскольку используется архитектура «Клиент-Сервер», возможно два варианта):

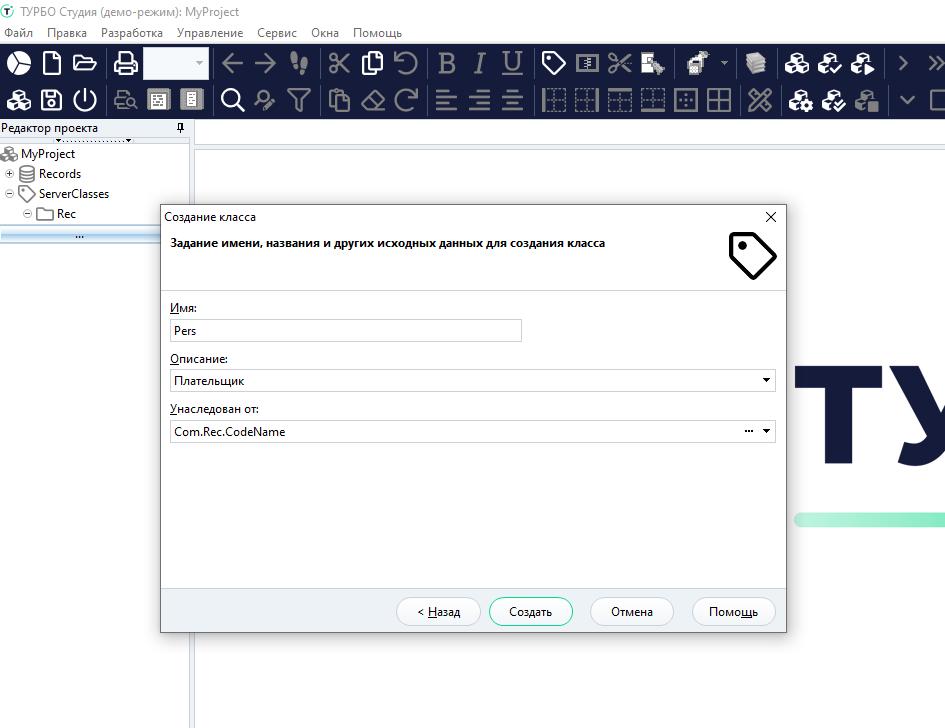




Добавим в группу папки: Rec, Frm, Tab - и элементы: Rec\Pers, Frm\Pers, Tab\Pers.

Добавление Rec-класса:





Результат создания – заготовка класса:

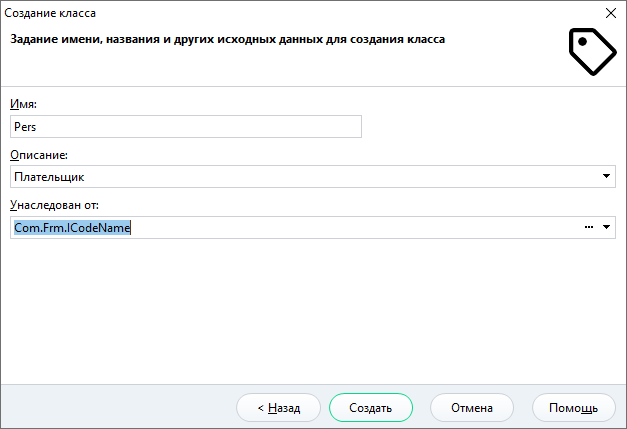
|  |
| --- |
| class inherited Com.Rec.CodeName "Плательщик";  inclass  inobject  end |

Внесем исправления для связи Rec-класса с описанием записи:

|  |
| --- |
| class inherited Com.Rec.CodeName "Плательщик";  inclass public  var RecordClass :Class Record := Refs.Person;  end |

В структуре Rec-класса (обертки записи) необходимо предусмотреть компоненты-обертки для полей. Для простого справочника это уже реализовано в базовом классе.

Добавление Frm-класса:



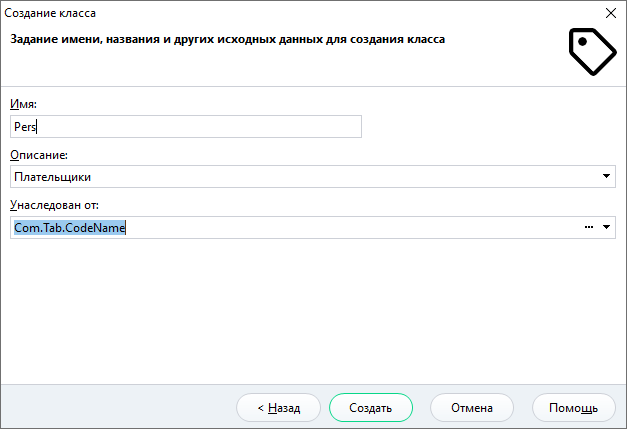
Результат создания – заготовка класса:

|  |
| --- |
| class inherited Com.Frm.ICodeName "Плательщик";  inclass  inobject  end |

Внесем исправления для связи Frm-класса с оберткой (Rec) записи и включения отображения элементов редактирования (введены в базовом классе, но по умолчанию не отображаются):

|  |
| --- |
| class inherited Com.Frm.ICodeName "Плательщик";  inclass public  var RecordClass :Class Com.Rec.CodeName := Rec.Pers;  inobject private  proc Init;  inherited Init;  Code.Visible = true;  Name.Visible = true;  end;  end |

Добавление Tab-класса:



Результат создания – заготовка класса:

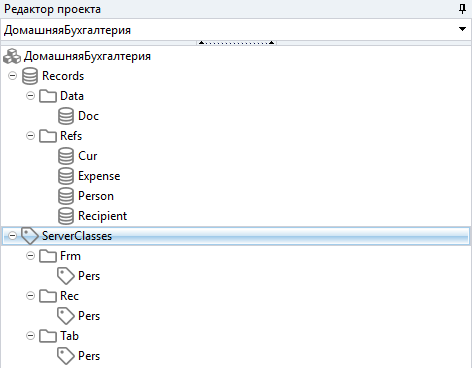
|  |
| --- |
| class inherited Com.Tab.CodeName "Плательщики";  inclass  inobject  end |

Укажем соответствующие связи и добавим элементы для отображения полей:

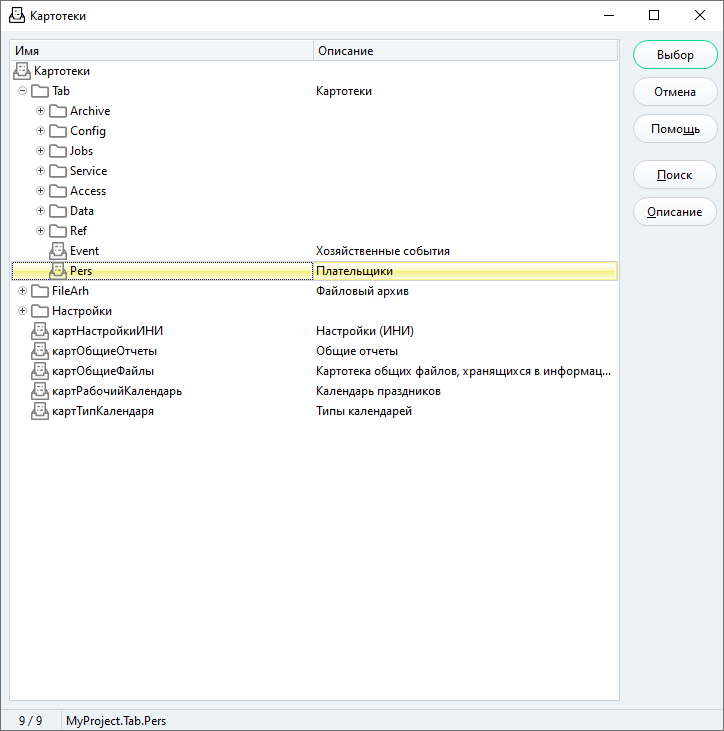
|  |
| --- |
| class inherited Com.Tab.CodeName "Плательщики";  inclass public  var RecordClass :Class Com.Rec.IRecord := Rec.Pers;  var EditorClass :Class Com.Frm.IEditor := Frm.Pers;  inobject private  proc Init;  inherited Init;  AddCode;  AddName;  end; |

Процедура Init вызывается в момент создания экземпляра класса и обеспечивает необходимые действия. В данном случае мы используем процедуры, включающие отображаемые в таблице поля записи Code и Namе. Вызов inherited Init обращается к соответствующей процедуре предка.

В результате вышеперечисленных действий дерево проекта будет иместь следующий вид:



Проверим работу со справочником «Плательщики» в режиме сессии. Для этого после запуска (F9) диалог выбора Tab-классов (Alt+T) и выберем класс Pers:



В базовых классах – предках реализован практически весь необходимый нам функционал – добавление, редактирование, поиск(фильтрация) и удление записей.

* 1. Использование редактора ресурсов для форм

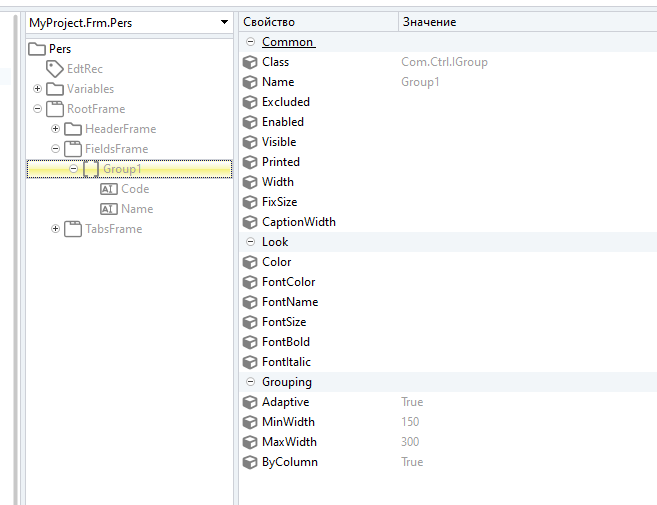
Форма – основной инструмент для работы с конкретной записью. На форме размещаются визуальные элементы – «контролы», для «вертикальной» группировки которых используются фреймы. При необходимости форма может содержать несколько закладок – для «горизонтальной» группировки фреймов и «контролов». Также на форме может отображаться табличная часть документа (строки или позиции).

Стркутура и «наполнение» формы оформляется в процедуре Init (этот вариант рассмотрен выше) либо с помощью специального инструмента – редактора ресурсов в дизайн-режиме. Перейти в него в режиме «Студия» можно двойным щелчком по заголовку на вкладке класса, в режиме «Сессия» - через пункт контекстного меню «Дизайн-режим».

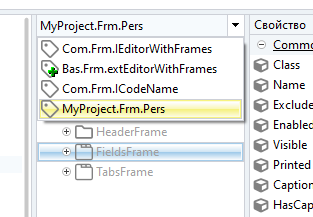
Для поддержки дизайн режима внесем в текст класса Frm.Pers следующие изменения:

|  |
| --- |
| class inherited Com.Frm.ICodeName "Плательщик";  inclass public  var UseResource :Logical := true;  var RecordClass :Class Com.Rec.CodeName := Rec.Cur;  end |

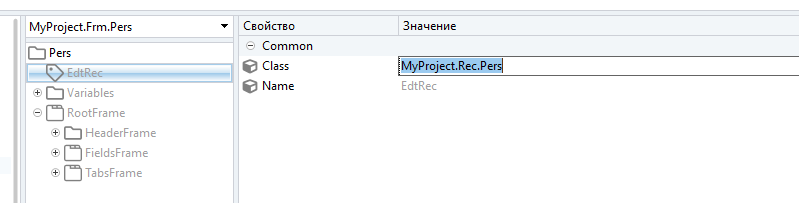
Перейдем в редактор ресурсов:



Серым цветом отображаются атрибуты, унаследованные от предка. Иерархию предков можно просмотреть:



Подключим к форме запись справочника Pers. Для этого в компоненте EdtRec в поле Class выберем ранее нами созданный ДомашняяБухгалтерия.Rec.Pers (перечень классов открывается нажатием кнопки ).

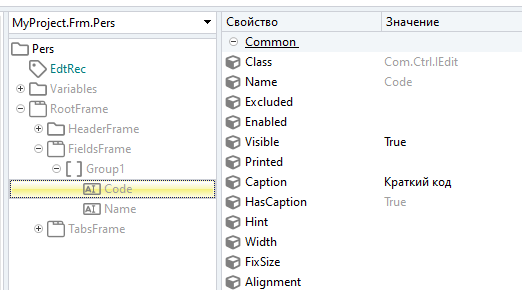


Основной элемент формы (фрейм) – RootFrame, на котором расположены:

* HeaderFrame - заголовок формы,
* FieldsFrame – поля редактирования,
* TabsFrame – дополнительные вкладки (для сложных форм).

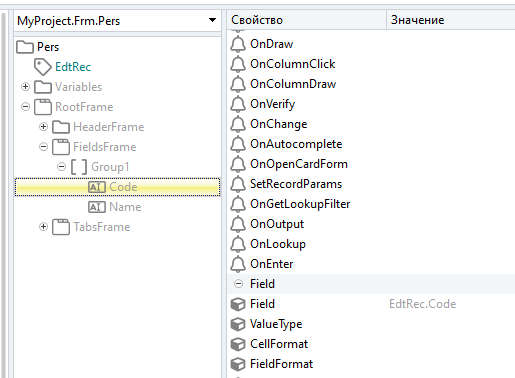
Нам понадобятся два унаследованных поля – Code и Name. Их описание уже находится в группе Group1 FieldsFrame.

Чтобы включить отображение поля Code на форме, в поле Visible указываем True. Также добавим заголовок «Кратко» (поле Caption).



Заголовок поля Name по умолчанию имеет значение «Наименование», что не очень нам подходит, поэтому изменяем его на приближенное ситуации «ФИО». Затем включаем видимость (Visible = true).

Ниже, в группе атрибутов Field в одноимённом поле, содержится редактируемое поле изменяемой записи. Для полей Code и Name формы его значение указано по умолчанию – соответствующие поля класса EdtRec (Rec.Pers).



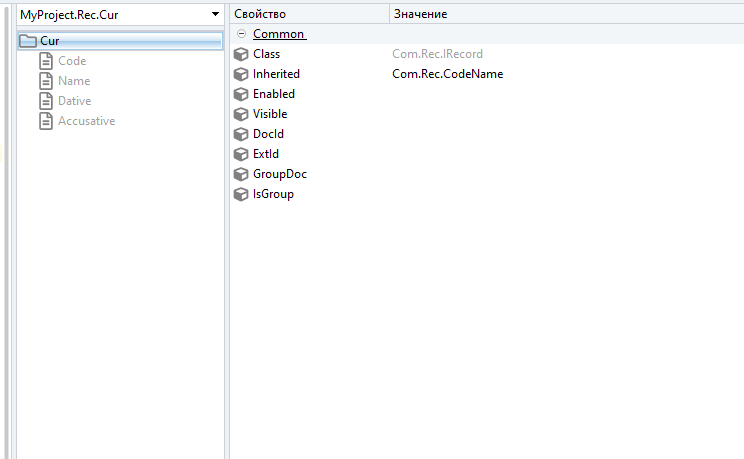
Еще два справочника (получатели - Recs и виды расходов – Exps ) реализуйте самостоятельно, Frm и Tab-классы с использованием редактора ресурсов.

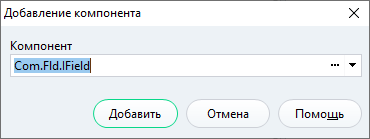
* 1. Реализация справочника «Валюты»
     1. Описание записи справочника

Создаем Rec-класс с именем Cur и модифицируем полученную заготовку:

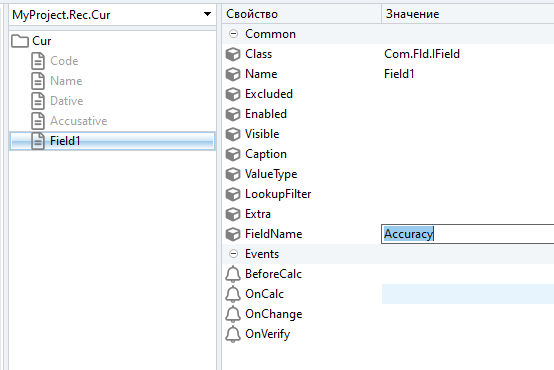
|  |
| --- |
| class inherited Com.Rec.CodeName "Валюты";  inclass public  var UseResource :Logical := true;  var RecordClass :Class Record := Refs.Cur;  inobject public  end |

Переходим в дизайн-режим и добавим компоненты для полей:





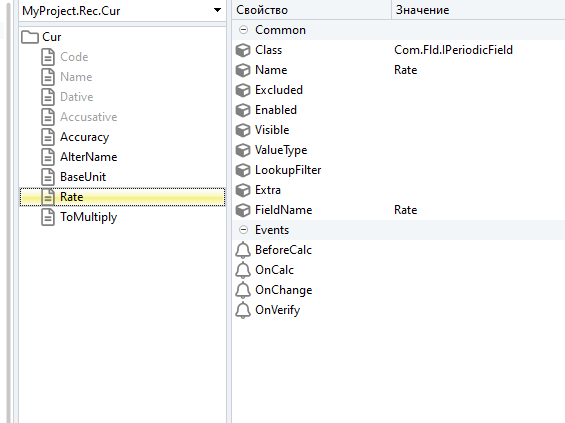
и укажем связь компонента с полем записи:



Для удобочитаемости рекомендуется поменять также атрибут Name (как правило совпадает с именем поля)

Для поля Rate выберем компонент Fld.IPeriodicField.

После добавления оберток для всех полей получаем такую структуру:



Добавленные компоненты также желательно описать в тексте класса (двойной клик по строке «Name» переносит описание автоматически). Описание понадобится для программного обращения к компонентам. В результате текст должен выглядеть так

|  |
| --- |
| class inherited Com.Rec.CodeName "Валюты";  inclass public  var UseResource :Logical := true;  var RecordClass :Class Record := Refs.Cur;  inobject public  Accuracy :Com.Fld.IField;  AlterName :Com.Fld.IField;  BaseUnit :Com.Fld.IField;  Rate :Com.Fld.IPeriodicField;  ToMultiply :Com.Fld.IField;  end |

Альтернативный вариант – программная инициализация. В этом случае добавление полей производится в процедуре Init:

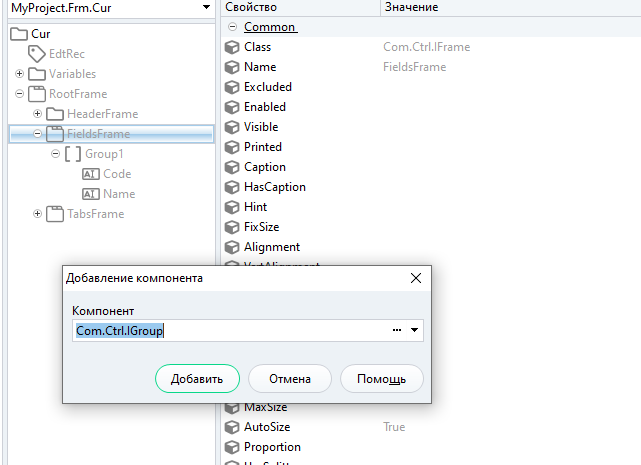
|  |
| --- |
| …  inobject private  proc Init;  inherited Init;  Accuracy = AddField(Accuracy , 'Точность' );  …  end;  … |

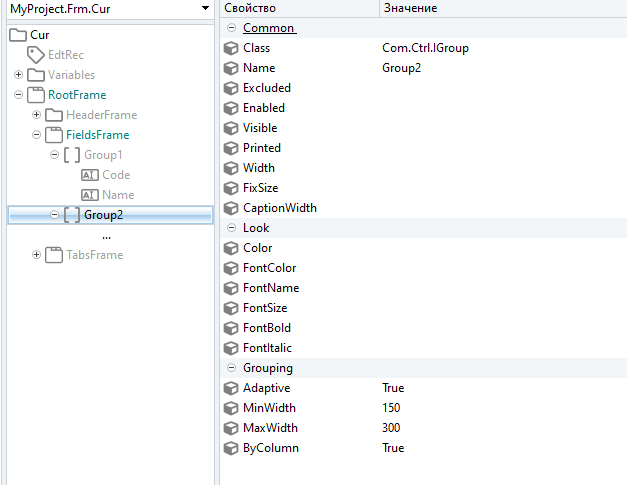
* + 1. Описание формы справочника валют

Создаем Frm-класс Cur и модифицируем заготовку:

|  |
| --- |
| class inherited Com.Frm.ICodeName "Валюта";  inclass public  var UseResource :Logical := true;  var RecordClass :Class Com.Rec.CodeName := Rec.Cur;  end |

В файле ресурсов в EdtRec указываем класс соответствующей записи (Rec.Cur), затем переходим во фрейм полей, включаем видимость полей Code и Name. Для размещения дополнительных полей справочника добавим группу:



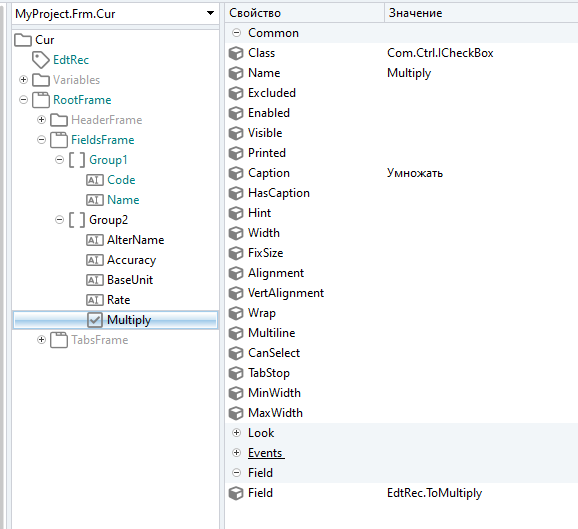


и зададим необходимые параметры:

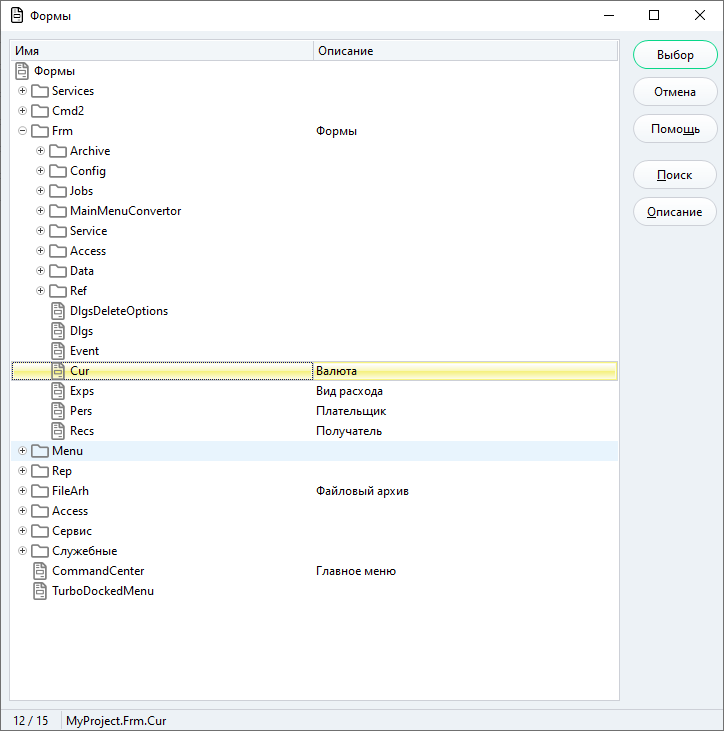
* адаптивность (автоматическое масштабирование по горизонтали)
* минимальную и максимальную ширину – пределы масштабирования
* расположение вложенных элементов «в колонку»

После этого в созданную группу добавляем компоненты для редактирования полей записи BaseUnit, Rate, Accuracy, ToMultiply и AlterName. Все они, кроме ToMultiply, будут класса Com.Ctrl.IEdit. Компонент ToMultiply – Com.Ctrl.ICheckBox. Имя компонента указывается в поле Name, заголовок (надпись на форме) – в Caption, в строке Field указывается редактируемое поле записи.

Описание компонента ToMultiply, например, выглядит следующим образом:



Для проверки результата запустим сессию (F9) и вызовем форму (Alt+B):



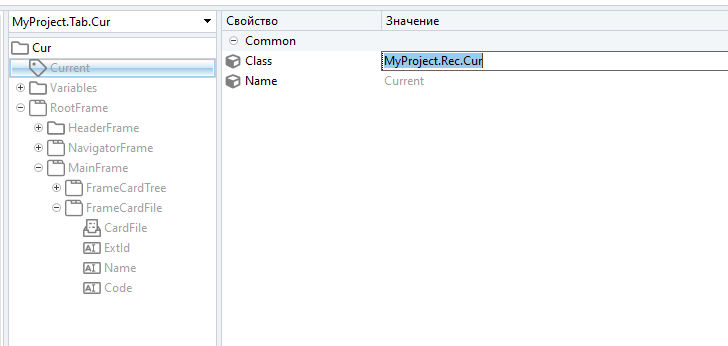
Внешний вид построенной формы (размеры, расположение элементов) предлагается улучшить самостоятельно

* + 1. Описание списочного представления

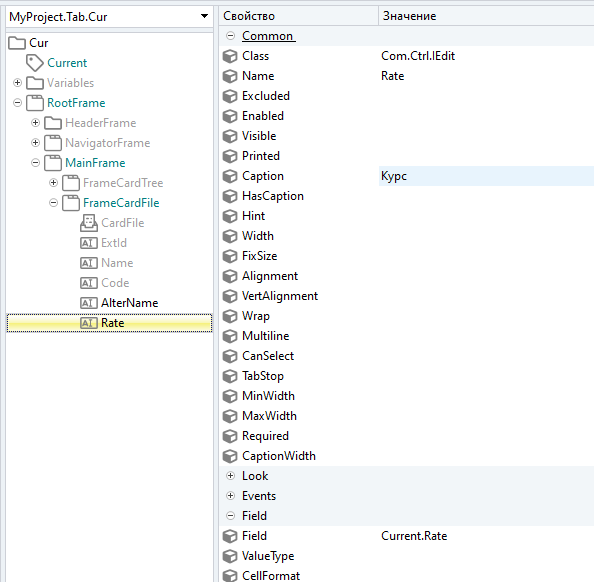
Создаем Tab - класс Cur и модифицируем заготовку:

|  |
| --- |
| class inherited Com.Tab.CodeName "Валюты";  inclass public  var UseResource :Logical := true;  var RecordClass :Class Com.Rec.IRecord := Rec.Cur;  var EditorClass :Class Com.Frm.IEditor := Frm.Cur;  end |

В редакторе ресурсов установим связь с записью:



и добавим отображаемые поля (практически так же, как на форме) – устанавливаем свойства Name, Caption и Field:



Для проверки результата запускаем сессию (F9) и открываем картотеку (Alt+T).

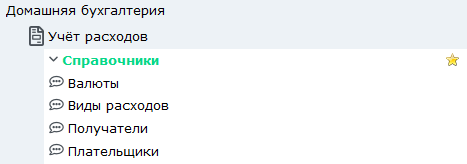
Последовательность колонок таблицы определяется последовательностью размещения элементов в редакторе ресурсов. Размер колонок устанавливается системой по умолчанию и может быть изменен разработчиком (совйства Width, MinWidth, MaxWidth). Пользователь также может в режиме сессии изменять порядок колонок («перетаскиванием») и их размер.

* 1. Командное меню

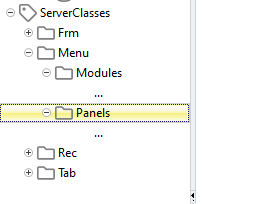
Командное меню обеспечивает пользователю простой доступ к функционалу проекта.

Меню в Turbo организовано иерархически: состоит из модулей (как правило, модуль содержит пункты, относящиеся к проекту), модули, в свою очередь, содержат панели (группы пунктов), панель включает в себя группы пунктов или команд. Каждая команда обеспечивает доступ к конкретной функции проекта, например – отображение картотеки, формы или отчета.

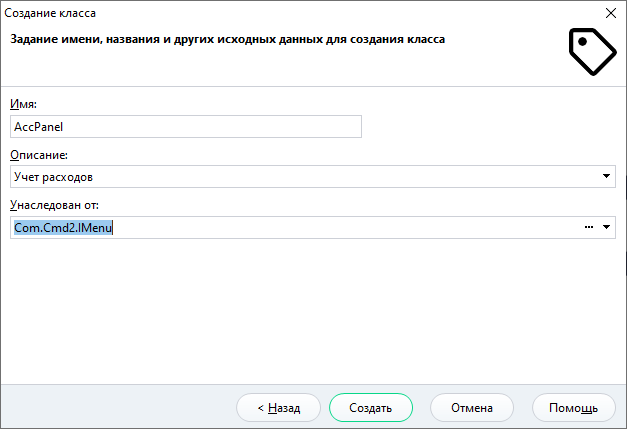
Создадим меню, состоящее из одного модуля (Домашняя бухгалтерия) с одной панелью (Учёт расходов) и группой команд для доступа к таблицам справочников (Справочники).



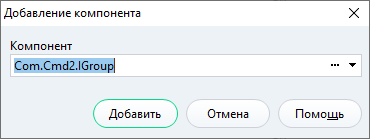
Для этого в группу объектов ServerClasses добавим папку Menu, в неё папки Modules и Panels



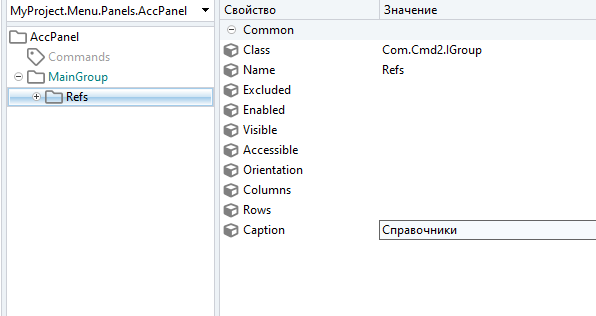
В папке Panels создадим класс AccPanel:



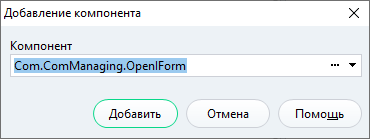
В редакторе ресурсов укажем заголовок (Caption) и добавим группу команд для вызова справочников (внутрь основной группы – MainGroup):



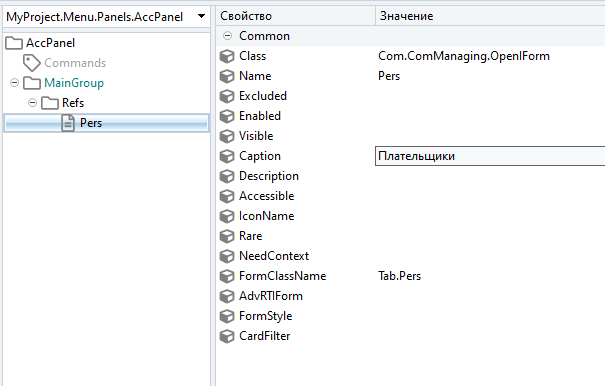
Зададим свойства Name и Caption



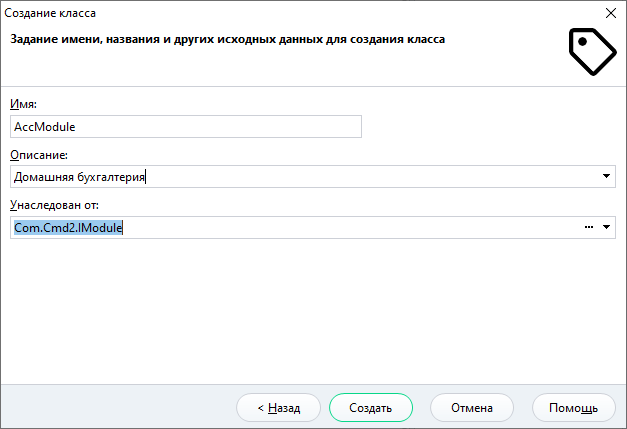
Теперь добавим команду открытия таблицы «Плательщики». Для этого в группу команд Refs добавим компонент класса Com.ComManaging.OpenIForm:

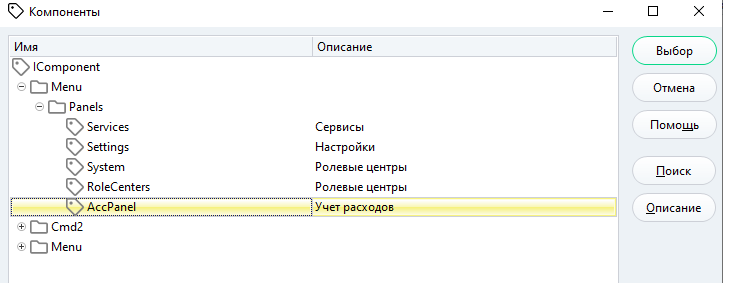


Как обычно, зададим свойства Name и Caption, также укажем имя класса, который будет вызван в этом пункте меню:

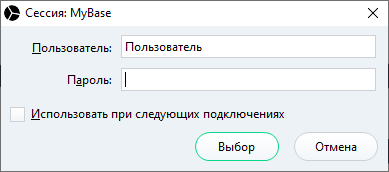


В папке Modules создадим модуль:

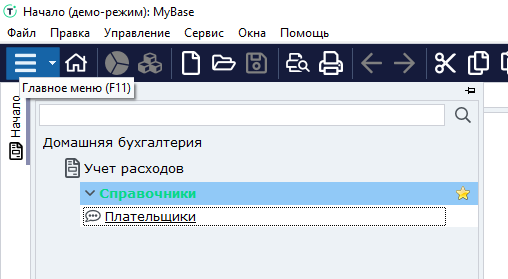


Перейдем к редактору ресурсов и добавим в модуль ранее созданную панель:

Для проверки войдем в сессию с логином «Пользователь»:



и вызовем меню:



Самостоятельно добавьте пункты для вызова остальных справочников и выберите подходящие иконки для элементов меню (свойство IconName для модуля и пунктов, свойство FormIcon для панели).

1. Урок 4. Реализация документа
   1. Описание записи документа

Поскольку документ имеет в своем составе табличную часть (строки), необходимо реализовать Rec-класс для документа в целом и Rec-класс для строки документа.

Класс Rec.Doc (для лучшего понимания структуры редактор ресурсов использовать не будем):

|  |
| --- |
| class inherited Com.Rec.CodeName "Счёт";  inclass public  var RecordClass :Class Record := Data.Doc;  inobject public  Date :Com.Fld.IField;  Person :Com.Fld.IField;  SumTotal :Com.Fld.IField;  Cont :Com.Fld.IField;  Cur :Com.Fld.IField;  Positions :Com.ISubtable;  inobject private  proc Init;  inherited Init;  Date = AddField( "Date", "Дата" );  Person = AddField( "Person", "Плательщик" );  Cont = AddField( "Cont", "Проведено" );  Cur = AddField( "Cur", "Валюта" );  SumTotal = AddField( "SumTotal", "Сумма чека" );  Positions = AddSubtable( "Positions", Rec.Doc\_Lines );  Cur.OnChange = proc (aSender: Com.Fld.IValue) SetCurForAllPositions; end;  end;  proc DoCreate;  inherited DoCreate;  Date.Value = Today;  Cont.Value = true;  end;  proc SetCurForAllPositions;  for var i = 1..Positions.Count do  (Positions.Items[i] as Rec.Doc\_Lines).SetCurr;  end;  end;  end |

Следует обратить внимание на процедуры DoCreate – пример обработчика события (создание новой записи) и SetCurForAllPositions – обработка строк при изменении валюты документа.

Класс Rec.Doc\_Lines:

|  |
| --- |
| class inherited Com.IStruct;  inclass public  type TDocPosition = Data.Doc.Positions;  var StructClass : Class Structure := TDocPosition;  inobject public  Recipient :Com.Fld.IField;  Sum :Com.Fld.IField;  Cur :Com.Fld.IField;  Expense :Com.Fld.IField;  Comment :Com.Fld.IField;  proc SetCurr;  Cur.Inner = Owner.Cur.Inner;  end;  inobject private  var Owner :Rec.Doc;  var FStruct :TDocPosition;  proc Init;  inherited Init;  Recipient = AddField ( "Recipient" );  Sum = AddField ( “Sum” );  Cur = AddField ( “Cur “ );  Expense = AddField ( “Expense” );  Comment = AddField ( “Comment” );  Sum.OnChange = proc (aSender: Com.Fld.IValue) SumOnChange; end;  AddOnDelete(@SumOnChange);  end;  proc SumOnChange;  Owner.SumTotal.Value = Subtable.SumOfField(“Sum”);  end;  proc DoCreate;  inherited DoCreate;  SetCurr;  end;  end |

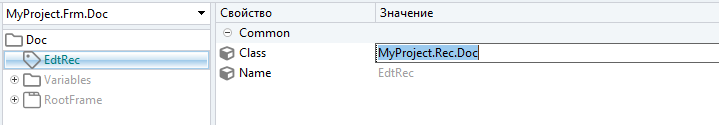
В этом классе обратите внимание на процедуры SetCurr и SumOnChange

* 1. Описание формы документа

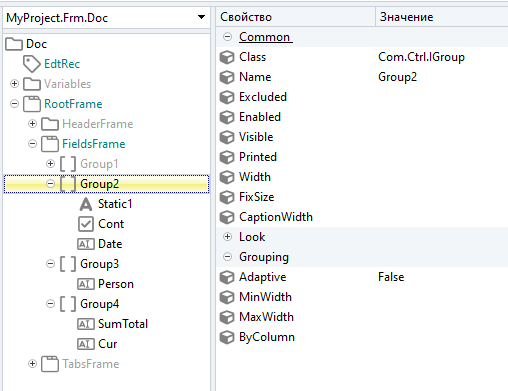
Создадим класс Frm.Doc и модифицируем так:

|  |
| --- |
| class inherited Com.Frm.ICodeName "Счет";  inclass public  var UseResource :Logical := true;  var RecordClass :Class Com.Rec.CodeName := Rec.Doc;  inobject private  EdtRec :Rec.Doc;  Positions :Rec.Doc\_Lines;  proc InitRecord;  inherited InitRecord;  Positions = EdtRec.Positions.DefStruct;  end;  end |

В файле ресурсов в компоненте EdtRec указываем класс Rec.Doc.



Переходим в RootFrame -> FieldFrame. Создадим следующую структуру:



Группа элементов Group1 входит по умолчанию, в ней мы ничего не меняем, на форме её видно не будет.

Каждая последующая группа представляет собой одну строку на форме. Чтобы поля имели постоянные размеры, в атрибуте Adaptive компонентов Group2/3/4 указываем False.

В группе элементов Group2 разместим:

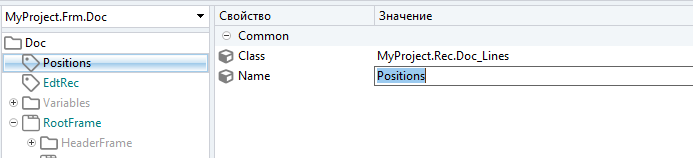
* «Пустой» элемент Static1 (Com.Ctrl.IStatic) - нужен для создания отступа в целях улучшения визуального представления. Установим его ширину (атрибут Width) равной 45.
* флаг Cont (Com.Ctrl.CheckBox). Добавим заголовок «Проводить» и установим ширину 25. Соответствующее поле изменяемого класса (атрибут Field) – EdtRec.Cont.
* Последним в строке будет поле Date (Com.Ctrl.IEdit). Укажем заголовок «Дата», ширину 35, редактируемое поле – EdtRec.Date.

В группе Group3 разместим элемент Person. Класс – Com.Ctrl.IEdit, заголовок – «Плательщик», ширина – 105, изменяемое поле – EdtRec.Person. Также необходимо подключить справочник «Плательщики» из которого будет делаться выбор. Для этого в поле CardFileClass указываем класс соответствующей картотеки – Tab.Pers.

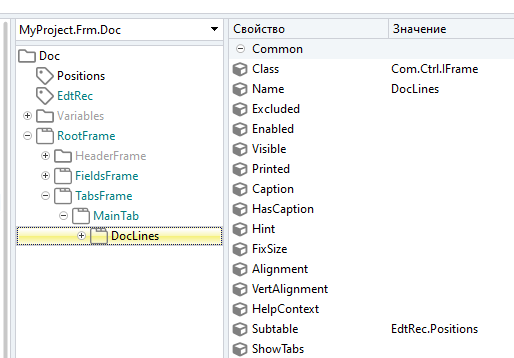
В группе Group4 будем отображать общую сумму и обозначение валюты:

* Поле SumTotal: class - Com.Ctrl.IEdit, Enabled – False (сумма по документу считается автоматически), Caption – «Итого», Width – 55, Field – EdtRec.SumTotal.
* поле Cur: class – Com.Ctrl.IEdit, HasCaption – False, Width – 50, Field – EdtRec.Cur, CardFileClass – Tab.Cur, LookupField – AlterName.

Для работы с подтаблицей необходим отдельный компонент («добавить библиотеку») «рядом» с EdtRec. Указываем класс записи Rec.Doc\_Lines и название Positions:



Перейдём теперь к созданию подтаблицы. Заходим во фрейм таблиц (TabsFrame). Включим его отображение на форме указанием значения True в поле Visible. Теперь откроем MainTab и добавим компонент класса Com.Ctrl.IFrame. Назовём его DocLines, в поле Subtable указываем EdtRec.Positions. Это основа нашей подтаблицы.

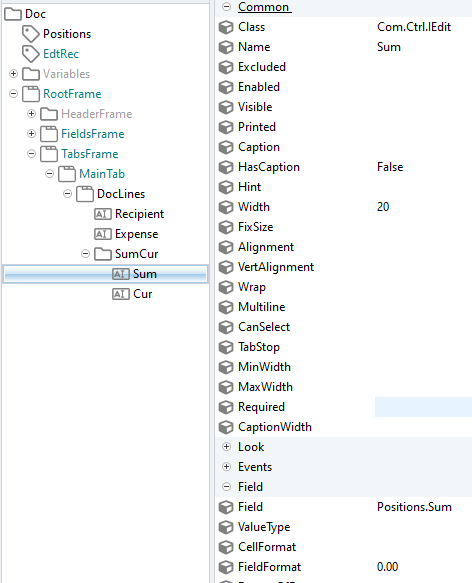


Теперь надо добавить в неё колонки. Добавляем элемент класса Com.Ctrl.IEdit с имененем Recipient. Укажем заголовок столбца «Получатель», ширина – 50, редактируемое поле подтаблицы (строка Field в настройках) – Positions.Recipient, класс картотеки – Tab.Reci.

Аналогичнгым образом добавим колонку Expense.

Объединим столбцы суммы и валюты под общим заголовком при помощи компонента класса Com.Ctrl.IColumnsGroup. Назовём его SumCur. В Caption укажем «Сумма», в HasCaption – True.

Для столбца «Сумма» зададим ширину = 20, формат поля = 0.00 и отсутствие заголовка:



В столбце Cur не забываем указать CardFileClass и AlterName.

Колонку для ввода комментария прделагается настроить самостоятельно

* 1. Описание списочного представления документа

Класс Tab.Docs:

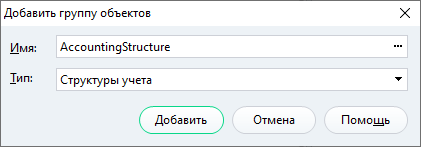
|  |
| --- |
| class inherited Com.Tab.CodeName "Счета";  inclass public  var RecordClass : Class Com.Rec.IRecord := Rec.Doc;  var EditorClass : Class Com.Frm.IEditor := Frm.doc;  inobject private  var Current : Rec.Doc;  proc Init;  inherited Init;  AddFieldCol( , 30, , Current.Cont );  AddFieldCol( , 30, , Current.Date );  AddFieldCol( , 80, , Current.Person , "Name" );  with AddFieldCol( , 27, , Current.SumTotal ) as Com.Ctrl.IEdit do  FieldFormat = '0.00';  end;  AddFieldCol( , 27, , Current.Cur , "AlterName" );  end;  end |

После описания таблицы документа читателю предлагается самостоятельно добавить в меню команду для доступа к ней (аналогично тому, как это делалось со справочниками).

1. Урок 5. Структура учета и отчет
   1. Проводки

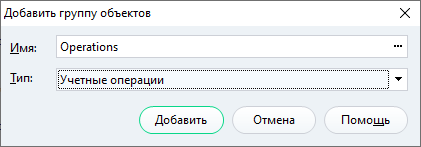
Формирование отчетов в Turbo базируется на механизме проводок. Проводка – запись, содержащая совокупность значений, которые необходимо отображать в отчетах. Как правило, проводка содержит количественныые показатели с единицей измерения (в нашем случае – сумма расходов и валюта) и набор аналитических признаков, позволяющих фильтровать и группировать данные (например, в нашей системе можно отобрать суммарные расходы по видам для конкретного плательщика и т.п.) и основной учетный элемент – «счет».

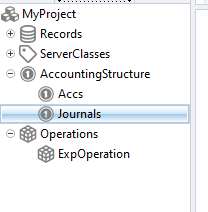
Проводки формируются системой по определенным разработчиком правилам автоматически. Правила описываются в отдельной папке проекта c типом «Структуры учета»:



Соответственно, в этой папке расположим описание счетов и аналитических признаков (область учета) и описание учетного журнала (совокупность формируемых проводок)

Правила формирования проводок располагаются в папке с типом «Учетные операции»:





Соответствующие тексты приведены ниже, синтаксис и семантика описаны в документации и справке.

* 1. Область учета

Для описания области учета в группе AccountStructures создадим элемент Accs:

|  |
| --- |
| Domain Home notaccumulated, bytrans, uncloseable;  reference refUnits synonym спрВалют :Cur, from Refs.Cur, card Tab.Cur, title "Валюта";  field Code synonym Name, Код :String, from Code;  field Description synonym Наименование :String, from Name, notrefresh, notloaded;  field Accur synonym Accuracy :Integer, from Accuracy;  field Base synonym BaseUnit :refUnits, from BaseUnit;  field Mult synonym ToMultiply :Logical, from ToMultiply;  field Rate :Numeric period Date, from Rate;  necessary  руб;  aud;  end;  reference refExpenses, from Refs.Expense , card Tab.Exps, title "Расходы";  field Code synonym Name, Код :String, from Code;  field Description synonym Наименование :String, from Name, notrefresh, notloaded;  end;  reference refPersons, from Refs.Person , card Tab.Pers, title "Плательщики";  field Code synonym Name, Код :String, from Code;  field Description synonym Наименование :String, from Name, notrefresh, notloaded;  end;  reference refRecipients, from Refs.Recipient ,card Tab.Pers, title "Получатели";  field Code synonym Name, Код :String, from Code;  field Description synonym Наименование :String, from Name, notrefresh, notloaded;  end;  accounts Accs;  type BaseAcc;  parameter Sum : unit refUnits = 0.00^руб;  parameter Expense : refExpenses = nil;  parameter Person : refPersons = nil;  parameter Recipient : refRecipients = nil;  separator Expense;  separator Person;  end;  acc 500 : BaseAcc Title "Money";  acc 600 : BaseAcc Title "Supplyers";  end; |

* 1. Проводки

Описания проводок хранятся в элементе ExpOperation группе объектов Operations»:

|  |
| --- |
| class "Exp";  oper ExpTrans(  SumO : numeric;  ExpenseO : refExpenses;  PersonO : refPersons;  RecipientO : refRecipients;  Cur : refUnits  );  Transaction Accs.600 , Accs.500,  +Sum = SumO^Cur,  -Sum = 0,  Expense = ExpenseO,  Person = PersonO,  Recipient = RecipientO;  end;  end |

* 1. Журнал операций

Описание журнала располагаем в папке AccountStructures, элемент Journals:

|  |
| --- |
| Domain Home notaccumulated, bytrans, uncloseable;  Journal Expenses title "Our Expences" : freetransenter ;  Priority 0;  Records Data.Doc;  Card com.TempoCard;  DateField Date;  EnableField Cont;  Operation ExpOperation.ExpTrans from Positions;  parameter SumO from Sum;  parameter ExpenseO from Expense;  parameter PersonO from "^Person";  parameter RecipientO from Recipient;  parameter Cur from Cur;  end;  end; |

* 1. Отчет

Рассмотрим пример простого отчета: предусмотрим возможность фильтрации данных по видам расходов, плательщику и плдучателю:

Класс ServerClasses\Rep\Report:

|  |
| --- |
| class inherited Com.Frm.ITurnReport "Отчет";  import Com classes Consts;  inobject public  var RC\_Плательщик :Rep.IRepControl;  var RC\_Получатель :Rep.IRepControl;  var RC\_ВидРасхода :Rep.IRepControl;  inobject private  proc InitParamFrame;  inherited InitParamFrame;  with ParamFrame do  RC\_Плательщик = AddRepControl( 'RC\_Плательщик' , 'Person' );  RC\_Получатель = AddRepControl( 'RC\_Получатель' , 'Recipient' );  RC\_ВидРасхода = AddRepControl( 'RC\_ВидРасхода' , 'Expense' );  SetColumns([-40, -120]);  AddEdit('Плательщик' , , RC\_Плательщик , , ReferenceFormat , 'Name' , Tab.Pers, true, frm.Pers);  AddEdit('Получатель' , , RC\_Получатель , , ReferenceFormat , 'Name' , Tab.Recs);  AddEdit('Вид Расхода' , , RC\_ВидРасхода , , ReferenceFormat , 'Name' , Tab.Exps);  end;  end;  proc InitReport;  inherited InitReport;  with FReport do  AccountPlan = 'Accs';  ExtFilter = 'Sum <> nil';  SetSumFormats(Report.sfNone, Report.sfRoll, Report.sfNone);  AddSplit(Report.rdRow ,, 'Date');  AddSplit(Report.rdRow ,, 'Person');  AddSplit(Report.rdRow ,, 'Expense');  AddSplit(Report.rdRow ,, 'Recipient');  AddContent(Report.Rdrow, 'Date', 'Дата');  AddContent(Report.Rdrow, 'Person.Name', 'Плательщик');  AddContent(Report.Rdrow, 'Expense.Name', 'Вид расходов');  AddContent(Report.Rdrow, 'Recipient.Name', 'Получатель');  AddIndicator('Sum',,'Сумма');  end;  end;  end |

Вызвать отчет можно с помощью диалога выбора формы (Alt+B).

Добавление в меню группы «Отчеты» и команды вызова отчета читателю предлагается выполнить самостоятельно.