ФГБОУ ВО Калининградский государственный технический университет

Институт цифровых технологий

Соловей Марина Викторовна

Учебно-методическое пособие по выполнению лабораторных работ по дисциплине

«АРХИТЕКТУРА И РАЗРАБОТКА ИНФОРМАЦИОННЫХ СИСТЕМ МАЛЫХ И СРЕДНИХ ПРЕДПРИЯТИЙ»

Калининград, 2022

Пособие рассмотрено и одобрено методической комиссией Института цифровых технологий. Протокол от «20» сентября 2022 г., № 6

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данное учебно-методическое пособие предназначено для студентов направления **9.03.03** «Прикладная информатика», изучающих дисциплину «Архитектура и разработка информационных систем малых и средних предприятий». Материалы для лабораторных работ взяты из источника: Официальный сайт интернет-униерситета «Интуит» (электронный ресурс) , режим доступа

http://www.intuit.ru/studies/courses/2318/618/lecture/17939.

Цель лабораторного практикума по дисциплине: изучить принципы проектирования ИС на платформе ««1С:Предприятие»».

Лабораторный практикум содержит 5 лабораторных работ. Лабораторные работы предназначены для студентов очного и заочного форм обучения.

Лабораторные работы проводятся в лабораториях 143, 256, 353 и других, где установлена платформа ««1С:Предприятие»» .

В результате выполнения лабораторных работ ожидается, что студенты будут:

Знать: роль и место информационных систем малых и средних предприятий в экономике; виды и классификацию автоматизированных информационных систем; историю развития, закономерности построения и функционирования АИС; состав, структуру и архитектуры АИС (функциональные подсистемы АИС, обеспечивающие подсистемы АИС, элементы и средства АИС); методологию и технологии обследования и разработки различных типов АИС и отдельных видов обеспечения, основные этапы жизненного цикла АИС и их особенности, стандартные этапы проектирования АИС, а также технологию и методологию внедрения АИС; методологические основы создания АИС в управлении предприятием (системный, информационный, стратегический и объектно-ориентированный подходы, разработка информационной модели системы управления предприятием); современные средства информационных и коммуникационных технологий обеспечения управленческой деятельности; основы документирования проектных решений по созданию АИС.

Уметь: обосновывать необходимость и целесообразность автоматизации ИС; составлять техническое задание на создание АИС; выбирать инструментальные средства создания АИС; осуществлять конфигурирование типовых проектных решений по созданию АИС и выполнять их адаптацию к конкретным условиям применения; документировать проектные решения по созданию АИС, готовить организационно-распорядительную документацию стадии ввода АИС в действие; осуществлять мероприятия по вводу АИС в действие.

Владеть: способностью составлять техническую документацию проектов информационных систем различного масштаба.

ЛАБОРАТОРНАЯ РАБОТА №1.

СОЗДАНИЕ ИНФОРМАЦИОННОЙ БАЗЫ. РАБОТА С КОНСТАНТАМИ. ОСНОВЫ КЛИЕНТ-СЕРВЕРНОГО ПРОГРАММИРОВАНИЯ

ОБЩИЕ СВЕДЕНИЯ

Цель: научиться информационную базу на платформе ««1С:Предприятие»»

Материалы, оборудование, программное обеспечение:
1. персональный компьютер (компьютерные классы ГУК)
2. программное обеспечение «1С:Предприятие»

Условия допуска к выполнению:
умение работать на ПК и знание техники безопасности

Критерии положительной оценки:
предоставление результатов работы в виде файла и прохождение защиты.

Планируемое время выполнения:
Аудиторное время выполнения (под руководством преподавателя): 4 ч.
Время самостоятельной подготовки: 2 ч.

Теоретическое введение

Создание новой информационной базы

Откроем окно запуска «1С:Предприятие», создадим новую пустую информационную базу. Для этого нажмем на кнопку **Добавить**, в появившемся окне выберем Создание новой *информационной базы*, в следующем окне, рис. 1.1., выберем вариант создания *информационной базы* без конфигурации.

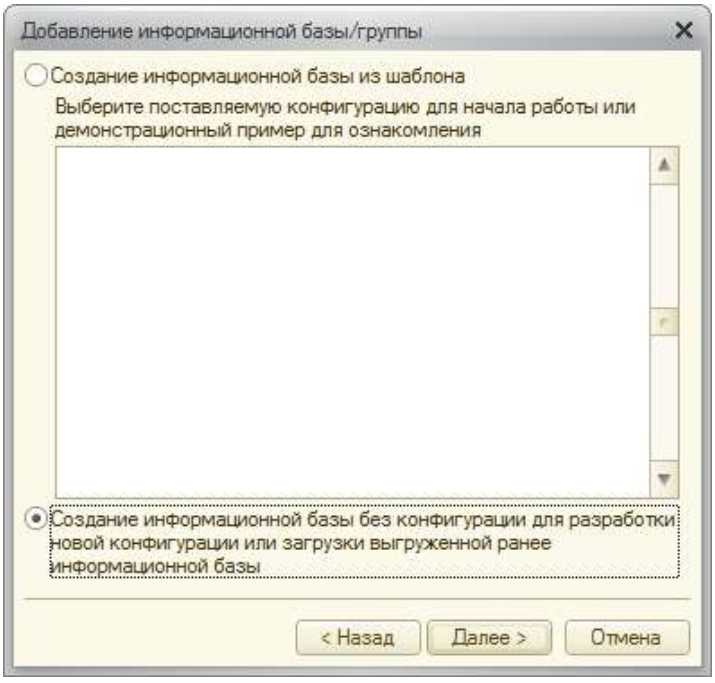


Рисунок 1.1. Создание информационной базы без конфигурации

Дадим информационной базе имя **Салон, управляемое приложение**, зададим в качестве папки *информационной базы* **C:\Salon1**, остальные параметры оставим по умолчанию.

После того, как база будет создана, откроем ее в **Конфигураторе** и, для того, чтобы открыть *дерево конфигурации*, выполним команду **Конфигурация > Открыть конфигурацию**. Вызовем контекстное *меню корневого элемента Конфигурация*, выберем в нем *пункт* **Свойства**, Рисунок 1.2.

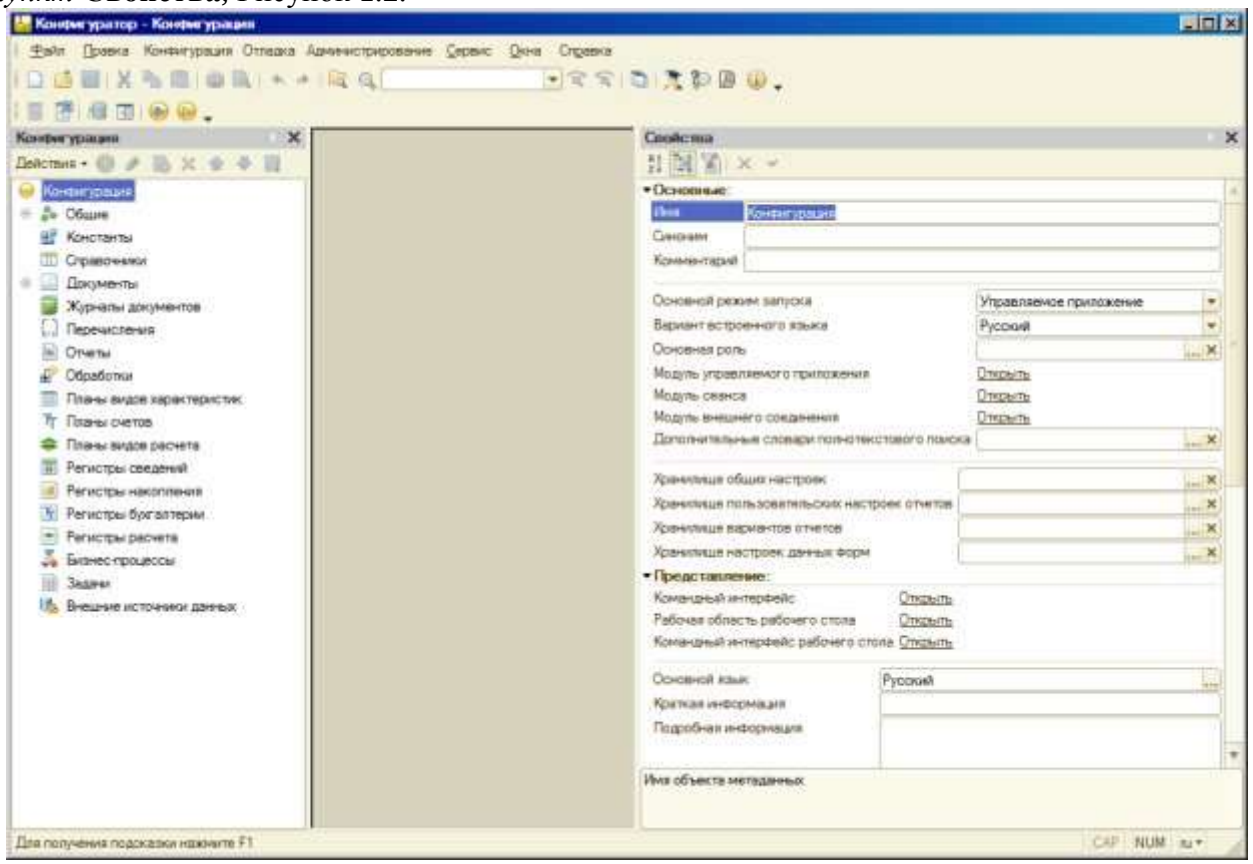


Рисунок 1.2. Свойства новой информационной базы

Обратите внимание на то, что свойство **Основной режим запуска** установлено в *значение* **Управляемое приложение**, в нижней части окна свойств расположено свойство **Режим совместимости**, которое установлено в *значение* **Не использовать**. В данном случае оно может принимать значения Версия 8.1. и Версия 8.2.13.

В качестве имени конфигурации введем **СалонКрасоты**, *поле* **Синоним** будет автоматически заполнено текстом **Салон красоты**.

Можно заметить, что изменились изображения интерфейсных элементов в **Конфигураторе**. Все говорит нам о том, что сейчас мы занимаемся разработкой конфигурации в режиме управляемого приложения. Среди нововведений платформы 8.2. можно отметить изменение состава объектов конфигурации. В частности, появились следующие новые объекты:

Общие реквизиты: здесь содержатся реквизиты, которые могут использоваться во многих объектах конфигурации. Например, если вы планируете добавить в документы своей конфигурации одинаковый *реквизит*, содержащий наименование организации, от имени которой составлен документ, это вполне логично реализовать с помощью общего реквизита. Кроме того, общие реквизиты используются в механизме разделения данных.

Функциональные опции: их используют для того, чтобы описывать возможности, которые можно включать и отключать в процессе эксплуатации системы. Функциональные опции могут влиять на командный *интерфейс*, например, скрывая или

отображая некоторые группы команд, а так же – на алгоритмы, написанные на встроенном языке.

Параметры функциональных опций: Содержит параметры, влияющие на функциональные опции

Хранилища настроек: Используется для сохранения и загрузки настроек.

Общие команды: Позволяет создавать команды, которые можно использовать в других объектах конфигурации, вызывая их, например, с помощью кнопок на формах.

Группы команд: Позволяет создавать группы для объединения команд

Элементы стиля: Позволяет создавать элементы стиля, такие, как цвет, *шрифт*, рамка, для организации единообразного оформления других объектов.

Внешние источники данных: эти объекты используются для получения информации из внешних источников и последующего использования ее в системе, в частности, в качестве источников данных для запросов, в качестве типов реквизитов*информационной базы* и так далее.

Теперь в *дереве конфигурации* нет следующих объектов:

Интерфейсы

Стили

Кроме того, некоторые объекты, в частности, это касается подсистем, теперь используются по-другому. Начнем разработку обновленной конфигурации с создания подсистем.

Подсистемы – основа командного интерфейса управляемого приложения

Ранее, когда понятия командного интерфейса не существовало, подсистемы при разработке *прикладных решений* для «1С:Предприятие», играли вспомогательную роль. В сущности, они были нужны лишь разработчику, помогая ему структурировать конфигурацию для своих целей и автоматизировать некоторые *операции*. С приходом командного интерфейса подсистемы приобрели новую роль – именно на их основе строится *интерфейс* приложений. Поэтому с них мы и начнем работу по созданию нашей конфигурации.

Создадим подсистемы в новой конфигурации:

- Оперативный учет материалов
- Учет работы мастеров
- Администрирование

Порядок создания новых объектов конфигурации выглядит так же, как он выглядел ранее. Для создания новой подсистемы нужно перейти в *ветвь* дерева конфигурации **Общие > Подсистемы**, после чего либо выбрать команду **Добавить** из контекстного *меню* ветви **Подсистемы**, либо выделить эту *ветвь* и нажать клавишу **Ins** на клавиатуре, либо воспользоваться кнопкой **Добавить** из командной панели дерева конфигурации. После этого появится окно редактирования объекта конфигурации, приведенное на Рисунок 1.3.

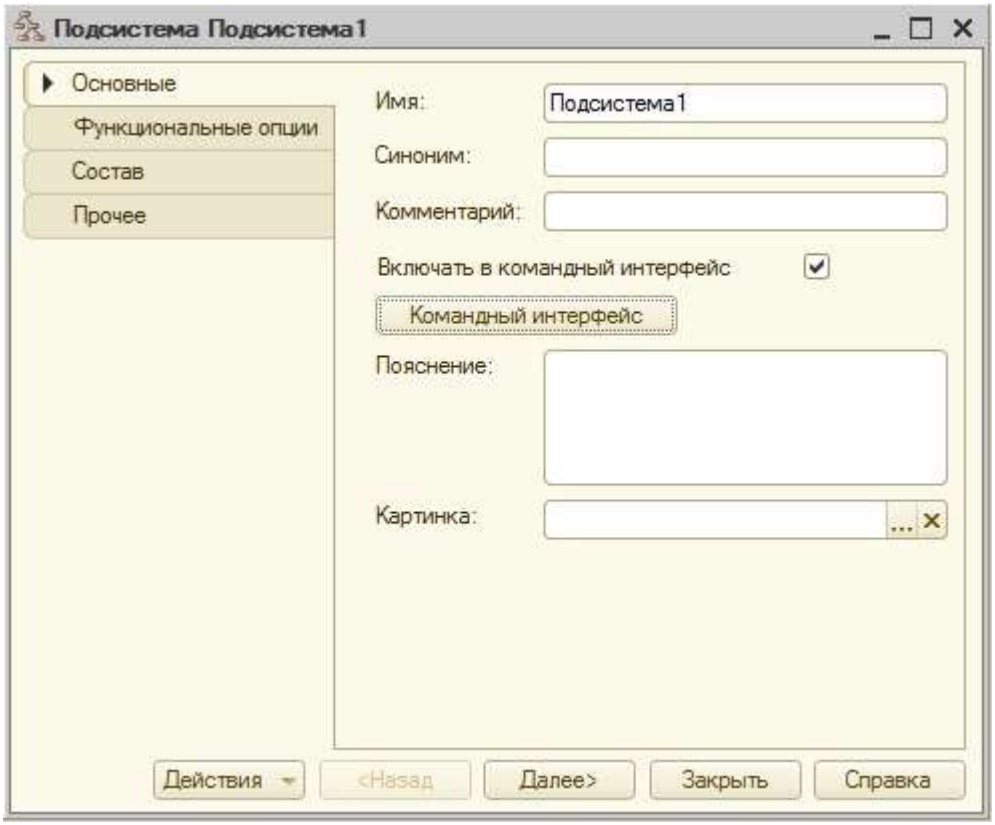


Рисунок 1.3. Окно редактирования объекта

Здесь можно либо перемещаться по вкладкам окна в произвольном порядке, либо, используя кнопку **Далее**, перемещаться по ним последовательно.

Зададим следующие параметры для нашей новой подсистемы:

Имя: УчетРаботыМастеров

Синоним: *Учет работы мастеров*

Синоним генерируется автоматически на основе имени, при необходимости его можно отредактировать вручную.

Поле **Картинка** можно использовать для того, чтобы задать подсистеме заранее созданную картинку. Это позволяет сделать *интерфейс* пользователя более удобным.

После того, как подсистема создана, посмотрим, на что будет похожа разрабатываемая *конфигурация* в режиме «1С:Предприятие». Запустим ее в этом режиме из Конфигуратора, воспользовавшись комбинацией клавиш **Ctrl+F5**, соответствующей командой *меню*(**Сервис > «1С:Предприятие»**), или кнопкой на панели инструментов **Конфигурация**.

То, что мы увидим после запуска конфигурации, разительно отличается от того, что мы привыкли видеть, Рисунок 1.4.

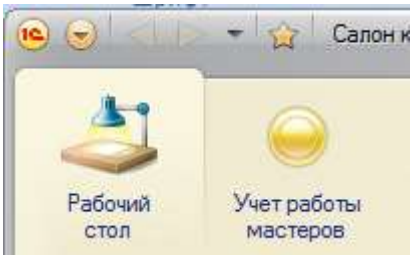


Рисунок 1.4. Разрабатываемая конфигурация в режиме «1С:Предприятие»

Рабочий стол нужен для ускорения доступа пользователя к наиболее часто используемым объектам системы. Это – одна из закладок командного интерфейса, которая появляется первой при открытии конфигурации в пользовательском режиме.

Наша подсистема видна в верхней части окна программы, в так называемой панели разделов. Она снабжена стандартным рисунком, назначаемым автоматически, подпись соответствует синониму. Щелчок по вкладке "**Бухгалтерский учет**" приведет нас к командам по работе с объектами конфигурации, которые включены в эту подсистему.

Здесь хочется обратить ваше внимание на кнопку **Главное меню**. Она открывает *меню*, содержащее стандартные для Windows-программ команды, Рисунок 1.5.

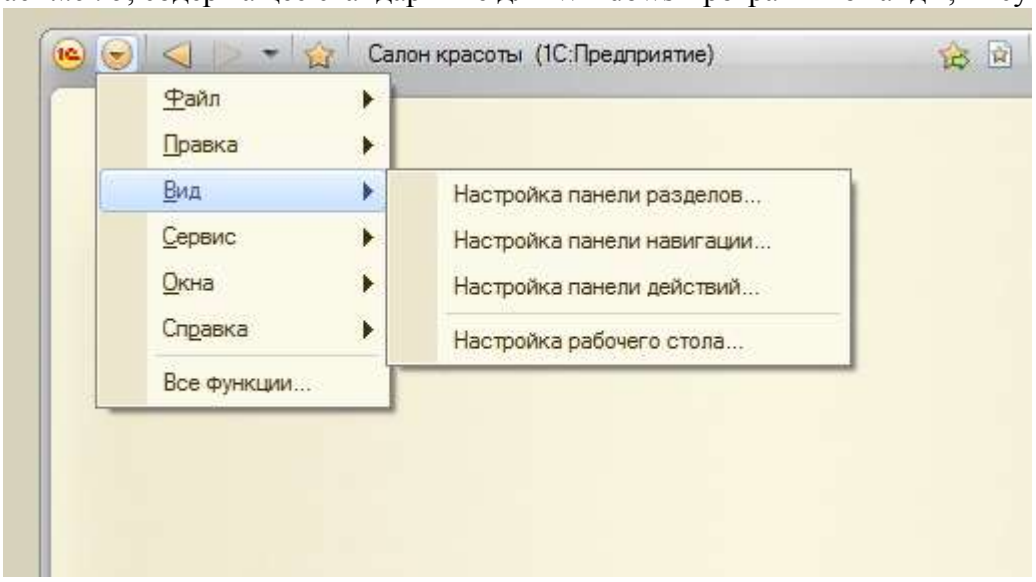


Рисунок 1.5. Главное меню в режиме «1С:Предприятие»

В сравнении с «1С:Предприятие» 8.1. в составе разделов этого *меню* многое поменялось (в особенности это касается разделов **Вид**,**Сервис**). В частности, обратите внимание на команду **Главное меню > Все функции**. Эта *команда*, Рисунок 1.6., открывает *доступ*к дереву объектов конфигурации, позволяет использовать некоторые стандартные команды.

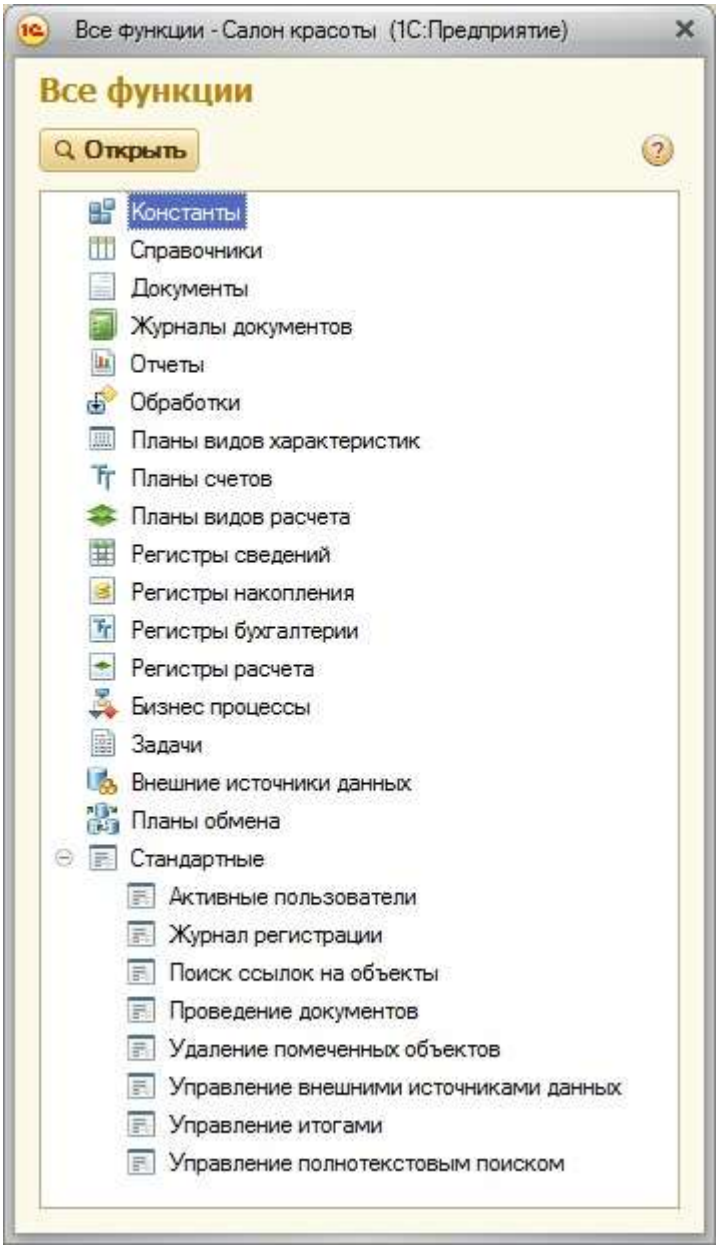


Рисунок 1.6. Окно Все функции

Особенно это окно полезно при разработке и отладке конфигурации – для быстрого поиска необходимых объектов без использования основного пользовательского интерфейса, для выполнения административных функций (таких, как удаление помеченных объектов, просмотр журнала регистрации). В законченной конфигурации есть смысл создать отдельную подсистему, которая будет содержать набор команд для вызова административных функций.

Вернемся в *Конфигуратор*, добавим к списку подсистем, которые следует создать, оставшиеся подсистемы – «Оперативный учет материалов» и «Администрирование».

Снова откроем конфигурацию в режиме «1С:Предприятие», Рисунок 1.7.

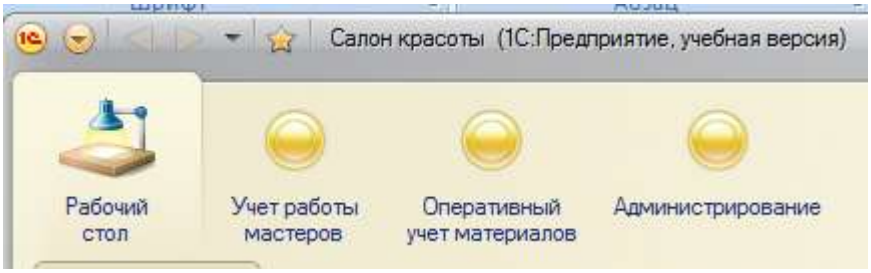


Рисунок 1.7. Панель разделов после добавления подсистем

Если порядок расположения разделов не соответствует желаниям пользователя, то, для того, чтобы изменить порядок следования подсистем в панели разделов нужно воспользоваться командой контекстного *меню* корневого объекта дерева конфигурации **Открыть командный интерфейс конфигурации**, Рисунок 1.8.

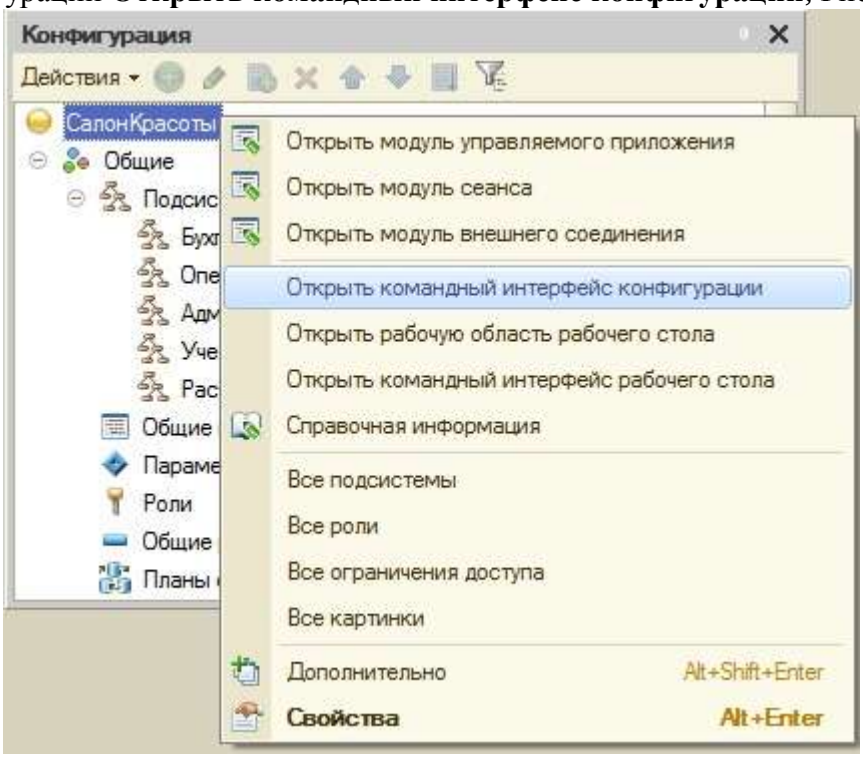


Рисунок 1.8. Открыть командный интерфейс конфигурации

В появившемся окне мы можем управлять порядком следования подсистем на панели разделов и их видимостью. Еще одной полезной возможностью настройки видимости подсистем является видимость по ролям. С помощью этого механизма можно конструировать интерфейсы для отдельных ролей, которые можно назначать пользователям, формируя, таким образом, рабочую среду, которая не содержит ничего лишнего. Настроим порядок следования подсистем с помощью кнопок **Переместить вверх** и**Переместить вниз** .

После нажатия на кнопку **ОК** и запуска конфигурации в пользовательском режиме, внесенные изменения можно будет наблюдать на панели разделов.

Теперь внесем в нашу конфигурацию еще одно изменение. Добавим в *ветвь* **Справочники** новый справочник, назовем его**Сотрудники** (*реквизит* Имя на вкладке **Основные**) и добавим во все подсистемы, Рисунок 1.9.

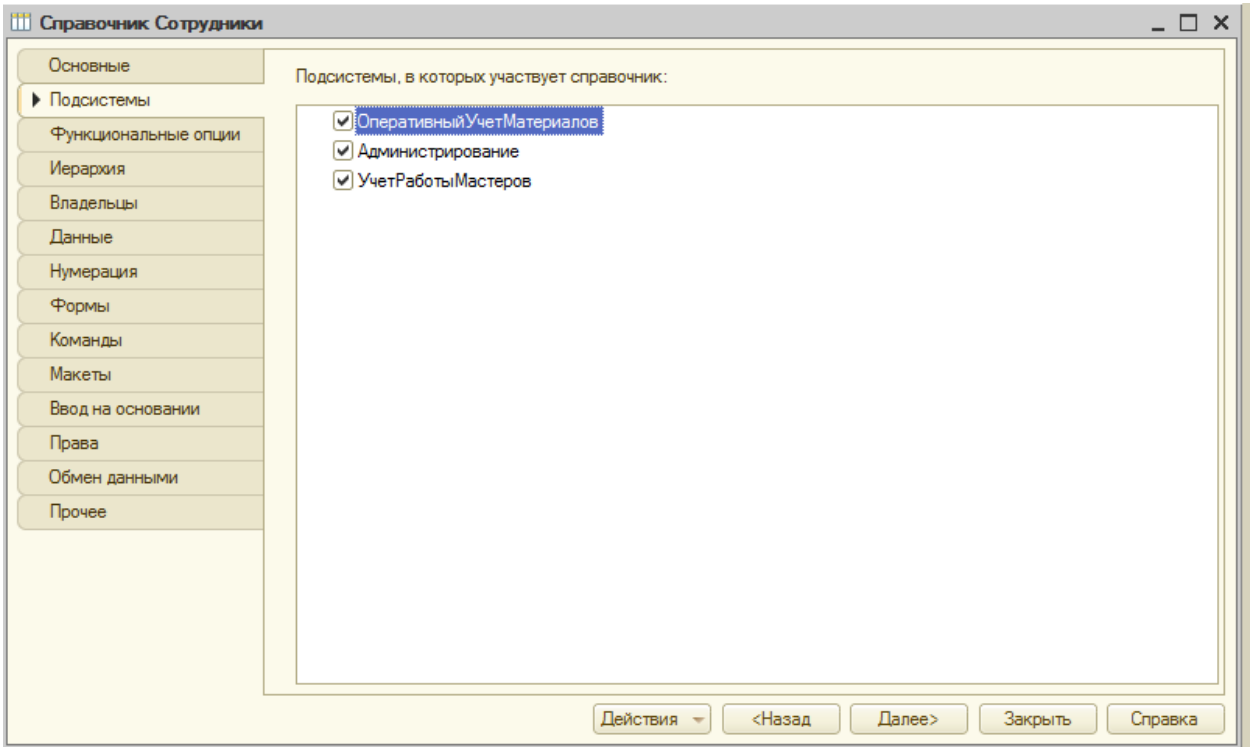


Рисунок 1.9. Новый справочник, добавленный во все подсистемы

Константы

Константы в «1С:Предприятие» используются для хранения информации, которая либо не меняется никогда, либо меняется – но очень редко. *Константы* содержатся в ветви *дерева конфигурации* **Константы**. Создадим новую константу (Рисунок 1.10), заполним ее параметры следующим образом:

Имя: ТекстСообщения

Тип: Строка

Длина: 50

Включим константу в состав подсистемы **УчетРаботыМастеров (Свойства-Дополнительно)**. Предполагается, что данная константа будет использоваться для показа сообщения пользователям, входящим в систему.

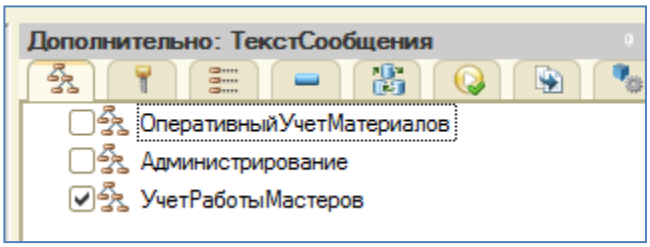


Рисунок 1.10. Настройка параметров новой константы

Посмотрим, как включение *константы* в подсистему **УчетРаботыМастеров**, отразится на интерфейсе нашего приложения в режиме «1С:Предприятие». Видно, Рисунок 1.11., что в разделе **Учет работы мастеров**, под панелью разделов, появилась еще одна панель. Она называется **панелью действий**. В панель действий автоматически включаются команды, разбитые на группы –**Сервис**, **Создать**, **Отчеты**. Группы в панели действий можно создавать и самостоятельно. В нашем случае в панели

действий видна *группа* **Сервис**, содержащая команду для работы с только что созданной константой.

Рисунок 1.11. Константа в панели действий в разделе Учет работы мастеров

В левой части окна программы можно видеть еще одну панель – она называется **панелью навигации.** Сейчас она отображает ссылку для доступа к справочнику **Сотрудники**, который мы создавали в предыдущей лекции. Свободная часть окна – это рабочая область, в которой, например, открываются списки справочников.

Щелкнем по команде **Текст сообщения** в панели действий. Отобразится окно, которое позволяет нам редактировать константу**ТекстСообщения**. Введем в *поле* **Текст сообщения** строку "Здравствуйте, уважаемый *пользователь*!", Рисунок 1.12. и нажмем на кнопку **Записать и закрыть.**

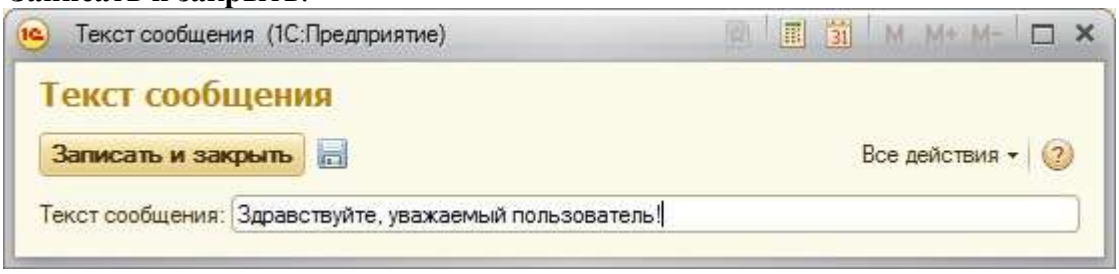


Рисунок 1.12. Форма редактирования константы Текст сообщения

Если мы не хотим сохранять внесенные изменения, можно просто закрыть окно с помощью стандартной кнопки **Закрыть**, для записи изменений без закрытия формы служит кнопка **Записать объект**.

Для того, чтобы воспользоваться дополнительными возможностями по работе с формой, можно использовать *меню* **Все действия,** Рисунок 1.13.

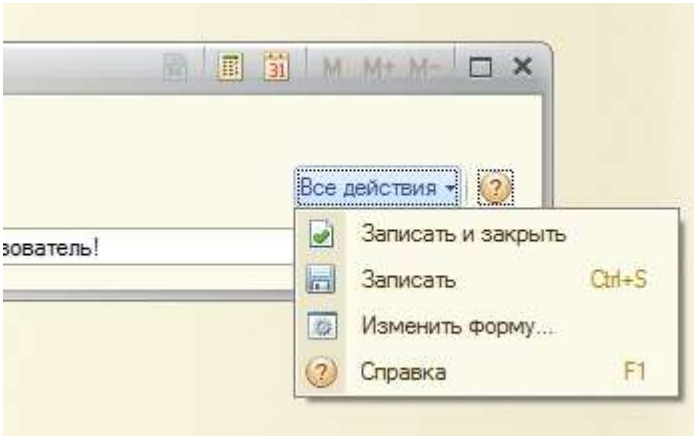


Рисунок 1.13. Меню Все действия

Отчасти оно дублирует кнопки, имеющиеся на форме, в нем так же имеется одна специфичная для платформы «1С:Предприятие» 8.2. *команда*. А именно, речь идет о команде **Изменить форму**.

Форма, которую мы видим, сформирована автоматически. Однако, в режиме «1С:Предприятие» мы можем вносить в нее некоторые изменения. Выполним команду **Изменить форму**, появится окно Настройка формы, Рисунок 1.14.

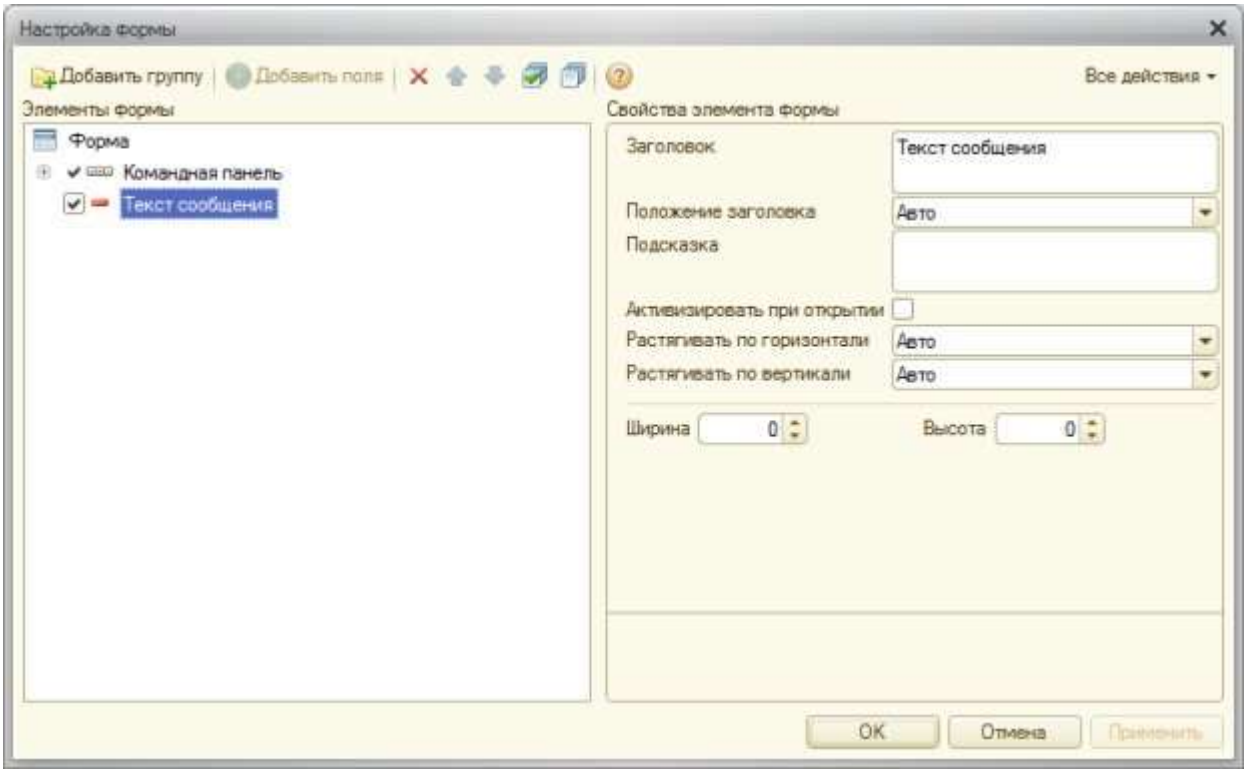


Рисунок 1.14. Окно Настройка формы

В нашем случае, Рисунок 1.14, в группе **Элементы формы** выделен элемент **Текст сообщения**, в группе **Свойства элемента формы** мы можем настраивать его свойства. Изменим свойство **Заголовок**, вместо "Текст сообщения" введем "Текст сообщения для пользователей", в итоге форма будет выглядеть так, как показано на Рисунок 1.15.

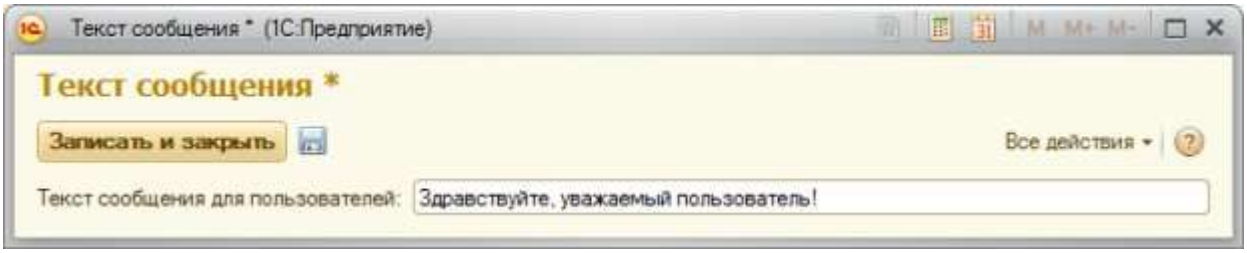


Рисунок 1.15. Отредактированный заголовок объекта

Перейдем в режим конфигурирования, создадим еще одну константу (она пригодится нам позже):

Имя: ПрефиксНомера

Тип: Строка

Длина: 2

Включим эту константу в подсистему **Администрирование**. В режиме «1С:Предприятие» *доступ* к этой константе будет организован в группе **Сервис** панели действий раздела **Администрирование**. Кроме того, мы можем организовать *доступ* к константам из других мест нашего приложения. Мы можем самостоятельно включить команду для вызова формы просмотра и редактирования*константы*, отредактировав командный *интерфейс*, можем так же создать специальную форму, называемую **формой констант**.

Форма констант

Для создания формы констант нужно вызвать контекстное *меню* ветви **Константы** *дерева конфигурации* и выбрать в нем команду **Создать форму констант**. В появившемся окне **Конструктор общих форм**, Рисунок 1.16., нужно оставить тип формы в значении**Форма констант**, при необходимости заполнить другие поля и нажать на кнопку Далее.

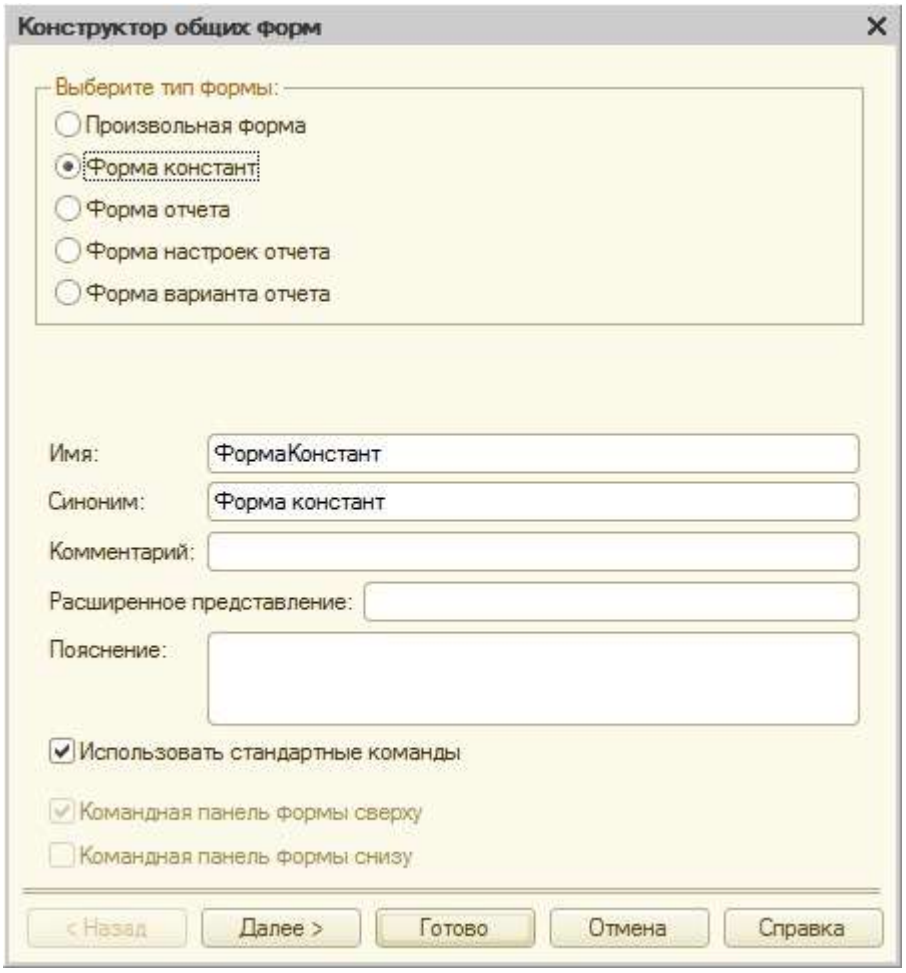


Рисунок 1.16. Конструктор общих форм

В появившемся окне, Рисунок 1.17. можно настроить состав формы констант, в нашем случае нас устраивает то, что в нее включены обе созданные в конфигурации *константы*, поэтому нажмем на кнопку **Готово**.

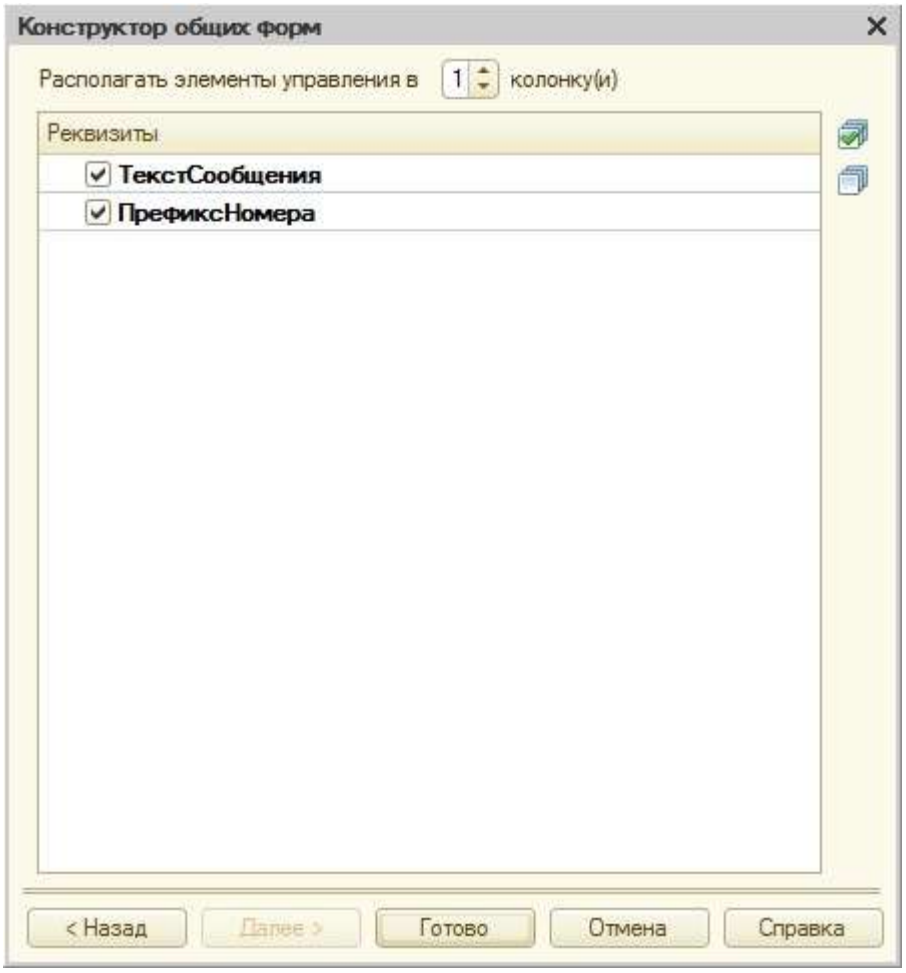


Рисунок 1.17. Конструктор общих форм, состав формы констант

В ветви **Общие формы** появится новая форма с именем **ФормаКонстант**, будет открыто окно редактирования формы, Рисунок 1.18.

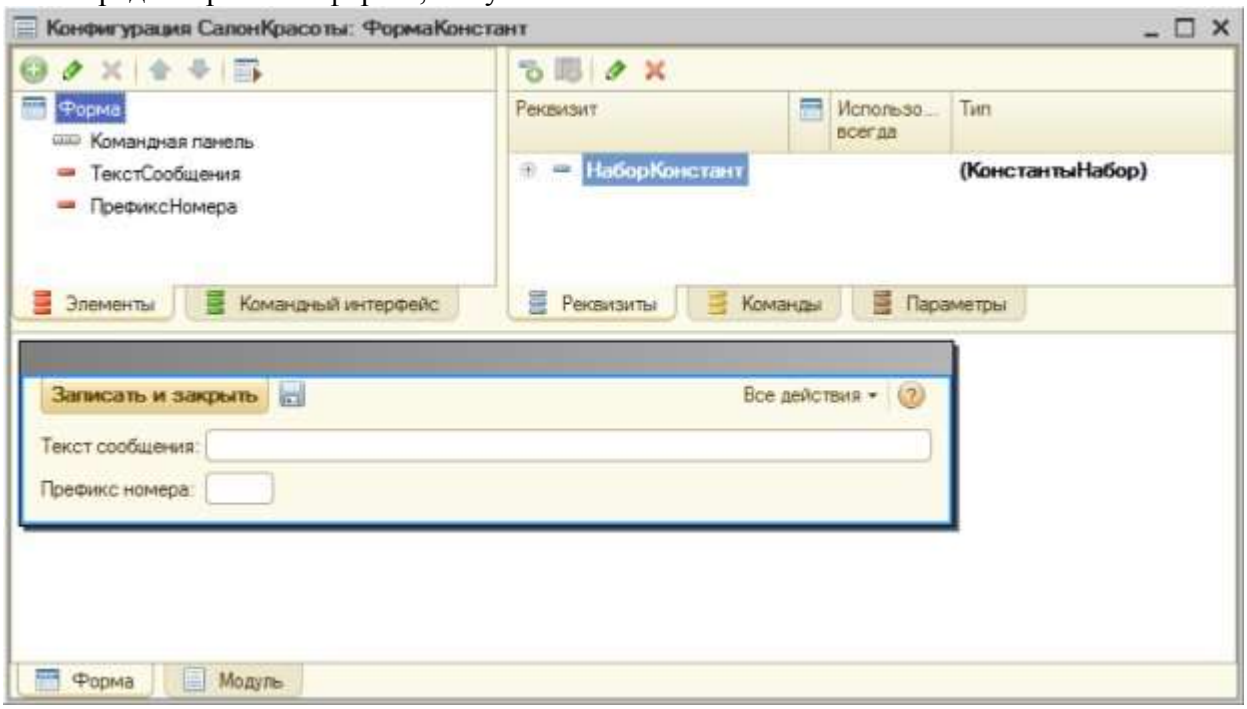


Рисунок 1.18. Окно редактирования формы

При разработке командного интерфейса в «1С:Предприятие» 8.2. используется концепция **декларативного описания**. Это означает, что разработчик задает схематичное описание элементов интерфейса, их группировку, свойства. При построении интерфейса для конкретного пользователя система создает его, опираясь на описание, сделанное разработчиком, учитывая различные дополнительные факторы, например, такие как *права* пользователя и пользовательские настройки интерфейса.

Форму констант также нужно включить в одну из подсистем. Включим ее в подсистему **Администрирование**, посмотрим, что у нас получилось, Рисунок 1.19.

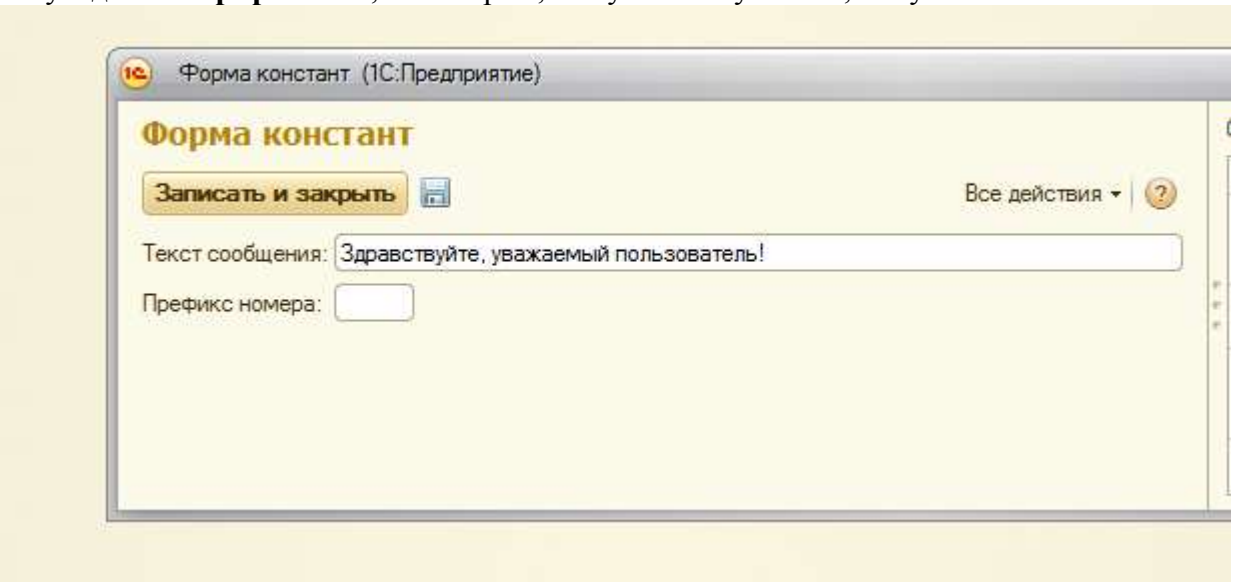


Рисунок 1.19. Окно редактирования формы

Мы видим, что форма констант доступна в группе **Сервис** панели действий раздела **Администрирование**. В текущей ситуации наличие в той же группе команды вызова окна *константы* **Префикс номера** может показаться избыточным. Для того чтобы убрать эту команду из панели действий, нам понадобится отредактировать командный *интерфейс*. Для этого мы можем выполнить команду **Открыть командный интерфейс** подсистемы **Администрирование** и в появившемся окне, Рисунок 1.20, снять флаг**Видимость** для команды **Префикс номера** группы **Сервис** панели действий.

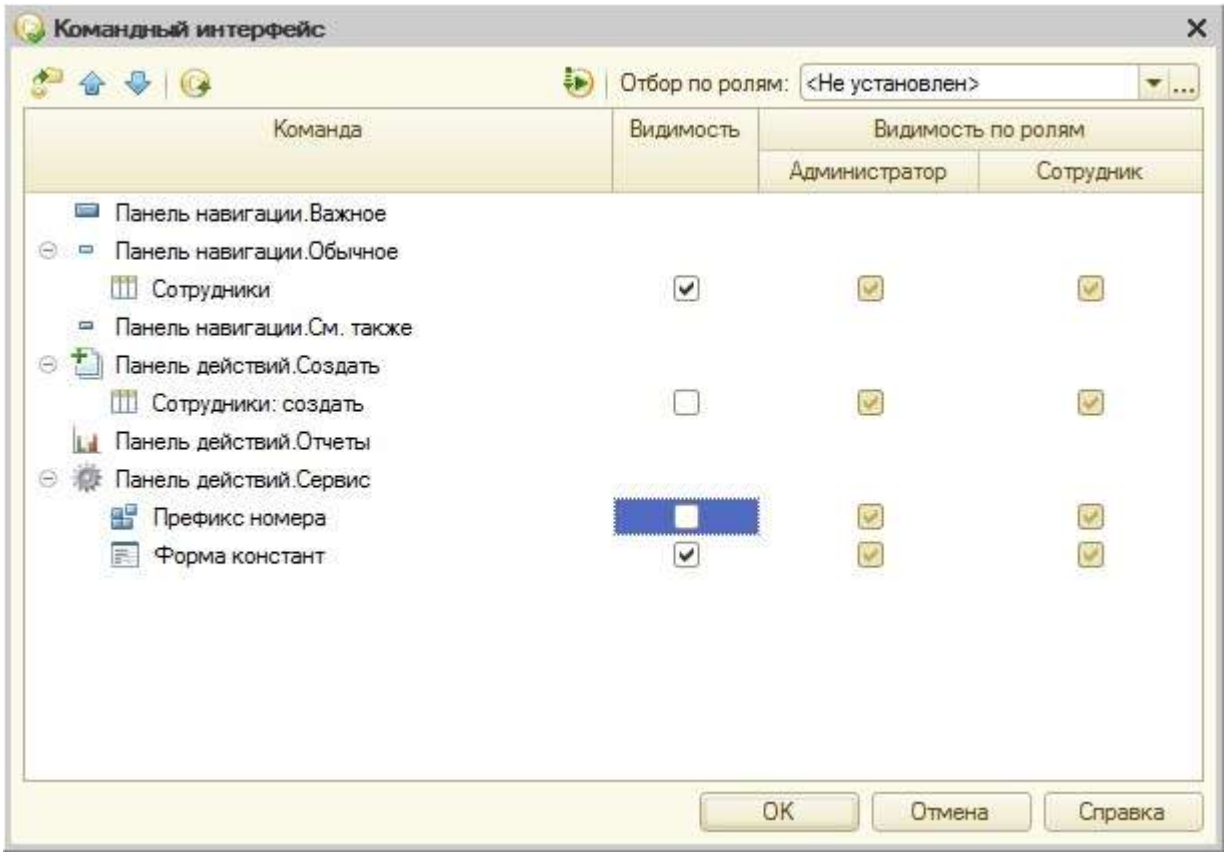


Рисунок 1.20. Настройка панели действий

Теперь при запуске в режиме «1С:Предприятие» ненужная *команда* отображаться не будет.

Выше мы создавали константу **Текст сообщения**, предполагая выводить заданный в ней текст в качестве сообщения для пользователей, входящих в систему. Реализуем эту функциональность. Для этого нам понадобится написать код в модуле управляемого приложения. Для того, чтобы открыть этот *модуль*, нужно воспользоваться командой **Конфигурация - Открыть модуль управляемого приложения** *корневого элемента* конфигурации. Для этого модуля предусмотрено несколько стандартных обработчиков событий, которые можно найти в панели инструментов **Модуль,** Рисунок 1.21. Нас интересует обработчик**ПриНачалеРаботыСистемы**.

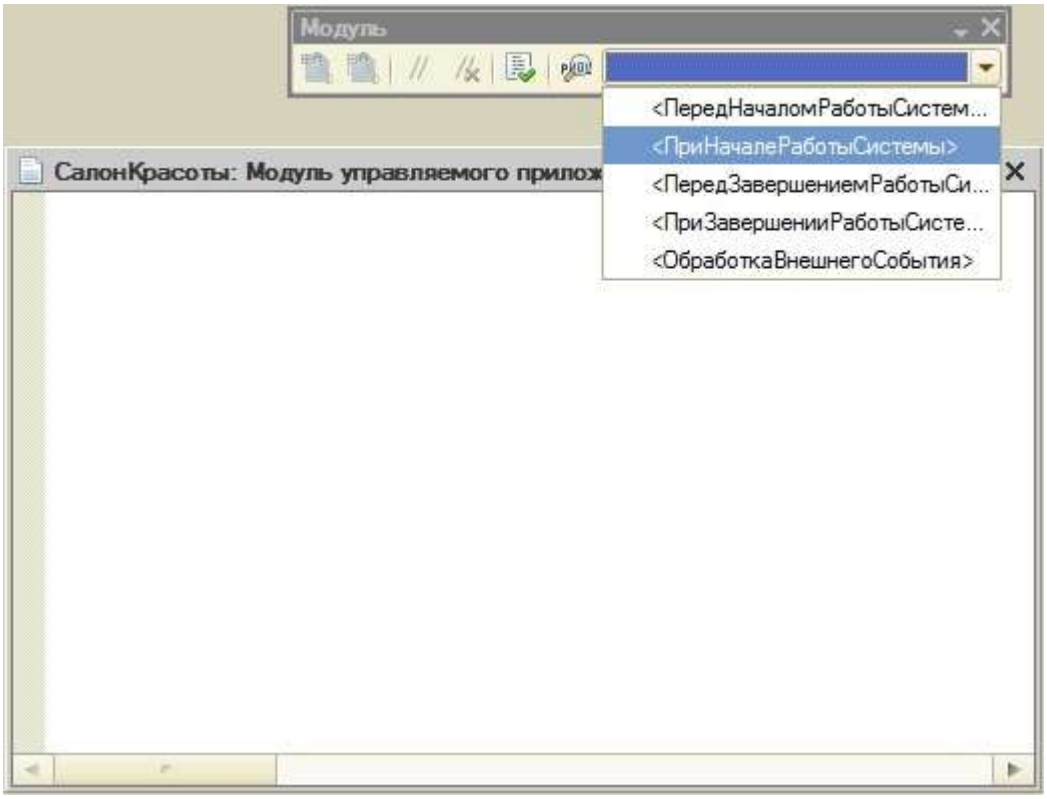


Рисунок 1.21. Выбор обработчика ПриНачалеРаботыСистемы

В модуле появится пустое тело обработчика, в которое нам нужно ввести команду для вывода сообщения пользователям. Если попытаться обратиться к константе напрямую из модуля управляемого приложения – мы столкнемся с ошибкой. Дело в том, что*исполнение* модуля управляемого приложения происходит на клиенте, в контексте которого нет доступа к константам. Поэтому нам понадобится код, который выполняется на сервере и возвращает *значение константы*. В данном примере мы можем возложить на серверную часть примера и *вывод* сообщения (*функция* **Сообщить**, которой можно здесь воспользоваться, работает и на клиенте, и на сервере). Но гораздо полезнее, в плане перспектив повторного использования кода серверной процедуры, "вытащить" серверные данные в метод, который выполняется на клиенте.

Создадим новый общий *модуль* (в ветви **Общие модули** *дерева конфигурации*), назовем его **СерверныеФункции**. Проследим за тем, чтобы в его свойствах были установлены флаги **Сервер** и **Вызов сервера**, Рисунок 1.22.

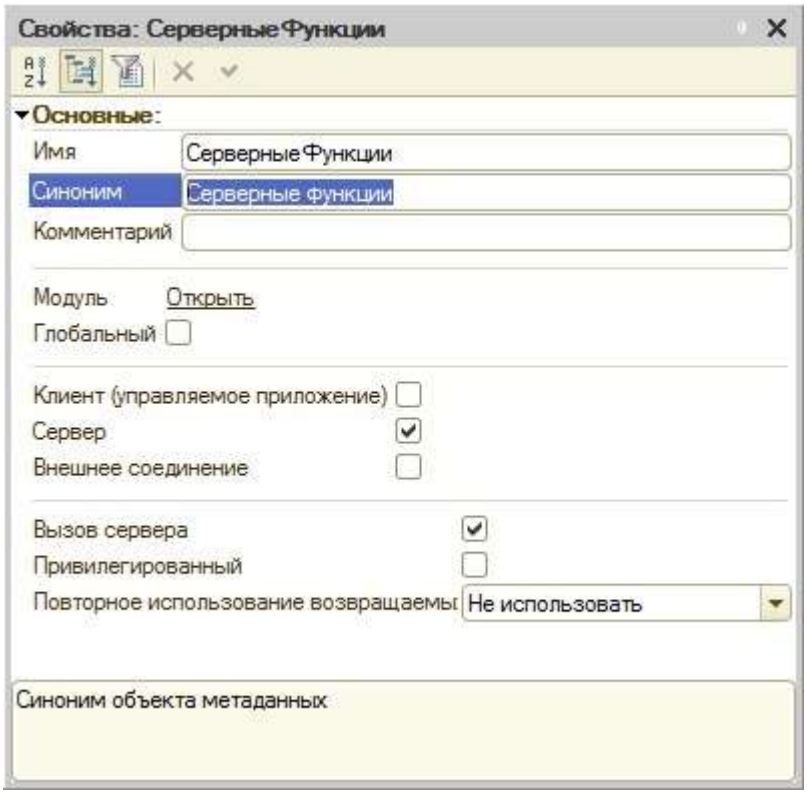


Рисунок 1.22. Общий модуль СерверныеФункции, свойства

Откроем *редактор кода* для кода модуля (например, двойным щелчком по модулю в *дереве конфигурации*) и введем следующий код, Рисунок 1.23.:

```
//Экспортная функция для вызова из других модулей
Функция ПолучитьКонстанту() Экспорт
   //Возвращаем полученное значение константы
   Возврат(Константы.ТекстСообщения.Получить());
КонецФункции
```

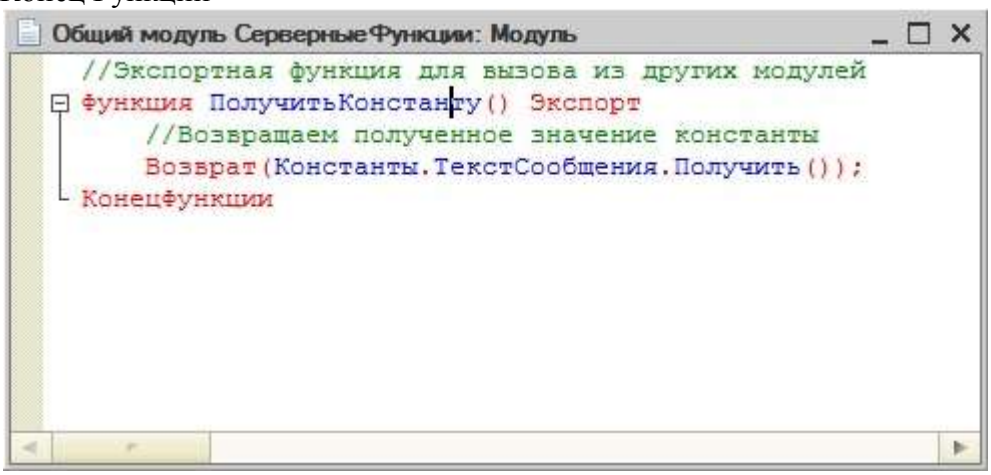


Рисунок 1.23. Общий модуль СерверныеФункции, код

Теперь нам нужно вызвать эту функцию в подходящем месте кода обработчика события **ПриНачалеРаботыСистемы** в модуле управляемого приложения. Например, это можно сделать так:

```
Процедура ПриНачалеРаботыСистемы()
```

```
    //Выводим сообщение пользователю
    Сообщить(СерверныеФункции.ПолучитьКонстанту());
КонецПроцедуры
```

В результате при входе в систему мы получим сообщение следующего вида, Рисунок 1.24.

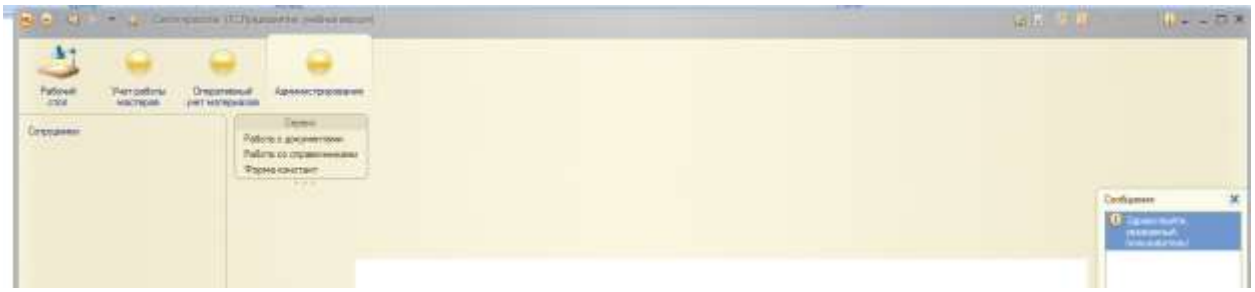


Рисунок 1.24. Вывод сообщения пользователю

Обратите внимание на то, что сообщение выводится в область **Сообщения** основного рабочего окна. Если сообщение вызвано из модуля какого-либо отдельного окна, например, из модуля формы констант, которая создана ранее, то, по умолчанию, сообщение будет выведено в этом окне.

Посмотрим на этот механизм в действии. Откроем окно редактирования формы констант (**Общие формы > ФормаКонстант**), перейдем на вкладку **Модуль**, на панели инструментов **Модуль** выберем стандартный обработчик события **ПриОткрытии**, отредактируем тело обработчика, чтобы оно приняло следующий вид, Рисунок 1.25:

```
&НаКлиенте
Процедура ПриОткрытии(Отказ)
    Сообщить("Вы открыли форму констант!");
КонецПроцедуры
```

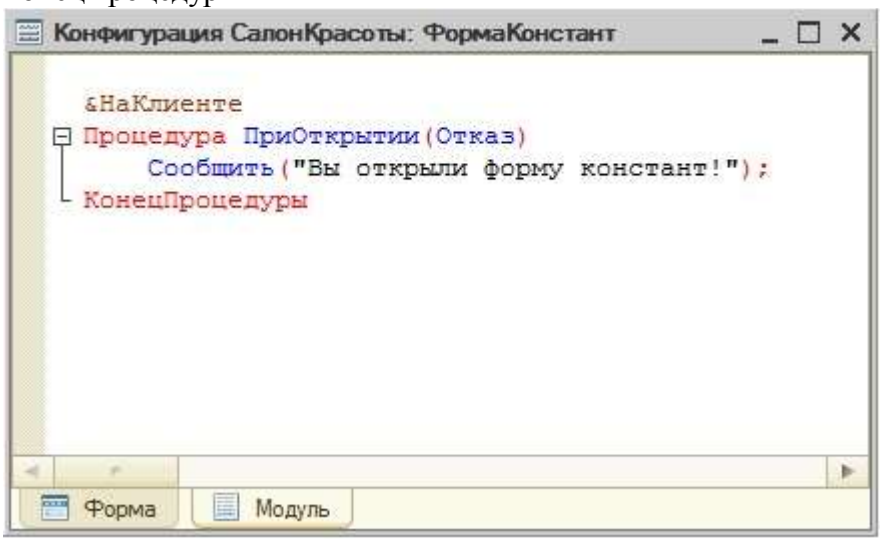


Рисунок 1.25. Вывод сообщения пользователю из модуля формы констант

Благодаря этому коду при открытии формы констант будет появляться следующее сообщение, Рисунок 1.26.

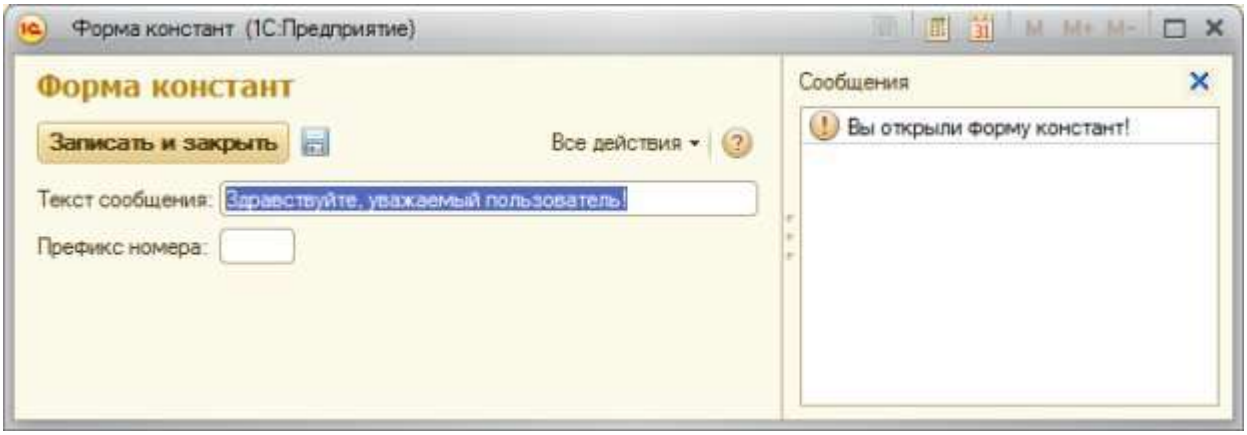


Рисунок 1.26. Вывод сообщения в форму констант

Основы клиент-серверного программирования

Обратите внимание на директиву компиляции **&НаКлиенте**, которая, в вышеописанном участке кода, автоматически размещена перед описанием процедуры **ПриОткрытии()**. Создавая решение для «1С:Предприятие» 8.2 разработчик должен четко разграничивать код, исполняемый в клиентской и серверной частях приложения. Причем, на клиенте (в контексте клиента) и на сервере (в контексте сервера) доступны различные объекты, различные программные *механизмы*. Основная задача серверного кода заключается во взаимодействии с базой данных, клиентский код занимается отображением этих данных и взаимодействием с пользователем. А задача разработчика заключается в том, чтобы создать код с учетом клиент-серверного взаимодействия.

Если перед описанием процедуры, функции или переменной в модуле формы отсутствует *директива* компиляции, по умолчанию считается, что код будет исполняться на сервере. В явном виде это задается указанием директивы **&НаСервере.**

Попытаемся в нашем модуле формы вывести в окно сообщения *значение константы*. Для этого мы можем добавить в *модуль*функцию, возвращающую *значение константы*, которая должна выполняться в контексте сервера. Например, это можно сделать одним из следующих способов – ниже приведена дополненная процедура ПриОткрытии и еще пара процедур, заданных в коде модуля формы:

```
&НаКлиенте
Процедура ПриОткрытии(Отказ)
   Сообщить("Вы открыли форму констант!");
   Сообщить(ПолучитьКонстанту()+" - из функции модуля формы без директивы");
   Сообщить(СерверныеФункции.ПолучитьКонстанту()+" - из общего модуля");
   Сообщить(ПолучитьКонстантуНаСервере()+" - из функции модуля формы с директивой &НаСервере");
 КонецПроцедуры
//По умолчанию функция считается серверной
 Функция ПолучитьКонстанту()
   //Возвращаем полученное значение константы
   Возврат(Константы.ТекстСообщения.Получить());
КонецФункции
//Директива компиляции задана явно
&НаСервере
 Функция ПолучитьКонстантуНаСервере()
```

```
    //Возвращаем полученное значение константы
    Возврат(Константы.ТекстСообщения.Получить());
КонецФункции
```

Здесь мы создали *пару функций* – одну назвали **ПолучитьКонстанту()**, при ее описании директиву компиляции мы не указывали. Вторую назвали **ПолучитьКонстантуНаСервере()** – при ее описании была указана *директива* **&НаСервере**. Мы вызвали эти функции для вывода сообщения в клиентской процедуре **ПриОткрытии()**. У нас уже есть серверная *функция* в общем модуле**СерверныеФункции** – здесь показан пример ее использования, в подобном случае, возникшем при реальной разработке, если действия, которые выполняются в серверной функции модуля формы, совпадают с действиями функции, описанной в общем модуле, можно и даже нужно пользоваться функцией общего модуля.

На Рисунок 1.27. вы можете видеть *вывод* сообщений, выполненный вышеприведенным кодом.

Рисунок 1.27. Вывод сообщения в форму констант, разные варианты работы с серверными данными

Вернемся к директивам компиляции. Они могут использоваться не только в модулях форм, но и в общих модулях, и в модулях команд. Таблица 1.1. содержит описание возможности применения директив компиляции в различных модулях

Таблица 1.1. Применение директив компиляции в модулях и переменных

Директивы компиляции	**Модули**			**Переменные**
	Модуль формы	**Модуль команды**	**Общий модуль**	
&НаКлиенте	Да	Да	Да	Да
&НаСервере	Да	Да	Да	Да
&НаКлиентеНаСервере	Нет	Да	Нет	Нет
&НаСервереБезКонтекста	Да	Нет	Нет	Нет
&НаКлиентеНаСервереБезКонтекста	Да	Нет	Нет	Нет

Опишем основные особенности их применения:

&НаКлиенте – эта *директива* предназначена для клиентских процедур и функций. Из такого метода могут быть вызваны любые процедуры и функции. Кроме того, с данной директивой можно объявлять переменные – их называют клиентскими.

Такая*переменная* существует столько же, сколько существует клиентская часть формы. Из метода с данной директивой доступны другие клиентские переменные модуля формы.

&НаСервере – эта *директива* предназначена для серверных процедур и функций. Из такой процедуры могут быть вызваны серверные и клиент-серверные внеконтекстные методы, а так же методы неглобальных серверных общих модулей. Допустимо объявление переменных с данной директивой – такие переменные существуют во *время выполнения* вызова сервера. Из серверных методов доступны серверные переменные, объявленные в модуле формы.

&НаКлиентеНаСервере – предназначена для описания процедур и функций, выполняемых на клиенте и на сервере. Такие процедуры и функции могут вызывать клиентские и серверные процедуры общих модулей. Не подходит для объявления переменных.

&НаСервереБезКонтекста – такая процедура или *функция* исполняется на сервере вне контекста формы, в модуле которой она описана. Из нее можно вызывать лишь другие внеконтекстные процедуры или функции. Использование этой директивы позволяет сократить объем данных, передаваемых между сервером и клиентом. Не подходит для объявления переменных.

&НаКлиентеНаСервереБезКонтекста – такая процедура или *функция* может исполняться на клиенте и на сервере, без доступа к контексту формы. Не подходит для объявления переменных.

Помимо директив компиляции в модуле управляемой формы можно пользоваться инструкциями препроцессору. Инструкции препроцессору обрабатываются до того, как будут обработаны директивы компиляции.

Литература:

Официальный сайт интернет-униерситета «Интуит» (электронный ресурс), режим доступа http://www.intuit.ru/studies/courses/2318/618/lecture/17939.

Контрольные вопросы для самопроверки:

1. Как создать информацинную базу?
2. Как установить свойства информационной базы?
3. Чем отличается директива «на клиенте» от директивы «на сервере»?

Задание к лабораторной работе

Создать информационную базу в соответствии с порядком, отраженным в теоретической части

Методические указания и порядок выполнения работы

База создается в точном порядке, как это указано на рисунках и в теоретической части

Индивидуальное задание

не предусмотрено

Требования к отчету и защите

1. Результатом выполнения лабораторной работы является сформированный в программе файл, содержащий выполненные задания. В ЭИОС результаты работы не выкладываются.

2. Планируется защита работы, где студент комментирует порядок выполнения заданий, а также отвечает на вопросы, представленные выше.

ЛАБОРАТОРНАЯ РАБОТА №2

РАБОТА СО СПРАВОЧНИКАМИ

ОБЩИЕ СВЕДЕНИЯ

Цель: научиться информационную базу на платформе ««1С:Предприятие»»

Материалы, оборудование, программное обеспечение:
1. персональный компьютер (компьютерные классы ГУК)
2. программное обеспечение «1С:Предприятие»

Условия допуска к выполнению:
умение работать на ПК и знание техники безопасности

Критерии положительной оценки:
предоставление результатов работы в виде файла и прохождение защиты.

Планируемое время выполнения:
Аудиторное время выполнения (под руководством преподавателя): 4 ч.
Время самостоятельной подготовки: 2 ч.

Теоретическое введение

Обсудив основы работы с константами и основные положения клиент-серверного программирования, перейдем к работе со справочниками. Обычно разработка системы справочников относится к начальному этапу разработки любой конфигурации, так как на типах данных, заданных справочниками, основываются другие *механизмы* системы. Но прежде чем начать разговор, о, собственно, справочниках, давайте поговорим об общих реквизитах.

Общие реквизиты

Общие реквизиты являются новой возможностью, которая была добавлена в 14-й *релиз* платформы «1С:Предприятие» 8.2. Общие реквизиты можно использовать двумя способами. Первый из них заключается в использовании их, как, собственно, реквизитов, которые присутствуют у всех (или у достаточно большого количества) объектов конфигурации. Второй способ предусматривает использование общих реквизитов в механизме разделения данных в качестве разделителей данных. В базе, использующей механизм разделения данных, могут работать несколько пользователей, набор данных каждого из которых не пересекается с набором данных других пользователей, то есть – каждый из них считает, что база содержит лишь "его" данные и ничего больше.

Сейчас мы создадим общие реквизиты, которые планируется использовать именно как реквизиты для других объектов конфигурации. В частности, общий *реквизит* может быть "подключен" к следующим объектам. Правильным будет и утверждение о том, что эти объекты могут "входить в состав" общего реквизита, так как, собственно, от настройки состава общего реквизита зависит его появление в других объектах. Итак, речь идет о следующих объектах:

- *планы обмена*
- справочники
- документы
- *планы видов характеристик*
- *планы счетов*
- *планы видов расчета*

- *регистры сведений*
- *регистры накопления*
- регистры бухгалтерского учета
- *регистры расчета*
- бизнес-процессы
- задачи

В нашем учебном примере мы собираемся вести в базе данных учет по нескольким организациям. Для этого нам понадобится, чтобы все объекты конфигурации, для которых уместен данный *реквизит*, содержали бы *реквизит* **Организация**, который содержит ссылку на организацию. Например, каждый документ будет оформляться от лица определенной организации, каждый элемент справочника будет относиться к той или иной организации, и так далее. Для того, чтобы не усложнять наши примеры, мы не будем в дальнейших лекциях курса развивать тему многофирменного учета в одной базе данных. Однако, в любом случае, общие реквизиты позволяют снизить трудоемкость разработки.

Второй *реквизит*, который предназначен для документов, будет использоваться для ввода комментариев к документу.

Прежде чем продолжать работу над общими реквизитами, создадим следующие объекты конфигурации, не настраивая их дополнительных свойств – справочник с именем **Организация**, и документ с именем **ПоступлениеМатериалов**. Включим их в подсистему **ОперативныйУчетМатериалов.**

Создадим новый **общий реквизит** со следующими параметрами, Рисунок 2.1.:

Имя: Комментарий

Тип: Строка, *длина* 50

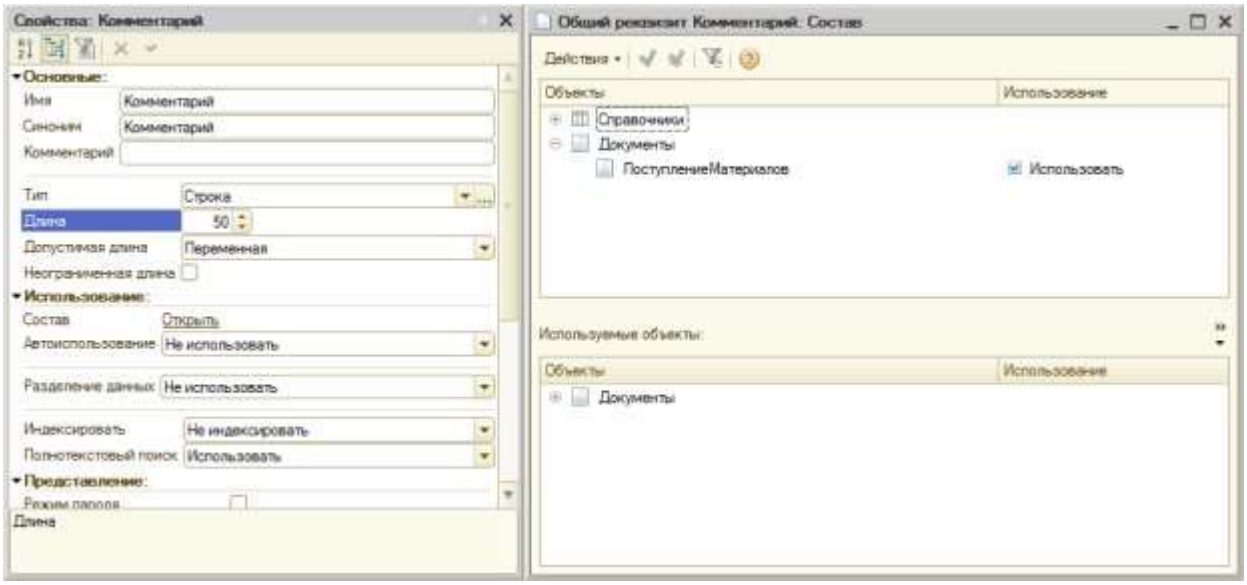


Рисунок 2.1. Настройка общего реквизита

Обратите внимание на *параметр* **Автоиспользование**. В данном случае мы оставляем его в значении по умолчанию – **Не использовать**. То есть – состав общего реквизита мы будем настраивать вручную. Этот общий *реквизит* мы планируем добавить ко всем документам, поэтому найдем свойство **Состав**, нажмем на ссылку **Открыть**, в появившемся окне выберем вариант **Использовать** для документа **ПоступлениеМатериалов**. При создании других документов мы сможем самостоятельно включать их в состав общего реквизита. Быстро проверить состав используемых объектов общего реквизита можно в нижней части окна настройки состава.

Преимущества использования общих реквизитов напоминают использование процедур и функций в общих модулях, к которым обращаются из многих других методов. Если возникает необходимость в изменении общего реквизита – например – в процессе работы в конфигурации оказалось, что длину комментария нужно увеличить – достаточно изменить параметры типа общего реквизита, и это изменение затронет все объекты конфигурации, включенные в его состав.

Создадим **второй** общий *реквизит*:

Имя: Организация

Тип: СправочникСсылка.Организации

Автоиспользование: Использовать

Этот *реквизит* мы планируем добавить ко всем объектам, допускающим использование общих реквизитов, за исключением справочника **Организации** и некоторых других. Перейдем в окно настройки состава общего реквизита и установим свойство **Использование** у справочника **Организации** в *значение* **Не использовать**, Рисунок 2.2.

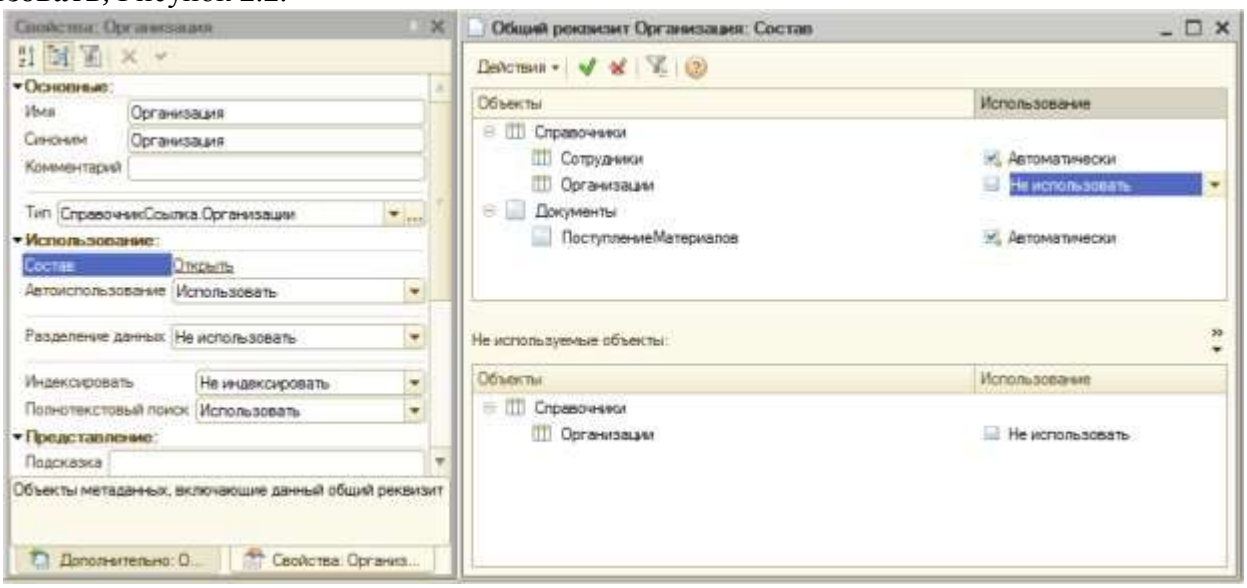


Рисунок 2.2. Настройка общего реквизита

Откроем нашу конфигурацию в режиме «1С:Предприятие» и посмотрим, как выглядит документ **ПоступлениеМатериалов** и справочники **Организации** и **Сотрудники.**

Для начала перейдем на вкладку **Оперативный учет материалов**. Обратите внимание на то, что в панель навигации раздела были автоматически добавлены ссылки для доступа к только что созданному справочнику **Организации** и к документу **Поступление материалов**. Щелкнем по ссылке **Организации**. В рабочей области окна появится *список* справочника. На данный момент он пуст, так как мы пока не заполняли справочник организациями, по которым будет вестись учет в базе. Щелкнем по кнопке **Создать**, которая расположена на командной панели списка – появится отдельное окно для заполнения свойств элемента справочника. Можно отметить, что помимо стандартных реквизитов (**Наименование**, **Код**) данный справочник не содержит ничего другого – это неудивительно, мы исключили его из состава общего реквизита **Организация.**

Теперь откроем *список* справочника **Сотрудники** и нажмем на кнопку **Добавить.** Общий *реквизит* **Организация** у данного справочника присутствует.

Откроем, наконец, окно создания документа **ПоступлениеМатериалов**. Здесь мы видим два общих реквизита – **Комментарий** и**Организация.**

Справочник «Организации»

Справочник можно сравнить с картотекой, с неким списком данных, каждая *запись* которого имеет определенную структуру. В организации – независимо от того, автоматизирован ли в ней учет или нет, присутствует множество таких списков. Это – списки сотрудников, клиентов, товаров.

В нашей конфигурации уже есть пара справочников. Один из них – это справочник **Организации**, который нужен для хранения списка организаций, *по* которым планируется вести учет. Справочник, сразу после его создания, имеет некоторые стандартные реквизиты. Это утверждение справедливо и для других объектов конфигурации. Для управления реквизитами объекта служит закладка **Данные** окна редактирования объекта, Рисунок 2.3.

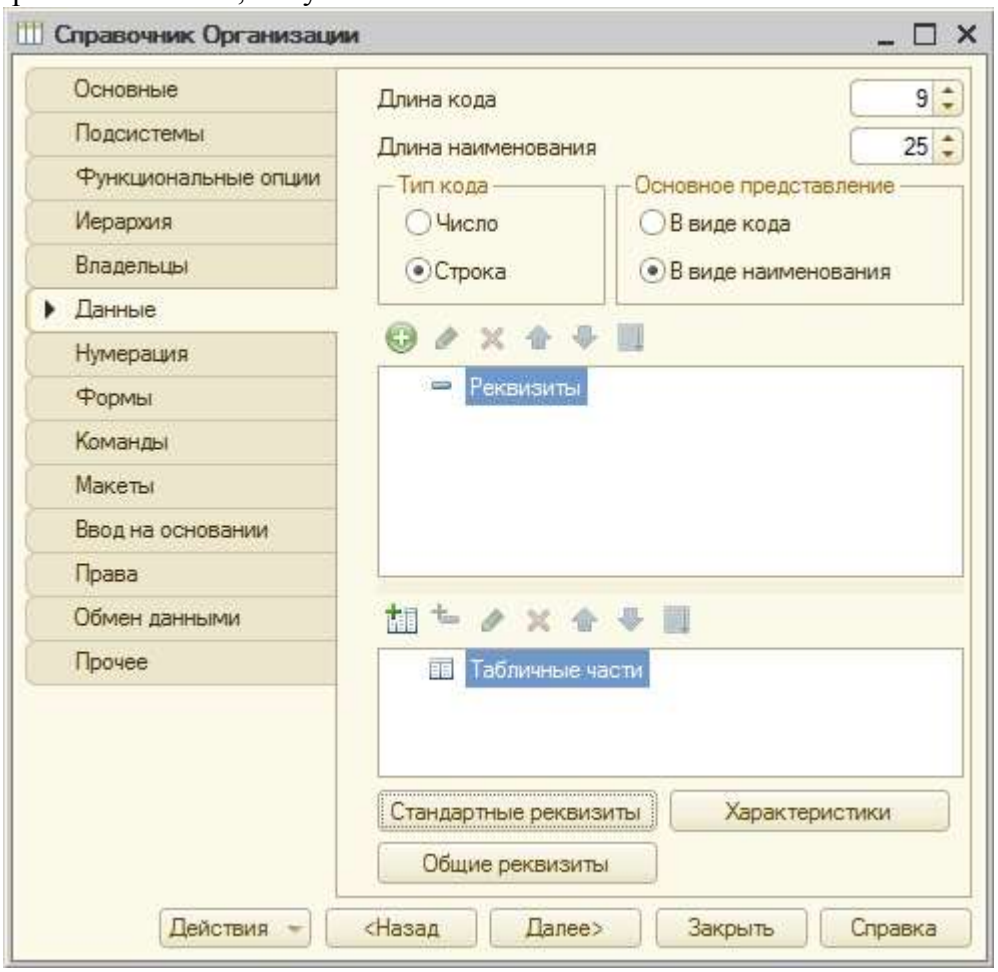


Рисунок 2.3. Настройка справочника

Ознакомиться со списком стандартных реквизитов можно, нажав на кнопку **Стандартные** реквизиты – появится окно, содержащее *список* таких реквизитов, Рисунок 2.4.

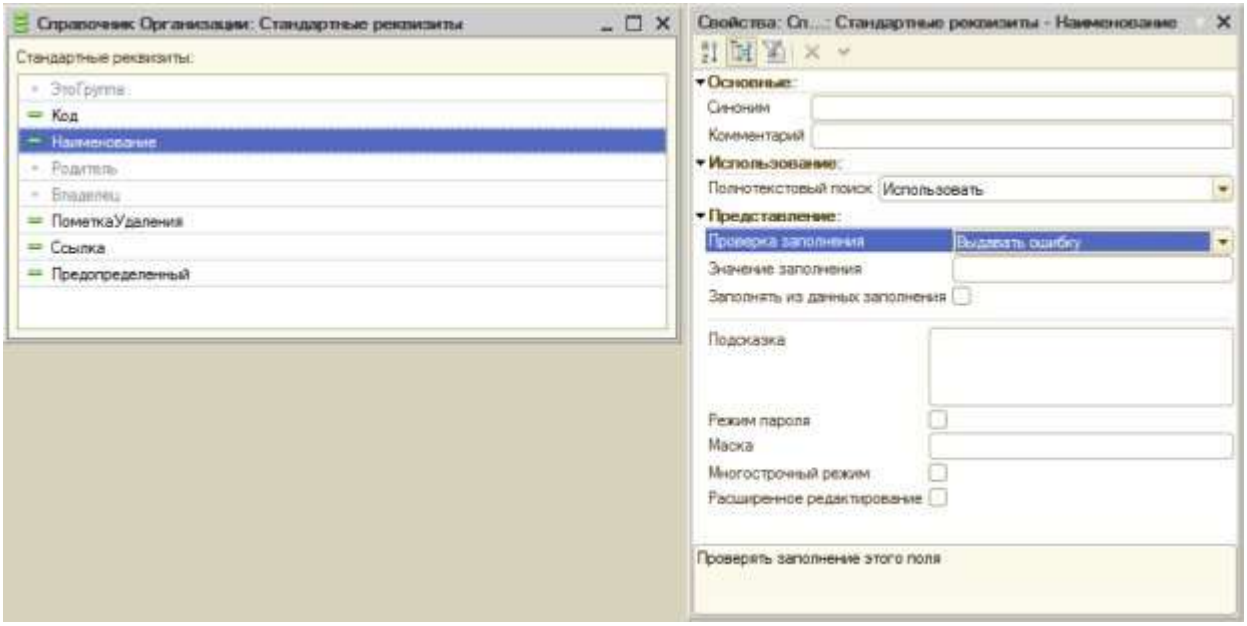


Рисунок 2.4. Стандартные реквизиты справочника и их свойства

Стандартные реквизиты поддерживают настройку некоторых свойств – для доступа к свойствам стандартного реквизита, достаточно выделить его в окне и обратиться к палитре **Свойства**.

Нашему справочнику **Организации** не хватает, для полноты его использования в системе, реквизита, который содержал бы полное наименование организации. Добавим этот *реквизит* к справочнику – на вкладке **Данные** окна редактирования объекта, нажмем на кнопку **Добавить**, параметры реквизита будут следующими:

Имя: ПолноеНаименование

Тип: Строка, *длина* – 50.

Проверка заполнения: Выдавать ошибку

Свойство **Проверка заполнения** *по* умолчанию для новых реквизитов установлено в *значение* **Не проверять**. Оно позволяет автоматически проверять заполненность поля – если *поле* не заполнено – система выдаст ошибку (Рисунок 2.5). Если нам нужны особые алгоритмы проверки содержимого поля перед записью элемента справочника, мы можем реализовать эти алгоритмы самостоятельно.

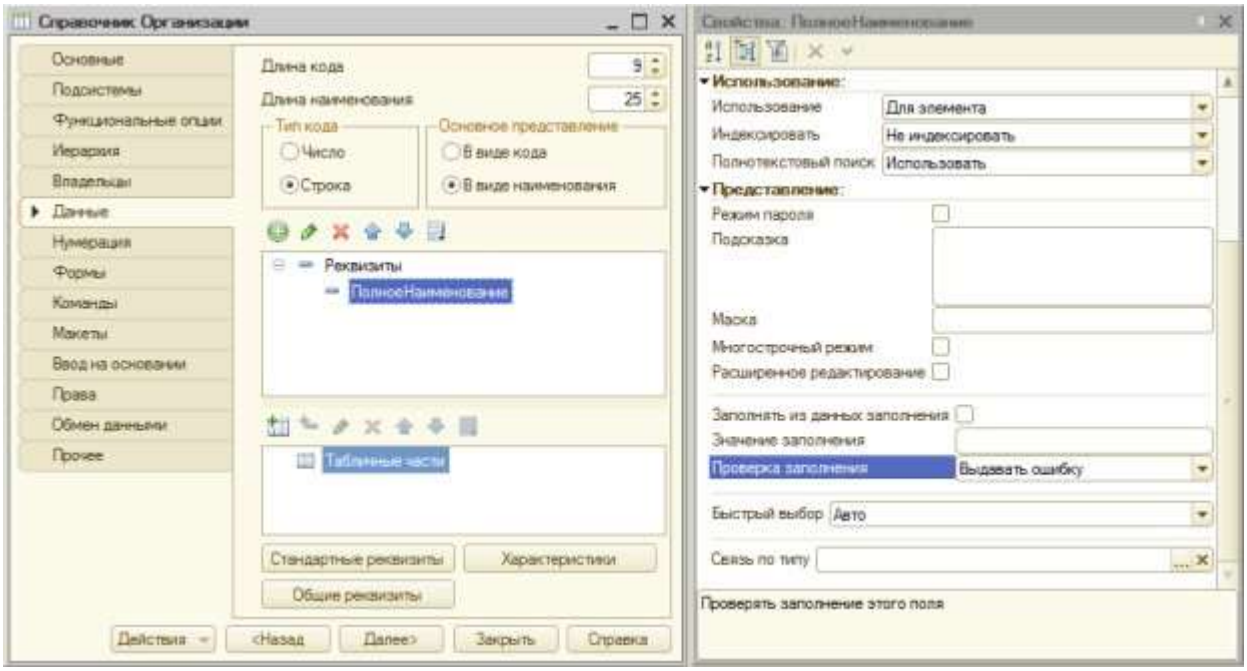


Рисунок 2.5. Настройка нового реквизита справочника

Посмотрим на наш справочник в режиме «1С:Предприятие». Создадим новый элемент, дадим ему наименование **Салон красоты**, а полное наименование заполнять не будем, и попытаемся записать элемент, нажав на кнопку **Записать и закрыть**. Элемент не будет записан, мы увидим *сообщение об ошибке* – в виде сообщения и в виде всплывающей подсказки, Рисунок 2.6.

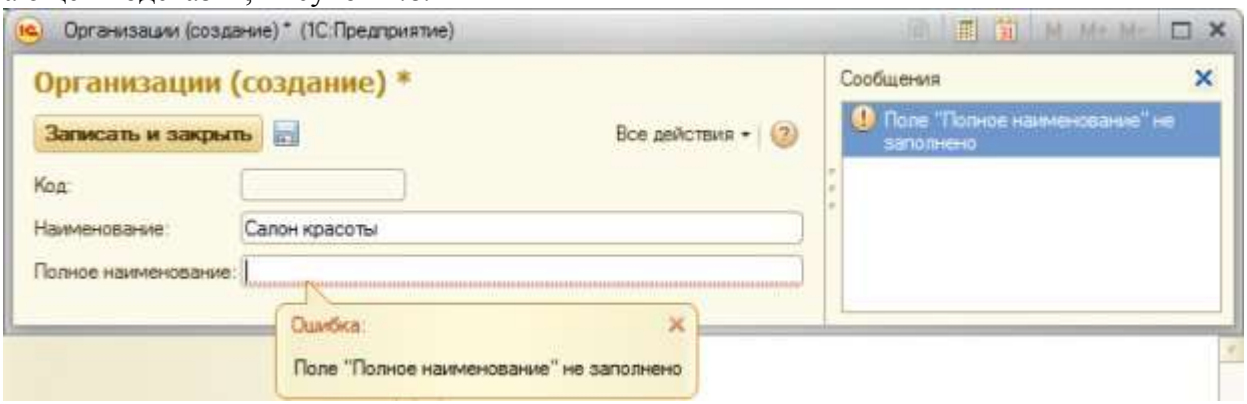


Рисунок 2.6. Сообщение об ошибке при попытке записи элемента справочника

Введем в *поле* **Полное наименование** текст *ООО* "Салон красоты" - после этого можно будет записать и закрыть элемент справочника. Он отобразится в списке справочника в рабочей области окна программы. В информационной панели, которая расположена в нижней части окна программы, появится *ссылка* для доступа к только что созданному элементу и будет сообщено о его создании.

Код элементу справочника будет присвоен автоматически.

Справочники в «1С:Предприятие» могут содержать предопределенные элементы. К их созданию можно перейти с вкладки **Прочее**, *по* кнопке **Предопределенные.**

Справочник ФизическиеЛица

Следующим нашим справочником будет справочник **ФизическиеЛица**. Он предназначен для хранения списка физических лиц и сведений о них. В частности, мы

хотели бы хранить данные о самом физическом лице (Фамилия, Имя, Отчество, дата рождения, пол, район проживания), а так же об истории его трудовой деятельности. Для хранения данных о физическом лице хорошо подойдут обычные реквизиты справочника, которыми мы уже занимались выше. А вот для того, чтобы хранить историю трудовой деятельности, нам понадобится другая *структура данных*, а именно – **табличная часть.**

Табличная часть – это *таблица*, состав и свойства полей (столбцов) которой мы задаем на этапе разработки. В пользовательском режиме создается необходимое количество строк. В нашем примере количество мест, в которых работало физическое лицо, заранее неизвестно.

Здесь надо отметить, что понятия "Сотрудник" и "Физическое лицо" - это разные вещи. Сотрудник – это тот, кто в настоящий момент работает в организации, и сотрудник обязательно является физическим лицом. А вот физическое лицо, сведения о котором могут храниться в базе данных организации, вполне может не являться сотрудником – например – это может быть кандидат на какую-либо должность, или, наоборот, уволенный сотрудник.

Создадим новый справочник, дадим ему имя **ФизическиеЛица**, включим его в состав подсистемы **УчетРаботыМастеров.**

На вкладке Данные создадим следующие реквизиты:

Имя: Фамилия
Тип: Строка, *длина* 30
Имя: Имя
Тип: Строка, *длина* 30
Имя: Отчество
Тип: Строка, *длина* 30
Имя: ДатаРождения
Тип: Дата, состав даты – Дата

Следующие реквизиты, которые мы планируем создать – это **Пол** и **РайонПроживания**. Строковые реквизиты, которые мы создавали выше, обычно заполняют вводом данных с клавиатуры. В случае же с указанием пола и района проживания заполнение с клавиатуры непременно приведет к появлению в базе различных наименований для одних и тех же показателей при использовании текстовых полей. Для мужского пола это вполне может быть, при ограничении длины строки одним символом, "М" и "м", для районов так же возможно различное написание. Для обеспечения единообразия при вводе подобных показателей рационально использовать для их хранения отдельные справочники или перечисления. Для хранения наименований пола мы воспользуемся перечислением.

Создадим новое *перечисление*, дадим ему имя **Пол**, включим в подсистему **УчетРаботыМастеров**. На вкладке **Данные** окна редактирования объекта для перечисления задаются значения перечисления. Зададим два значения – **Мужской** и **Женский**, Рисунок 2.7.

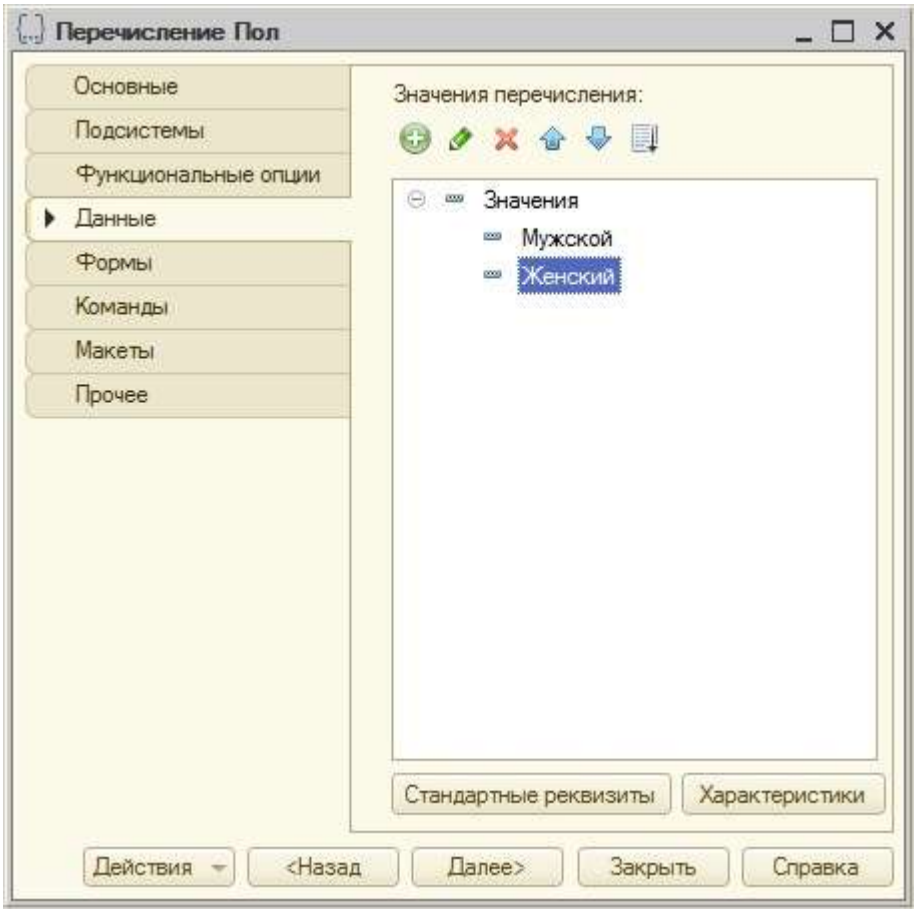


Рисунок 2.7. Создание перечисления **Пол**

Теперь создадим новый справочник – дадим ему имя **Районы**, включим в состав подсистемы **УчетРаботыМастеров**, на вкладке **Данные** изменим длину наименования до **100** символов, этот справочник не будет иметь дополнительных реквизитов, так же мы можем исключить его из состава общего реквизита **Организация**, Рисунок 2.8.

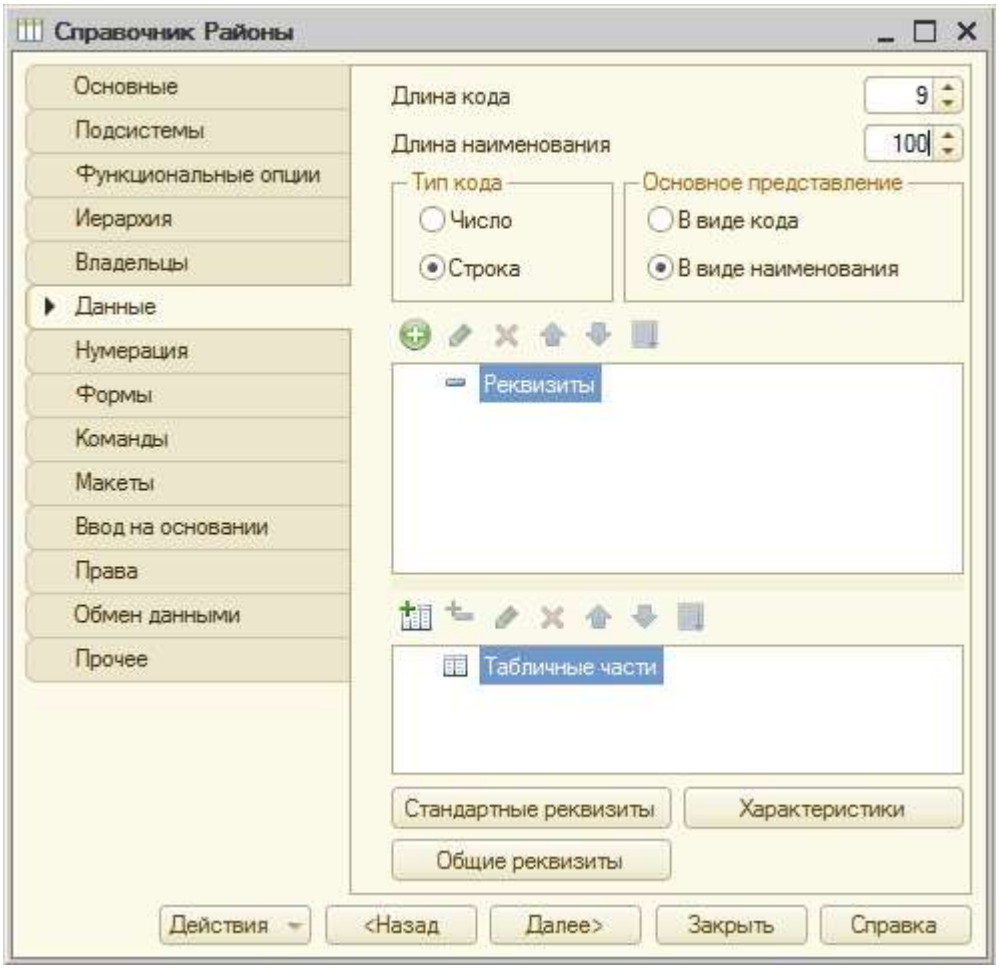


Рисунок 2.8. Создание справочника **Районы**

Вернемся к настройке справочника **ФизическиеЛица**. Добавим еще два реквизита:
Имя: Пол
Тип: ПеречислениеСсылка.Пол
Имя: РайонПроживания
Тип: СправочникСсылка.Районы
Теперь займемся табличной частью справочника. При необходимости, справочники могут иметь несколько табличных частей. Сначала нажмем на кнопку **Добавить табличную часть**, зададим имя табличной части **ТрудоваяИстория**. В табличную часть добавим следующие реквизиты (поля), выделив табличную часть и нажав на кнопку **Добавить реквизит:**
Имя: Организация
Тип: Строка, *длина* 30
Имя: ДатаНачалаРаботы
Тип: Дата, состав даты – Дата
Имя: ДатаОкончанияРаботы
Тип: Дата, состав даты – Дата.
В итоге окно редактирования нашего справочника будет выглядеть так, как показано на Рисунок 2.9.

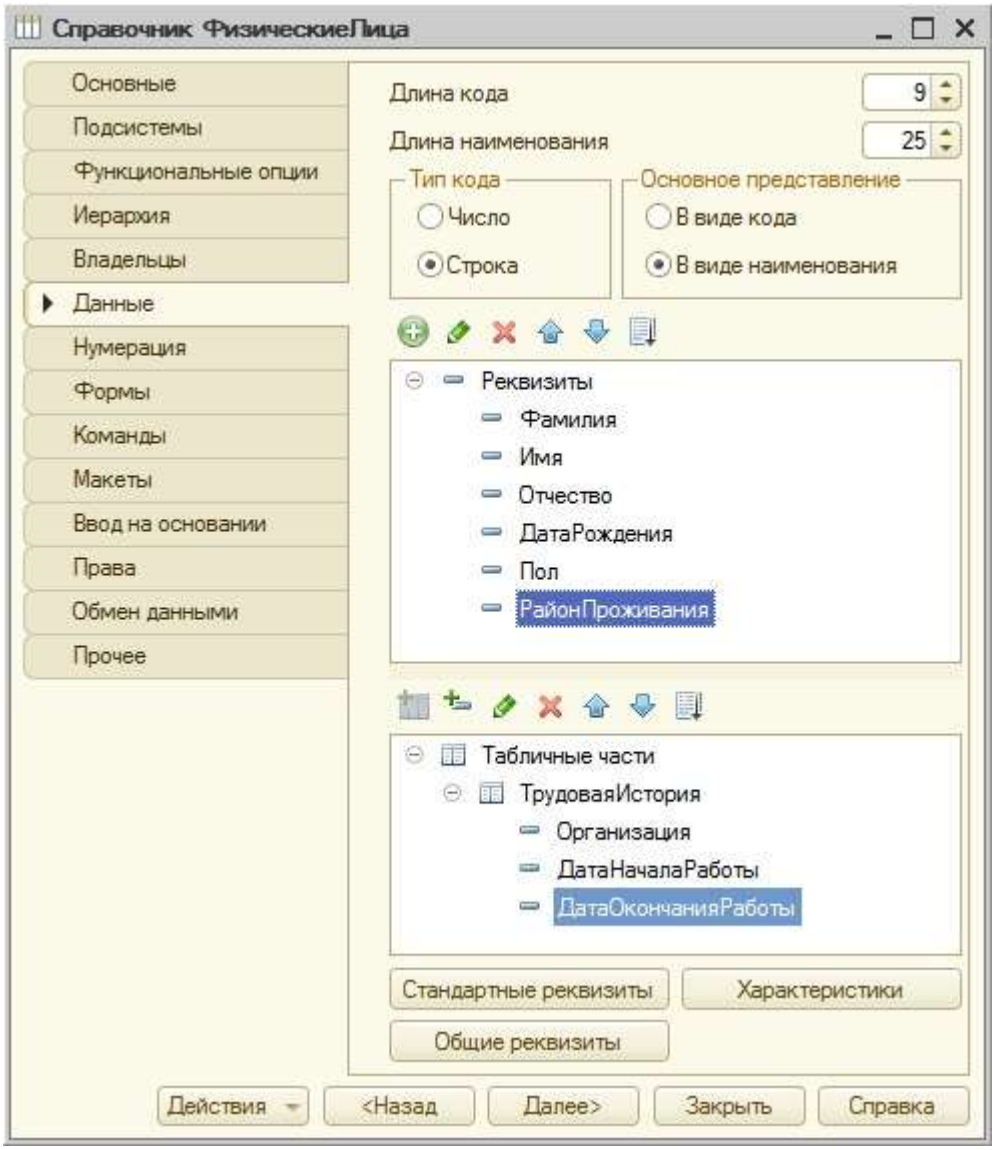


Рисунок 2.9. Состав справочника ФизическиеЛица

В предыдущей лекции мы создавали общий *реквизит* **Организация**, который планировалось добавлять ко многим объектам конфигурации. Справочник **ФизическиеЛица** имеет смысл вести *по* всем организациям. Как вы уже видели, настроить состав общего реквизита можно в ветви **Общие реквизиты**. Сделать это можно и в окне редактирования объекта, нажав кнопку **Общие реквизиты** на вкладке **Данные**. Нажмем эту вкладку и установим для общего реквизита **Организация** *значение* **Не использовать**, Рисунок 2.10.

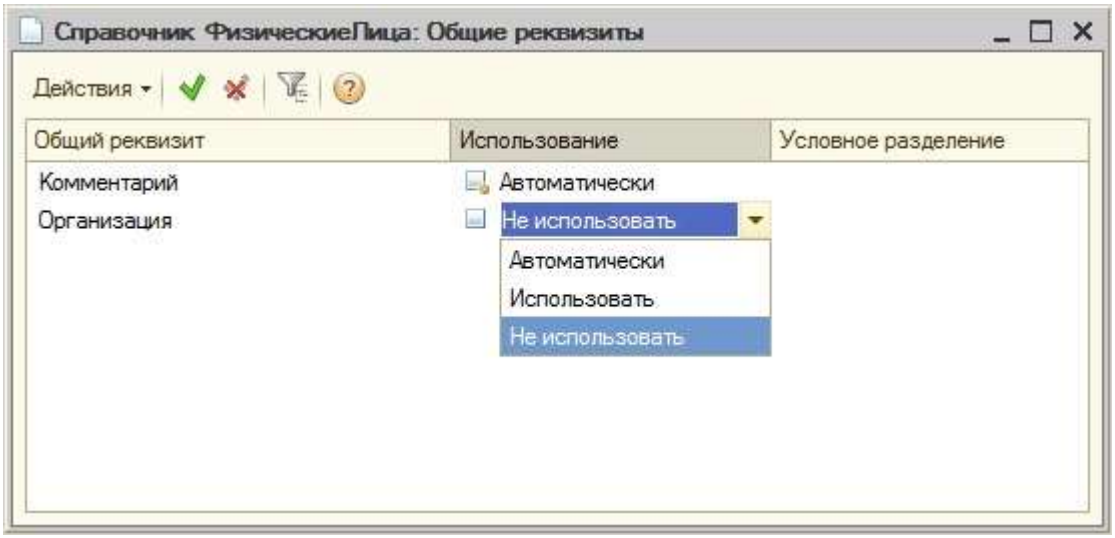


Рисунок 2.10. Настройка общих реквизитов из окна редактирования объекта

Если мы попытаемся открыть справочник в режиме «1С:Предприятие» – с ним можно будет работать, так как система автоматически сгенерирует его форму, Рисунок 2.11. – с автоматически созданными формами мы уже встречались ранее. Такие формы подходят в том случае, если мы не планируем каким-либо образом вмешиваться в функционирование формы из *Конфигуратора*.

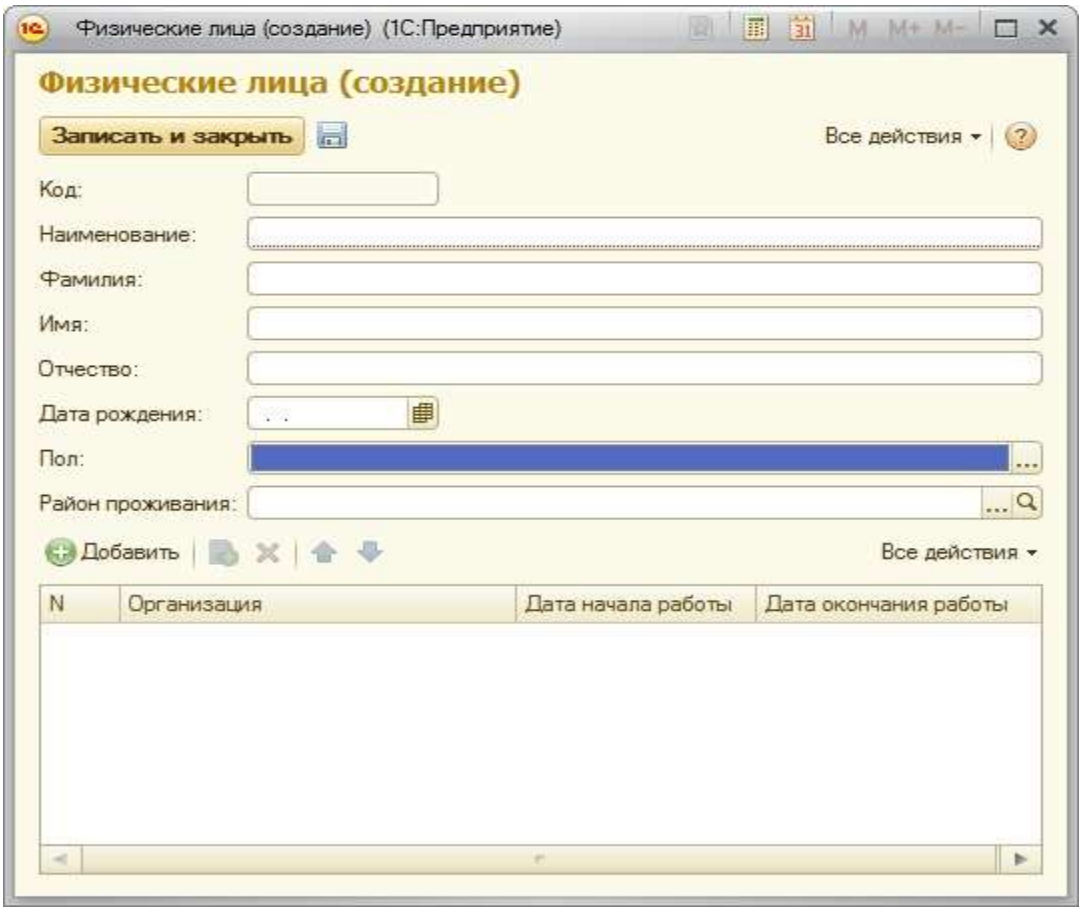


Рисунок 2.11. Форма справочника, сгенерированная автоматически

Если же решаемая нами задача требует каких-то особенных приемов работы с формой объекта, нам понадобится собственная форма. Например, это нам понадобится, если мы хотим автоматически заполнять *поле* **Наименование** на основе полей **Фамилия, Имя** и **Отчество**. А именно, мы хотели бы, чтобы наименование содержало фамилию и инициалы физического лица.

Разработка формы справочника ФизическиеЛица

Откроем закладку **Формы** окна редактирования справочника **ФизическиеЛица**. Можно отметить, Рисунок 2.12, что ни одной формы не задано – то есть все они создаются системой автоматически. Нам же нужна собственная форма **элемента** справочника.

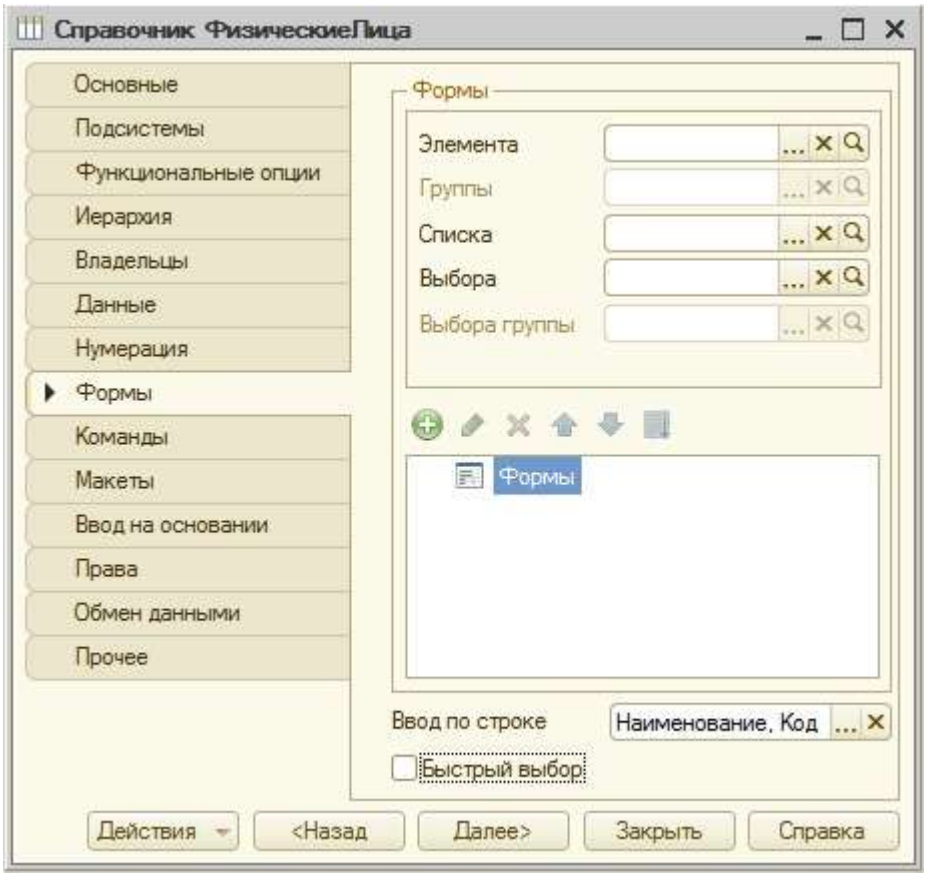


Рисунок 2.12. Вкладка **Формы** окна редактирования объекта

Нажмем на кнопку с увеличительным стеклом напротив поля **Элемента** в группе **Формы**. Появится окно **Конструктора формы справочника**, в его первом окне оставим все *по* умолчанию – а именно – нас интересует **Форма элемента справочника**, Рисунок 2.13.

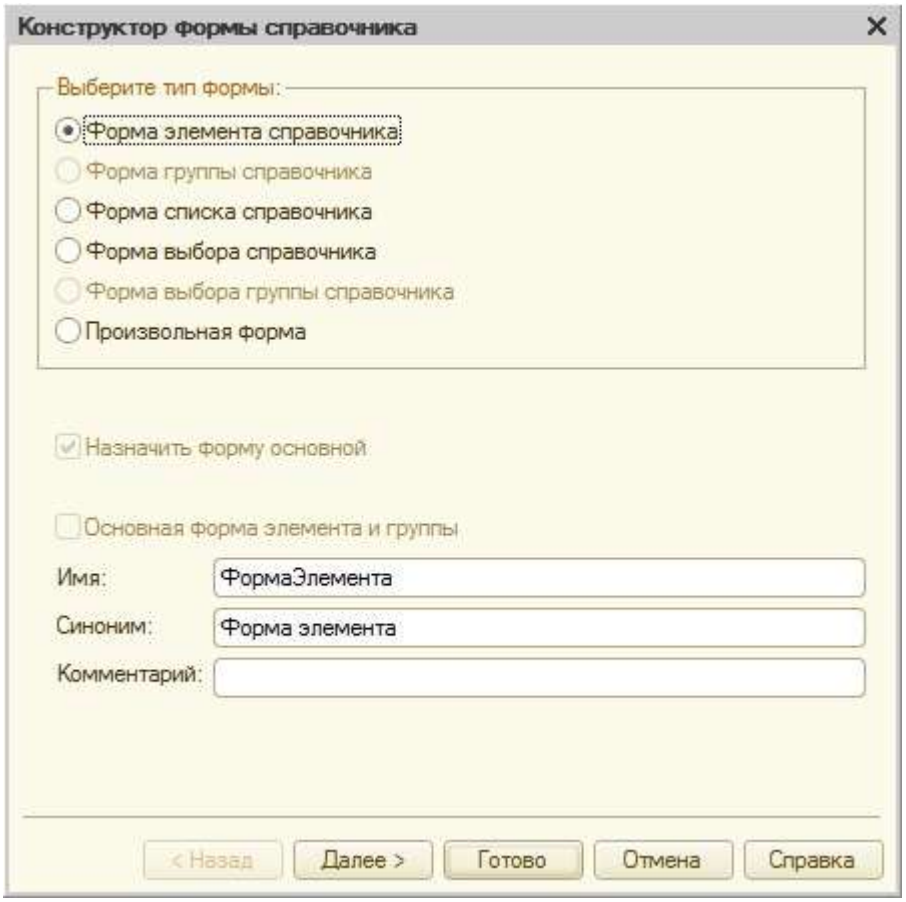


Рисунок 2.13. Первое окно конструктора форм справочника

В следующем окне, Рисунок 2.14., мы можем указать состав реквизитов для расположения на форме, а так же указать количество колонок, которое нужно для расположения элементов управления на форме. Оставим здесь все так же *по* умолчанию и нажмем на кнопку **Готово**.

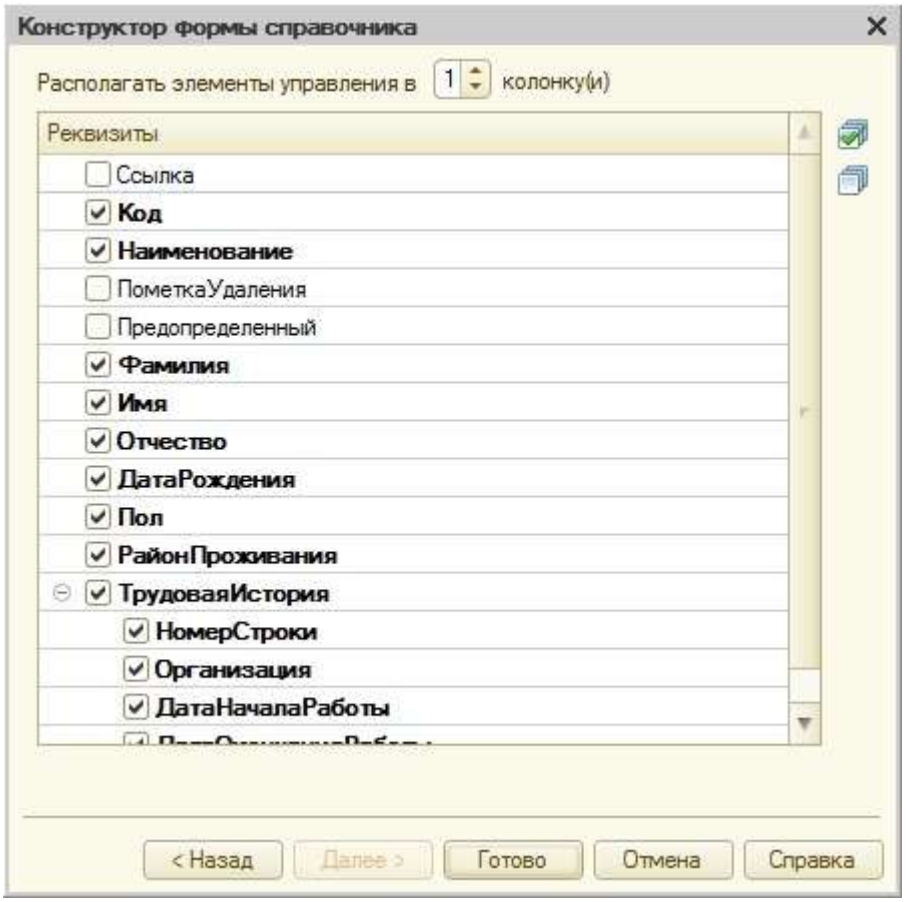


Рисунок 2.14. Второе окно конструктора форм справочника

После этого нужно открыть окно редактора форм для формы элемента справочника, Рисунок 2.15. Ранее мы уже сталкивались с этим окном, теперь рассмотрим его подробнее.

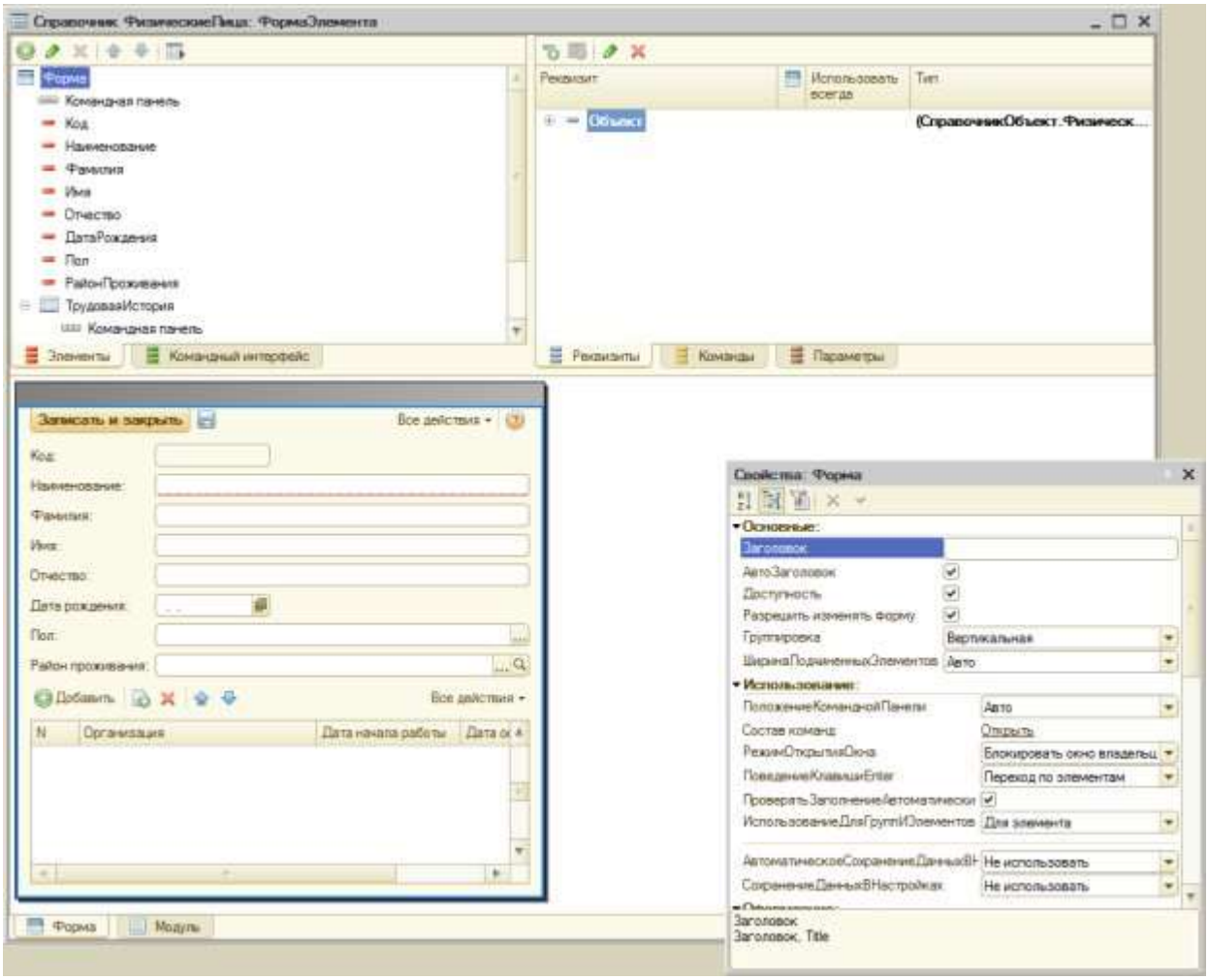


Рисунок 2.15. Окно редактирования формы элемента справочника

На самом деле, это окно объединяет в себе несколько редакторов и окон. В частности, это следующие:

Редактор элементов формы (закладка **Элементы** в верхней левой части окна) – с его помощью можно контролировать*элементы управления*, которые будут расположены на форме. Выделив элемент в данном окне, мы можем настраивать его свойства в стандартной палитре свойств. Обратите внимание на кнопку **Проверить**, находящуюся в правой части командной панели закладки **Элементы**. Нажатие на нее приводит к выводу конструируемой формы в интерактивном виде, что позволяет лучше оценить ее внешний вид в пользовательском режиме, но, конечно, не дает возможности работать с данными *информационной базы*.

Окно просмотра формы (закладка **Форма** в нижней части окна) – здесь представлена форма в том виде, который она примет после настроек. Кроме того, выделяя элементы формы в данном окне, мы, не имея возможности, как это было ранее, произвольно перемещать их, можем вызывать их контекстное *меню*, Рисунок 2.16, с помощью которого можно перемещать элемент вверх или вниз (то же самое можно делать в окне **Элементы**), открывать окно его свойств, назначать обработчики событий (их можно назначать и в окне **Свойства**, открытом для данного элемента).

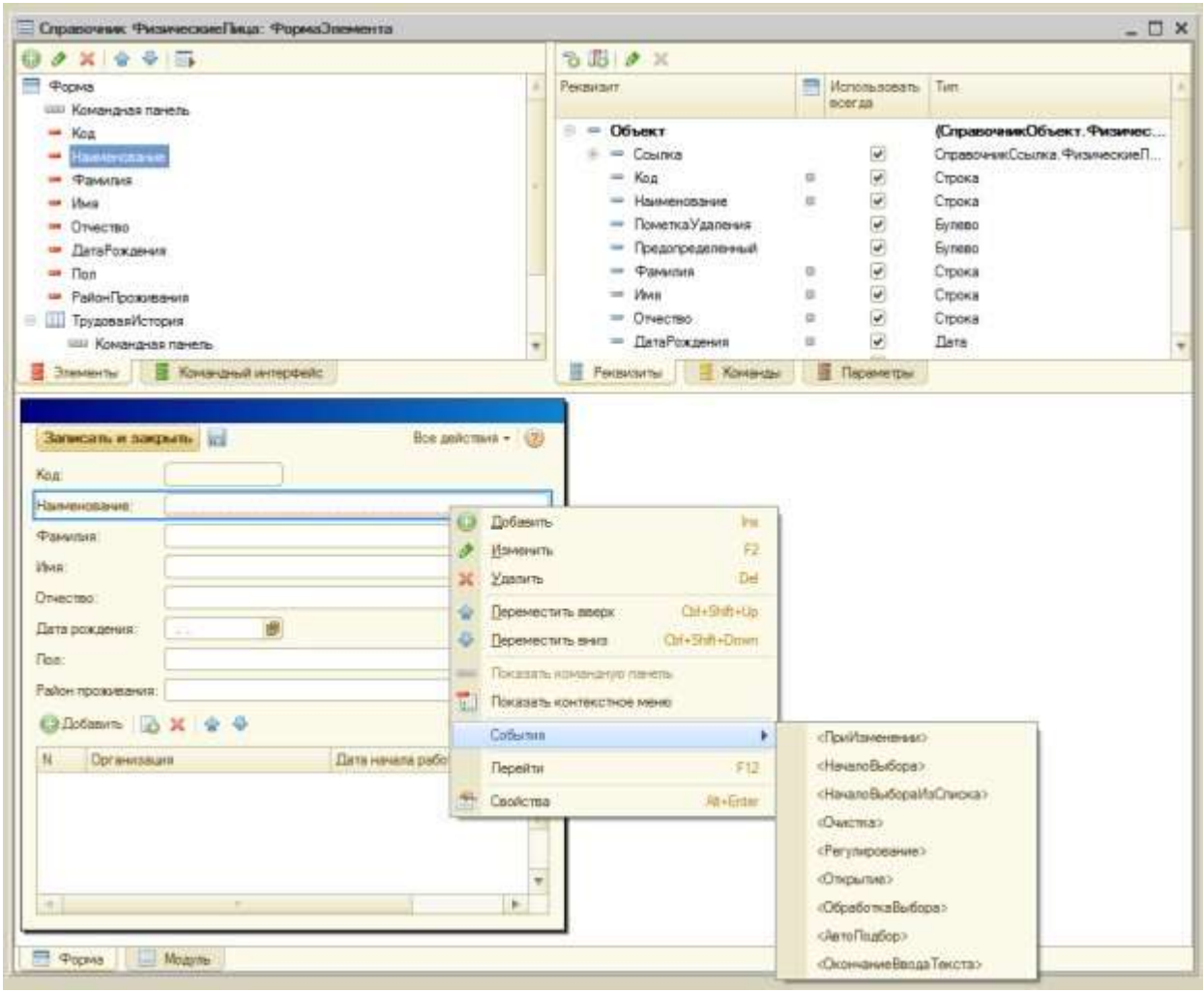


Рисунок 2.16. Работа с элементами формы

Редактор реквизитов представлен вкладкой **Реквизиты** (Рисунок 2.16.). Для того, чтобы добавить *реквизит* объекта на форму (то есть – создать элемент управления, связанный с данным реквизитом), достаточно перетащить элемент из окна **Реквизиты** в окно**Элементы**. Реквизиты, уже присутствующие на форме, отмечены серым квадратиком.

Редактор команд можно открыть, нажав на вкладку **Команды**. Здесь доступны три дополнительные вкладки. Вкладка **Команды формы** (*по* умолчанию пустая) содержит команды формы, их можно сравнить с *командными кнопками*, которые в версии «1С:Предприятие» 8.1. можно было размещать на форме. Теперь последовательность действий выглядит так – сначала создать команду формы, потом перетащить ее в окно **Элементы**, настроить свойства, задать обработчики событий. Вкладка**Стандартные команды** (Рисунок 2.17) содержит стандартный набор команд – в нашем случае – стандартный для формы и табличного поля, размещенного на форме.

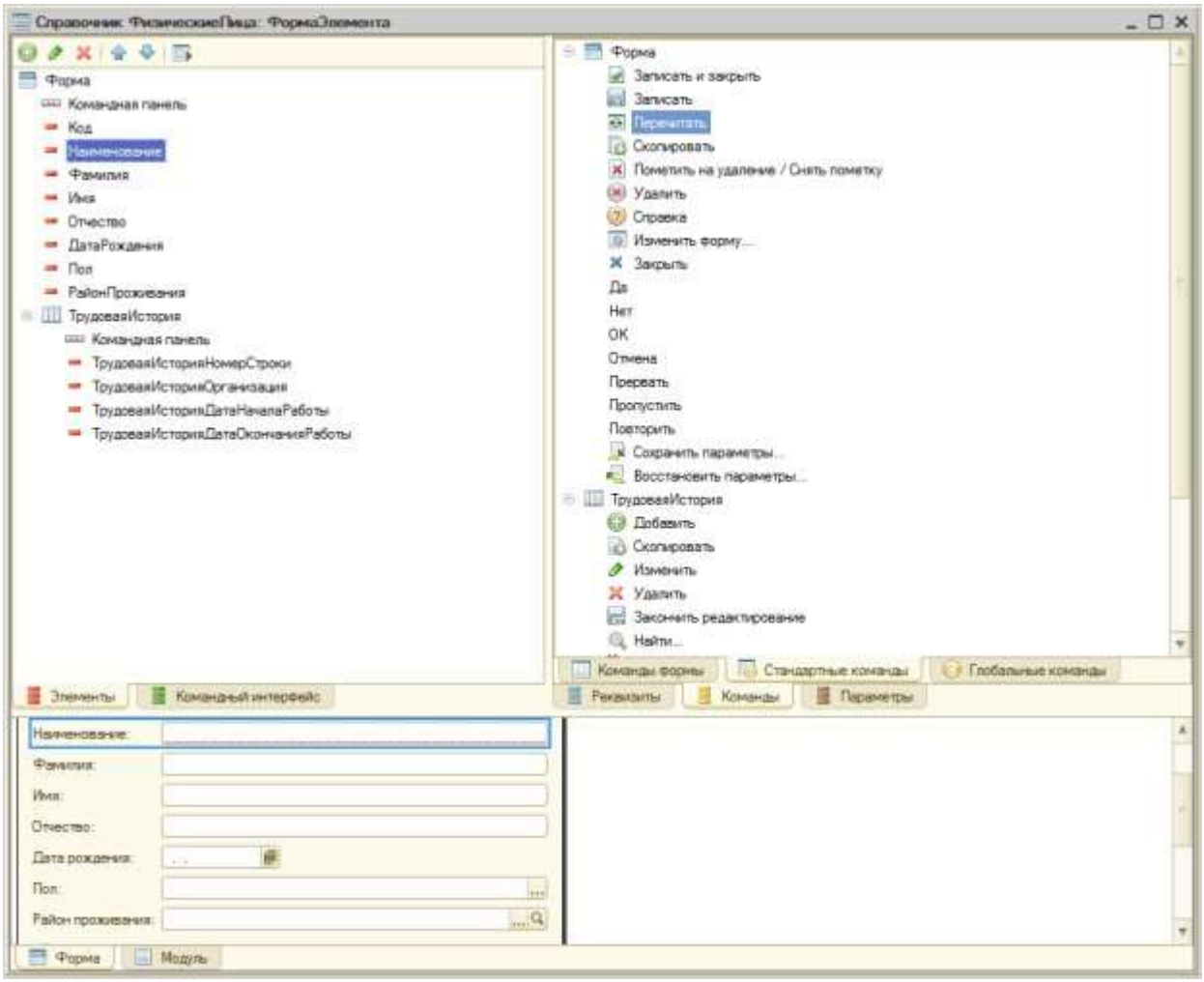


Рисунок 2.17. Стандартные команды

Вкладка **Глобальные команды** содержит набор команд уровня *прикладного решения*.

Вкладка **Параметры** предоставляет *доступ* к редактору параметров.

Вкладка **Командный интерфейс** позволяет редактировать командный *интерфейс*.

Реализуем автоматическое заполнение поля **Наименование** на основе полей **Фамилия**, **Имя** и **Отчество**.

Для этого сначала настроим элемент управления, отображающий наименование на форме, таким образом, чтобы его нельзя было редактировать. Выделим элемент управления в панели **Элементы**, откроем окно его свойств и установим свойство**ТолькоПросмотр**, Рисунок 2.18. Благодаря этому свойству *пользователь* не сможет отредактировать текст в *поле* ввода. Похожего эффекта можно достичь и другими способами, например, указав в свойстве **Вид элемента** элемента **Наименование** вместо **Поле ввода – Поле надписи**.

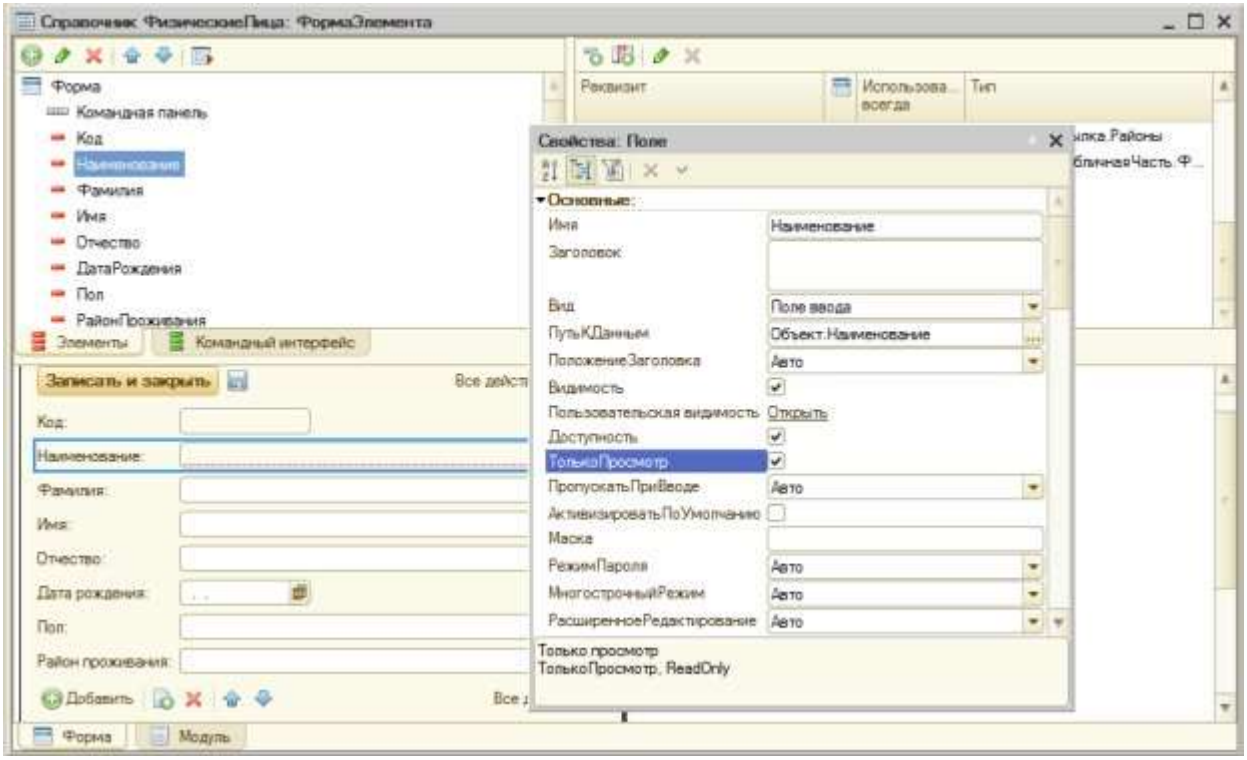


Рисунок 2.18. Настройка элемента **Наименование**

Для правильного формирования наименования важно, чтобы *пользователь* ввел данные в поля **Фамилия**, **Имя** и **Отчество**.

Клиентские методы в модуле формы

Теперь перейдем к написанию кода, в котором будем формировать наименование. Для этого нам нужно понимать, что конфигурации «1С:Предприятие» управляются событиями – и сейчас нас интересуют события формы.

Выделим форму в окне Элементы, откроем окно ее свойств и рассмотрим группу свойств **События**, Рисунок 2.19.. Наименование должно быть сформировано до того, как данные объекта будут записаны. Для достижения нашей цели нам вполне подойдет событие **ПередЗаписью**. Здесь же можно выполнить какие-либо пользовательские проверки полей перед формированием наименования. Хотя, если говорить о производительности решения, лучше подобные проверки производить на сервере, например, с помощью обработчика события **ОбработкаПроверкиЗаполнения**, который создается в модуле объекта.

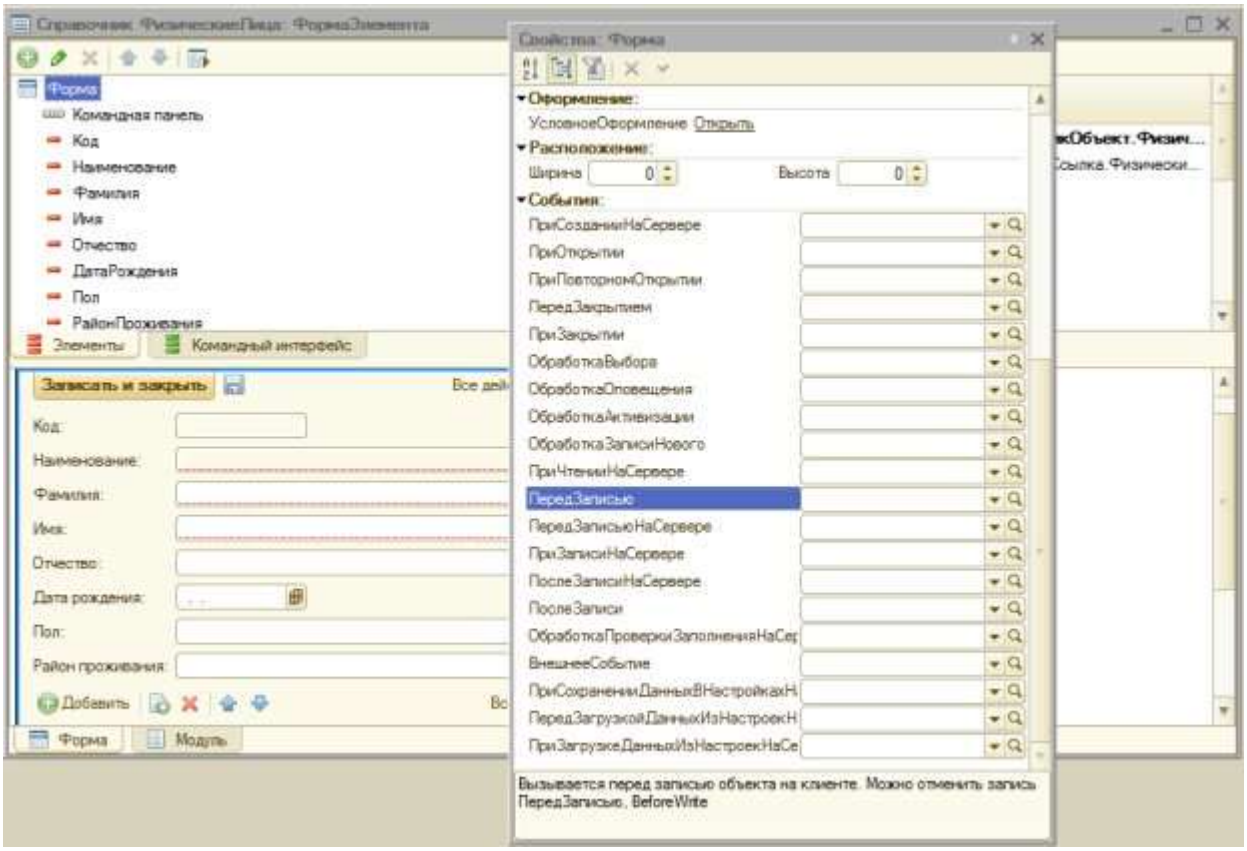


Рисунок 2.19. Выбор события для выполнения запланированных действий

Нажмем на кнопку с увеличительным стеклом в *поле* события **ПередЗаписью** – автоматически будет открыт *модуль* формы и создан пустой обработчик события **ПередЗаписью**. Он имеет следующий вид:

```
&НаКлиенте
Процедура ПередЗаписью(Отказ, ПараметрыЗаписи)
  // Вставить содержимое обработчика.
КонецПроцедуры>
```

Из директивы компиляции **&НаКлиенте** понятно, что процедура это клиентская, она имеет два параметра – нас сейчас интересует *параметр* **Отказ** – благодаря этому параметру, а именно, установив его в *значение* **Истина**, мы можем отказаться от записи объекта в том случае, если выполняется какое-либо условие, препятствующее записи. В нашем случае записи объекта могут воспрепятствовать незаполненные или неправильно заполненные поля **Фамилия, Имя** или **Отчество**. Проверку на незаполненность реквизита мы можем доверить и системе – для этого можно установить свойство **Проверка заполнения** для нужных реквизитов в *значение* **Выдавать ошибку,** делается это в списке реквизитов объекта в окне редактирования объекта или в *дереве конфигурации*, Рисунок 2.20. Не будем включать проверку заполнения, выполним ее и еще некоторые проверки самостоятельно.

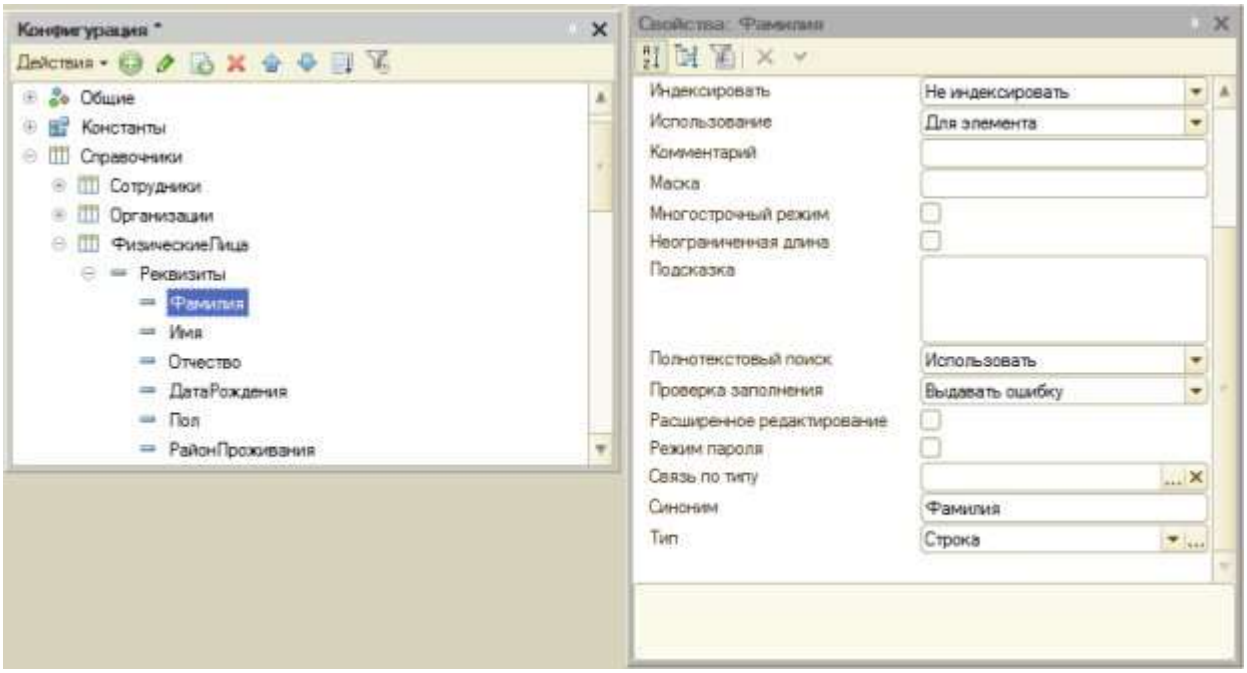


Рисунок 2.20. Настройка проверки заполнения

Параметры и процедуры в системе «1С:Предприятие» *по* умолчанию передаются *по* ссылке – передав в процедуру некую переменную, мы, на самом деле, передаем ссылку на нее, то есть – при модификации соответствующего этой переменной параметра внутри процедуры, фактически, происходит и модификация переменной. Вернемся к нашей процедуре**ПередЗаписью.**

В этой процедуре мы сначала проверим поля **Фамилия**, **Имя** и **Отчество** на заполненность (возможны и более сложные проверки), после чего, если хотя бы одно *поле* не заполнено – сообщим об этом пользователю и выйдем из процедуры, если все поля заполнены – сформируем наименование. Вот какой код позволяет реализовать эту задачу:

```
&НаКлиенте
Процедура ПередЗаписью(Отказ, ПараметрыЗаписи)
 //Переменная для хранения текста сообщения пользователю
 Перем ТекстСообщения;
 //Запишем пустую строку в переменную
 ТекстСообщения="";
  //Если не введена фамилия...
 Если ПустаяСтрока(Объект.Фамилия) Тогда
   //Формируем строку сообщения
   ТекстСообщения=ТекстСообщения+"Не заполнено поле Фамилия;";
 КонецЕсли;
  //Если не введено имя...
 Если ПустаяСтрока(Объект.Имя) Тогда
   ТекстСообщения=ТекстСообщения+" Не заполнено поле Имя;";
 КонецЕсли;
 //Если не введено отчество...
 Если ПустаяСтрока(Объект.Отчество) Тогда
   ТекстСообщения=ТекстСообщения+" Не заполнено поле Отчество;";
 КонецЕсли;
```

```
//Если строка сообщения не пуста, то есть - содержит
//сообщения о незаполненных полях
Если НЕ ПустаяСтрока(ТекстСообщения) Тогда
 //Выводим сообщение
 Сообщить(ТекстСообщения);
 //Отказываемся от записи объекта
 Отказ=Истина;
 //Выходим из процедуры
 Возврат;
КонецЕсли;
//Если все поля заполнены, выхода из процедуры не произошло,
//формируем наименование
Объект.Наименование=Объект.Фамилия+" "+ ВРег(Лев(Объект.Имя,1))+". "+ВРег(Лев(Объект.Отчество,1))+".";
КонецПроцедуры
```

Строковая функция **Лев** позволяет получить заданное количество символов из строки, начиная с самого левого. *Строковая функция* **ВРег** переводит символы в верхний *регистр* – на тот случай, если *пользователь* случайно ввел имя, фамилию или отчество с маленькой буквы. Конечно, здесь можно предусмотреть еще множество проверок и автоматических корректировок (например, можно исправить первую букву во введенных фамилии, имени и отчестве, если она случайно введена в нижнем регистре), мы ограничимся тем, что сделано сейчас.

В итоге мы получаем следующие сообщения об ошибках при незаполненности полей, Рисунок 2.21.

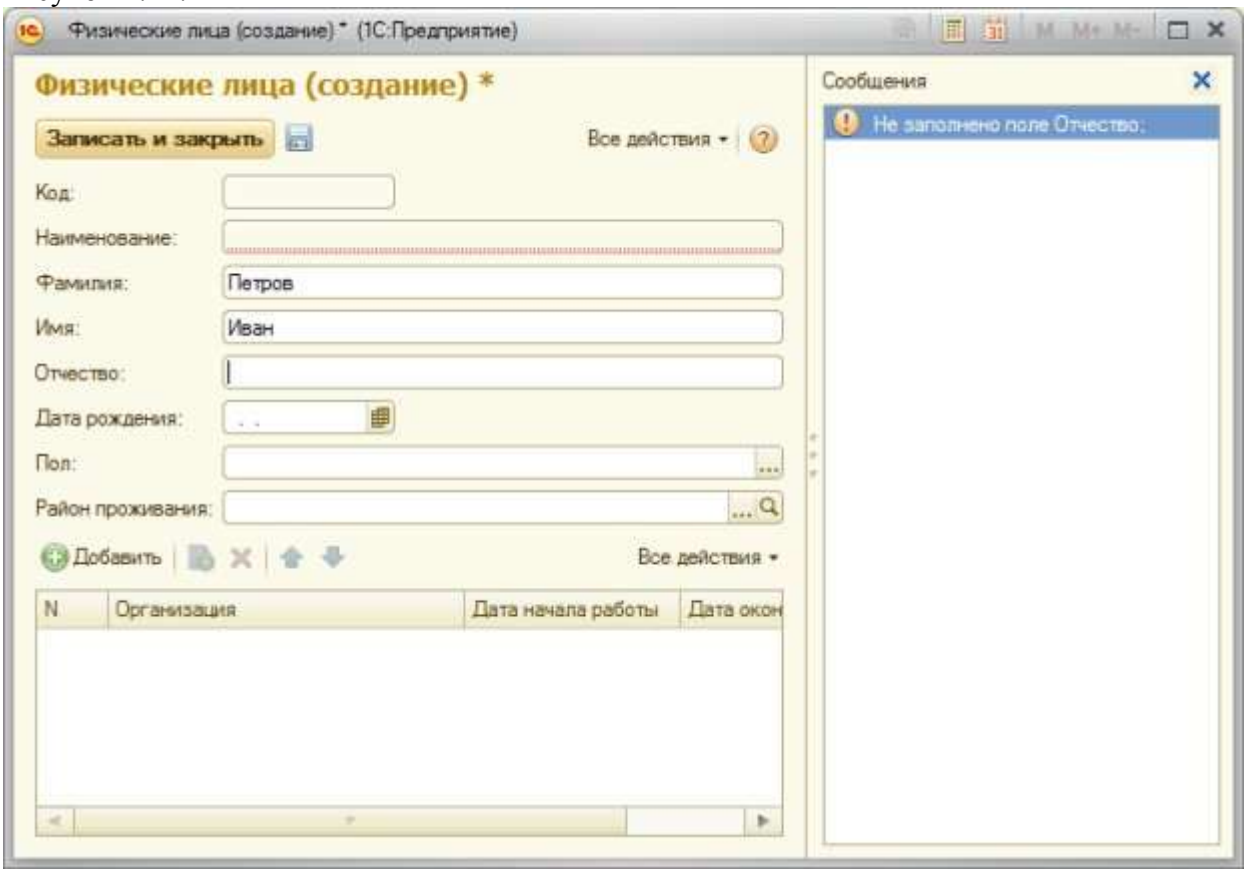


Рисунок 2.21. Сообщение об ошибке

После успешного выполнения процедуры **ПередЗаписью**, наименование выглядит следующим образом. Мы намеренно ввели отчество с маленькой буквы – как было пояснено выше, наш код готов к такому повороту событий, Рисунок 2.22.

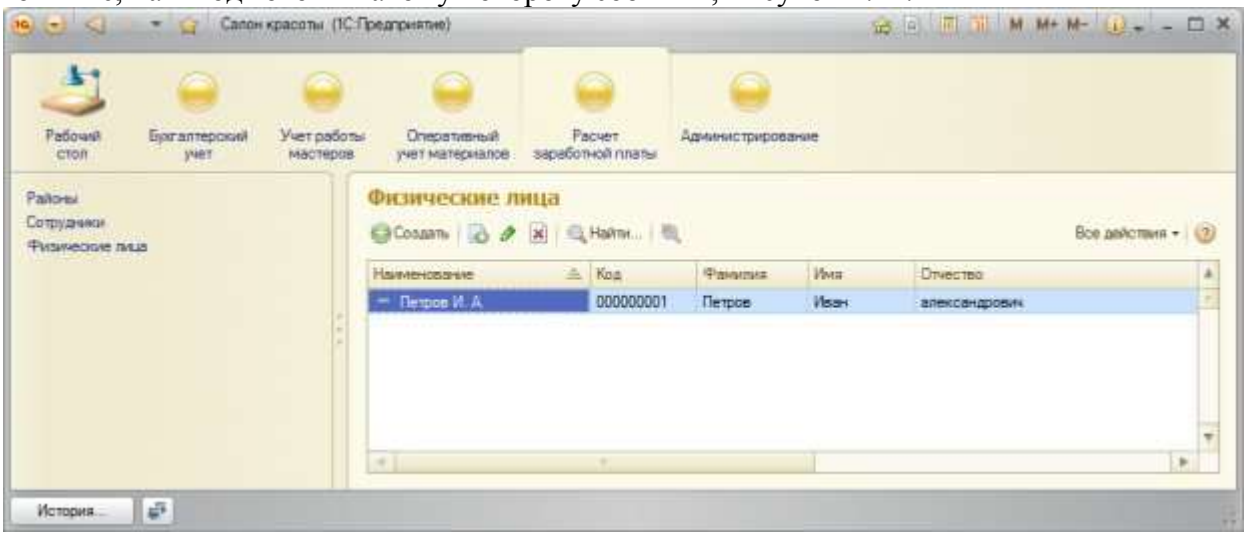


Рисунок 2.22. Новая запись в справочнике Физические лица

Объект СообщениеПользователю

Обратите внимание на то, что здесь мы пользуемся обычным методом **Сообщить** – мы выводим в окно сообщения одно сообщение, содержащее необходимые сведения. В «1С:Предприятие» 8.2. мы можем поступить *по*-другому – вывести сообщения об ошибках или другие сведения, "привязав" их к полям, которые вызвали ошибки. Для этого можно воспользоваться объектом**СообщениеПользователю**. Он, помимо прочих полезных возможностей, позволяет формировать сообщения и "привязывать" их к реквизитам формы. Перепишем код таким образом, чтобы сообщения об ошибках (то есть, о незаполненных полях **Фамилия**,**Имя**, или **Отчество**), выявленных в процедуре ПередЗаписью, выводились бы в привязке к соответствующим элементам формы. Вот какой код позволяет этого добиться:

```
&НаКлиенте
Процедура ПередЗаписью(Отказ, ПараметрыЗаписи)
 //Если не введена фамилия...
 Если ПустаяСтрока(Объект.Фамилия) Тогда
   СообщитьПользователю("Объект.Фамилия", "Заполните поле Фамилия", Отказ);
 КонецЕсли;
 //Если не введено имя...
 Если ПустаяСтрока(Объект.Имя) Тогда
   СообщитьПользователю("Объект.Имя", "Заполните поле Имя", Отказ);
 КонецЕсли;
 //Если не введено отчество...
 Если ПустаяСтрока(Объект.Отчество) Тогда
   СообщитьПользователю("Объект.Отчество", "Заполните поле Отчество", Отказ);
 КонецЕсли;
 //Если флаг Отказ не был установлен - формируем наименование
 Если НЕ Отказ Тогда
   Объект.Наименование=Объект.Фамилия+" "+ ВРег(Лев(Объект.Имя,1))+".
"+ВРег(Лев(Объект.Отчество,1))+".";
 КонецЕсли;
```

```
КонецПроцедуры

&НаКлиенте
//Процедура, формирующая и выводящая сообщение с переданными ей параметрами
Процедура СообщитьПользователю(ПутьКРеквизиту, Текст, Отказ)
 Сообщение=Новый СообщениеПользователю;
 Сообщение.Поле=ПутьКРеквизиту;
 Сообщение.Текст=Текст;
 Сообщение.Сообщить();
 Отказ=Истина;
КонецПроцедуры
```

Поясним приведенный код. Для начала, мы создали новую клиентскую процедуру **СообщитьПользователю**. Эта процедура принимает три параметра. Первый – **ПутьКРеквизиту** содержит *строковый путь* к полю, к которому должно быть привязано сообщение. Второй – **Текст** – содержит текст сообщения. Третий – **Отказ** – используется для установки в *значение* **Истина**параметра **Отказ** процедуры **ПередЗаписью** в том случае, если процедура **СообщитьПользователю** будет вызвана хотя бы один раз. А хотя бы однократный ее вызов означает, что одно из полей не заполнено, то есть наименование сформировать невозможно, соответственно, записать *объект* так же не получится.

Когда процедура вызывается, мы сначала создаем новый *объект* типа **СообщениеПользователю**. Затем его свойство **Поле**устанавливаем в *значение* параметра **ПутьКРеквизиту**. Этот *параметр* должен быть строковым и имеет, в нашем случае вид "*Объект*.Фамилия", "*Объект*.Имя", "*Объект*.Отчество" - это позволяет правильно "привязать" сообщение к полям формы. Свойство**Текст** объекта **СообщениеПользователю** содержит текст для вывода.

Мы, кроме того, полностью переработали процедуру **ПередЗаписью**. А именно, если проверка на заполнение поля указывает на то, что *поле* пустое, вызывается процедура **СообщитьПользователю.** *По* окончании проверок мы проверяем, установлен ли*параметр* **Отказ** в *значение* **Истина** – если не установлен – ни одна из проверок не завершилась обнаружением пустого поля и мы можем формировать наименование. Если установлен – наименование мы не формируем – и процедура заканчивает работу, а записи объекта, естественно, не происходит – *пользователь* видит лишь сообщения об ошибках.

Если было сформировано несколько сообщений типа **СообщениеПользователю** – *пользователь* видит одно окно сообщения около поля, но это окно снабжено кнопками для перемещения вперед и назад – щелчок *по* кнопке приводит к "переходу" сообщения от одного поля с ошибкой к другому, Рисунок 2.23, 2.24.

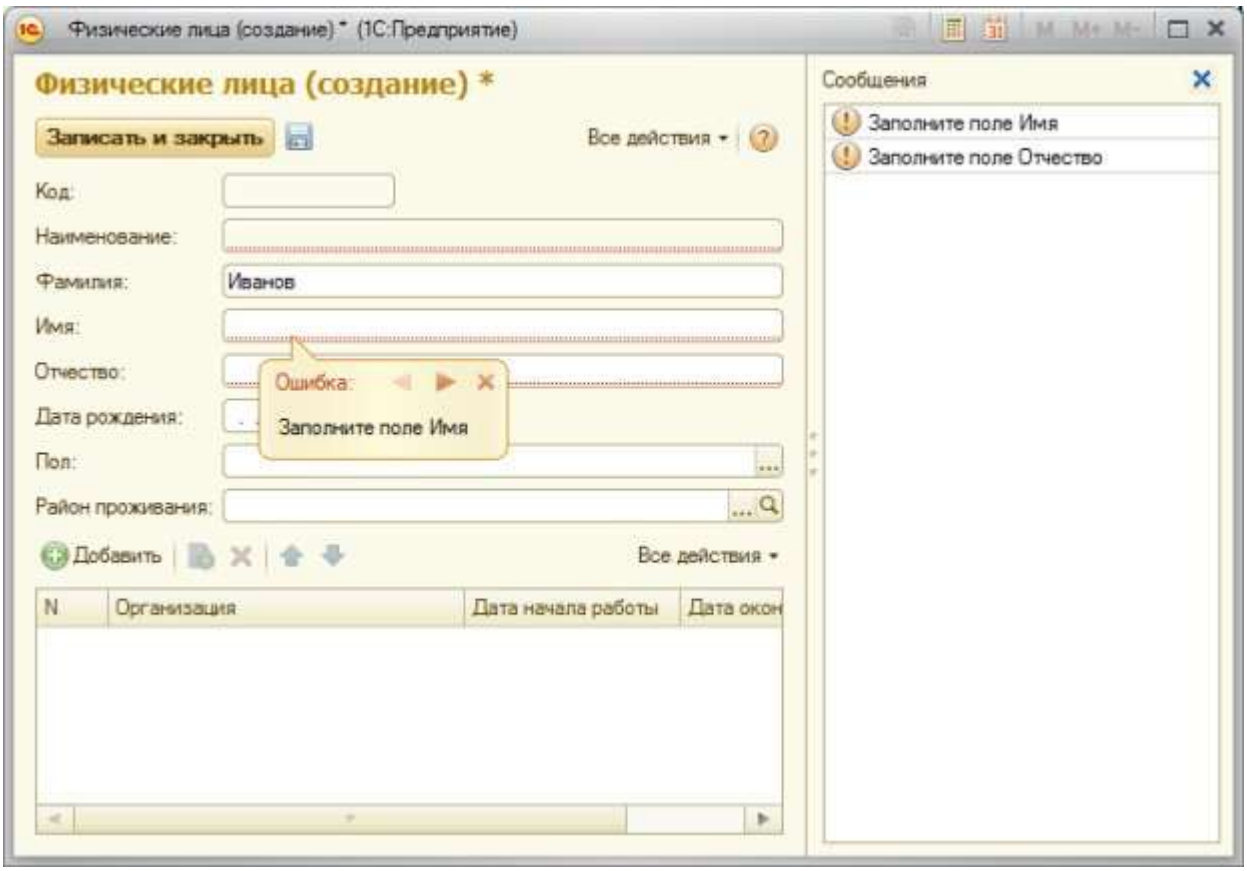


Рисунок 2.23. Сообщение об ошибке, привязанное к полю Имя

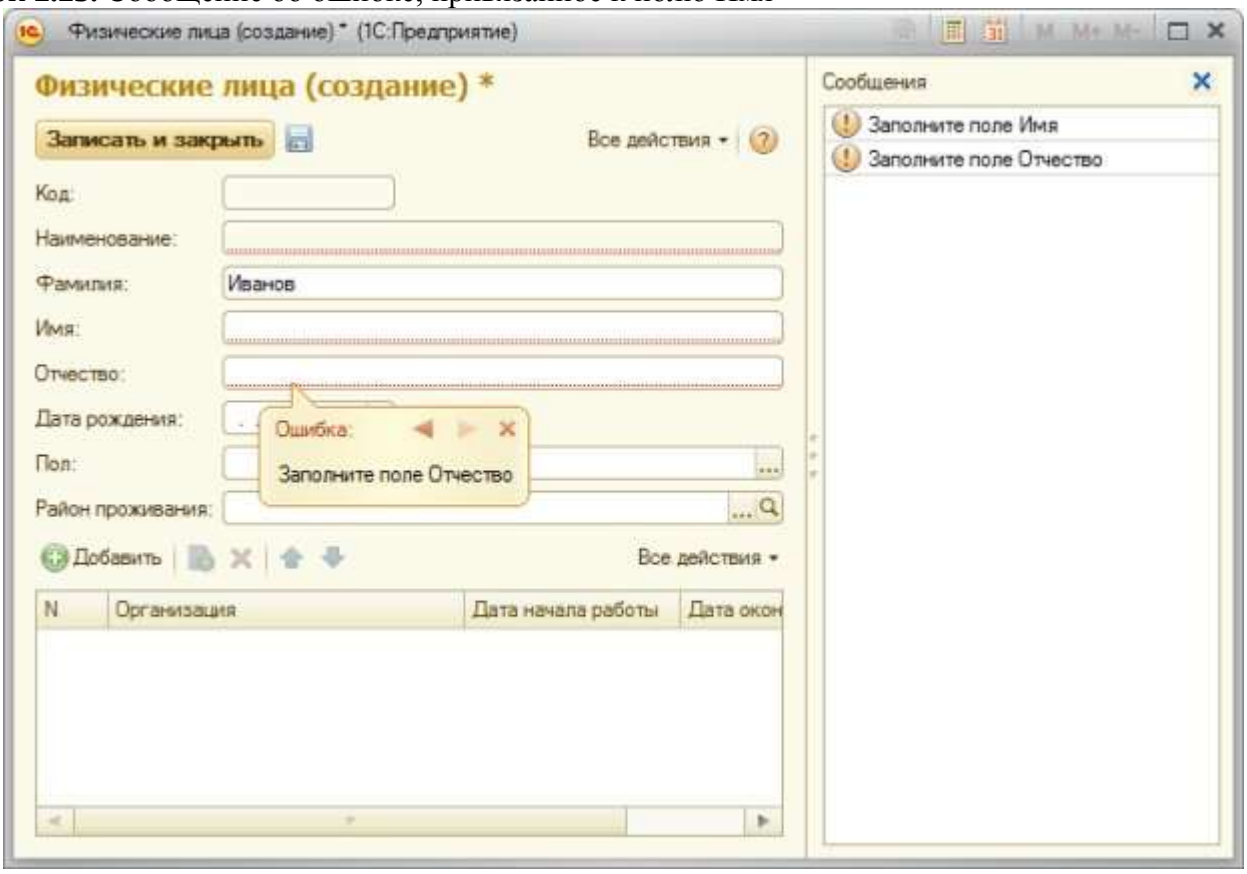


Рисунок 2.24. Сообщение об ошибке, привязанное к полю Отчество

Доведем до логического завершения пример со справочником **ФизическиеЛица**. Для этого заполним справочник Районы и введем в *информационную базу* сведения о следующих физических лицах (Таблица 2.1).

Таблица 2.1 – Физические лица

Фамилия	**Имя**	**Отчество**	**Дата рождения**	**Пол**	**Район**
Иванов	Иван	Иванович	27.02.1984	Мужской	Ленинский
Петров	Петр	Петрович	12.06.1985	Мужской	Ленинский
Васильев	Павел	Петрович	17.05.1985	Мужской	Ленинский
Расчетчиков	Александр	Иванович	12.03.1980	Мужской	Октябрьский
Александров	Александр	Александрович	17.09.1970	Мужской	Октябрьский
Бухгалтерова	Василиса	Владимировна	13.08.1976	Женский	Уральский

Обратите внимание на то, что справочник **ФизическиеЛица** – это пример справочника, с которым пользователям нашей *информационной базы* придется работать достаточно часто. В данный момент для того, чтобы создать новый элемент справочника, нам нужно выполнить несколько действий – перейти в раздел **Расчет заработной платы**, щелкнуть *по* ссылке, открывающей *список* справочника, после чего нажать на кнопку **Создать новый элемент списка**. Для того, чтобы сократить количество действий, необходимых для выполнения часто используемых операций, мы можем соответствующим образом настроить *интерфейс* нашего *прикладного решения*, в частности, поработать с панелью действий соответствующего раздела и с**Рабочим столом**.

Настройка командного интерфейса для ускорения доступа к справочнику

Добавим команду создания нового элемента справочника **ФизическиеЛица** в панель действий раздела **УчетРаботыМастеров**. Для этого откроем окно Все подсистемы командой контекстного *меню* ветви Подсистемы *дерева конфигурации* и установим флаг **Видимость** напротив команды **Физические лица: Создать** в области **Панель действий.Создать**, Рисунок 2.25 .

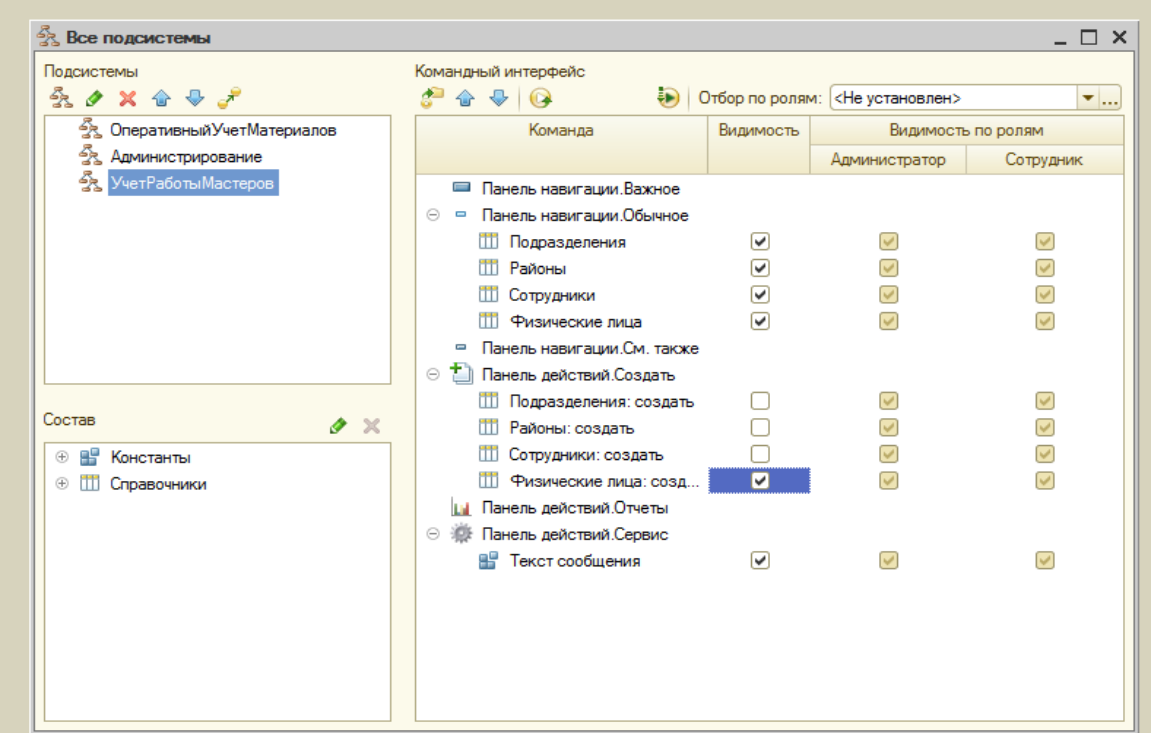


Рисунок 2.25. Настройка панели действий раздела Расчет заработной платы

Мы можем включить команду добавления нового физического лица в командный *интерфейс* **Рабочего стола (Рисунок 2.26)**.

Для этого выполним команду контекстного *меню корневого элемента* конфигурации **Открыть командный интерфейс рабочего стола**

Выделим в *поле* **Доступные команды** команду **Физические лица: создать**, в *поле* состава командного интерфейса – **команду Панель действий.Создать** и нажмем на кнопку со значком ">" (Добавить команду на *рабочий стол*), которая находится между полями, после чего установим флаг Видимость для добавленной команды, рис 3.25.

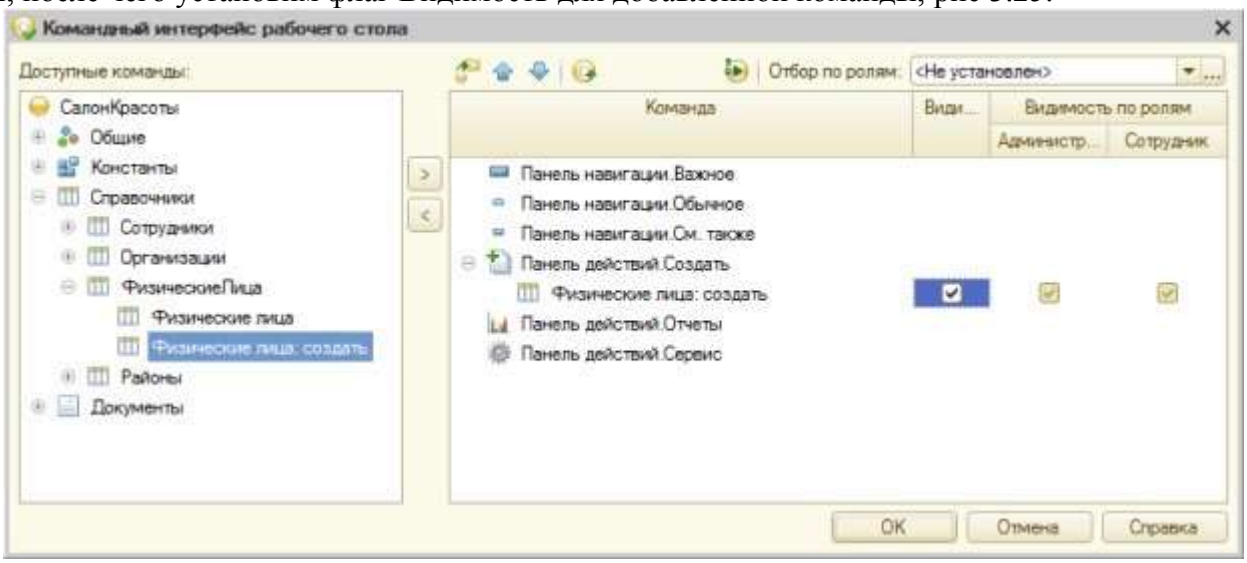


Рисунок 2.26. Настройка панели действий Рабочего стола

Теперь, Рисунок 2.27., *команда* для быстрого создания элементов справочника **ФизическиеЛица** добавлена в панель действий **Рабочего стола**.

Рисунок 2.27. Новая команда в панели действий рабочего стола

Литература:

Официальный сайт интернет-униерситета «Интуит» (электронный ресурс), режим доступа http://www.intuit.ru/studies/courses/2318/618/lecture/17939.

Контрольные вопросы для самопроверки:

4. Как создаются справочники?
5. Виды справочников в «1С:Предприятие»?
6. Как настроить интерфейс рабочего стола

Задание к лабораторной работе

Создать справочники в системе в соответствии с порядком, отраженным в теоретической части

Методические указания и порядок выполнения работы

Работа выполняется в точном порядке, как это указано на рисунках и в теоретической части

Индивидуальное задание

не предусмотрено

Требования к отчету и защите

1. Результатом выполнения лабораторной работы является сформированный в программе файл, содержащий выполненные задания. В ЭИОС результаты работы не выкладываются.

2. Планируется защита работы, где студент комментирует порядок выполнения заданий, а также отвечает на вопросы, представленные выше

ЛАБОРАТОРНАЯ РАБОТА № 3

О РАЗЛИЧНЫХ ВИДАХ СПРАВОЧНИКОВ

ОБЩИЕ СВЕДЕНИЯ

Цель: научиться информационную базу на платформе ««1С:Предприятие»»

Материалы, оборудование, программное обеспечение:
1. персональный компьютер (компьютерные классы ГУК)
2. программное обеспечение «1С:Предприятие»

Условия допуска к выполнению:
умение работать на ПК и знание техники безопасности

Критерии положительной оценки:
предоставление результатов работы в виде файла и прохождение защиты.

Планируемое время выполнения:
Аудиторное время выполнения (под руководством преподавателя): 2 ч.
Время самостоятельной подготовки: 2 ч.

Теоретическое введение

Многие справочники, используемые на практике, являются иерархическими. Это означает, что каждому из элементов справочника может быть сопоставлен другой элемент, который называется родителем.

Возможна *иерархия* различных видов, в частности, при иерархии групп и элементов в справочник, помимо обычных элементов можно включать группы, которые, в свою *очередь*, могут включать в себя другие группы и элементы.

Примером такого справочника является справочник, хранящий списки товаров, материалов, услуг. Обычно такой справочник носит название Номенклатура. Как правило, подобный справочник имеет множество групп – например – Товары – для хранения записей о товарно-материальных ценностях, и Услуги – для хранения списков услуг.

Второй вид иерархии – это *иерархия* элементов. Хорошим примером справочника, для которого естественна *иерархия* элементов, является справочник для хранения сведений о подразделениях организаций. Такой справочник обычно так и называется – *Подразделения*. Логика иерархии элементов заключается в том, что отдельные элементы справочника, описывающие отдельные *подразделения* организации, могут являться родителями для других элементов – так же подразделений. Например, подразделение Администрация вполне может включать в себя *подразделения* Бухгалтерия, Отдел кадров и так далее.

Справочники, в сущности, являются хранилищами аналитических признаков учета. Например, в случае со справочником Номенклатура можно организовать учет таким образом, чтобы иметь сведения об остатках каких-либо номенклатурных позиций на складе, об их стоимости. В случае со справочником, поддерживающим иерархию элементов, любой элемент справочника можно использовать как аналитический "разделитель" учета (то есть, например, выбирать данные элементы при заполнении табличных частей других объектов, например, документов). А вот при иерархии групп и элементов использовать в качестве аналитического разделителя группу не получится.

Если родительские отношения существуют внутри справочника, такие справочники называются иерархическими. Существует еще один вид взаимоотношений между

справочниками, который называется подчинением. Предположим, у нас имеется справочник Контрагенты, содержащий *список* организаций, с которыми наша организация имеет какие-то взаимоотношения. Каждый из контрагентов, описанных в этом справочнике, имеет некоторое количество контактных лиц – представителей контрагента. Можно сказать, что каждый представитель контрагента "принадлежит" определенному контрагенту. Такие взаимоотношения между справочниками реализуются при помощи механизма указания владельцев справочника и настройки подчинения.

Рассмотрим работу с *иерархическими справочниками*.

Иерархические справочники

Создадим новый справочник **Единицы измерения**, зададим следующие его параметры:

Имя: ЕдиницыИзмерения

Длина наименования: 100 символов

Подсистемы: ОперативныйУчетМатериалов

Это будет очень простой справочник, стандартный *реквизит* которого **Наименование** будет использоваться для хранения информации о наименовании единицы измерения.

Теперь создадим очередной справочник – **Номенклатура**. Зададим следующие параметры:

Имя: Номенклатура

Подсистемы: БухгалтерскийУчет, ОперативныйУчетМатериалов

На вкладке окна редактирования объекта **Иерархия**, Рисунок 4.1., установим следующие параметры:

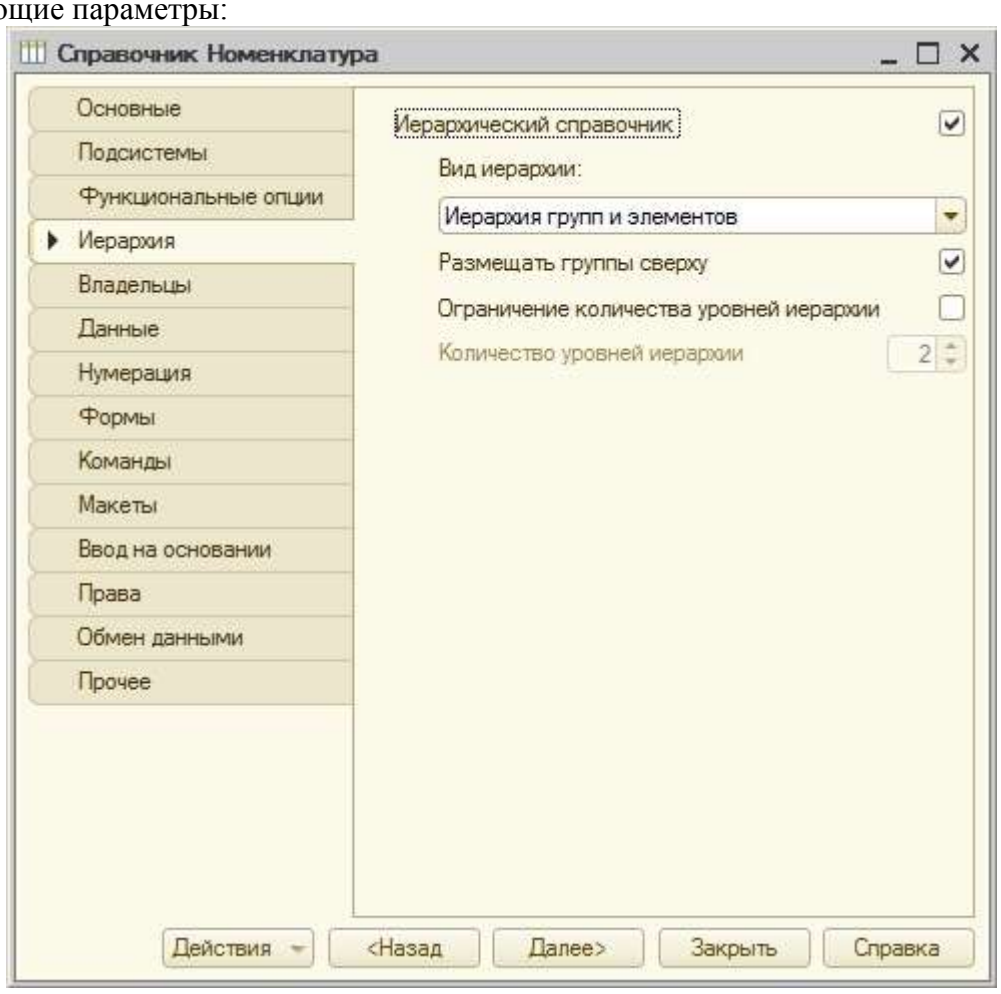


Рисунок 3.1. Настройка иерархического справочника

Иерархический справочник: Установлено

Вид иерархии: *Иерархия* групп и элементов (Рисунок3.1).

Этот *параметр* может принимать *значение* **Иерархия элементов**. В нашем случае справочник сможет содержать отдельные элементы, собранные, в зависимости от их вида, в группы. Эту структуру можно сравнить с папками и файлами в файловой системе компьютера. Группы – это папки, отдельные элементы – это файлы.

На вкладке **Данные**, Рисунок 3.2., добавим следующие реквизиты:

ЕдиницаИзмерения: Тип СправочникСсылка.ЕдиницыИзмерения.

Услуга: Тип Булево, **Использование**: Для группы и элемента. Эта установка позволит задавать данный *реквизит* и для элементов и для групп.

Заполнять из данных заполнения: *Истина*

Отдельные группы нашего справочника планируется использовать для хранения исключительно услуг, и подобная установка (в частности, истинность параметра **Заполнять из данных заполнения**) позволит нам реализовать автоматический механизм заполнения данного реквизита для элементов, входящих в группы.

Длина наименования: 100

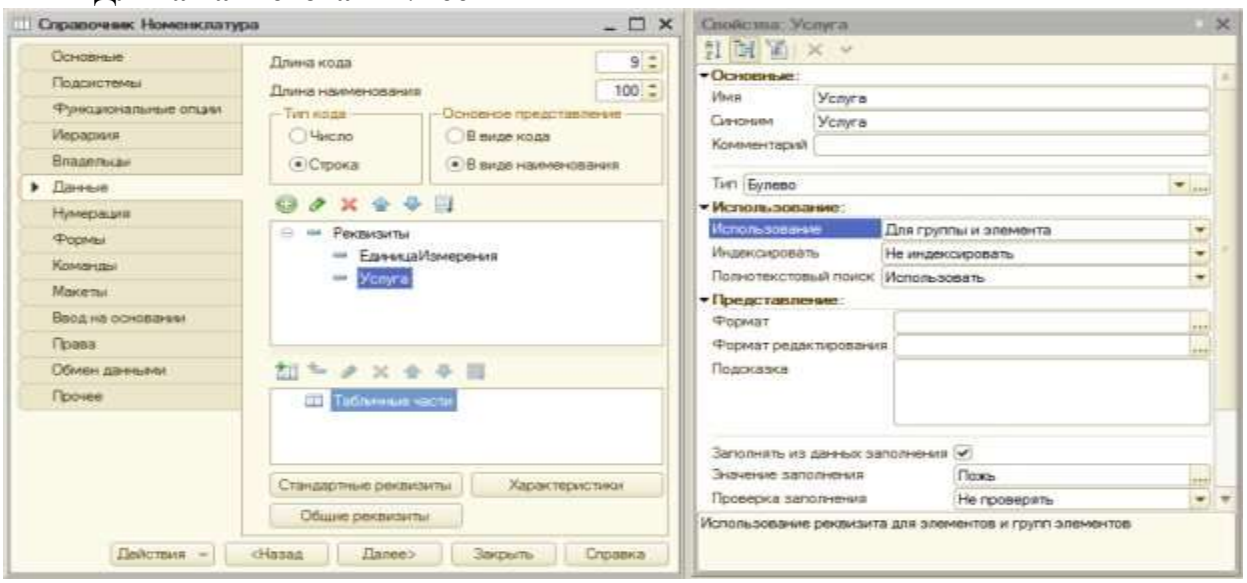


Рисунок 3.2. Состав реквизитов справочника Номенклатура

Таким образом, при создании элемента справочника мы будем задавать название элемента в стандартном *реквизите***Наименование**, указывать единицу измерения, а так же, для услуг, устанавливать флаг **Услуга**, причем, установка этого флага для группы будет означать, что в ней хранятся списки услуг, а для элемента – то, что он является услугой.

Автозаполнение реквизитов

Реализуем функцию автоматического заполнения реквизита **Услуга** для элементов, входящих в группы. Нам нужно, чтобы элемент, создаваемый в группе с установленным флагом **Услуга**, при его создании, автоматически бы получал установленный флаг **Услуга**, соответственно, если данный флаг у группы не установлен, у элемента он так же не должен быть установлен. При этом нам нужно предусмотреть ситуацию, когда элемент создается вне группы – на верхнем уровне справочника **Номенклатура**. Для решения этой задачи мы можем воспользоваться обработчиком события **ОбработкаЗаполнения**, его процедура располагается в модуле объекта.

Перейдем в *модуль* объекта (кнопка **Модуль объекта** на закладке **Прочие** окна редактирования объекта), из списка процедур и выберем **ОбработкаЗаполнения**, Рисунок 3.3.

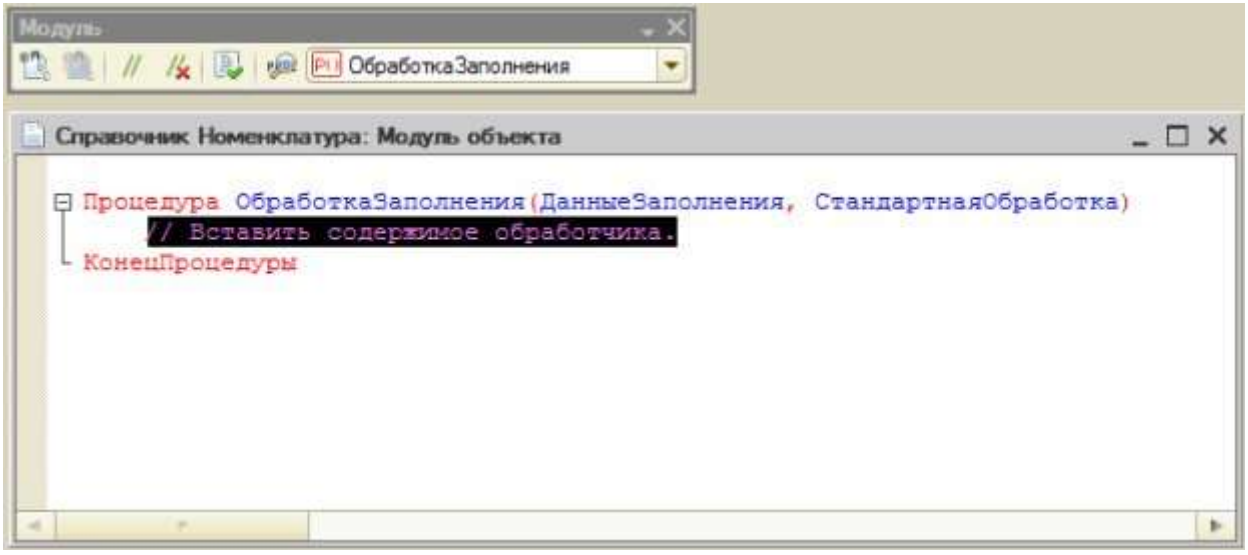


Рисунок 3.3. Процедура обработки заполнения справочника

Процедура будет исполняться на стороне сервера, причем, вызываться она будет при различных способах создания элемента справочника – например, при интерактивном создании пользователем, при копировании, при программном создании. *Параметр*процедуры **СтандартнаяОбработка** позволяет включать или отключать стандартную обработку процесса заполнения реквизитов,*параметр* **ДанныеЗаполнения** содержит данные, которые система использует для заполнения элемента.

В режиме **«1С:Предприятие»** откроем справочник **Номенклатура**, создадим две группы – **Товары** – флаг **Услуги** в этой группе не устанавливаем, и **Услуги** – флаг установлен, Рисунок 3.4.

Рисунок 3.4. Две группы в справочнике **Номенклатура**

Исследуем процедуру обработки заполнения, прежде чем продолжать работу над ней. Для этого вставим в нее какую-нибудь команду, например: **ТестоваяПеременная = 0;** и установим на строку с данной командой точку останова. Для этого либо выполним *двойной щелчок* левой кнопкой мыши на сером *поле* слева от команды, либо, установив *курсор* в строку с командой, выполним команду **Отладка > Точка останова**, либо – установив *курсор* в нужную строку, нажмем **F9**. Для установки точки останова нужно, чтобы строка, на которую мы пытаемся ее установить, присутствовала в *конфигурации базы данных*, то есть – написав код, нужно нажать на кнопку **Обновить конфигурацию** *базы данных*. В итоге у нас должно получиться следующее, Рисунок 3.5.

Рисунок 3.5. Точка останова в процедуре

Запустим конфигурацию в режиме отладки (кнопка **Начать отладку**, *команда меню* **Отладка > Начать отладку**, или клавиша **F5**на клавиатуре). Перейдем в группу **Услуги** и создадим в ней новый элемент. Когда управление будет передано в **Конфигуратор**, установим *курсор* на имя параметра **ДанныеЗаполнения**, вызовем контекстное *меню* и выберем команду **Вычислить выражение**. Появится окно *Выражение*, из которого можно понять, что *переменная* **ДанныеЗаполнения** – это структура, в которой присутствуют сведения о родителе создаваемого элемента – то есть – о группе **Услуги**, Рисунок 3.6.

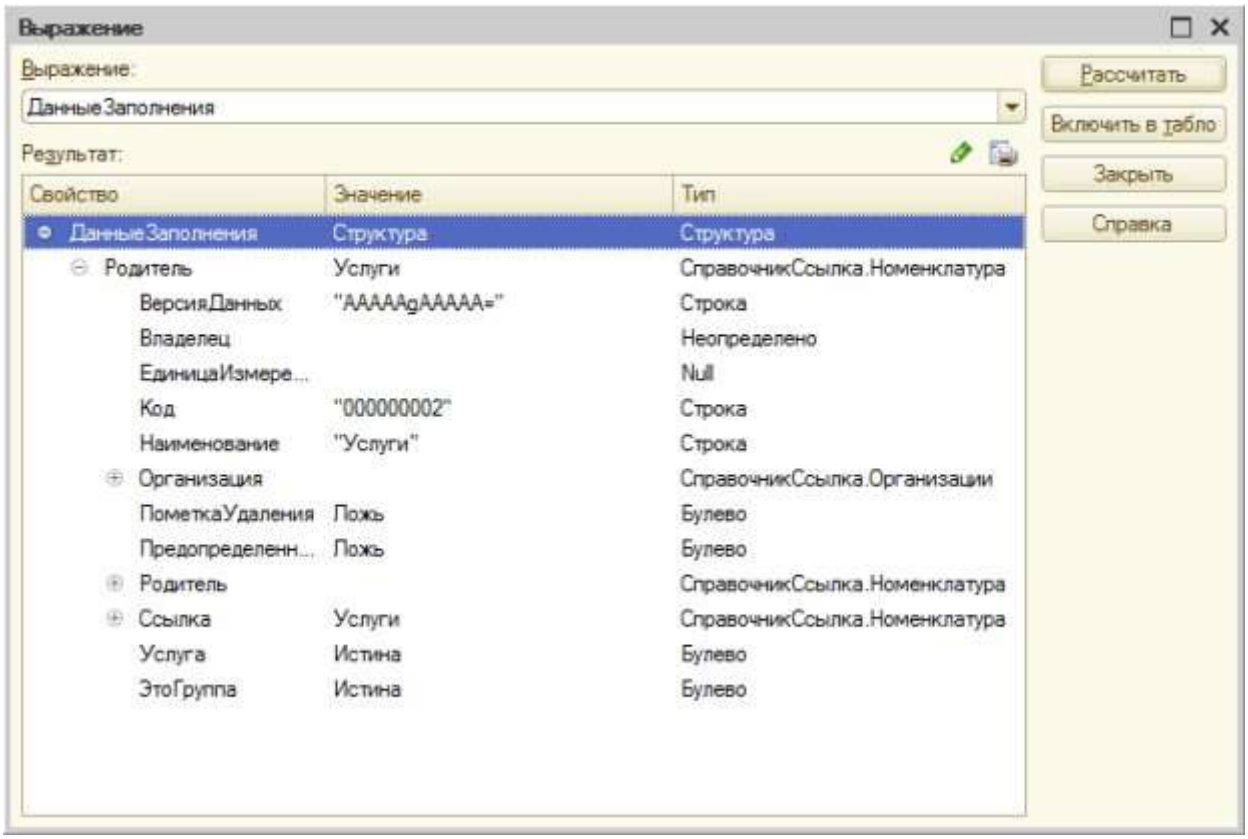


Рисунок 3.6. Структура ДанныеЗаполнения

Структура – это *таблица*, которая содержит пары вида *Ключ – Значение*.

В нашем случае, если процедура **ОбработкаЗаполнения** отработает – она заполнит лишь *поле* **Родитель** для создаваемого элемента. А нам хотелось бы установить и флаг **Услуга** в соответствии с данными родителя.

Рассмотрим некоторые составляющие данных, к которым мы можем получить *доступ* посредством структуры.

Родитель – здесь хранится родитель элемента – в нашем случае – *группа* **Услуги** типа **СправочникСсылка.Номенклатура**. То есть, при заполнения поля **Родитель** создаваемого элемента, окажется, что он будет хранить ссылку на другой элемент (в нашем случае – группу), входящий в справочник **Номенклатура.**

Владелец – данное *поле* у нашей группы, находящейся в справочнике, имеет *значение* **Неопределено**. Такое *значение*присваивается тем свойствам, которые, в принципе, могут быть установлены, но в данном случае значения не имеют.

ЕдиницаИзмерения имеет *значение* **Null**. При настройке состава реквизитов справочника **Номенклатура**, мы указали, что**ЕдиницаИзмерения** может задаваться только для элемента. Но в структуре справочника в *информационной базе* подобное *поле*присутствует и у группы. Однако значения оно содержать не может – поэтому в качестве типа значения мы видим **Null**. Типы значений **Неопределено** и **Null** кажутся похожими, но это – разные вещи. *Значение* с типом **Неопределено** может быть задано, а *значение* **Null** не может быть задано в принципе.

Свойство **Услуга** установлено в *значение* **Истина** – этот флаг мы устанавливали при создании группы **Услуги.**

Свойство **ЭтоГруппа** так же истинно – оно устанавливается в истинность для групп.

Для того чтобы установить свойство **Услуга** у создаваемого элемента, мы могли бы напрямую обратиться к свойству элемента**Услуга** и установить его в *значение* флага Услуга у его родителя. Выглядеть это может, например, так:

Услуга = ДанныеЗаполнения.Родитель.Услуга.

Однако процедура ОбработкаЗаполнения предусматривает автоматический механизм заполнения реквизитов на основе переданной структуры. Так как стандартный механизм нас вполне устраивает, мы можем поступить *по*-другому. А именно, для установки свойства **Услуга** нам нужно лишь дополнить структуру необходимой записью. Сделать это можно с помощью стандартных операций *по* работе со структурой. А именно, следующим образом:

ДанныеЗаполнения.Вставить("Услуга", ДанныеЗаполнения.Родитель.Услуга);

В итоге у нас получается такой код, Рисунок 3.7.

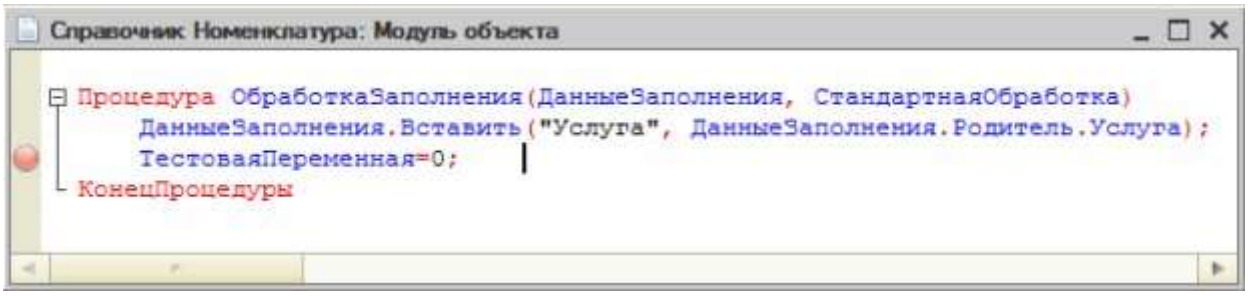


Рисунок 3.7. Заполнение реквизита Услуга на основании параметров элемента родителя

Здесь мы установили точку останова для того, чтобы посмотреть, как изменится структура при выполнении данной процедуры. Опробуем решение в пользовательском режиме, можно заметить, что, во-первых, структура **ДанныеЗаполнения** действительно теперь содержит *ключ* **Услуга** со значением **Истина**, а так же то, что элементы,

создаваемые в группе с установленным флагом**Услуга**, имеют данный *реквизит* в установленном положении, Рисунок 3.8.

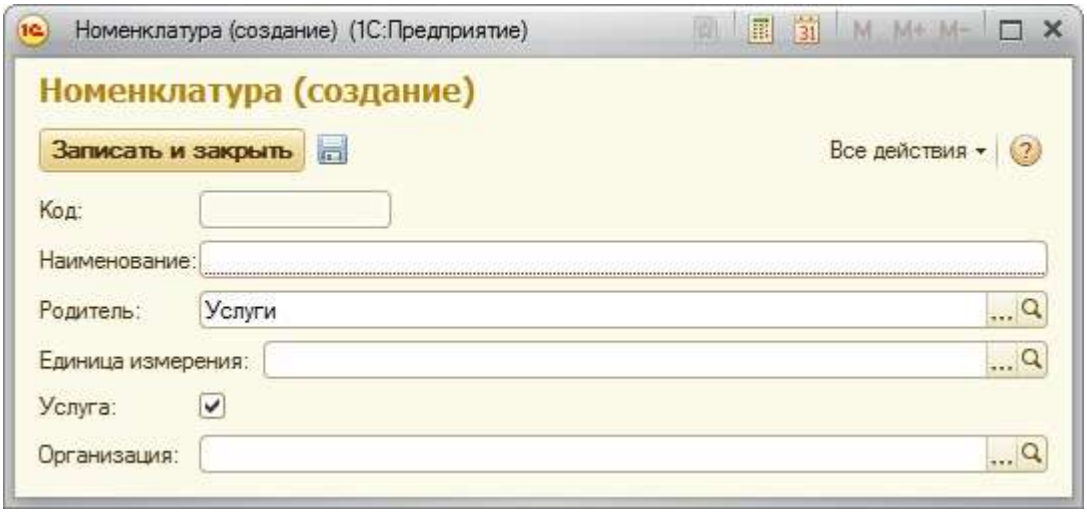


Рисунок 3.8. Результат заполнения реквизита Услуга на основании параметров элемента родителя

Подойдет ли *созданная процедура* для практического использования? На этот вопрос может ответить ее тестирование. А именно, попробуем создать еще одну группу в корневой части справочника. Очевидно, что у такой группы родителя не будет. То же самое касается создания элемента. Попытка приводит к появлению ошибки, Рисунок 3.9.

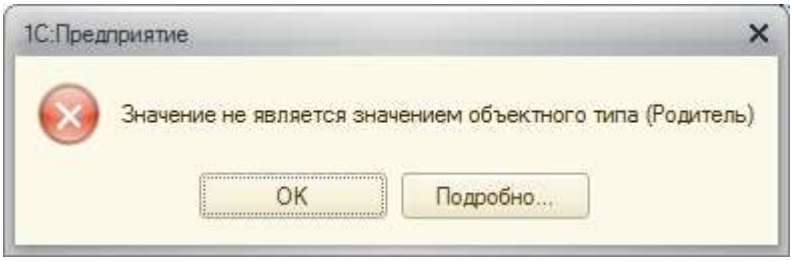


Рисунок 3.9. Ошибка при попытке создать элемент, у которого нет родителя

Нажав на кнопку **Подробно**, видим, что ошибка произошла при попытке добавить в структуру новую *запись*, Рисунок 3.10.

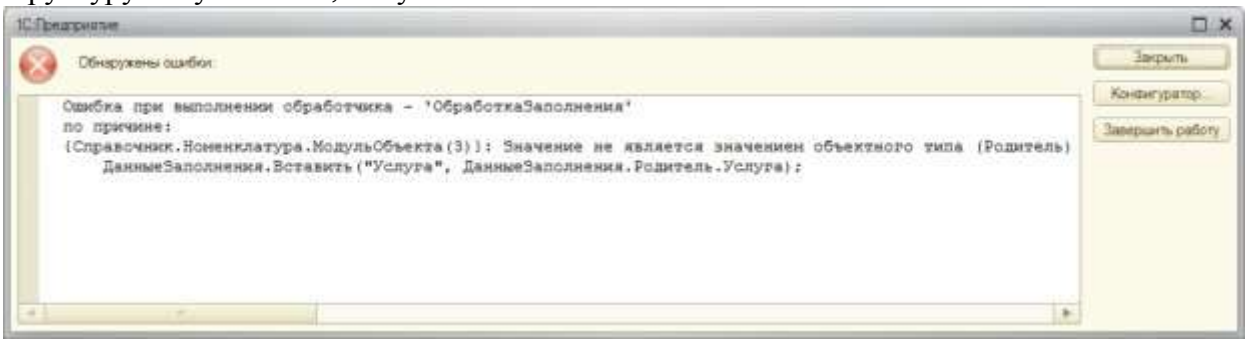


Рисунок 3.10. Более подробное описание ошибки

Нажав на кнопку **Конфигуратор**, мы попадаем в *Конфигуратор*. Прежде чем обращаться к элементу структуры Родитель, нужно убедиться в том, что **Родитель** в

структуре присутствует. Если **Родителя** нет – в структуру не нужно ничего добавлять, если есть – можно добавить. Попробуем такой код:

```
Если ДанныеЗаполнения.Свойство("Родитель") Тогда
 ДанныеЗаполнения.Вставить("Услуга", ДанныеЗаполнения.Родитель.Услуга);
КонецЕсли;
```

Как кажется, все должно работать правильно – если в структуре обнаружилось *поле* Родитель – мы можем обращаться к свойству Услуга. Но попытка выполнить эту процедуру снова приводит к ошибке

Эта ошибка возникает при проверке условия на наличие в структуре свойства Родитель. Здесь у нас возникает вопрос о том, чем является передаваемый *параметр* **ДанныеЗаполнения** при создании элемента или группы на верхнем уровне справочника. Для ответа на этот вопрос мы можем снова прибегнуть к отладке. Как видно, *значение* параметра неопределено, Рисунок 3.11.

Рисунок 3.11. Параметр ДанныеЗаполнения при создании элемента или группы на верхнем уровне справочника

Тип **Неопределено** говорит нам о том, что перед нами лишь *переменная*, тип которой программе не известен. Мы не можем обращаться к ней как к структуре, поэтому, прежде чем проверять, есть ли в структуре **ДанныеЗаполнения** свойство**Родитель**, нам нужно проверить, является ли передаваемый *параметр* **ДанныеЗаполнения** структурой. Мы знаем, что этот*параметр*, когда он заполнен данными, имеет тип **Структура,** следовательно, нам нужно исключить вариант, когда его тип равняется **Неопределено.**

Вышеприведенные рассуждения приводят нас к следующему коду:

```
Процедура ОбработкаЗаполнения(ДанныеЗаполнения, СтандартнаяОбработка)
Если ДанныеЗаполнения<>Неопределено Тогда
 Если ДанныеЗаполнения.Свойство("Родитель") Тогда
  ДанныеЗаполнения.Вставить("Услуга", ДанныеЗаполнения.Родитель.Услуга);
 КонецЕсли;
КонецЕсли;
КонецПроцедуры
```

В данной редакции обработчика события **ОбработкаЗаполнения** все работает верно.

Чисто теоретически (предположим, при изменении кем-либо нашего кода) возможна ситуация, когда **ДанныеЗаполнения** будут являться структурой и в этой структуре, в то же время, не будет свойства **Родитель**. Поэтому наряду с проверкой на неопределенность значения мы оставляем и проверку на наличие свойства **Родитель.**

Мы обсудили и проиллюстрировали родительские отношения в справочнике, рассмотрим теперь пример работы с *подчиненными справочниками*.

Подчиненные справочники

Создадим новый справочник, назовем его **Контрагенты**.
Добавим его в подсистемы **ОперативныйУчетМатериалов**.

Справочник будет иерархическим, с иерархией групп и элементов.

В состав реквизитов справочника добавим следующие (Рисунок 3.12.):

Имя: ПолноеНаименование, тип – Строка, *длина* 100.

Имя: КонтактныеСведения, тип – Строка, *длина* 100

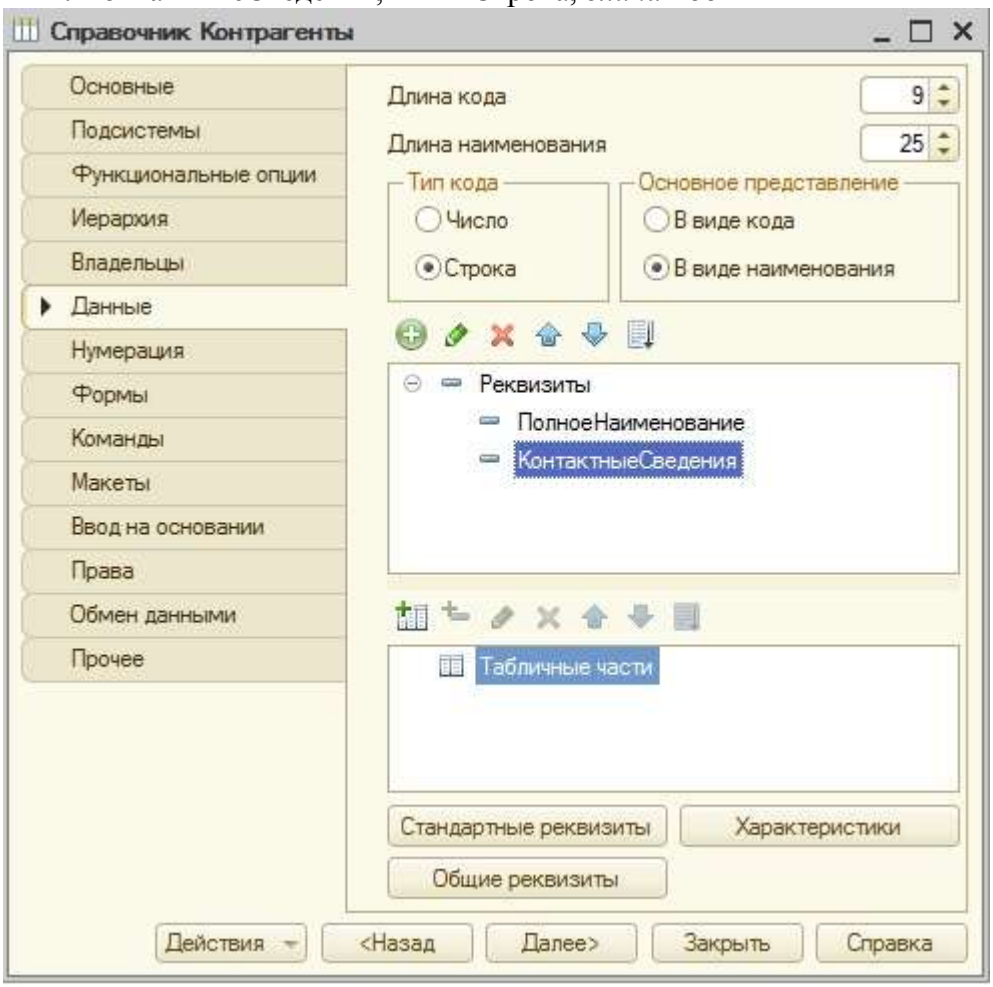


Рисунок 3.12. Справочник Контрагенты

Создадим еще один справочник. Назовем его **ПредставителиКонтрагентов**. Главная черта этого справочника – то, что он подчинен справочнику **Контрагенты**. Для настройки подчинения используется вкладка окна настройки объекта конфигурации **Владельцы**. Здесь мы должны добавить в **Список владельцев справочника** справочники-владельцы, в нашем случае – справочник **Контрагенты**. После того, как владелец добавлен в этот *список*, мы можем настроить для него *параметр* **Использование подчинения**. Здесь возможны три варианта:

Элементам – элементы *подчиненного справочника* подчинены элементам справочника-владельца. В нашем случае выбор этого параметра означает, что в качестве "владельца" представителя контрагента выступает сам *контрагент*.

Группам – подчинение группам справочника-владельца. Нас это не устроит – так как группы справочника будут содержать контрагентов, сгруппированных *по* достаточно общим показателям (Покупатели, Заказчики и т.д.), и сопоставление одного контактного лица нескольким разным, например, покупателям, смысла не имеет. В другой ситуации эта установка могла бы быть вполне оправданной.

Группам и элементам – подчинение как группам, так и элементам справочника-владельца.

Мы укажем в параметре **Использование подчинения** вариант **Элементам**, Рисунок 3.13.

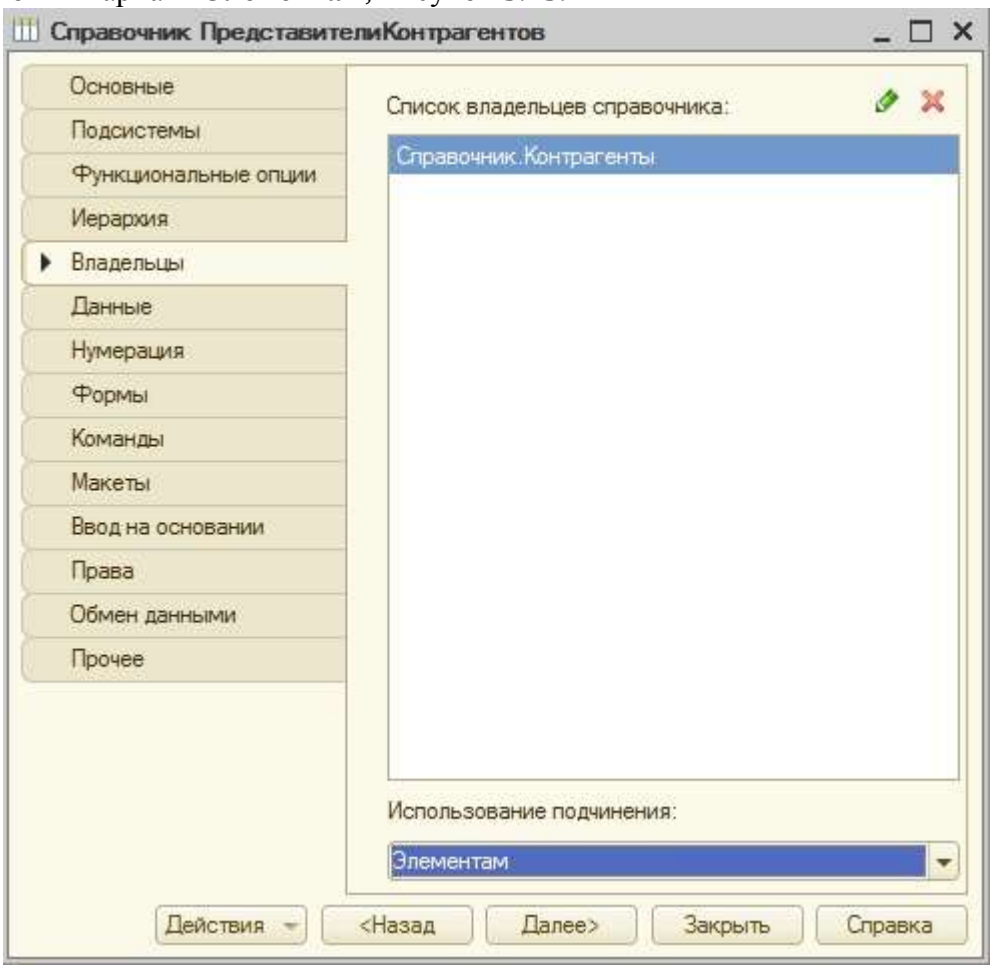


Рисунок 3 .13. Настройка подчинения

Добавим справочник в состав подсистемы **ОперативныйУчетМатериалов**.

В состав реквизитов справочника добавим следующие:

Имя: ФИО, тип – Строка, *длина* 100

Имя: КонтактныеСведения, тип – Строка, *длина* 100.

Имя: ПредставительРаботает, тип – Булево.

Посмотрим теперь, как выглядит работа с созданными справочниками в режиме **»1С:Предприятие»**. Особенность здесь заключается в том, что, открывая карточку контрагента, в левой ее части мы видим область **Перейти**, где можно найти ссылку для перехода в справочник**ПредставителиКонтрагентов**, Рисунок 3.14.

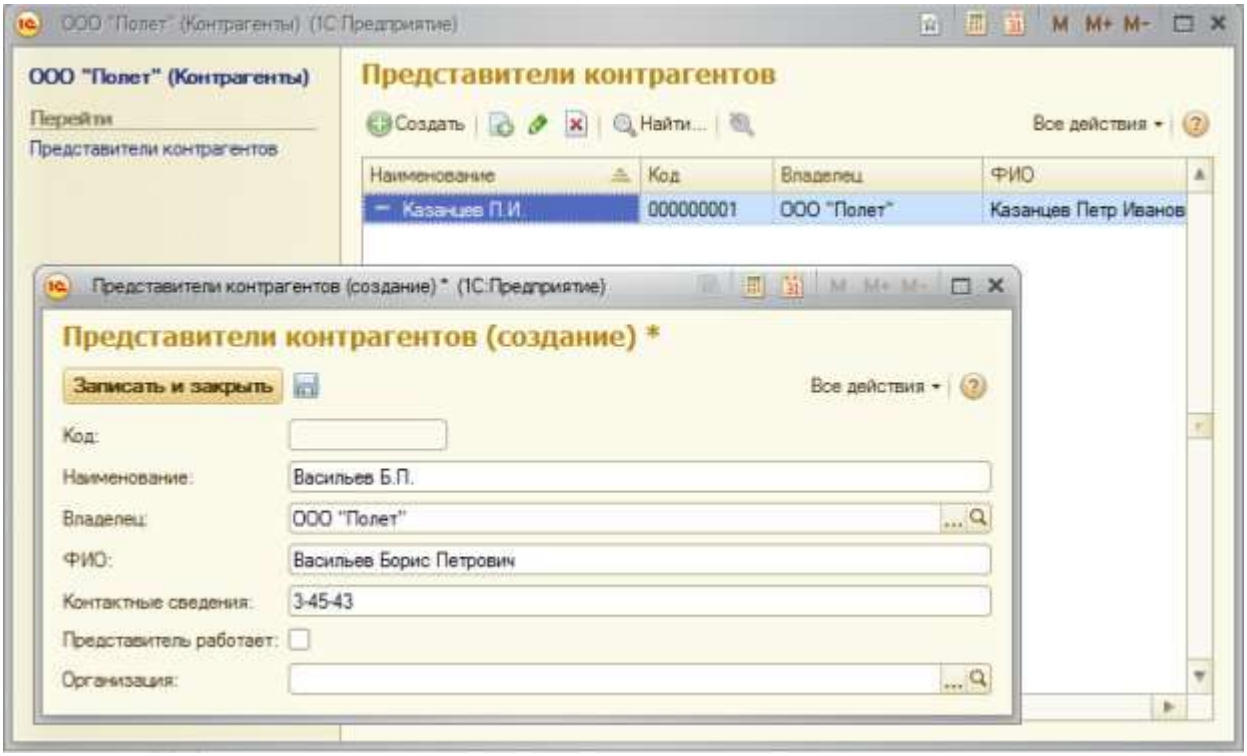


Рисунок 3.14. Форма элемента справочника Контрагенты

При переходе в этот справочник мы будем видеть в открывшемся окне лишь тех представителей, которые относятся к контрагенту, с которым мы в данный момент работаем. При создании новой записи о представителе он автоматически будет "привязываться" к тому контрагенту (*поле* владелец будет заполнено должным образом), из формы элемента которого мы перешли в справочник **ПредставителиКонтрагентов**. В форме списка справочника будет отображаться *ссылка* для перехода к форме элемента справочника-владельца, Рисунок 3.15.

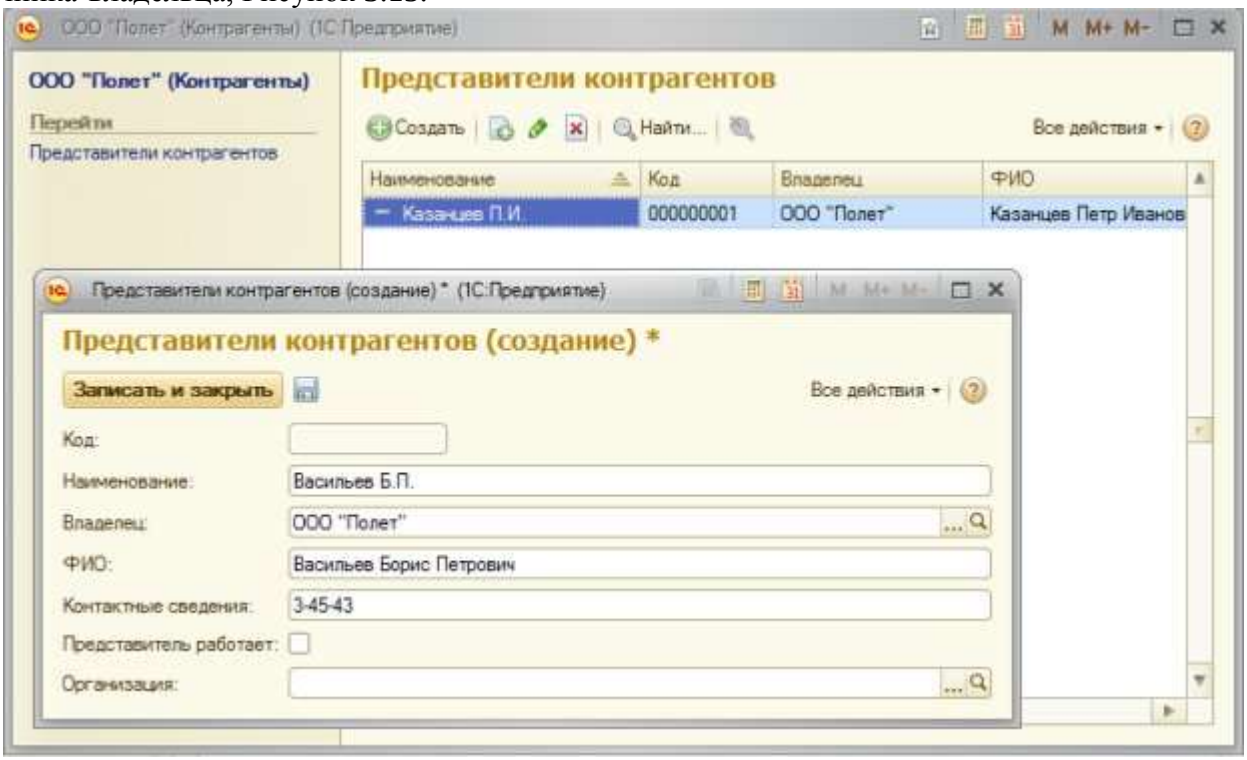


Рисунок 4.15. Формы списка и элемента справочника ПредставителиКонтрагентов

Мы можем создавать элементы справочника **ПредставителиКонтрагентов** и непосредственно перейдя в него, тогда нам придется самостоятельно указывать его владельца – элемент справочника **Контрагенты**. При переходе в *подчиненный справочник* не из формы элемента справочника-владельца, мы можем просматривать все его элементы, Рисунок 4.16.

Рисунок 3.16. Просмотр формы списка справочника ПредставителиКонтрагентов

Перейдем в окно редактирования объекта конфигурации справочника **Контрагенты**, перейдем на его закладку **Формы**, создадим новую форму списка. При работе с конструктором форм можно заметить, что на закладке управления реквизитами присутствуют два элемента – **Дерево** и**Список**. *Список* мы с вами уже видели, а элемент **Дерево** характерен для *иерархических справочников*, он позволяет облегчить навигацию по большим справочникам, выводя их иерархическую структуру в *дополнение* к списку. Установим флаг в *поле* **Дерево**, из списка реквизитов, отображаемых в дереве элементов, выберем **Наименование**, Рисунок 3.17.

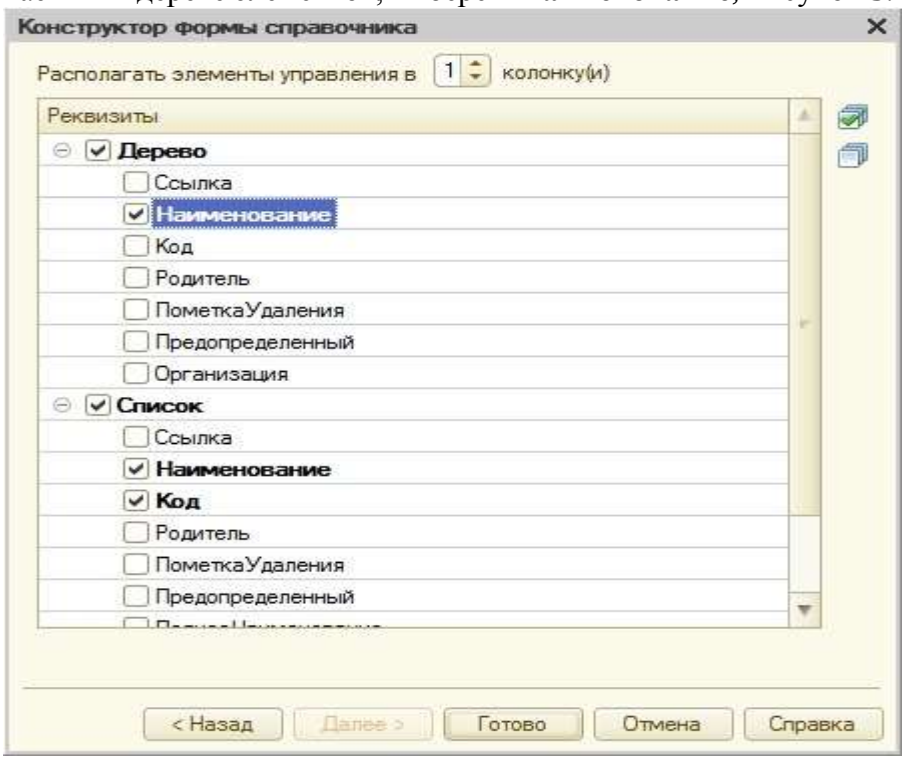


Рисунок 3.17. Конструктор формы справочника Контрагенты

Вот как будет выглядеть форма списка справочника в режиме «1С:Предприятие», 4.18.

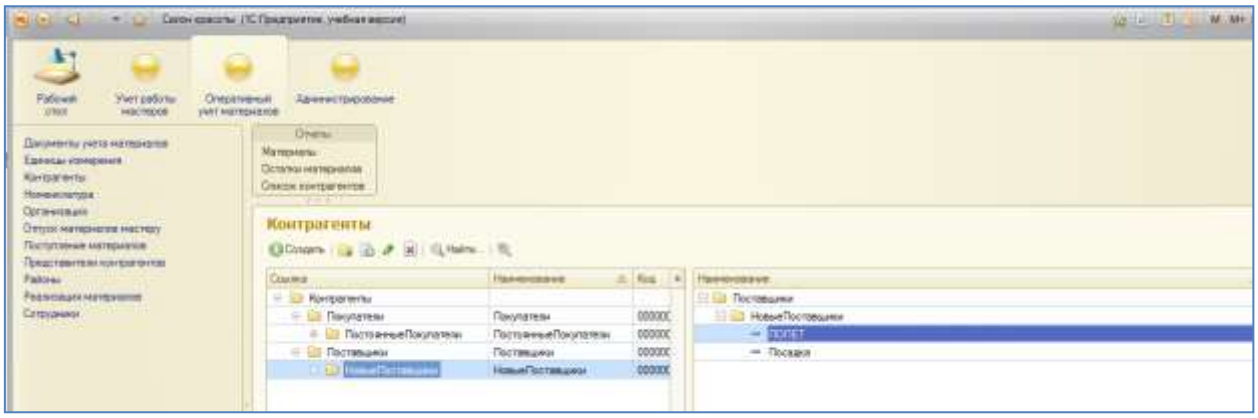


Рисунок 3.18. Форма справочника со списком и деревом элементов

Перед тем, как перейти к дальнейшему рассмотрению материала – заполните справочник «Контрагенты» и подчиненный справочник «Представители контрагентов» тремя-четырьмя записями!!.

Проверка заполнения реквизита справочника, фильтрация

Расширим справочник **Контрагенты**, добавим в состав его реквизитов еще один – назовем его **ОсновноеКонтактноеЛицо**, тип – **СправочникСсылка.ПредставителиКонтрагентов**. Смысл этого поля заключается в хранении ссылки на представителя контрагента, который является "основным" для данного контрагента. Если нужно связаться с контрагентом, можно открыть его карточку и тут же увидеть, какой представитель является основным. Создадим форму элемента справочника **Контрагенты** и посмотрим на нее, попытавшись установить новое *поле* – **Основное контактное лицо**, Рисунок 3.19.

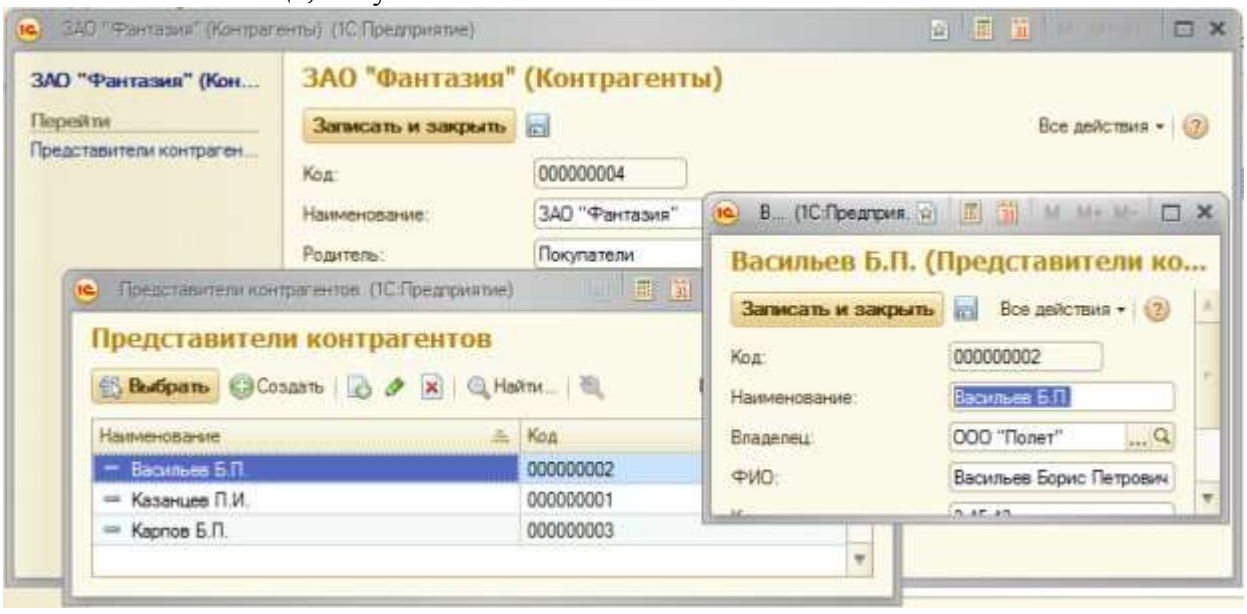


Рисунок 3.19. Попытка заполнения реквизита Основное контактное лицо

Видно, что при попытке подбора элемента в данное *поле* нам показывают не только те элементы справочника**ПредставителиКонтрагентов**, владельцем которых является редактируемый элемент, но и все остальные. Так работать неудобно – это значит, что нам нужно настроить фильтрацию выводимых элементов. Для того, чтобы это сделать,

удобнее всего будет воспользоваться свойством **Связи параметров выбора** реквизита **ОсновноеКонтактноеЛицо**. Для открытия палитры свойств реквизита мы можем сделать *двойной щелчок по* реквизиту в окне редактирования объекта конфигурации, в *дереве конфигурации*, или воспользоваться командой контекстного *меню* Свойства.

В открывшейся палитре свойств найдем свойство **Связи параметров выбора** и нажмем на кнопку с тремя точками около этого поля. Появится окно **Связи параметров выбора**, Рисунок 3.20.

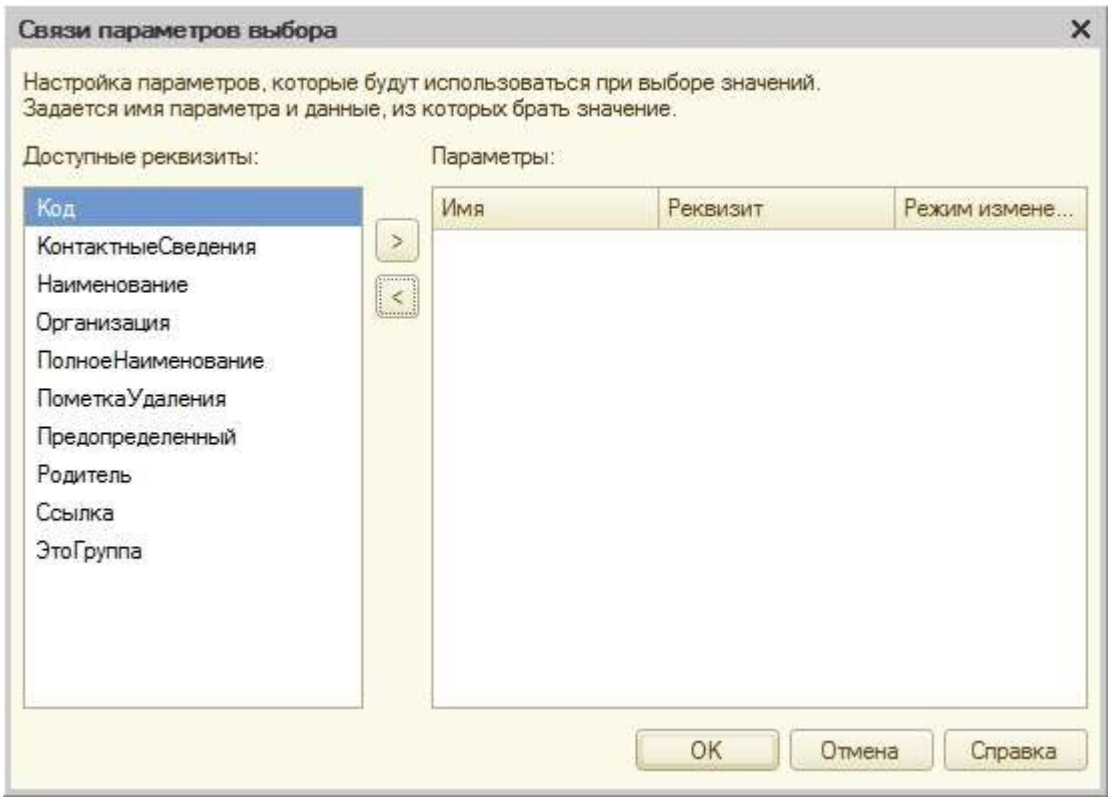


Рисунок 3.20. Окно **Связи параметров выбора**

В левой части окна можно видеть доступные реквизиты (это реквизиты открытого элемента справочника **Контрагенты**), в правом – параметры, влияющие на отбор элементов в появляющемся окне выбора элементов при заполнении поля представителя контрагента. Выделим *реквизит* **Ссылка** и нажмем на кнопку **Добавить выбранный реквизит в параметры выбора** (она находится между полями). Окно примет следующий вид, Рисунок 3.21.

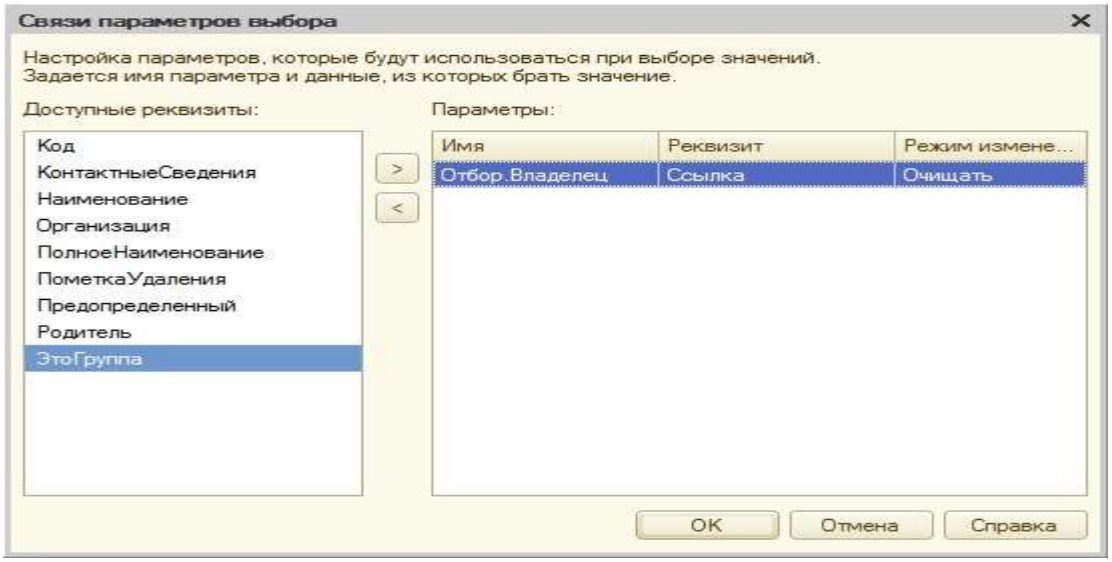


Рисунок 3.21. Окно Связи параметров выбора с настроенным параметром

В данном случае в строке области **Параметры** отображается как раз то, что нам нужно – нам нужно, чтобы отбор в раскрывающемся списке происходил *по* владельцу, а именно – *по* текущему открытому элементу справочника **Контрагенты**, на который и указывает *реквизит* **Ссылка**. В *поле* имя можно выбрать другие варианты отбора, Рисунок 3.22.

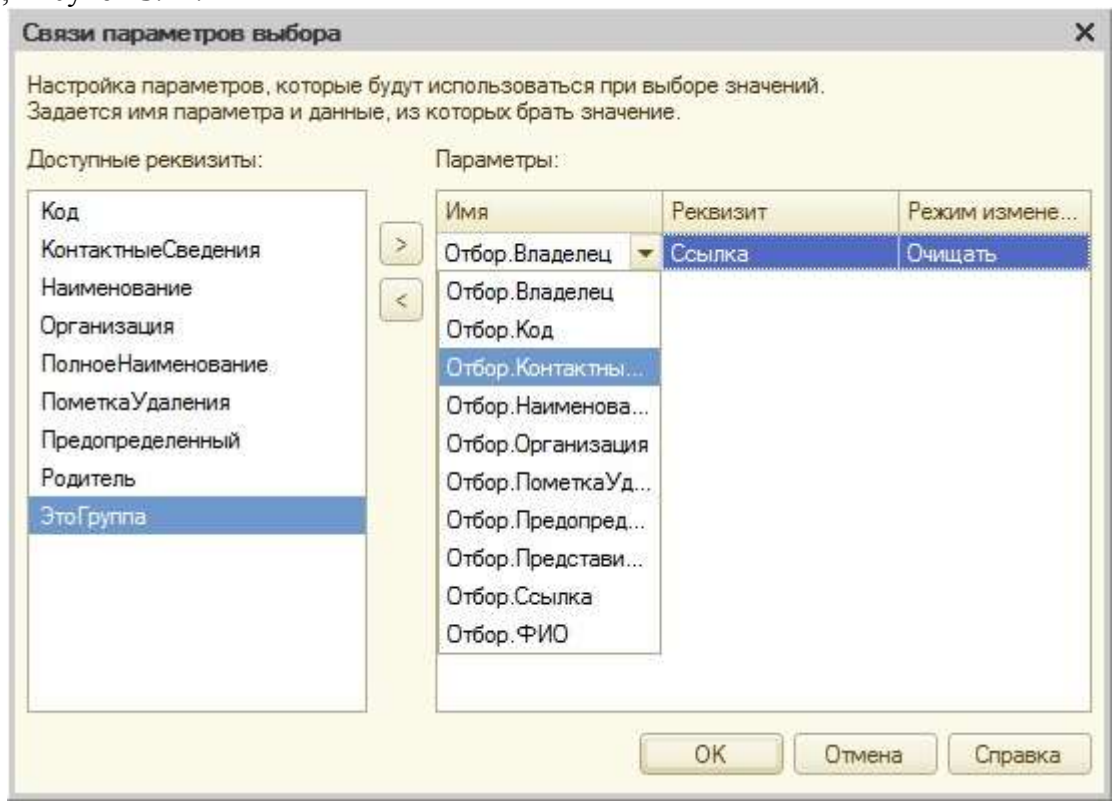


Рисунок 3.22. Настройки в окне Связи параметров выбора

После того, как эта настройка выполнена, мы можем нажать **ОК** в окне **Связи параметров выбора** и проверить функциональность решения – при заполнении поля **ОсновноеКонтактноеЛицо** *список* выбора ограничивается подчиненными элементами.

Литература:

Официальный сайт интернет-униерситета «Интуит» (электронный ресурс), режим доступа http://www.intuit.ru/studies/courses/2318/618/lecture/17939.

Контрольные вопросы для самопроверки:

1. Как связать два справочника?
2. Виды справочников в «1С:Предприятие»?
3. Особенности иерархических справочников

Задание к лабораторной работе

Создать справочники в системе в соответствии с порядком, отраженным в теоретической части

Методические указания и порядок выполнения работы

Работа выполняется в точном порядке, как это указано на рисунках и в теоретической части

Индивидуальное задание

не предусмотрено

Требования к отчету и защите

1. Результатом выполнения лабораторной работы является сформированный в программе файл, содержащий выполненные задания. В ЭИОС результаты работы не выкладываются.

2. Планируется защита работы, где студент комментирует порядок выполнения заданий, а также отвечает на вопросы, представленные выше

ЛАБОРАТОРНАЯ РАБОТА №4

РАБОТА С ОТЧЕТАМИ

ОБЩИЕ СВЕДЕНИЯ

Цель: научиться информационную базу на платформе ««1С:Предприятие»»

Материалы, оборудование, программное обеспечение:
1. персональный компьютер (компьютерные классы ГУК)
2. программное обеспечение «1С:Предприятие»

Условия допуска к выполнению:
умение работать на ПК и знание техники безопасности

Критерии положительной оценки:
предоставление результатов работы в виде файла и прохождение защиты.

Планируемое время выполнения:
Аудиторное время выполнения (под руководством преподавателя): 2 ч.
Время самостоятельной подготовки: 2 ч.

Теоретическое введение

Простой отчет

Конечная цель любой учетной системы – формирование отчетов. «1С:Предприятие» 8.2 предоставляет разработчику множество инструментов для *создания отчетов* – от достаточно простых механизмов, позволяющих создавать несложные отчеты, до комплексных средств, таких, как *система компоновки данных*. Сейчас мы рассмотрим пример создания простого отчета. Нам нужно вывести отчет в виде списка контрагентов *по* группам с указанием наименования контрагента, основного контактного лица и телефона этого контактного лица.

Отчет нужного нам вида можно сформировать различными способами. Так, вполне можно реализовать эту функциональность непосредственно в справочнике **Контрагенты**, добавив в него соответствующие программные *механизмы*, выведя кнопку "**Сформировать список контрагентов**" в форму списка справочника. Можно сделать это с помощью специализированного прикладного объекта **Отчет**. Обычно для создания подобного рода отчетов так и поступают.

Отчет, в нашем случае, будет строиться на основе макета, с областями которого работают в программном коде, формируя готовый отчет. Как правило, если речь идет о неких общих отчетах, их создают в виде отдельных объектов, если же, например, нам нужно создать печатную форму для отдельного элемента справочника или отдельного документа – вполне можно "поместить" всю функциональность требуемого отчета внутри прикладного объекта. В частности, прикладные объекты могут иметь в числе подчиненных объектов макеты, которые и используются при создании отчетов. В то же время, внешний отчет вполне можно использовать и для создания *печатной формы* отдельного элемента справочника или документа.

Создадим в ветви *дерева конфигурации* **Отчеты** новый отчет, дадим ему имя **СписокКонтрагентов**, Рисунок 5.1.

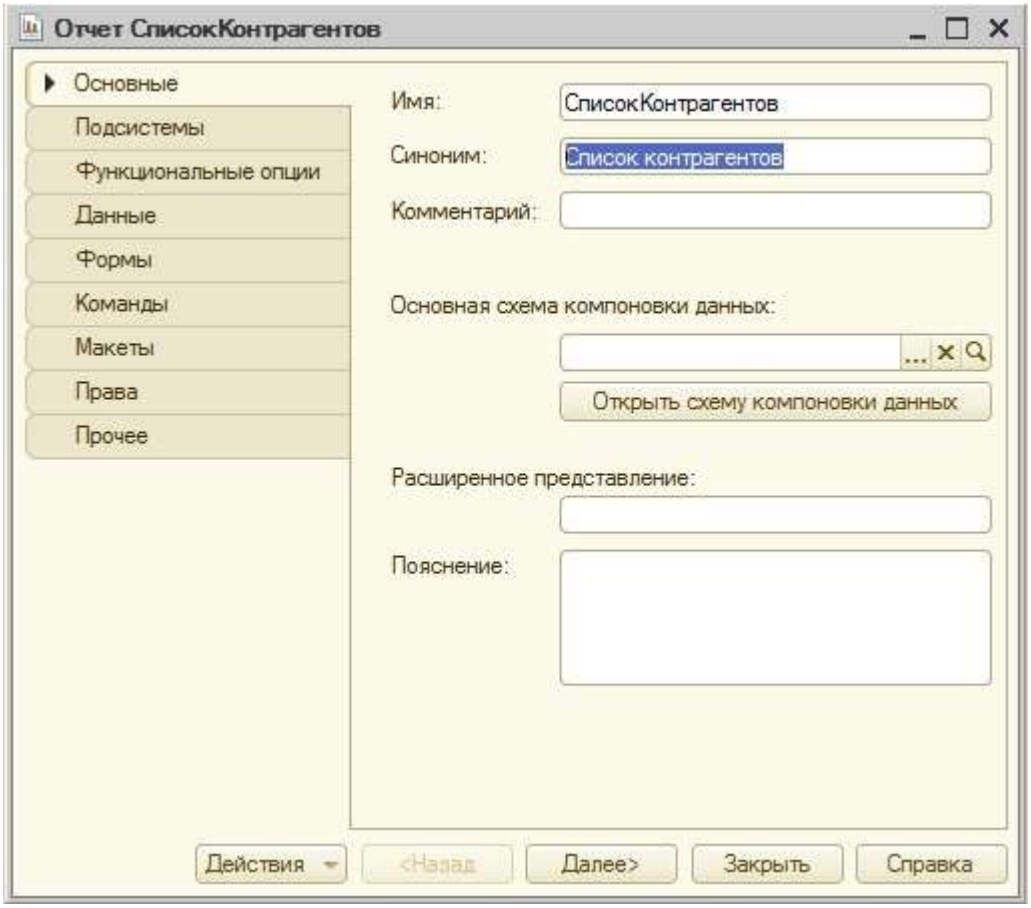


Рисунок 5.1. Создание нового отчета

Первым этапом работы над отчетом станет создание макета отчета. *Макет* позволяет заранее определить и оформить "блоки", из которых будет построен отчет.

Следует отметить, что во всех возможных случаях при разработке *прикладных решений* для «1С:Предприятие» 8.2. следует создавать их на основе схемы компоновки данных. Однако умение работать с макетами в форме табличных документов может пригодиться в том случае, если вам понадобится отредактировать сторонний отчет, выполненный в таком стиле.

Перейдем на закладку формы редактирования объекта **Макеты** и нажмем на кнопку **Добавить**. Появится окно конструктора макета, где нам предложат задать его имя (оставим имя *по* умолчанию – **Макет**), и тип макета – нас устроит **Табличный документ**, Рисунок 5.2.

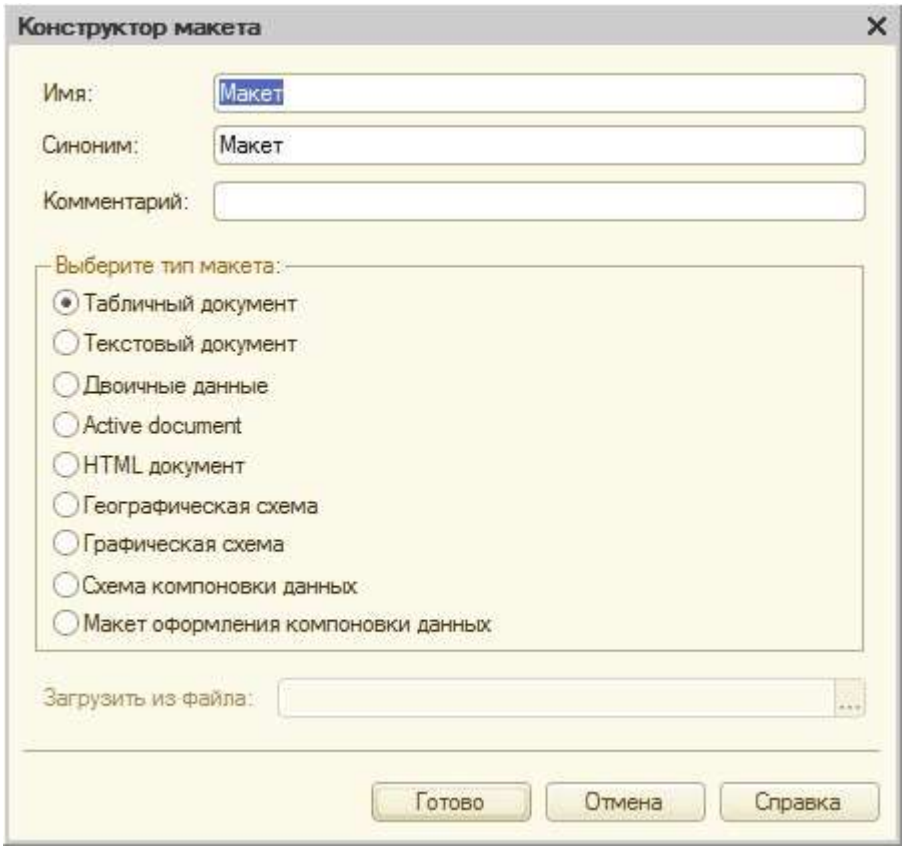


Рисунок 5.2. Создание макета для отчета

После нажатия на кнопку **Готово**, мы видим табличный редактор, Рисунок 5.3., очень напоминающий Microsoft *Excel*. Работая с ним, мы можем пользоваться стандартной палитрой свойств, а так же – панелями инструментов, в частности – **Форматирование**,**Табличный документ**, **Имена**. Наша задача сейчас – создать и отформатировать области, которые позже будут использованы для формирования готового отчета.

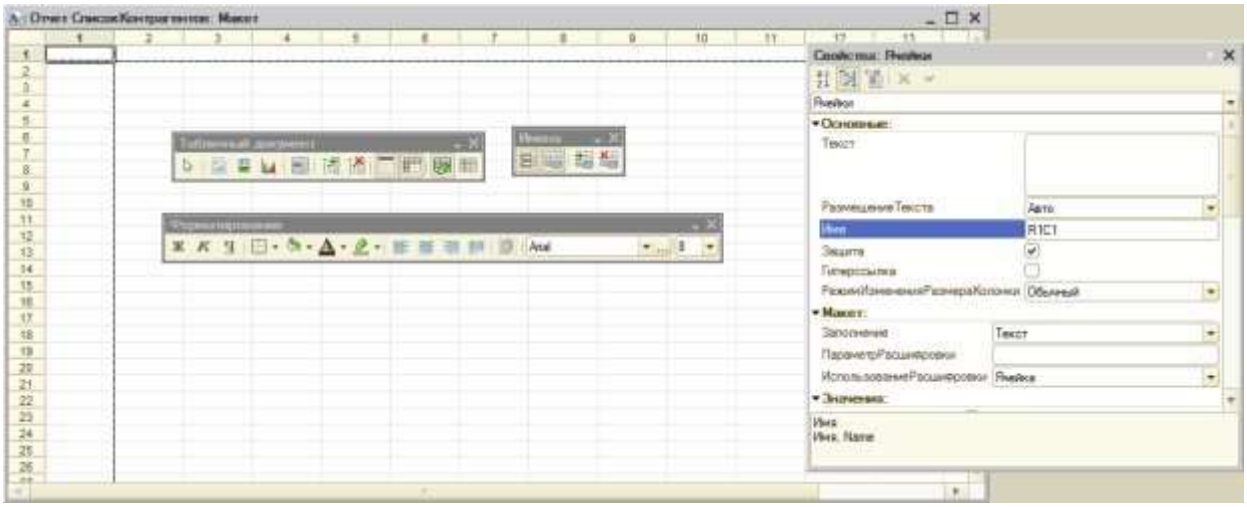


Рисунок 5.3. Средства редактирования макета отчета

При создании макета мы можем вводить в ячейки обычный текст – такой текст отображается в ячейке без каких-либо дополнительных знаков. *Ячейка* может содержать именованный *параметр*, который будет заполнен при формировании отчета. Так же

ячейки могут содержать шаблоны, состоящие из обычного текста и параметров, которые так же можно заполнить.

На Рисунок 5.4. показан готовый *макет*.

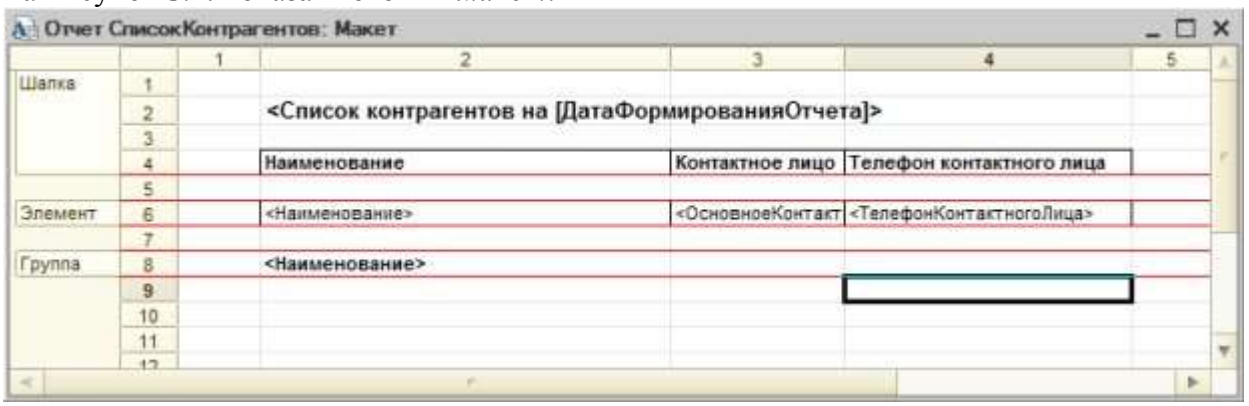


Рисунок 5.4. Готовый макет отчета

Ячейка 2,2 заполнена следующим образом: в нее сначала введен текст **"Список контрагентов на [ДатаФормированияОтчета]"**, после чего вызвано окно свойств этой ячейки, в которых, в свойстве **Заполнение** выбрано**Шаблон**, Рисунок 5.5.

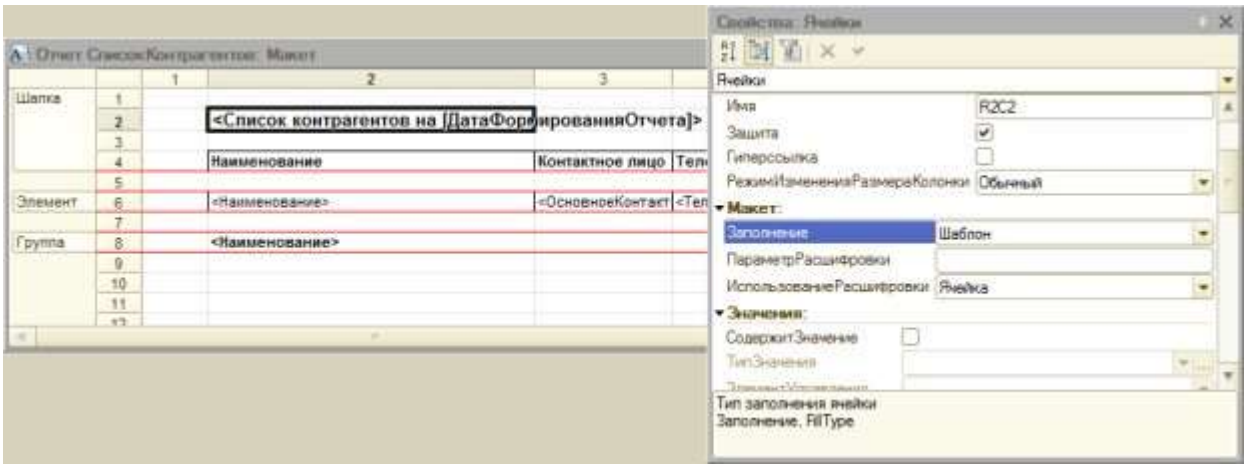


Рисунок 5.5. Настройка ячейки, содержащей шаблон

Параметр **ДатаФормированияОтчета** мы установим в текущую дату программно при формировании отчета.

Ячейки с 4,2 *по* 4,4 содержат обычный текст – он будет выводиться в качестве шапки таблицы.

И заголовок отчета и шапка таблицы объединены в область с именем **Шапка**. Для задания имени области достаточно выделить нужные ячейки (выделять нужно *по* заголовкам строк) и отредактировать в палитре свойств *параметр* **Имя выделенного диапазона**, или воспользоваться кнопкой **Назначить имя** панели инструментов **Имена.**

Область **Элемент** содержит три параметра – **Наименование**, **ОсновноеКонтактноеЛицо** и **ТелефонКонтактногоЛица**. После ввода в каждую из ячеек имен параметров, нужно выделить их (все вместе или *по* одной) и в окне свойств в *поле* **Заполнение**указать **Параметр**. К тексту в ячейках будут автоматически добавлены угловые скобки (<>), что позволяет визуально определить наличие в ячейке параметра.

Область **Группа** содержит лишь *параметр* **Наименование.**

Обратите внимание на имена параметров – они соответствуют именам реквизитов справочника, которыми мы собираемся их заполнять.

Ячейки в шаблоне можно форматировать – задавать их границы, оформление текста, выравнивание и т.д.

Теперь приступим к созданию формы отчета. Перейдем на вкладку **Формы** окна редактирования объекта, добавим новую форму отчета, оставим все настройки в состоянии *по* умолчанию и нажмем **Готово**. Добавим, на вкладке **Реквизиты** редактора форм новый *реквизит*, назовем его **ТабличныйДокумент**, выберем для него тип **ТабличныйДокумент**. Перетащим созданный *реквизит*в *поле* **Элементы**. В состав команд формы добавим новую команду, зададим ей имя **СформироватьОтчет** и так же переместим в*поле* **Элементы**. В итоге у нас получится форма, выглядящая так, как показано на Рисунок 5.6.

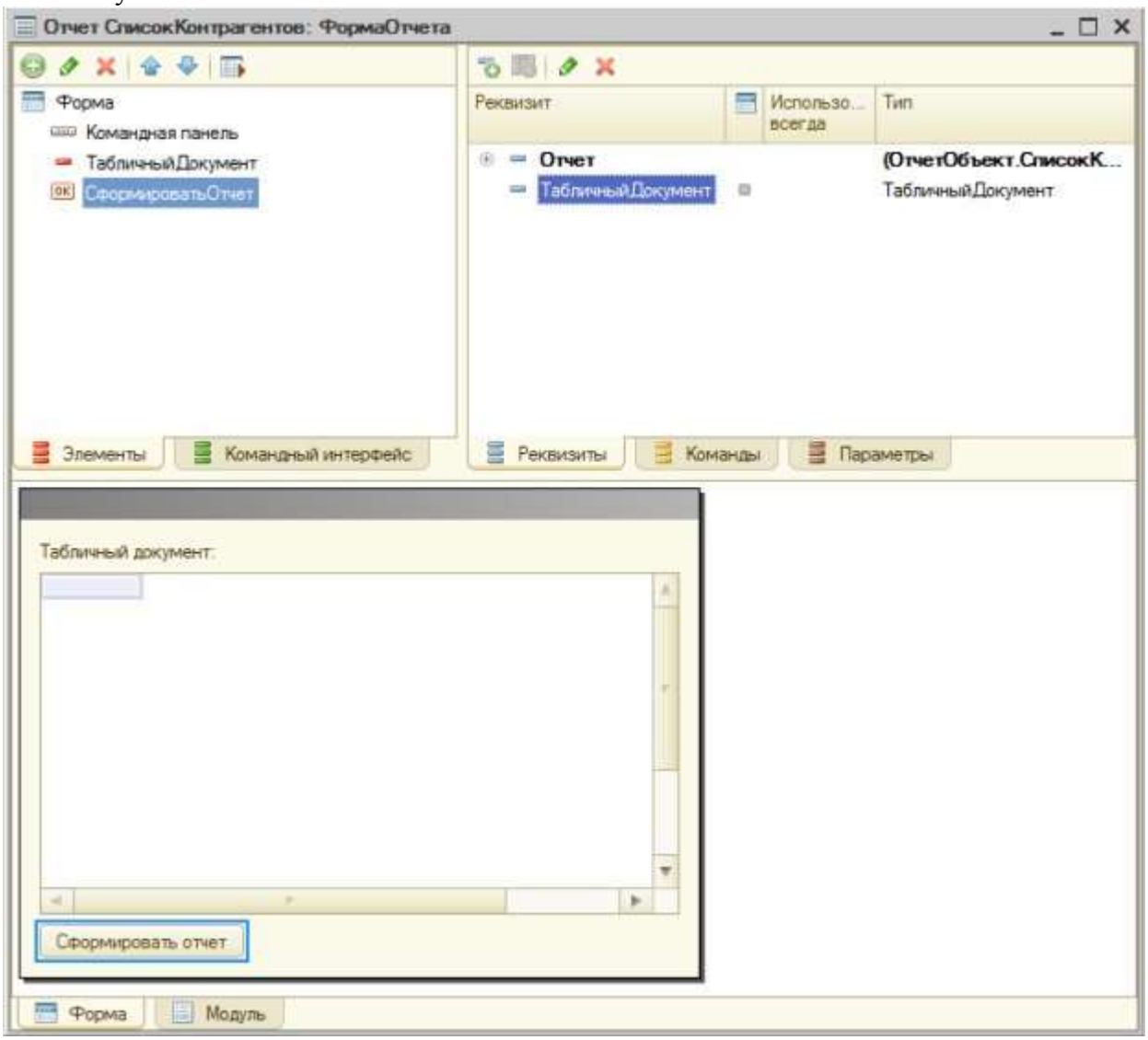


Рисунок 5.6. Настройка формы отчета

Для построения отчета мы должны будем получать данные из справочника. Это можно сделать с помощью уже знакомого вам объекта **СправочникВыборка**, можно получить данные с помощью запроса. В любом случае, это предусматривает работу с базой данных, то есть, нам понадобится процедура, выполняемая на сервере. До настоящего времени мы пользовались лишь директивой компиляции **&НаСервере** – при вызове методов, объявленных с этой директивой, мы имеем *доступ* к контексту формы, при этом между клиентом и сервером происходит передача дополнительных данных – как

при вызове серверного метода с клиента на*сервер*, так и в обратном направлении. Сейчас мы воспользуемся серверным внеконтекстным методом для формирования отчета. В этом методе мы собираемся формировать табличный документ, содержащий данные отчета.

При реализации метода в виде процедуры, нам придется передать в него в качестве параметра наш *реквизит***ТабличныйДокумент**. *По* умолчанию параметры передаются *по* ссылке, то есть, работать процедура будет непосредственно с нашим реквизитом.

При реализации метода в виде функции мы можем ничего не передавать в него, сформировать внутри функции табличный документ и вернуть уже заполненный документ в точку вызова, присвоив его нашему реквизиту **ТабличныйДокумент.**

Реализуем метод в виде функции. Готовый код формирования отчета (Рисунок 5.7) будет выглядеть следующим образом:

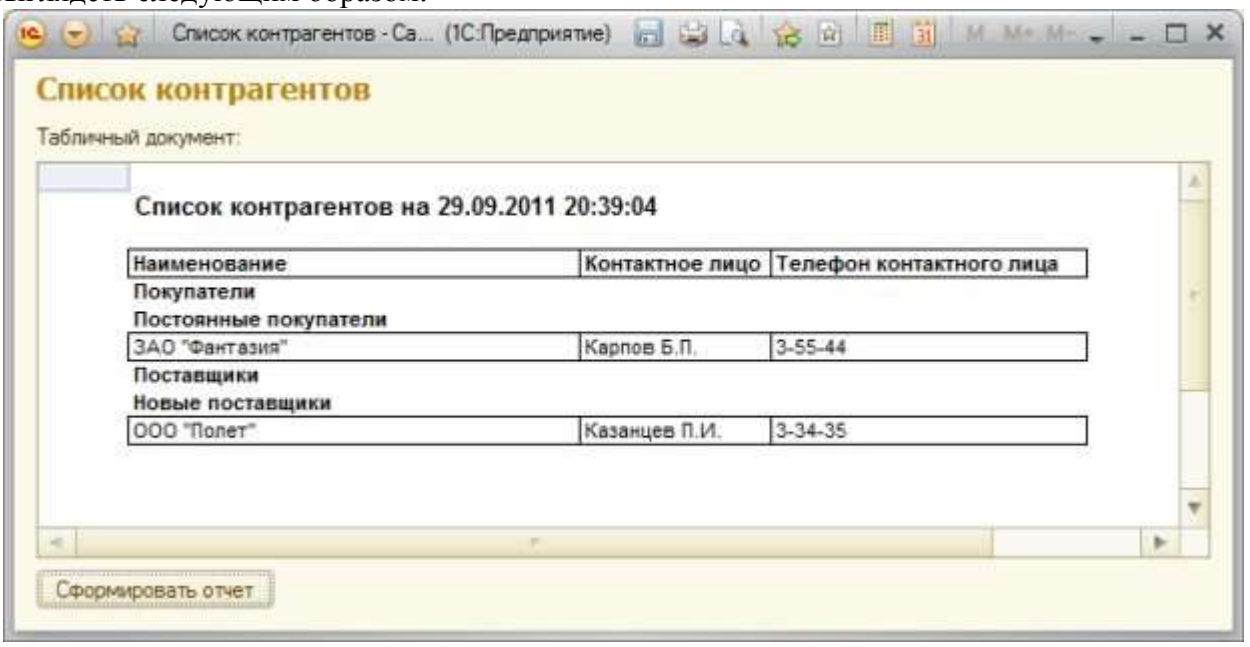


Рисунок 5.7. Готовый отчет

```
&НаКлиенте
Процедура СформироватьОтчет(Команда)
  ТабличныйДокумент=СформироватьОтчетНаСервере();
КонецПроцедуры

&НаСервереБезКонтекста
Функция СформироватьОтчетНаСервере()
  ТабличныйДокумент=Новый ТабличныйДокумент();
  Макет=Отчеты.СписокКонтрагентов.ПолучитьМакет("Макет");
  Шапка=Макет.ПолучитьОбласть("Шапка");
  Элемент=Макет.ПолучитьОбласть("Элемент");
  Группа=Макет.ПолучитьОбласть("Группа");
  Шапка.Параметры.ДатаФормированияОтчета=ТекущаяДата();
  ТабличныйДокумент.Вывести(Шапка);
  Выборка=Справочники.Контрагенты.ВыбратьИерархически();
  Пока Выборка.Следующий() Цикл
    Если Выборка.ЭтоГруппа Тогда
      Область=Группа;
```

```
    Иначе
      Область=Элемент;
    КонецЕсли;
    Область.Параметры.Заполнить(Выборка);
    ТабличныйДокумент.Вывести(Область);
  КонецЦикла;
  Возврат (ТабличныйДокумент);
КонецФункции
```

В клиентской процедуре **СформироватьОтчет()** мы вызываем серверную внеконтекстную функцию**СформироватьОтчетНаСервере()**, присваивая возвращаемое ей *значение* реквизиту **ТабличныйДокумент.**

В серверной функции мы создаем новую переменную с типом **ТабличныйДокумент** и именем **ТабличныйДокумент**. Именно в него мы будем выводить данные и именно его будем возвращать в точку вызова. Несмотря на то, что имя реквизита формы и имя данной переменной совпадают, между ними нет никакой связи. Это – отдельные объекты.

Далее, мы получаем *макет* из нашего отчета, пользуясь методом **ПолучитьМакет()** и задавая имя макета. Отчет может иметь несколько макетов, их выбор осуществляется *по* имени.

В переменную **Шапка** мы записываем область макета **Шапка**, соответственно поступаем с переменными **Элемент** и **Группа.**

Командой **Шапка.Параметры.ДатаФормированияОтчета=ТекущаяДата();** мы заполняем ранее заданный в макете *параметр***ДатаФормированияОтчета**, записав в него текущую дату. Дата, возвращаемая функцией **ТекущаяДата**, содержит, помимо года, месяца и дня, так же часы, минуты и секунды. При необходимости дату перед выводом можно отформатировать при помощи функции **Формат().**

После того, как *параметр*, находящийся в шапке, заполнен, мы можем вывести шапку в табличный документ командой**ТабличныйДокумент.Вывести(Шапка);**

Следующим этапом нашей работы будет получение иерархической выборки справочника. Такая *выборка* позволяет получить элементы и группы справочника в иерархическом порядке, учитывая родительские отношения между элементами.

В цикле обхода выборки мы сначала проверяем, является ли текущий элемент справочника группой. Если является, присваиваем переменной **Область** ранее полученную область отчета **Группа**. Если не является – присваиваем ей область **Элемент.**

Благодаря совпадению имен параметров и имен реквизитов справочника, для помещения данных из выборки в область макета, достаточно воспользоваться конструкцией **Область.Параметры.Заполнить(Выборка);**. После заполнения параметров мы выводим сформированную область в табличный документ.

Когда цикл перебора выборки будет завершен, мы возвращаем сформированный табличный документ в точку вызова и*пользователь* видит готовый отчет.

Для формирования простейших отчетов, *пользователь* может воспользоваться стандартной функциональностью, присутствующей в «1С:Предприятие» 8. Для этого, открыв, например, *список* справочника, он может выполнить команду **Все действия > Вывести список**. Появится окно **Вывести список**, Рисунок 5.8, где в *поле* **Выводить в** можно выбрать либо **Табличный документ** (его обычно и используют), либо – **Текстовый документ.**

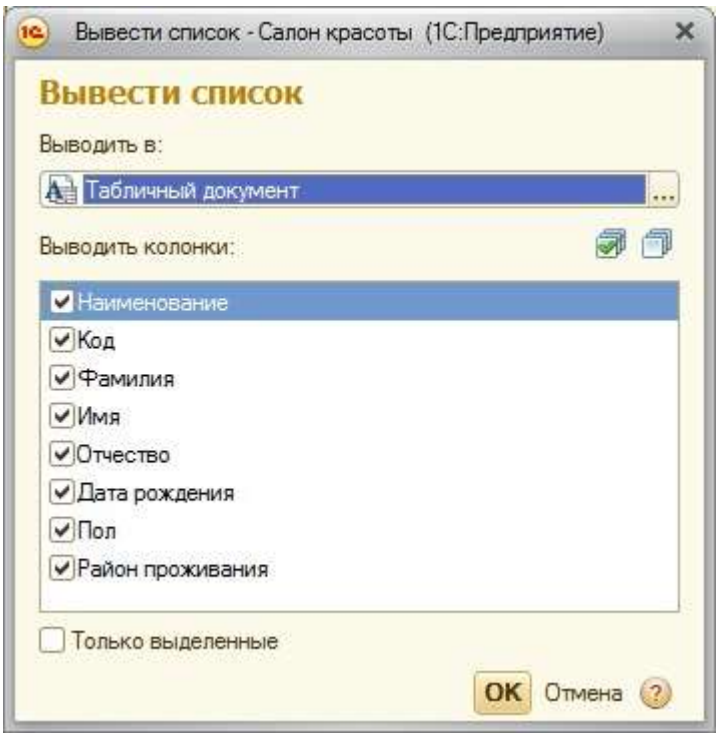


Рисунок 5.8. Окно Настройка списка

В *поле* **Выводить колонки** можно настроить состав выводимых в документ колонок (в нашем случае *команда* выполнена для справочника **ФизическиеЛица**). После нажатия на **ОК** выбранные данные оформляются в виде табличного документа, а с помощью команды **Файл > Сохранить как**, Рисунок 5.9., этот документ можно сохранить в нужном формате для дальнейшей обработки в других приложениях.

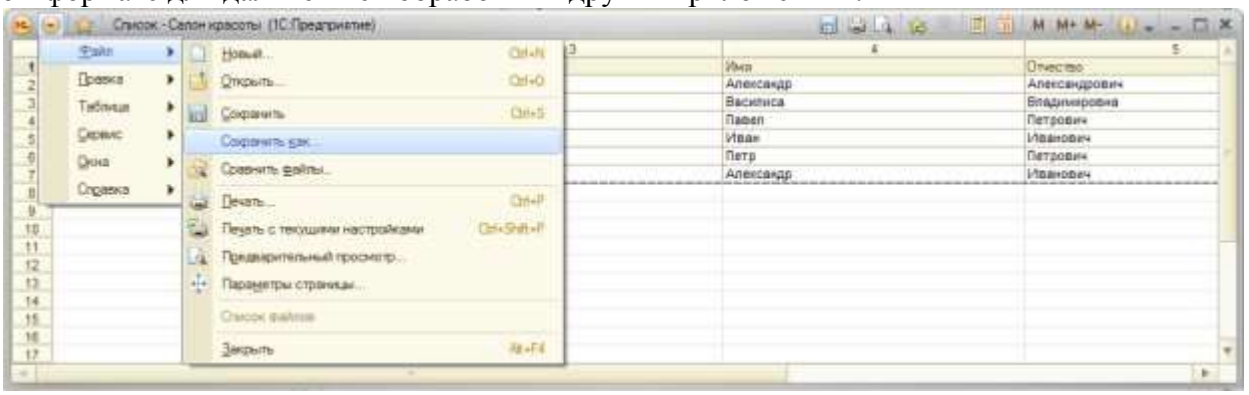


Рисунок 5.9. Вывод данных в табличный документ

Справочник с иерархией элементов

Добавим в нашу конфигурацию еще один справочник. Дадим ему имя **Подразделения**, добавим в состав подсистем **БухгалтерскийУчет**, **УчетРаботыМастеров** и **РасчетЗаработнойПлаты**. Увеличим длину наименования на закладке **Данные**до 100 символов. Сделаем справочник иерархическим – на закладке **Иерархия** установим флаг **Иерархический справочник**,*параметр* **Вид иерархии** установим в *значение* **Иерархия элементов**, Рисунок 5.10.

Иерархия элементов вполне логична для справочника **Подразделения**, так как одни *подразделения* могу включать в себя другие, и, при этом, вполне самостоятельны, их

можно выбирать при заполнении, например, реквизитов других справочников, в то время, как при иерархии групп и элементов, группы играют лишь вспомогательную роль для организации информации внутри справочника.

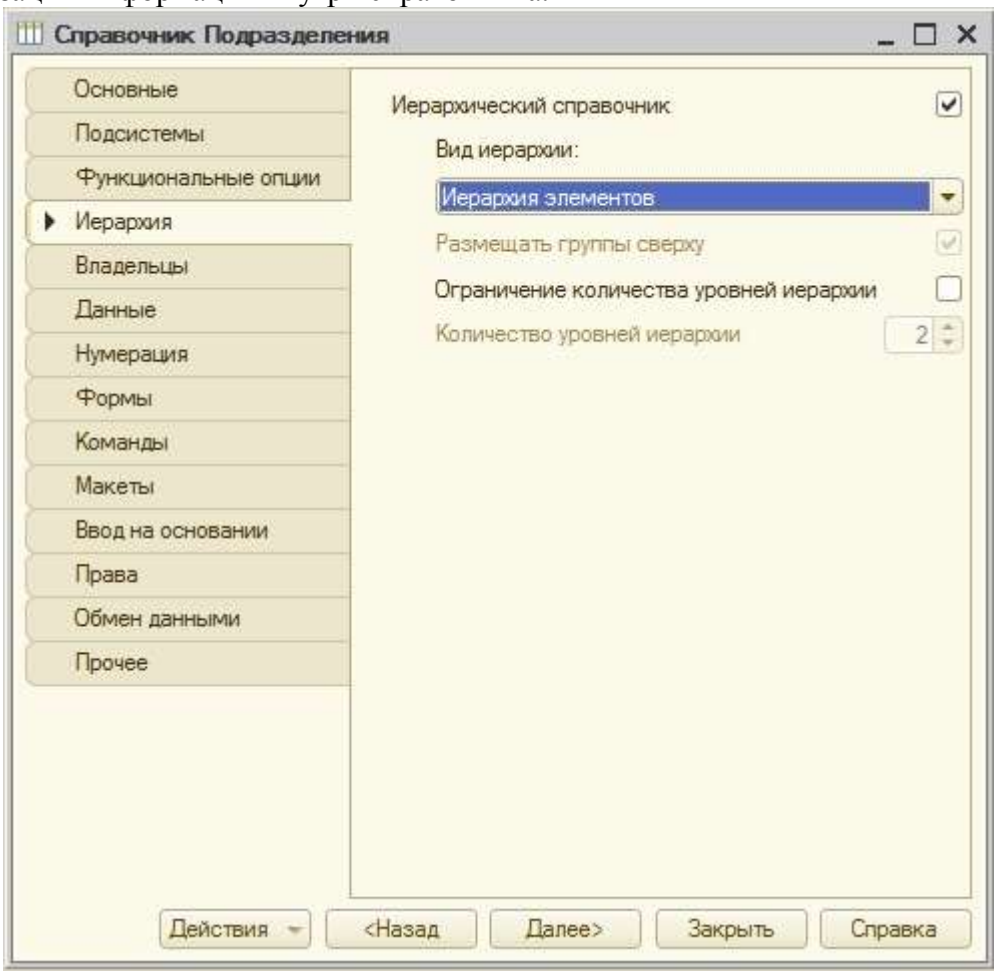


Рисунок 5.10. Настройка иерархии справочника Подразделения

Кроме того, в справочник **Подразделения** мы добавим несколько предопределенных элементов. Эти элементы справочника задаются в **Конфигураторе**, *пользователь* обладает лишь ограниченными возможностями *по* управлению ими, в частности, не может их удалить. Такие элементы обычно создают для того, чтобы ими можно было удобно и надежно оперировать в программном коде, не опасаясь того, что *пользователь* удалит их.

Для этого перейдем на вкладку окна редактирования объекта **Прочее** и нажмем на вкладку **Предопределенные**. В окне ввода предопределенных элементов справочника введем следующие (Рисунок 5.11.):

- Администрация
- Бухгалтерия
- Парикмахерская

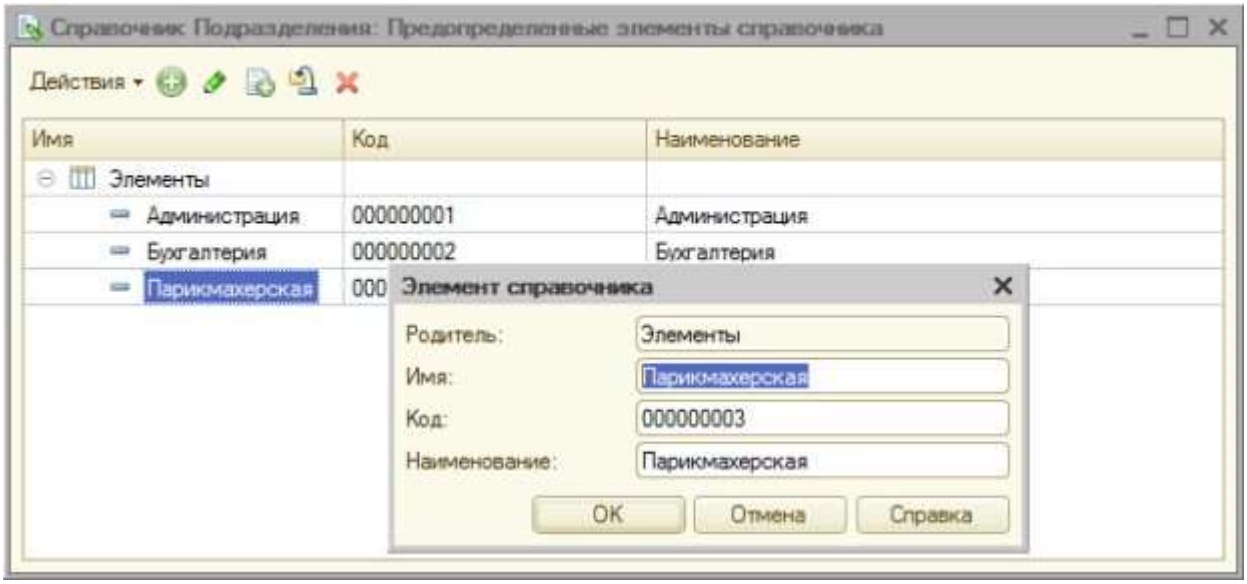


Рисунок 5.11. Создание предопределенных элементов справочника Подразделения

Доработаем справочник **Сотрудники**. Снабдим его следующими реквизитами, Рисунок 5.12.:

Имя: ФизическоеЛицо, **Тип**: СправочникСсылка.ФизическиеЛица

Имя: Подразделение, **Тип**: СправочникСсылка.*Подразделения*

Имя: Расчетчик, **Тип**: Булево

Имя: *Пользователь*, **Тип**: Строка, *длина* 50.

Увеличим длину **наименования** до 50 символов.

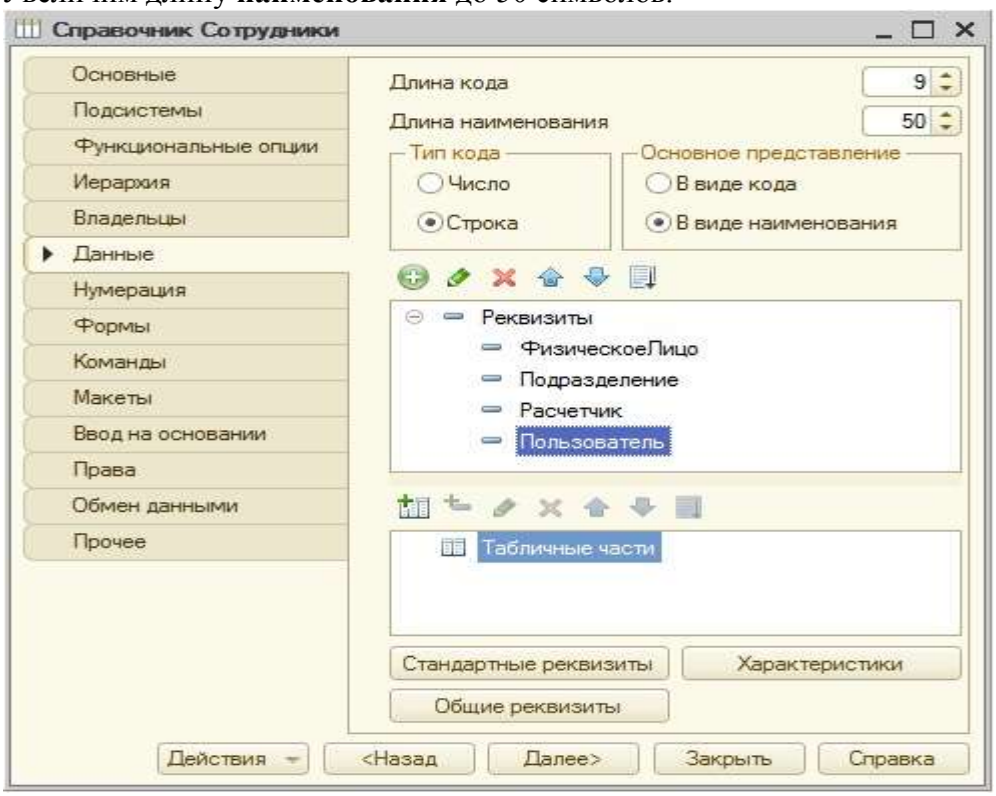


Рисунок 5.12. Реквизиты справочника Сотрудники

Мы хотели бы, чтобы наименование сотрудника в данном справочнике формировалось бы автоматически и состояло бы из ФИО физического лица

и *подразделения*, в котором работает сотрудник. Создадим форму элемента справочника и, для элемента формы **Наименование**, снимем флаг **Доступность,** Рисунок 5.13.

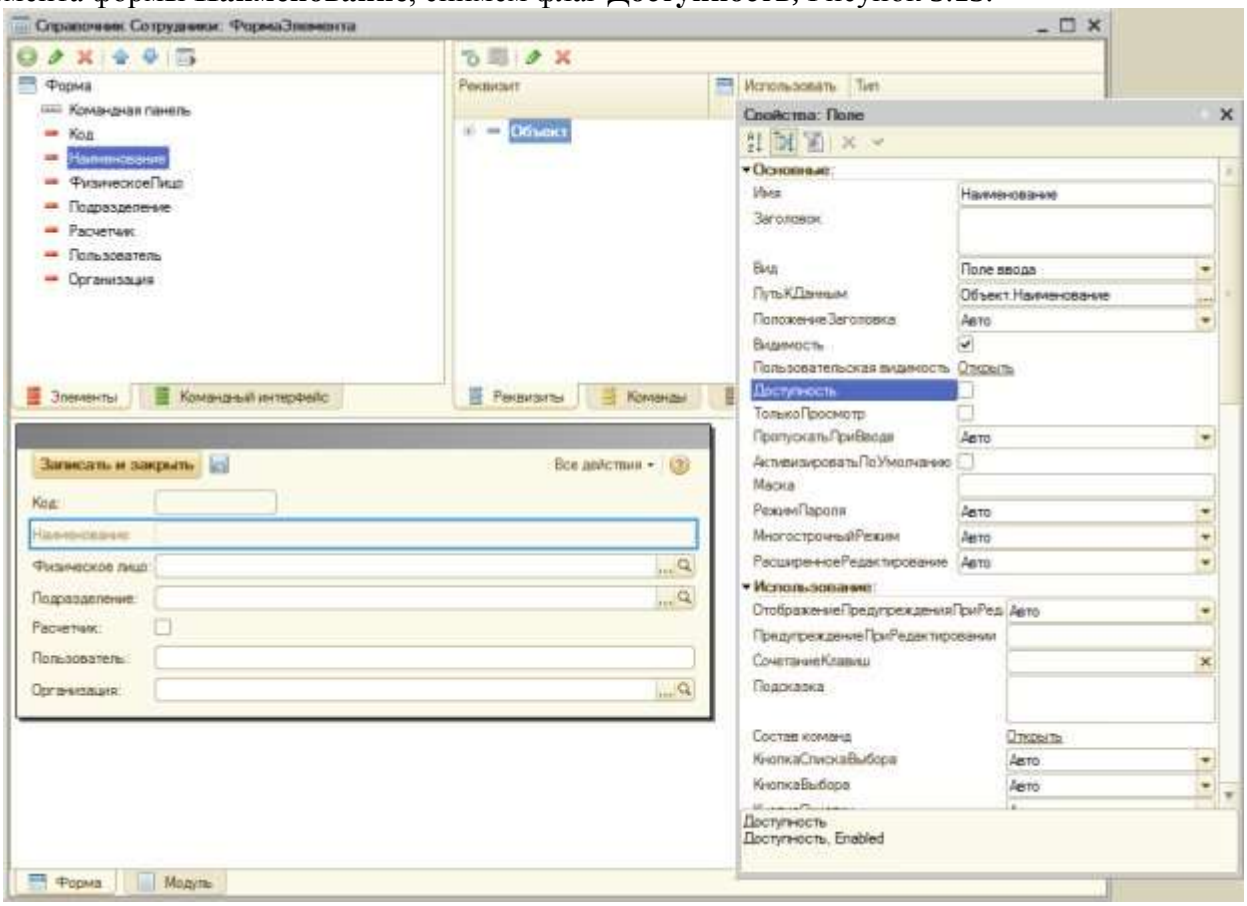


Рисунок 5.13. Настройка формы элемента справочника Сотрудники

Теперь подумаем над тем, как автоматически заполнить *поле* **Наименование** на основе данных полей **Физическое лицо** и**Подразделение**. Сделать это можно различными способами, мы реализуем следующую функциональность: перехватим события изменения полей **Физическое лицо** и **Подразделение** и вызовем в обработчике каждого из этих событий процедуру, заполняющую *поле* **Наименование**. Так *пользователь*, заполняющий элемент справочника, сможет сразу же увидеть результаты формирования наименования.

Нашей задаче отвечает следующий код:

```
&НаКлиенте
Процедура ФизическоеЛицоПриИзменении(Элемент)
  СформироватьНаименование();
КонецПроцедуры

&НаКлиенте
Процедура ПодразделениеПриИзменении(Элемент)
  СформироватьНаименование();
КонецПроцедуры

Процедура СформироватьНаименование()
  Объект.Наименование=Объект.ФизическоеЛицо.Наименование +" ("+
  Объект.Подразделение.Наименование+") ";
КонецПроцедуры
```

Результаты работы созданного нами механизма показаны на Рисунок 5.14.

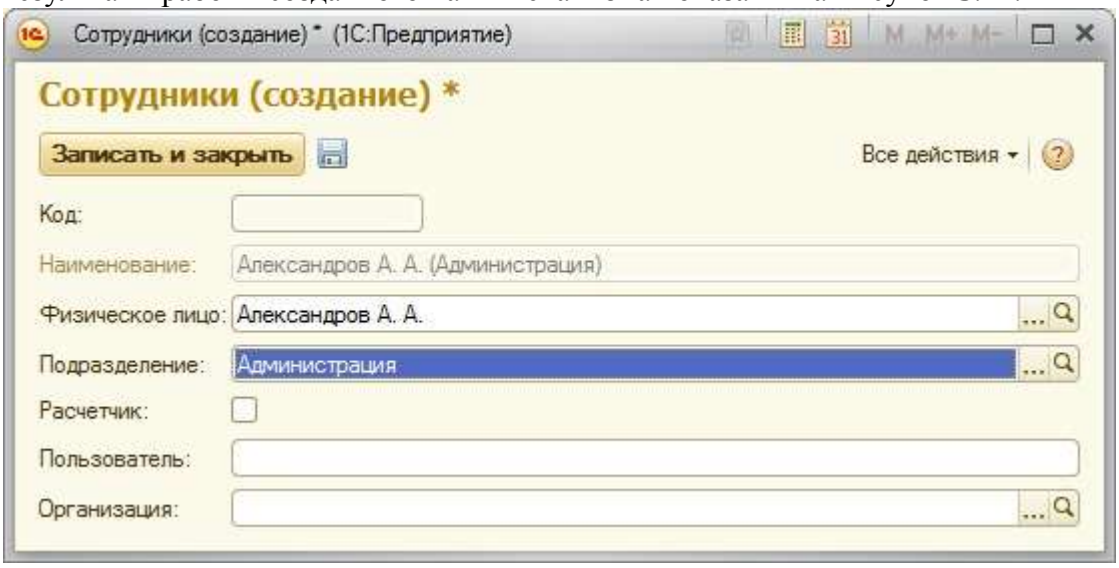


Рисунок 5.14. Настройка формы элемента справочника Сотрудники

Литература:

Официальный сайт интернет-униерситета «Интуит» (электронный ресурс), режим доступа http://www.intuit.ru/studies/courses/2318/618/lecture/17939.

Контрольные вопросы для самопроверки:

4. Для чего нужны отчеты в системе?
5. Виды отчетов в «1С:Предприятие»?
6. Как настроить формы справочников и отчетов?

Задание к лабораторной работе

Создать отчеты в системе в соответствии с порядком, отраженным в теоретической части

Методические указания и порядок выполнения работы

Работа выполняется в точном порядке, как это указано на рисунках и в теоретической части

Индивидуальное задание

не предусмотрено

Требования к отчету и защите

1. Результатом выполнения лабораторной работы является сформированный в программе файл, содержащий выполненные задания. В ЭИОС результаты работы не выкладываются.

2. Планируется защита работы, где студент комментирует порядок выполнения заданий, а также отвечает на вопросы, представленные выше

ЛАБОРАТОРНАЯ РАБОТА № 5

Документы и регистры

ОБЩИЕ СВЕДЕНИЯ

Цель: научиться создавать документы и регистры на платформе ««1С:Предприятие»»

Материалы, оборудование, программное обеспечение:
1. персональный компьютер (компьютерные классы ГУК)
2. программное обеспечение «1С:Предприятие»

Условия допуска к выполнению:
умение работать на ПК и знание техники безопасности

Критерии положительной оценки:
предоставление результатов работы в виде файла и прохождение защиты.

Планируемое время выполнения:
Аудиторное время выполнения (под руководством преподавателя): 2 ч.
Время самостоятельной подготовки: 2 ч.

Теоретическое введение

Любая учетная система получает исходные данные из документов. В классическом бухгалтерском учете основа всего – первичн ые документы, автоматизированные системы учета – не *исключение*.

Для описания документов в *дереве конфигурации* имеется отдельная *ветвь* – **Документы**. В одной из предыдущих лекций мы создали один документ – **ПоступлениеМатериалов**. Сейчас мы займемся работой с ним. Для начала определимся с целью использования этого документа. Мы планируем с его помощью отражать в системе поступление материалов. Исходя из этих целей, нам понадобятся следующие реквизиты документа (Рисунок 6.1.), которые мы зададим на вкладке **Данные** окна редактирования объекта:

Имя: *Контрагент*, Тип: СправочникСсылка.*Контрагенты*

Имя: **ОтветственныйСотрудник**: Тип: СправочникСсылка.Сотрудники

Добавим в состав табличных частей нашего документа новую табличную часть с именем **Материалы** и следующими реквизитами:

Имя: Номенклатура, Тип: СправочникСсылка.Номенклатура

Имя: Цена, Тип: Число, *длина* 10, *точность* 2

Имя: Количество, Тип: Число, *длина* 10, *точность* 3

Имя: Сумма, Тип: Число, *длина* 10, *точность* 2

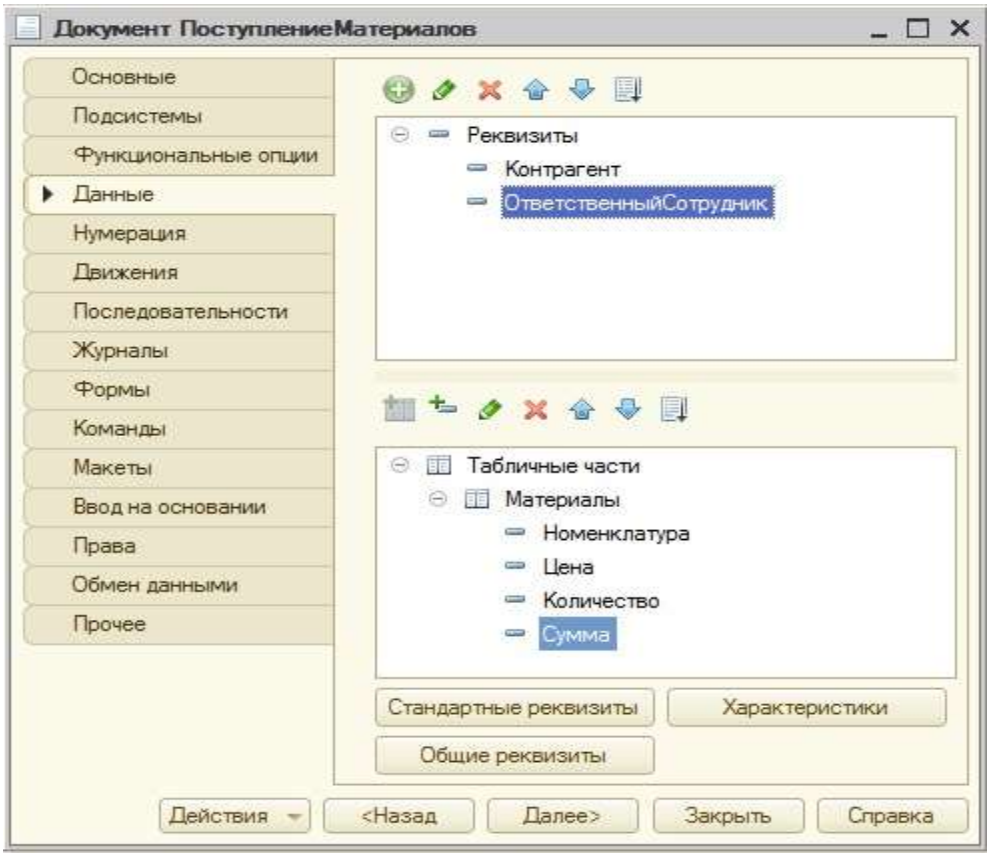


Рисунок 6.1. Настройка состава реквизитов документа

На закладке **Нумерация**, Рисунок 6.2., можно задать параметры нумерации документов.

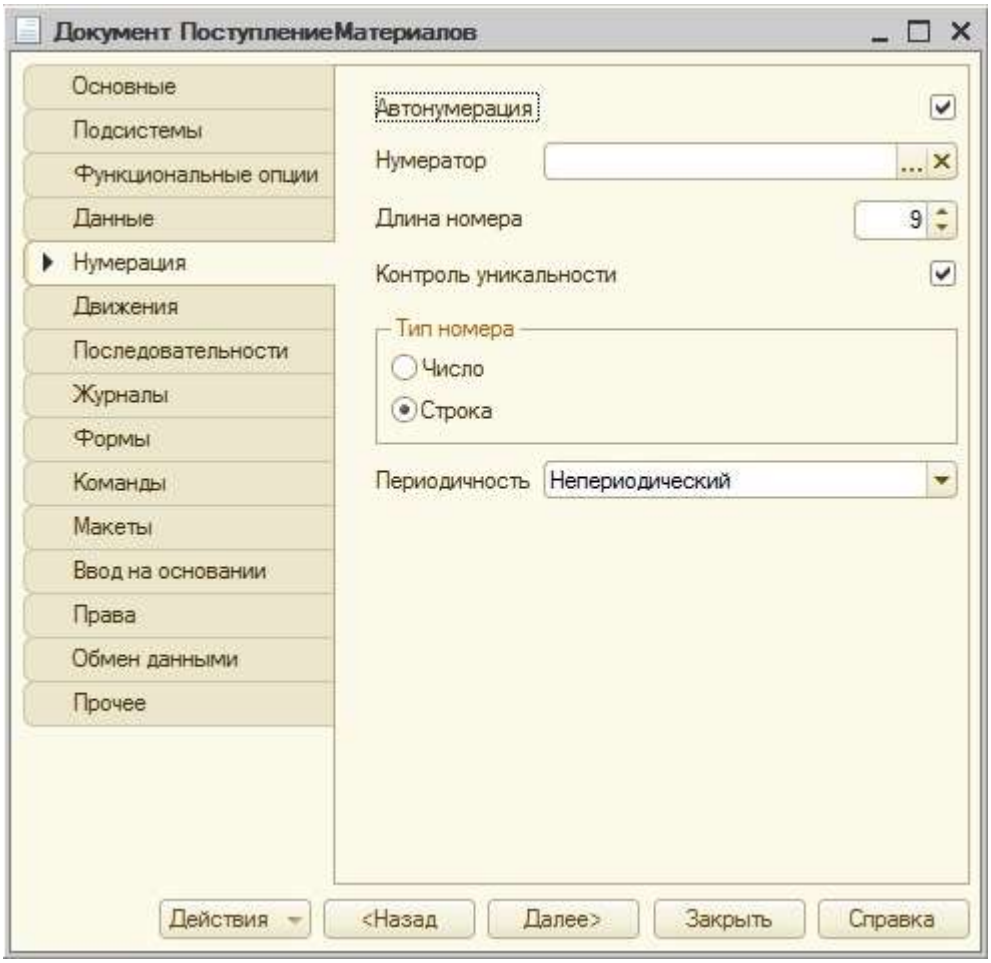


Рисунок 6.2. Настройка параметров нумерации документа

В данном случае документы будут нумероваться автоматически с контролем уникальности номеров.

Нескольким различным видам документов можно назначить сквозную нумерацию с одинаковыми настройками благодаря использования нумератора.

Если в качестве типа номера использована строка – это позволит, при возникновении необходимости, добавлять к номеру различные символьные префиксы.

Настройка периодичности в данном случае установлена в *значение* **Непериодический**, то есть, независимо от срока работы с информационной базой, *нумерация* документов будет продолжаться, а при установке некоторой периодичности (в пределах дня, месяца, квартала, года) – уникальность номеров будет соблюдаться в течение указанного периода.

Закладка **Движения,** Рисунок 6.3., позволяет управлять проведением документа.

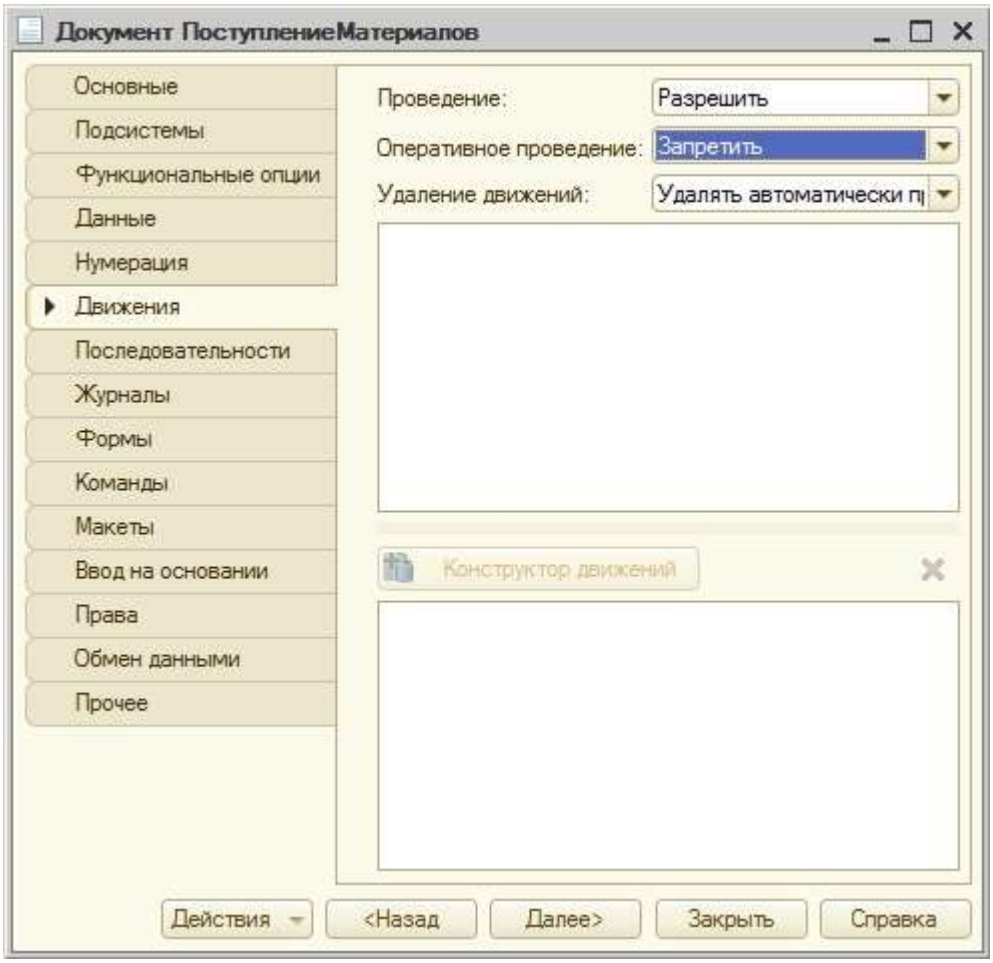


Рисунок 6.3. Настройка параметров проведения документа

Если документ планируется проводить, то есть – он будет формировать движения *по* регистрам, проведение следует разрешить.**Оперативное проведение** позволяет разработчику настроить различное реагирование кода, отвечающего за *проведение документа*, при *проведении документа*, скажем, более поздней или ранней датой, чем текущая дата.

Если мы не планируем реализацию подобной функциональности, мы можем отключить оперативное проведение.

Автоматическое удаление движений документа предусматривает, при отмене проведения, автоматически удалять движения, которые документ сформировал *по* регистрам. В данный момент в нашей конфигурации пока нет регистров, *по* которым будет проводиться документ. "Формирование движений *по* регистру", более простым языком, означает то, что документ при проведении делает записи в регистре. Регистры можно сравнить с таблицами, содержащими ключевые данные о документах, которые *по* ним проводятся. Работу с различными регистрами мы рассмотрим ниже.

Вкладка **Журналы** позволяет настраивать включение документа в так называемые **журналы документов**. Журнал позволяет организовать совместную работу с документами различных типов, которые включены в этот журнал. Такие документы отображаются в едином списке.

Если *проведение документа* запрещено – то *пользователь* сможет лишь сохранить документ в базе данных. Других воздействий на информационную базу такой документ не произведет. Например, такое поведение может быть характерно для документов, вроде выписанных счетов, которые сами *по* себе воздействия на учет не производят, но их важно

хранить в системе для того, чтобы "помнить" о том, какие счета выписаны, важно иметь возможность формировать их *печатные формы*.

Но то, что счет выписан, еще не гарантирует то, что счет будет оплачен, то, что товары, указанные в выписанном счете будут действительно отгружены покупателю. Если продолжить пример с выписанным счетом и перейти на вкладку **Ввод на основании**,Рисунок 6.4., то окажется, что эта вкладка позволяет настроить ввод одного документа на основании другого.

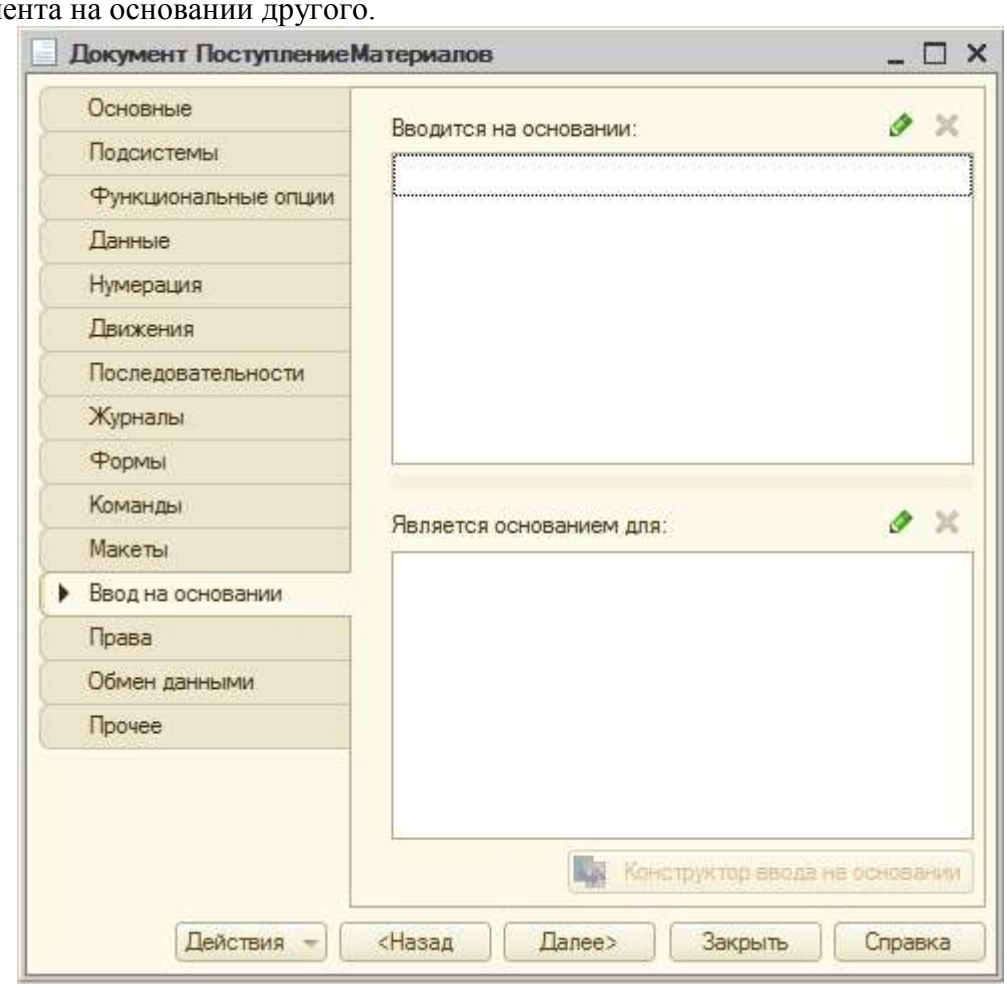


Рисунок 6.4. Настройка параметров ввода на основании

Если, например, мы имеем дело с документом наподобие "**Отгрузка материалов**", окажется, что такой документ вполне логично будет вводить на основании документа "**Счет**" - после оплаты этого счета и фактической отгрузки материалов. Документ отгрузки, в отличие от счета, фиксирует уже свершившийся *факт хозяйственной жизни*, который должен оказать воздействие на состояние информационной базы. Такой документ должен проводиться – то есть – делать записи в соответствующие регистры.

На данном этапе мы можем запустить систему, попытаться поработать с документом, используя автоматически сгенерированную форму, и посмотреть, все ли в данной форме нас устраивает.

Прежде чем продолжать работу с документом **ПоступлениеМатериалов**, приведите данные справочника **Номенклатура** в вашей *информационной базе* к виду, показанному в таблице 6.1.

Таблица 6.1. Данные справочника Номенклатура

Наименование	**Единица измерения**	**Услуга**	**Группа**
Парикмахерские услуги		Да	Да
Завивка	Час	Да	
Стрижка	Час	Да	
Парфюмерия		Нет	Да
Духи	Штука	Нет	
Одеколон	Штука	Нет	
Прочие материалы		Нет	Да
УФ-гель	Упаковка	Нет	
Спецодежда		Нет	Да
Одежда для парикмахера	Штука	Нет	
Уход за волосами		Нет	Да
Бальзам для волос	Штука	Нет	
Лак для волос	Упаковка	Нет	

На Рисунок 6.5. вы можете видеть форму документа после ввода в нее некоторых данных.

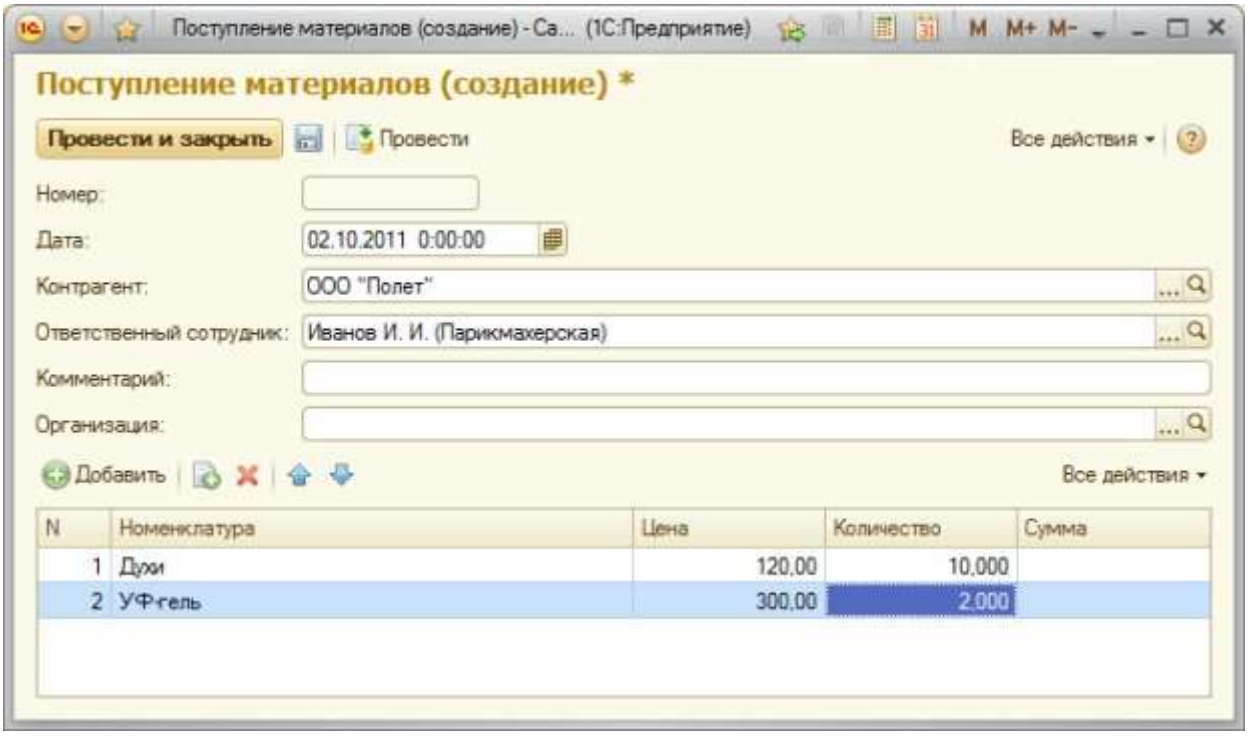


Рисунок 6.5. Заполнение документа **Поступление товаров**

Подобный документ, очевидно, заполняется в *информационной базе по* данным некоего бумажного документа (*приходной накладной*, например), поступившей от поставщика вместе с поступившими товарами. Цена и количество каждой товарной позиции вводится из документа вручную, с этим здесь ничего поделать нельзя. Но вот ввод суммы *по* каждой из строк табличной части вполне поддается автоматизированному расчету на основе данных о цене и количестве. Еще один важный момент, на который можно обратить внимание – в нашем документе не отображается итоговый показатель *по* табличной части. Как правило, в подобных случаях пользователю важно увидеть общую сумму документа. Общая сумма позволит быстро сверить данные,

введенные в *электронный документ*, с его бумажным аналогом. Если общая сумма совпадает – то, почти наверняка, все строки табличной части введены верно.

Реализуем эту функциональность.

Для того, чтобы автоматически заполнить *поле* сумма *по* каждой из строк табличной части, редактируемой пользователем, очевидно, что рассчитывать сумму имеет смысл либо после заполнения поля **Цена**, либо – после заполнения поля **Количество,** перехватив какие-либо события, имеющие *отношение* к редактируемой табличной части.

В нашем случае это должны быть события, генерируемые при изменении полей **Цена** или **Количество** при вводе данных в определенной строке. Для того, чтобы назначить обработчики подобных событий для определенных элементов табличной части, можно поступить так же, как мы поступали, назначая обработчики событий для любых других элементов формы (Рисунок 6.6.). Для начала, конечно же, нам нужно будет создать собственную форму документа, делается это на закладке **Формы** окна редактирования объекта. С параметрами, предложенными конструктором форм *по* умолчанию, можно согласиться.

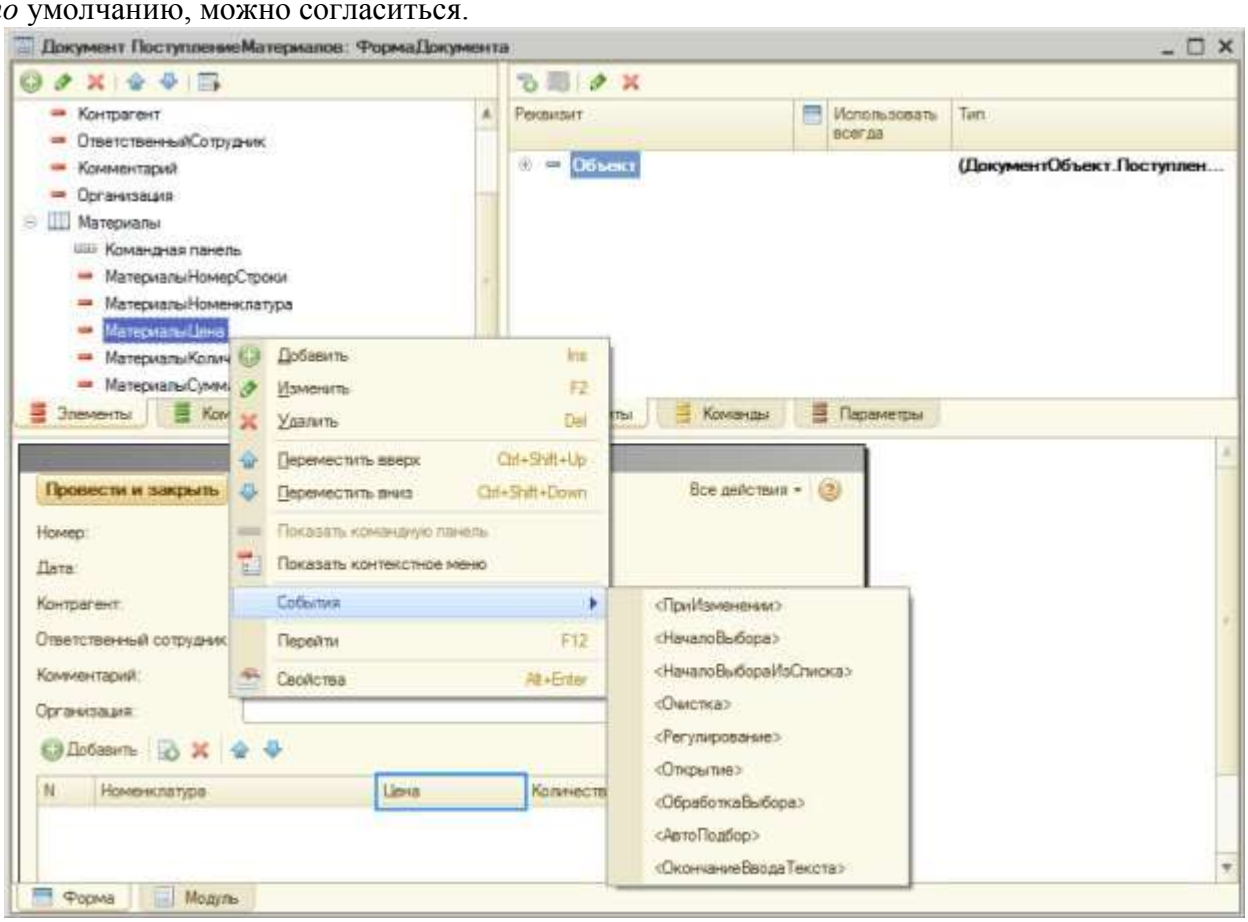


Рисунок 6.6. Назначение обработчика полю табличной части

Назначим обработчики событий **ПриИзменении** для полей **МатериалыЦена** и **МатериалыКоличество.**

Теперь нам нужно реализовать следующее: при работе в определенной строке таблицы, при вводе в нее данных, получить эту строку, и, при изменении цены или количества номенклатуры рассчитать сумму.

У табличных полей есть свойство **ТекущиеДанные**, которое, как раз, позволяет обращаться к текущей редактируемой строке. Данные редактируются на клиенте, поэтому мы вполне можем обойтись здесь без вызова серверных процедур, выполнив все необходимые действия на клиенте. Если вы хотите побольше узнать о том, что можно сделать с табличным полем из кода, как и в других случаях, помочь вам в этом могут

инструменты отладки. Вот как, например, выглядит свойство **ТекущиеДанные** при срабатывании точки останова в коде модуля нашей формы при отладке кода, который будет представлен ниже, Рисунок 6.7.

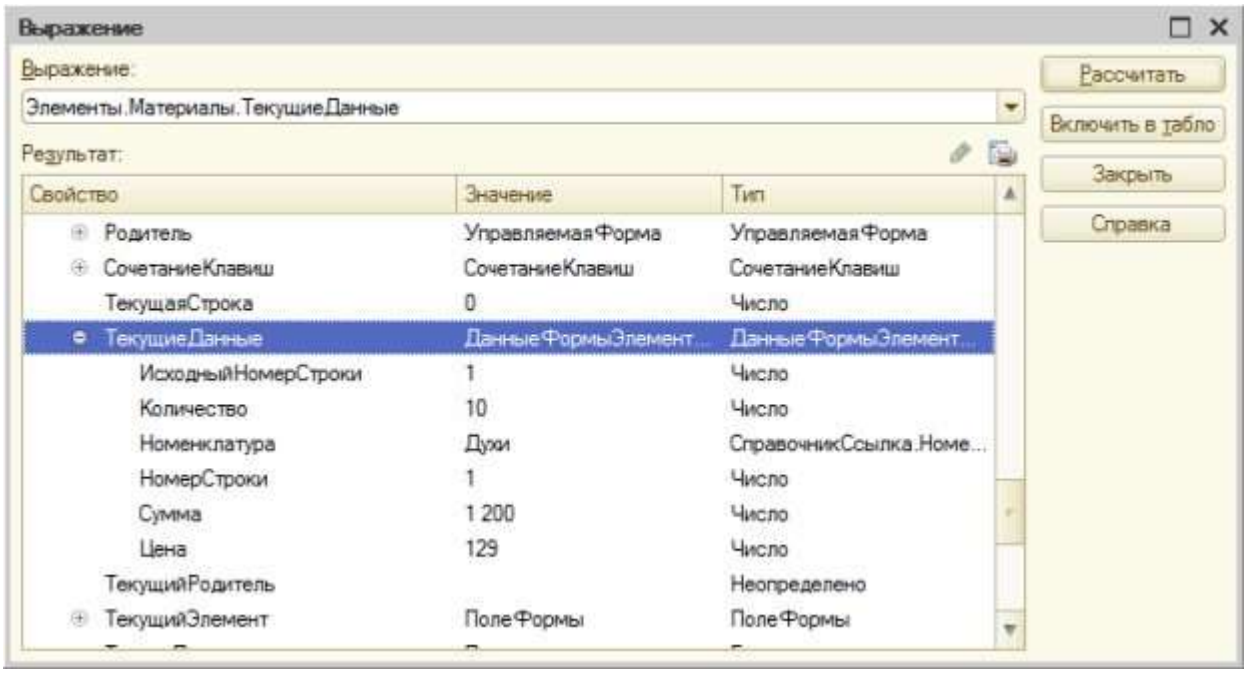


Рисунок 6.7. Просмотр свойства ТекущиеДанные в окне Выражение при отладке кода

Итак, наша задача может быть решена следующим образом:

```
&НаКлиенте
Процедура МатериалыЦенаПриИзменении(Элемент)
 РассчитатьСумму();
КонецПроцедуры

&НаКлиенте
Процедура МатериалыКоличествоПриИзменении(Элемент)
 РассчитатьСумму();
КонецПроцедуры

&НаКлиенте
Процедура РассчитатьСумму()
 ТекущаяСтрока=Элементы.Материалы.ТекущиеДанные;
 ТекущаяСтрока.Сумма=ТекущаяСтрока.Количество*ТекущаяСтрока.Цена;
КонецПроцедуры
```

Из пары обработчиков событий **ПриИзменении** вызывается клиентская процедура **РассчитатьСумму()**. Здесь мы получаем данные текущей строки через свойство **ТекущиеДанные** и вычисляем *поле* **Сумма**, перемножая данные в полях **Количество** и **Цена**.

При необходимости, мы можем редактировать *поле* **Сумма** независимо от значений полей **Цена** и **Количество**.

Вторая задача из тех, которые мы поставили себе выше, заключается в выводе на форму итоговых сведений *по* табличному полю. Ее можно реализовать различными способами, но лучше всего воспользоваться стандартными итоговыми показателями

табличного поля, которые можно найти в составе табличного поля на закладке **Реквизиты** редактора форм, Рисунок 6.8.

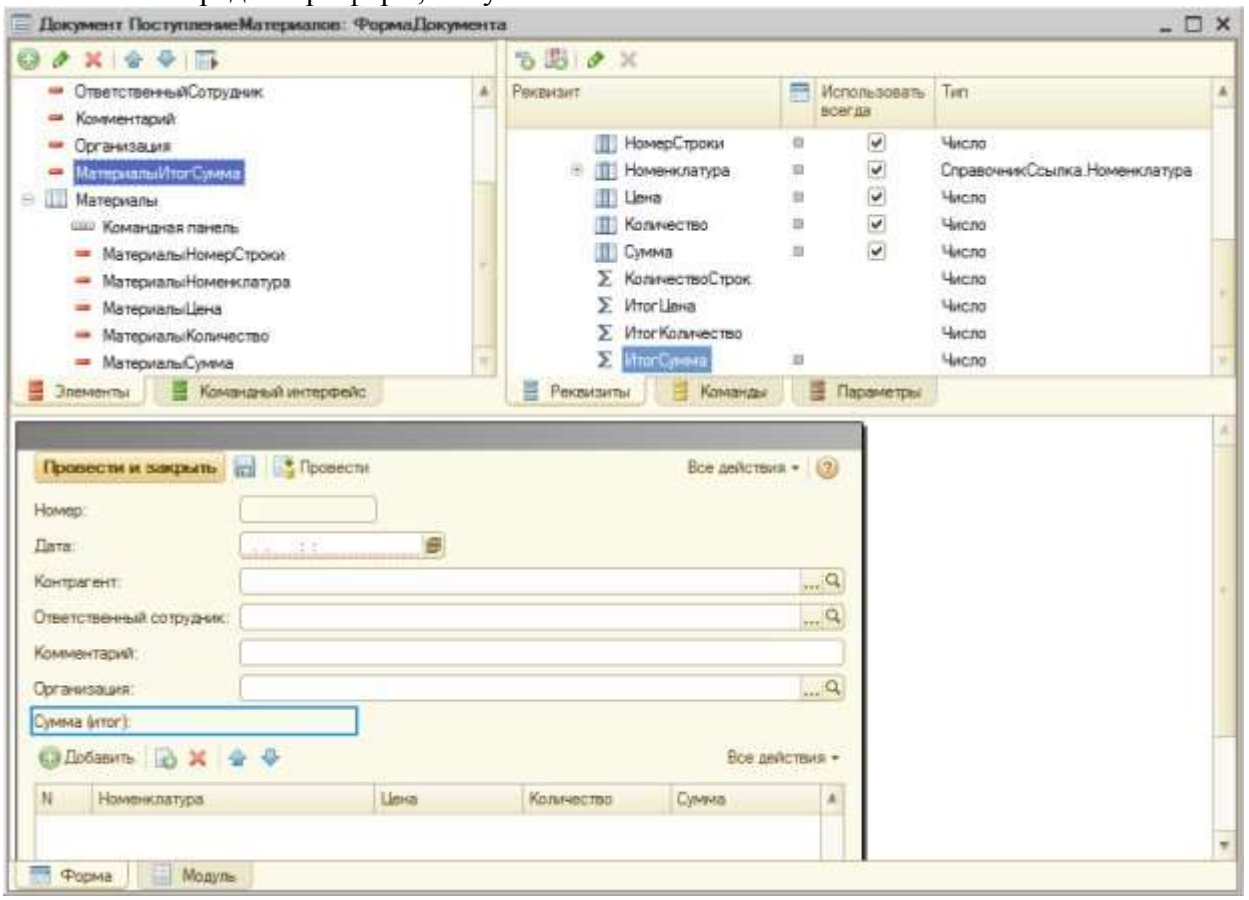


Рисунок 6.8. Вывод итогового показателя для поля Сумма на форму

Этот *реквизит* – **ИтогСумма** – нужно перетащить на вкладку **Элементы**. Он будет отображаться на форме, изменяясь при изменениях суммы в строках табличной части, Рисунок 6.9.

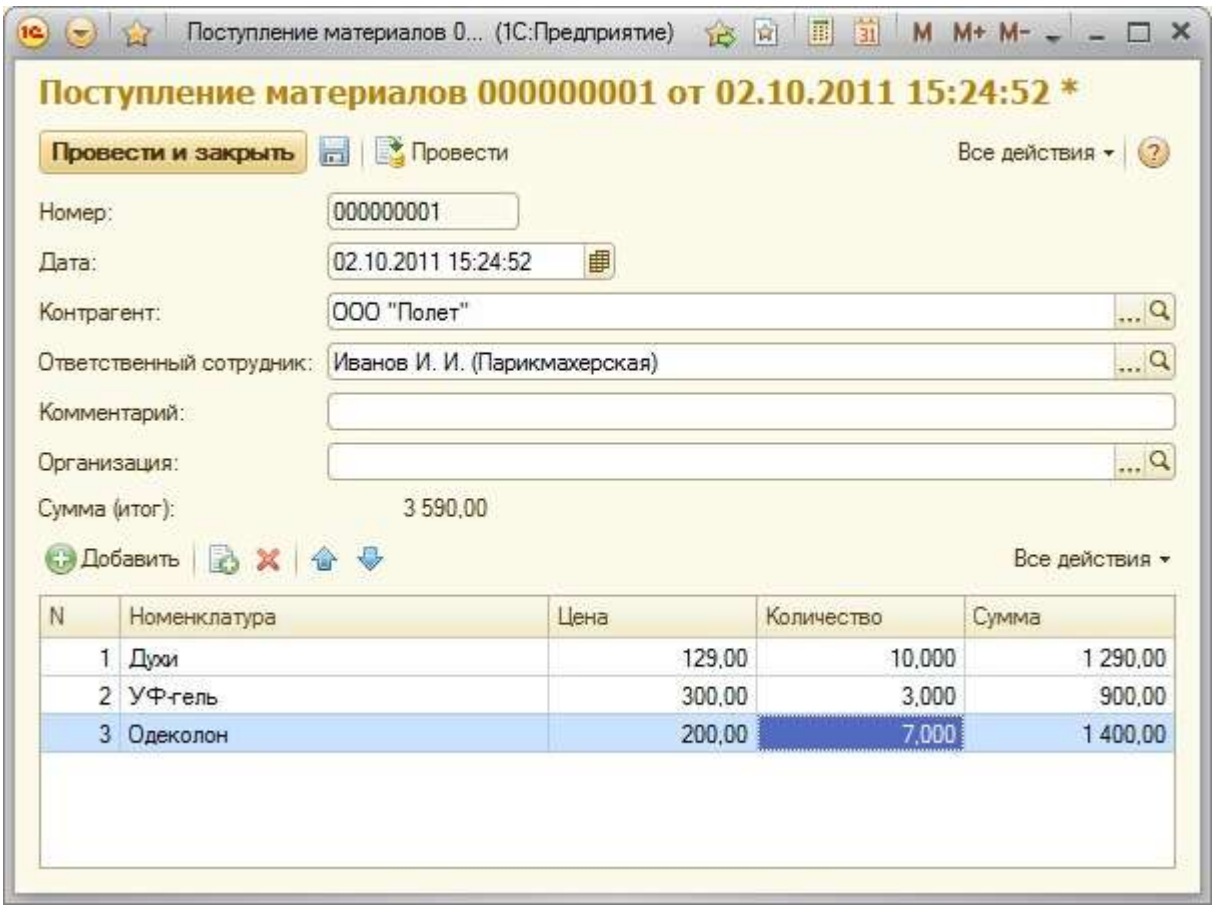


Рисунок 6.9. Форма после модификации

Документ **ПоступлениеМатериалов** мы разработали, настроили его форму. Документ, хотя его проведение и разрешено, пока, фактически, не проводится – мы не реализовали *механизмы* проведения, у нас нет регистров, *по* которым он будет проводиться.

Данную функциональность мы реализуем ниже, а сейчас займемся еще одним документом, который, являясь, *по* составу реквизитов и *по* особенностям устройства формы, очень похожим на документ **ПоступлениеМатериалов**, выполняет противоположную ему функцию – а именно – отвечает за списание материалов. В нашей системе материалы выбывают при передаче их в производство. Мы вполне можем создать новый документ копированием предыдущего и изменением некоторых его реквизитов. Так и поступим. Скопируем документ и приведем состав его реквизитов к показанному на Рисунок 6.10.

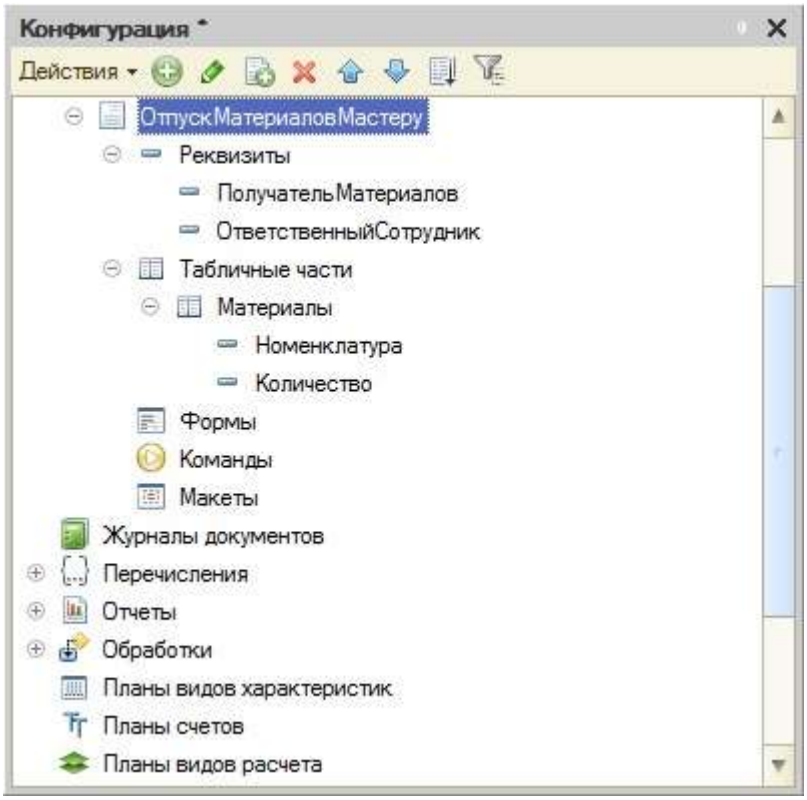


Рисунок 6.10. Создание документа ОтпускМатериаловМастеру

Включим данный документ в состав подсистемы **ОперативныйУчетМатериалов**, вместо реквизита **Контрагент** у него будет *реквизит* **ПолучательМатериалов** с типом **СправочникСсылка.Сотрудники**.

В табличной части документа мы используем лишь два реквизита – это **Номенклатура** и **Количество**. Показатели стоимости списываемой номенклатуры мы будем рассчитывать автоматически. При работе с этим документом нас вполне устроит форма, генерируемая автоматически.

Форма нашего нового документа будет выглядеть так, как показано на Рисунок 6.11.

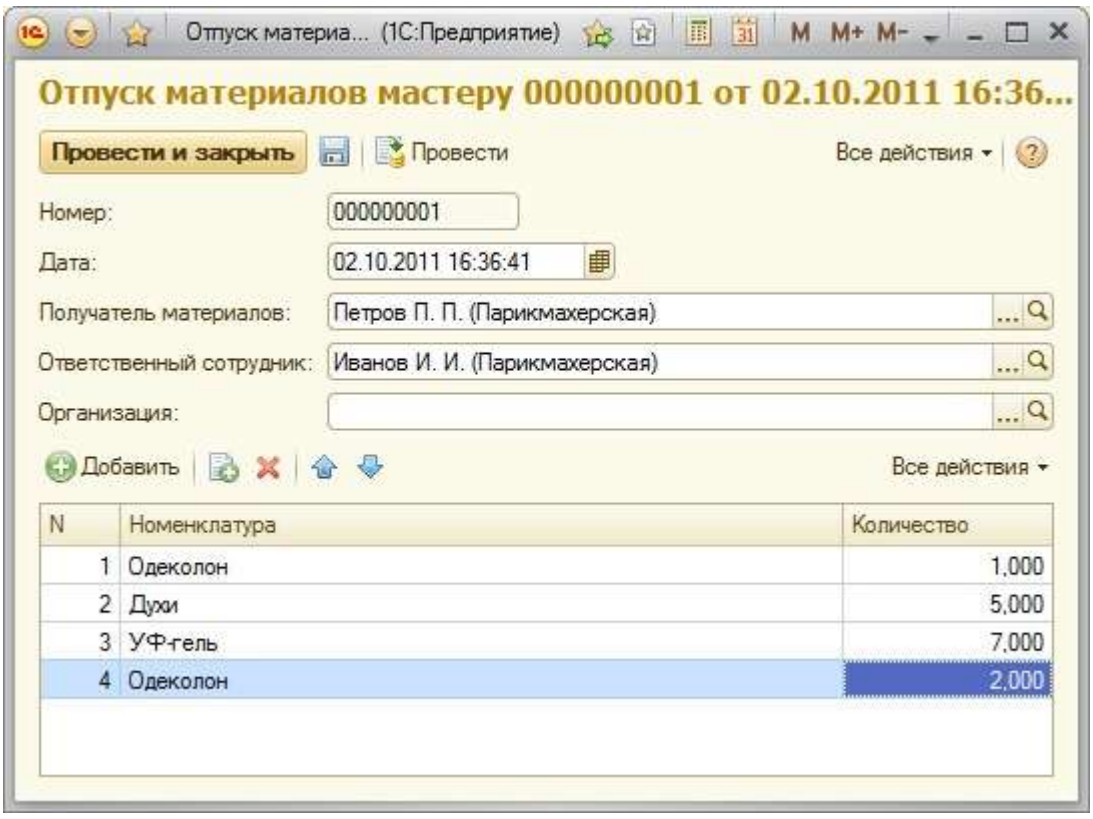


Рисунок 6.11. Документ ОтпускМатериаловМастеру в работе

Документы созданы, но сейчас они, во-первых, заполняются практически полностью вручную (за исключением поля **Сумма** в табличной части документа **ПоступлениеМатериалов**), а, во-вторых, нам сейчас довольно сложно будет понять, каков *остаток* материалов, числящихся за определенным сотрудником.

Единственный способ, которым можно сделать это сейчас – программно или "вручную" просмотреть все документы поступления материалов, выводя некоторые итоговые показатели. При заполнении документа отпуска материалов мы вынуждены заранее самостоятельно проверять остатки материалов *по* существующим документам, самостоятельно решать, какова цена этих материалов, контролировать остатки. Это не очень удобно, не производительно, и решением этих задач мы сейчас займемся.

Регистры накопления

Представим себе организацию, в которой все сведения о приходе материалов хранятся лишь в виде приходных документов. Для того, чтобы узнать количество и *стоимость* имеющихся в организации материалов, нам понадобится обращаться к документам, просматривать каждый из них, выписывая нужные данные, после чего суммировать их, получая нужные данные. Такой подход неудобен – он слишком медленный как для нашего воображаемого "ручного" случая, так и для автоматизированного учета.

Логичнее было бы, в *дополнение* к документам, вести специальные таблицы, в которые, при приеме материалов и при их списании, вносить краткие сведения об этом. Если, скажем, ежедневно, подводить итоги *по* этим таблицам и выводить остатки материалов, то, для того, чтобы сказать, сколько и каких материалов имеется в организации на определенную дату, достаточно обратиться к соответствующей графе таблицы. В системе «1С:Предприятие» такими таблицами являются *регистры накопления*. Как следует из названия, они предназначены для отражения, накопления, неких

показателей. И отражение в регистрах прихода и расхода материалов – один из типичных примеров их использования.

В нашей организации ведется учет материалов в привязке к ответственным лицам, которые их получают и с которых эти материалы списывают при отпуске в производство. Нам нужно хранить информацию о количестве материалов и об их стоимости, а так же, при списании в производство, иметь сведения о том, какому именно мастеру эти материалы переданы. Эти соображения позволяют нам спроектировать структуру *регистра накопления*, который мы сейчас будем создавать.

При планировании состава *регистра накопления* нужно понять, какие именно данные мы собираемся в нем хранить, после чего "разложить" эти данные *по* измерениям, ресурсам и реквизитам регистра.

Итак, нам нужно хранить следующие данные:

- Номенклатурная позиция
- Ответственный сотрудник, на котором числится данная позиция
- Количество номенклатуры
- Стоимость номенклатуры
- Данные о мастере, которому переданы материалы для использования.

Измерения регистра, или разрезы, в которых хранятся данные, позволяют нам ответить на вопросы о том, какие именно данные хранятся в регистре. В нашем случае нам нужно знать две основных характеристики – это, за каким **ответственным лицом**закреплена та или иная **номенклатурная позиция**. Очевидно, измерениями из нашего списка данных будут номенклатурная позиция и ответственный сотрудник.

Ресурсы регистра – это всегда числовые значения, характеризующие хранимые данные. Числовые значения – это количество и сумма, и именно они будут ресурсами нашего регистра.

Реквизиты регистра играют вспомогательную роль, и, в нашем случае, логично будет в *реквизите* регистра хранить сведения о мастере, получившем материалы для работы – на тот случай, если нам понадобится узнать – кто именно эти материалы использовал.

Еще один важный вопрос, который нужно решить, проектируя *регистр*, заключается в том, будет ли этот *регистр* **регистром остатков** или **регистром оборотов**. Нас интересуют и сведения об остатках материалов, и сведения об оборотах, поэтому при настройке регистра следует указать вид регистра – **Остатки**. *Регистр* с видом **Остатки** позволяет нам работать и с остатками и с оборотами

Предложенная здесь структура *регистра накопления* – это лишь один из вариантов того, как можно организовать хранение описываемых данных. Подобную схему учета можно реализовать, скажем, с помощью пары регистров, один из которых используется исключительно для целей хранения суммовых остатков материалов – то есть, те данные, которые нужны для финансовых отчетов, другой – для хранения данных *по* центрам ответственности. В любом случае, каждая конкретная схема учета может потребовать и собственной структуры регистров, и наш пример – лишь демонстрация одного из возможных вариантов.

Обсудив теоретическую часть вопроса, перейдем к практике. Создадим новый *регистр накопления*, назовем его**ОстаткиМатериалов**, *параметр* **Вид регистра** оставим в значении **Остатки**, Рисунок 6.12.

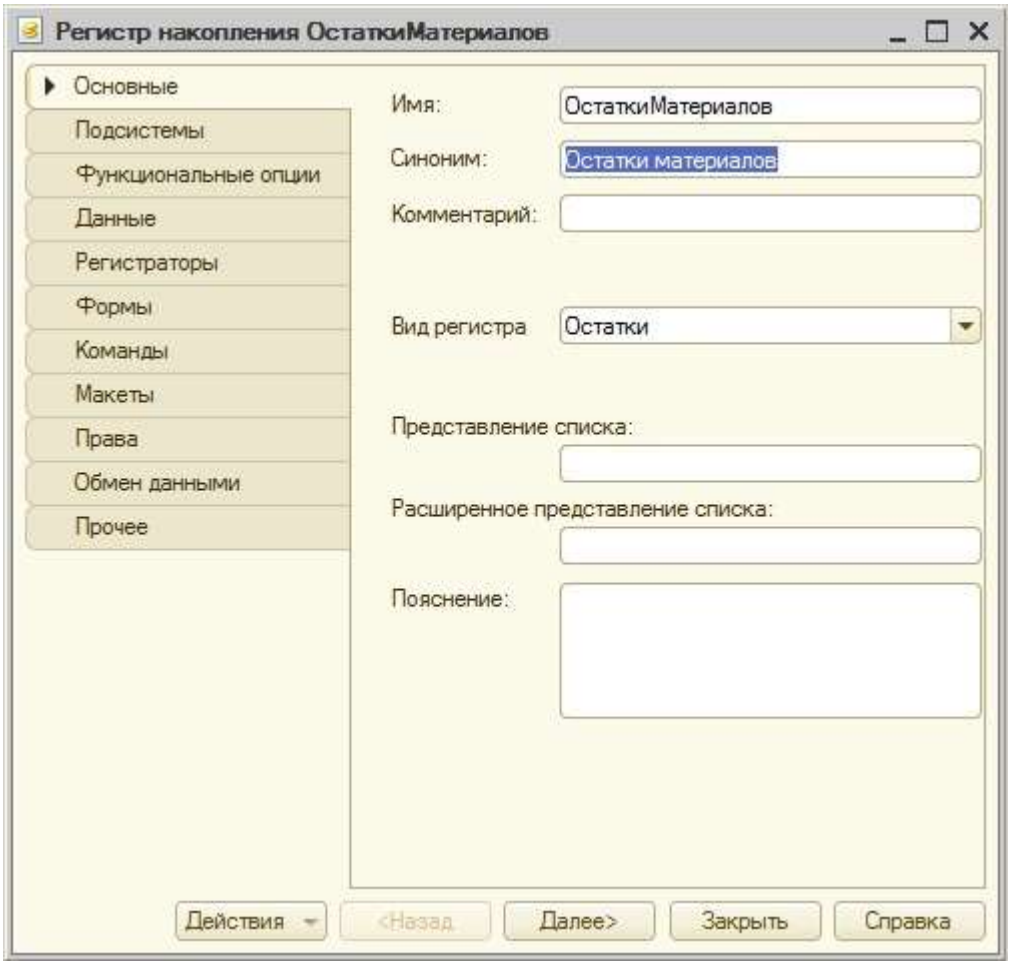


Рисунок 6.12. Регистр накопления ОстаткиМатериалов

Включим *регистр накопления* в состав подсистемы **ОперативныйУчетМатериалов**.

На вкладке **Данные** создадим следующие измерения, ресурсы и реквизиты:

Измерения:

Имя: Номенклатура, Тип: СправочникСсылка.Номенклатура, Запрет незаполненных значений – установлено.

Имя: ОтветственныйСотрудник, Тип: СправочникСсылка.Сотрудники, Запрет незаполненных значений – установлено.

Ресурсы

Имя: Количество, Тип: число, *длина* 10, *точность* 3

Имя: Сумма, Тип: число, *длина* 10, *точность* 2

Реквизиты:

Имя: ПолучательМатериалов, Тип: СправочникСсылка.Сотрудники

Обратите внимание на имена этих реквизитов, на их типы, а так же – на стандартные реквизиты регистра (Рисунок 6.13.) – эти данные пригодятся нам при работе над процедурой *проведения документа*.

Исключим из состава реквизитов регистра общий *реквизит* **Организация**. Сейчас в нем нет необходимости. Для организации хранения данных в регистре в разрезе различных организаций нам понадобилось бы новое измерение – Организация, благодаря наличию которого мы смогли бы работать с материалами различных организаций.

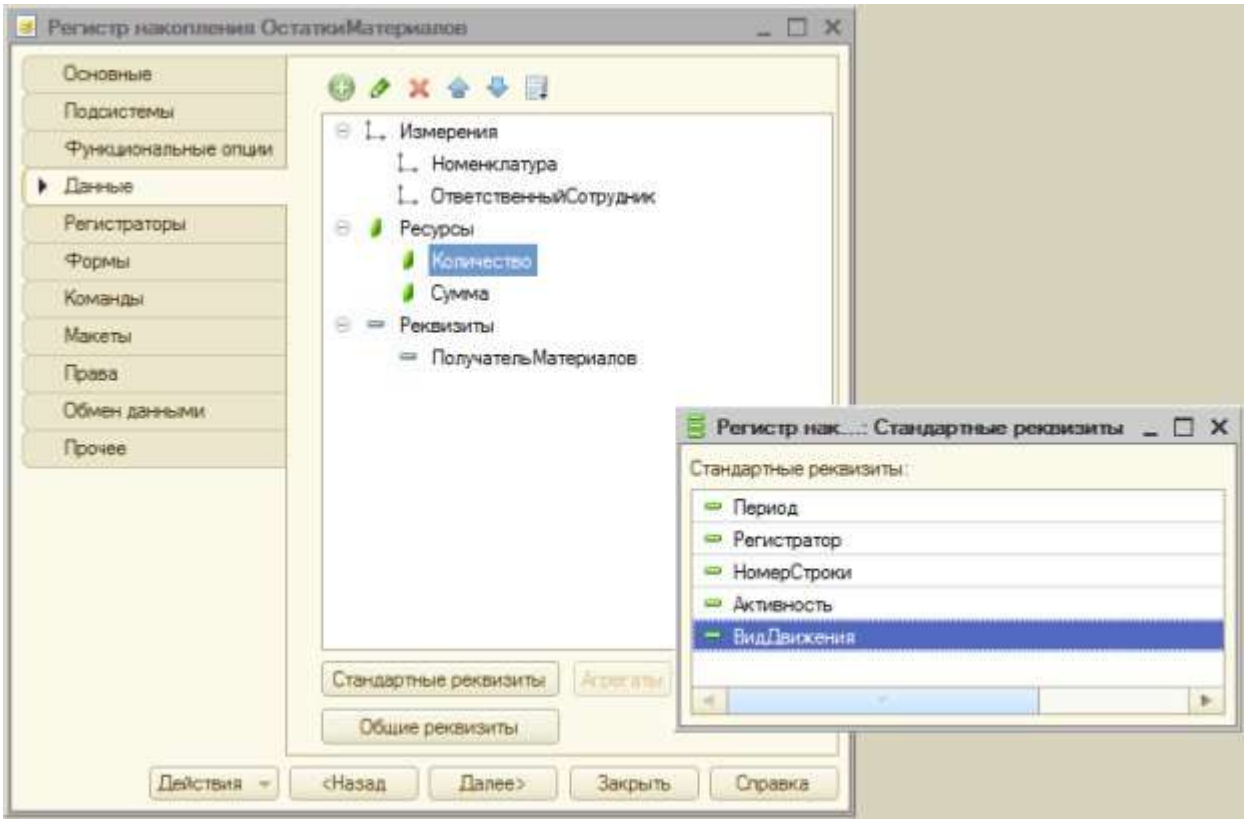


Рисунок 6.13. Регистр накопления ОстаткиМатериалов, состав данных

Перейдем на вкладку **Регистраторы** окна редактирования объекта и выберем в качестве документов-регистраторов документы – **ПоступлениеМатериалов** и **ОтпускМатериаловМастеру**.

На данном этапе настройка *регистра накопления* окончена, перейдем к настройкам документов. Начнем с документа**ПоступлениеМатериалов**.

Откроем окно редактирования объекта для этого документа, перейдем на вкладку **Движения** (Рисунок 6.14.) и нажмем на кнопку**Конструктор движений**

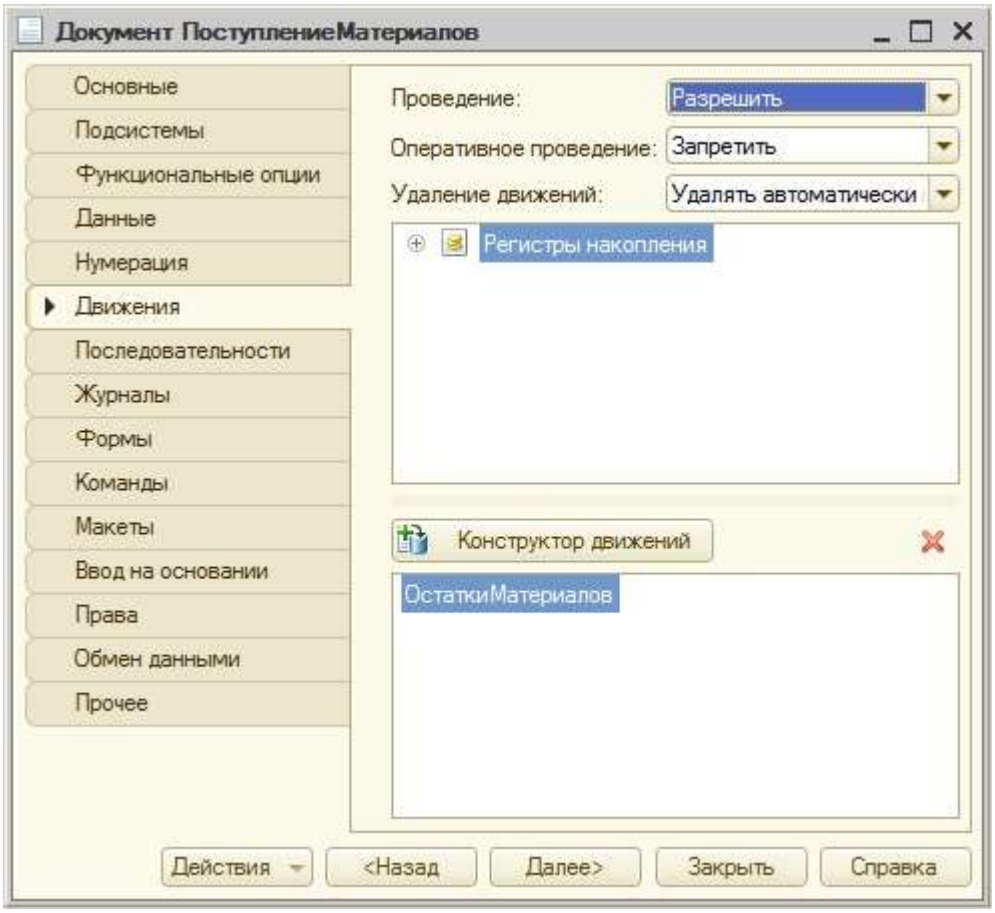


Рисунок 6.14. Документ ПоступлениеМатериалов, вкладка Движения

В конструкторе, выберем тип движения регистра – **Приход**, в *поле* **Табличная часть** укажем табличную часть документа**Материалы**, нажмем на кнопку **Заполнить выражения**. Автоматический механизм установления соответствия между данными документа и регистра не всегда работает правильно (в том случае, если не может однозначно определить соответствия, или тогда, когда соответствие, определенное им *по* его логике, отличается от желаемого), поэтому проверим правильность установленных соответствий. В итоге окно **Конструктора движения регистра** должно выглядеть так, как показано на Рисунок 6.15.

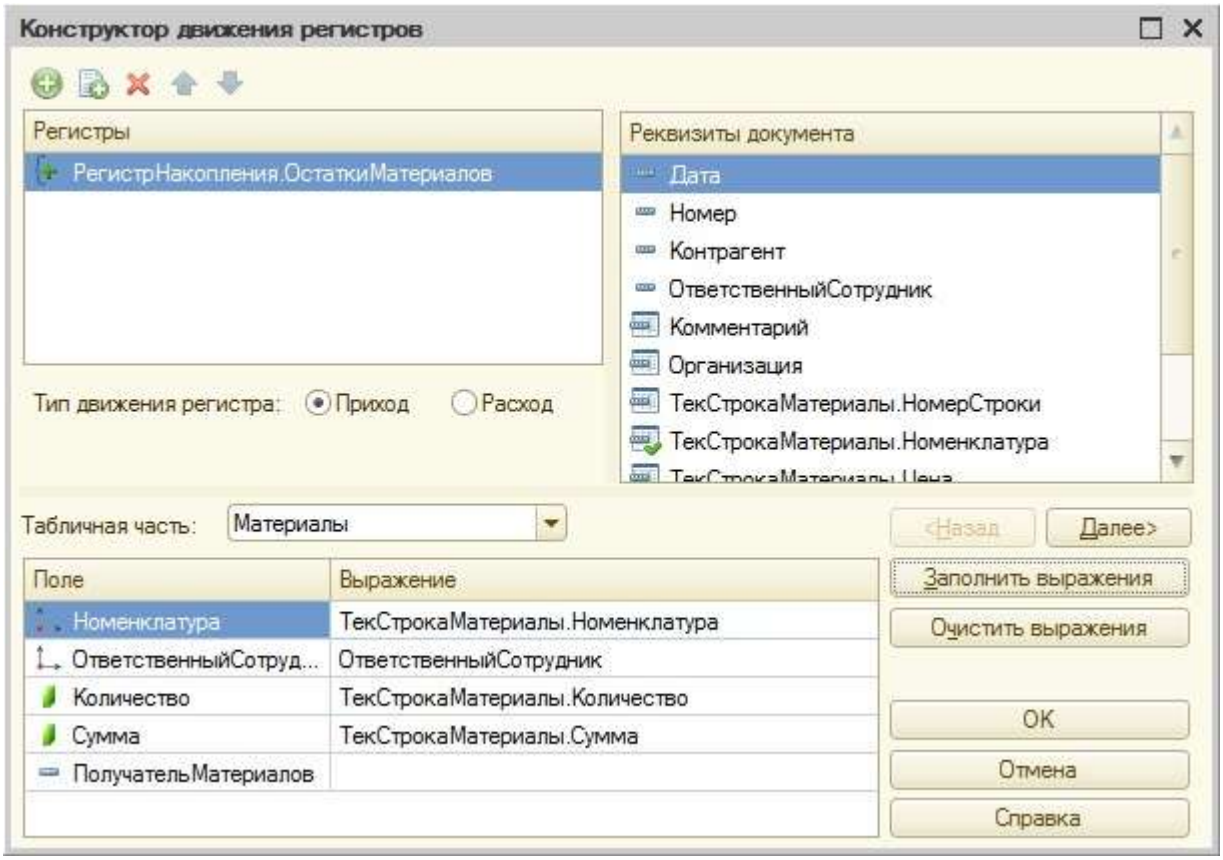


Рисунок 6.15. Конструктор движений

После нажатия на кнопку ОК, в модуле объекта документа будет сформирована такая процедура обработки проведения (так она выглядит после удаления комментариев о том, что код построен конструктором движений):

```
Процедура ОбработкаПроведения(Отказ, Режим)
 // регистр ОстаткиМатериалов Приход
 Движения.ОстаткиМатериалов.Записывать = Истина;
 Для Каждого ТекСтрокаМатериалы Из Материалы Цикл
   Движение = Движения.ОстаткиМатериалов.Добавить();
   Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
   Движение.Период = Дата;
   Движение.Номенклатура = ТекСтрокаМатериалы.Номенклатура;
   Движение.ОтветственныйСотрудник = ОтветственныйСотрудник;
   Движение.Количество = ТекСтрокаМатериалы.Количество;
   Движение.Сумма = ТекСтрокаМатериалы.Сумма;
 КонецЦикла;
КонецПроцедуры
```

Эта процедура объявлена в модуле объекта, она исполняется на *сервере. Установка* свойства**Движения.ОстаткиМатериалов.Записывать** в *значение* **Истина** говорит системе о том, что она должна записать в *регистр* движения, сформированные в процедуре. Движения заполняются в цикле обхода табличной части документа. Они будут физически записаны в *регистр* после того, как будет сформирован полный набор записей. Процедура проведения вполне может использовать и другие *механизмы* обращения к данным. Например, нужные данные могут быть получены с помощью запроса.

Так как мы проводим документ, отвечающий за приход товаров, *параметр* **Движение.ВидДвижения** устанавливается в *значение***ВидДвиженияНакопления.Приход**. Период устанавливается равным дате документа (мы напрямую работаем с реквизитами документа). Остальные данные заполняются, опять же, либо из реквизитов документа, либо из реквизитов текущей строки табличной части.

Запустим конфигурацию в режиме «1С:Предприятие», проверим работу механизма на практике. Для этого перепроведем существующие документы **ПоступлениеМатериалов**, можем ввести и новые документы этого вида. При проведении или перепроведении данные из документов попадают в *регистр накопления* **ОстаткиМатериалов**, Рисунок 6.16.

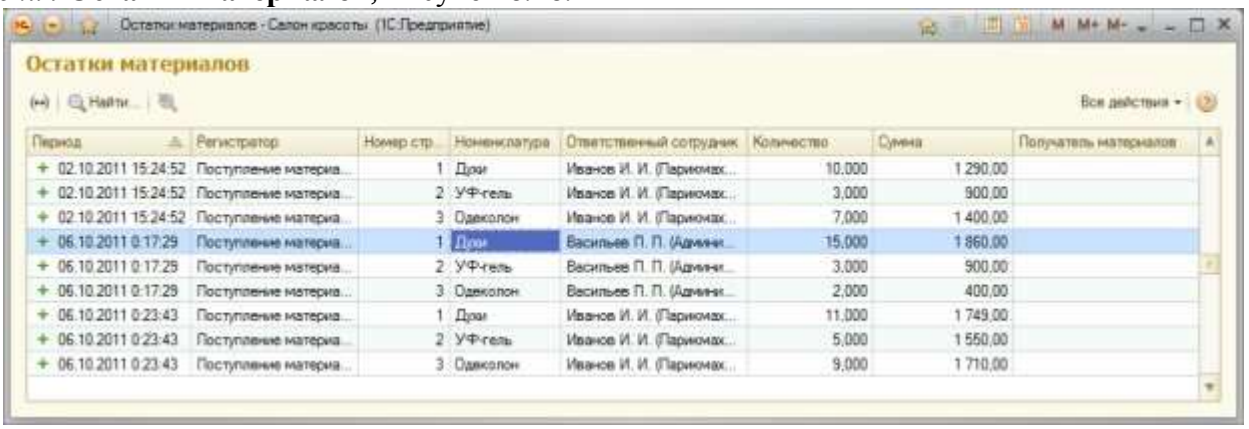


Рисунок 6.16. Данные в регистре накопления

Обратите внимание на то, что *регистры накопления*, как объекты, которые не предназначены изначально для "ручной" работы пользователя, не выводятся в командном интерфейсе даже при указании подсистем, в которые они входят. Нам, при разработке, понадобится просматривать регистры.

Для того, чтобы открыть окно регистра можно либо воспользоваться командой **Главное меню > Все функции** и в появившемся окне **Все функции** найти нужный *регистр*, либо открывать его с помощью команды в интерфейсе, предварительно самостоятельно добавив эту команду в нужный раздел интерфейса.

При записи данных о приходе материалов нам, в нашем случае, нет нужды в каких-либо дополнительных проверках вводимых данных, поэтому нас вполне устроит стандартная процедура проведения.

Следующим документом, проведение которого мы хотим организовать, является документ, списывающий материалы –**ОтпускМатериаловМастеру**. Им мы займемся в следующей лекции, а сейчас построим отчет **ОстаткиМатериалов.**

Отчет ОстаткиМатериалов, введение в СКД

Ранее мы рассматривали построение простейшего отчета самостоятельно формируя *макет* отчета, код вывода данных. Такой подход имеет право на жизнь, им нужно владеть, *по* крайней мере, на тот случай, если вам придется редактировать отчет в какой-либо конфигурации, где применяется такой подход. Однако, наиболее правильным способом построения отчетов в системе «1С:Предприятие» 8.2. является использование *системы компоновки данных*, или, сокращенно, СКД. Начиная с этой лекции мы будем строить отчеты именно с использованием СКД.

Итак, прежде чем строить отчет, поймем, чего мы ждем от этого отчета.

Нам хотелось бы получать сведения о количественных и суммовых остатках номенклатуры *по* ответственным лицам на заданную дату. Эти данные мы можем найти в регистре накопления **ОстаткиМатериалов.**

Добавим в *дереве конфигурации* новый отчет, назовем его **ОстаткиМатериалов**. Включим его в состав подсистемы**ОперативныйУчетМатериалов**.

На закладке **Основные** нажмем на кнопку с увеличительным стеклом в *поле* **Основная схема компоновки данных**. Появится окно конструктора макета, где мы можем задать имя (нас устроит имя *по* умолчанию – **ОсновнаяСхемаКомпоновкиДанных**), тип макета ограничен единственным – **Схема компоновки данных**. Нажмем в этом окне **Готово** и попадем в окно**конструктора СКД**. Здесь нам, в первую *очередь*, нужно добавить новый **источник данных**, в нашем случае это будет **Запрос**,Рисунок 6.17.

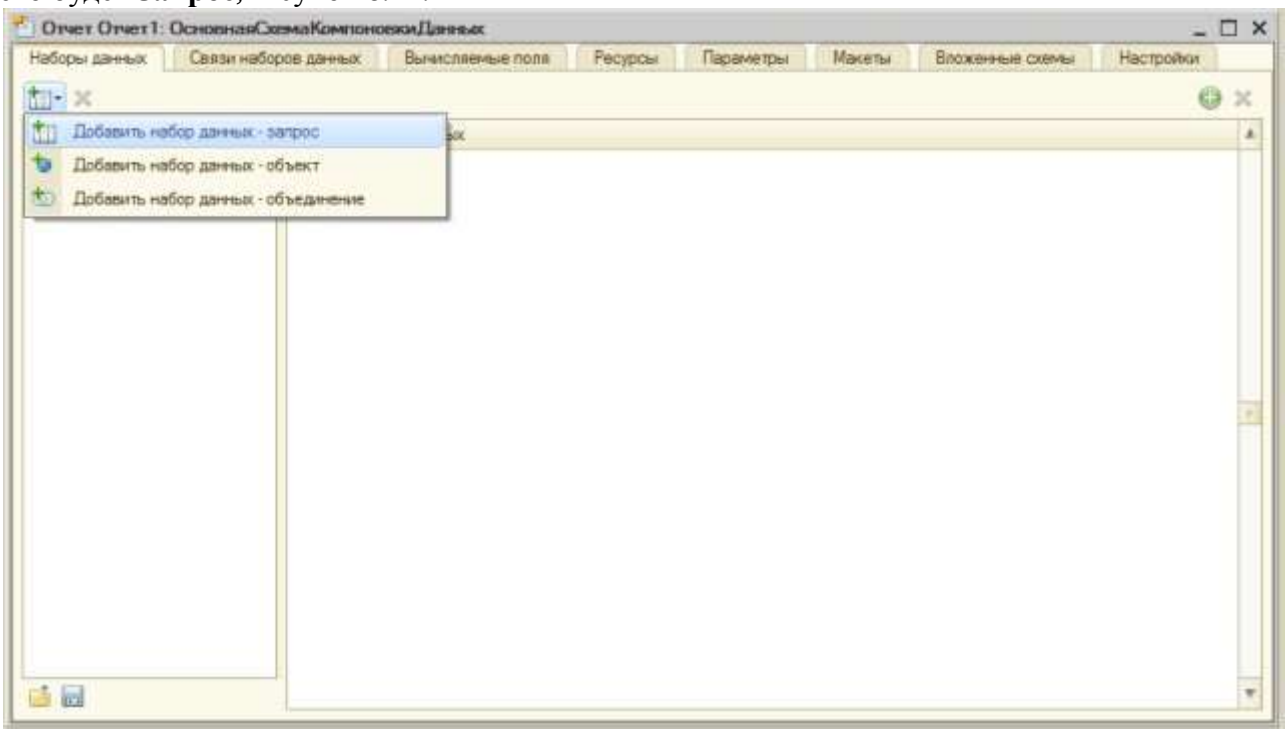


Рисунок 6.17. Добавление нового набора данных – запроса

Когда набор данных, названный **НаборДанных1**, будет добавлен, мы можем нажать на кнопку **КонструкторЗапроса**, находящуюся над полем **Запрос** в нижней части окна. Это приведет к открытию окна конструктора запроса.

Из виртуальной таблицы *регистра накопления* **ОстаткиМатериалов** выберем следующие поля, Рисунок 6.18.

- Номенклатура
- ОтвественныйСотрудник
- КоличествоОстаток
- СуммаОстаток

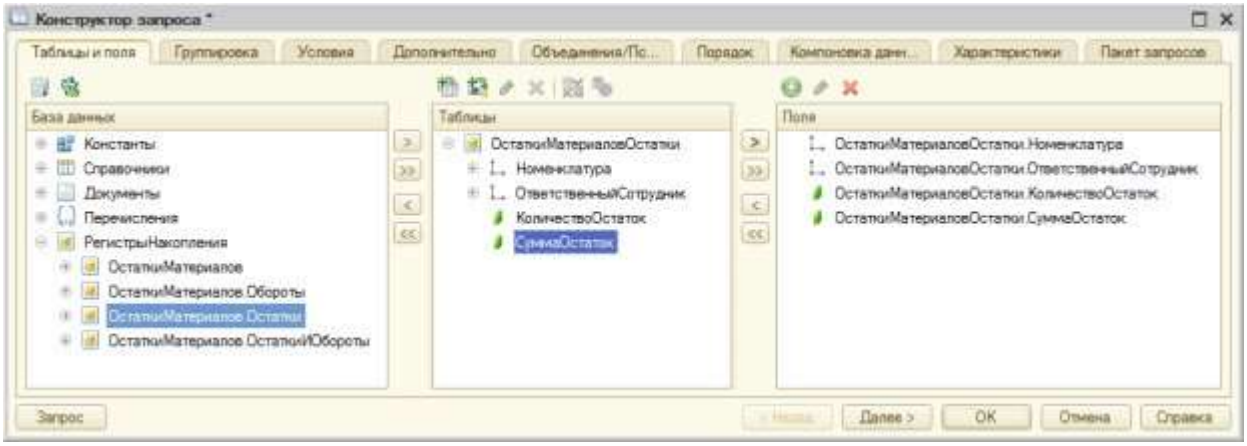


Рисунок 6.18. Создание запроса

На этом работа с конструктором запроса завершена – остальные настройки мы будем делать в конструкторе СКД. Благодаря установленному *по* умолчанию флагу **Автозаполнение**, на вкладке **Наборы данных** после создания запроса мы можем видеть заполненный *список* полей, Рисунок 6.19. – с этими полями мы сможем работать при создании отчета.

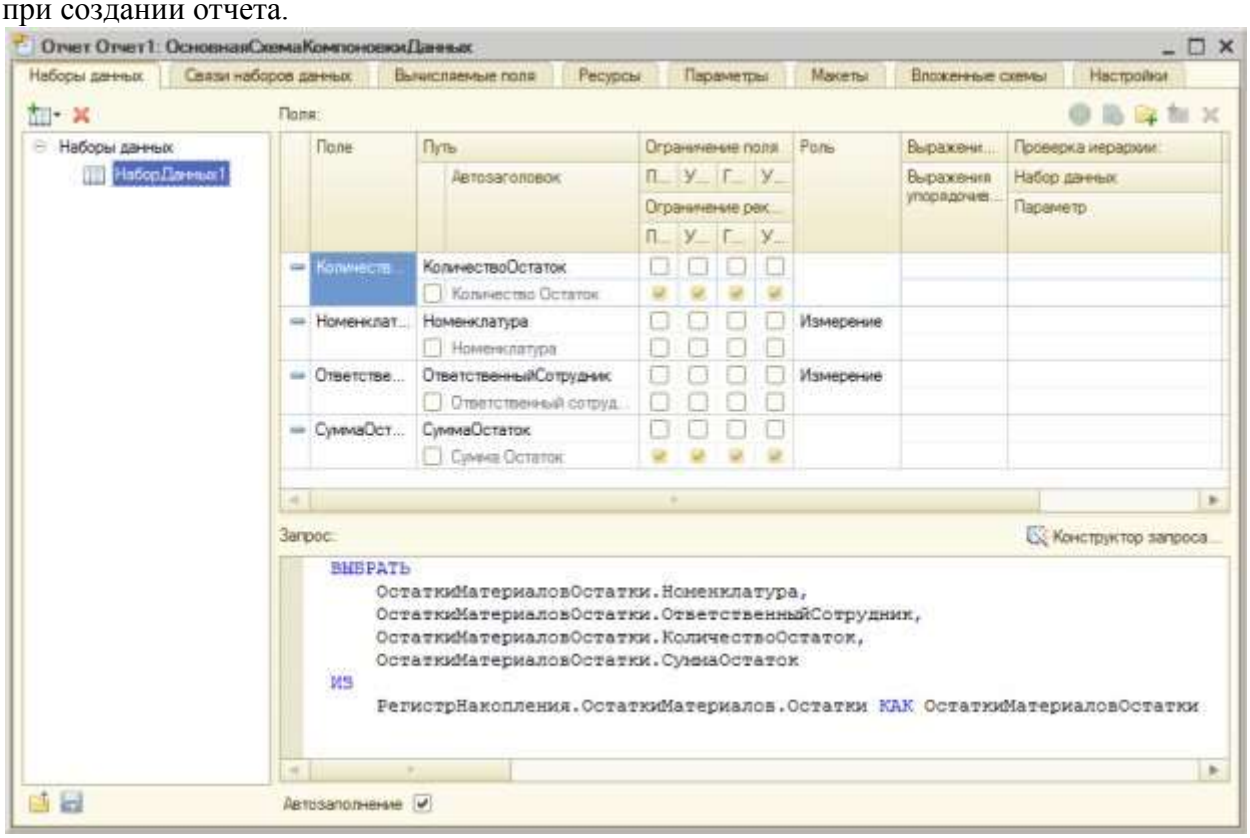


Рисунок 6.19. Автозаполнение списка полей на закладке Наборы данных

Переместимся в окне редактора СКД на вкладку **Ресурсы**, из списка **Доступные поля** перенесем в *список*, находящийся в правой части окна, поля, *по* которым можно вычислять итоги. В нашем случае это поля **КоличествоОстаток** и **СуммаОстаток**. *По*умолчанию этим полям в *поле* **Выражение** будет назначена агрегатная *функция* **Сумма**, нас устроит такое положение дел, Рисунок 6.20.

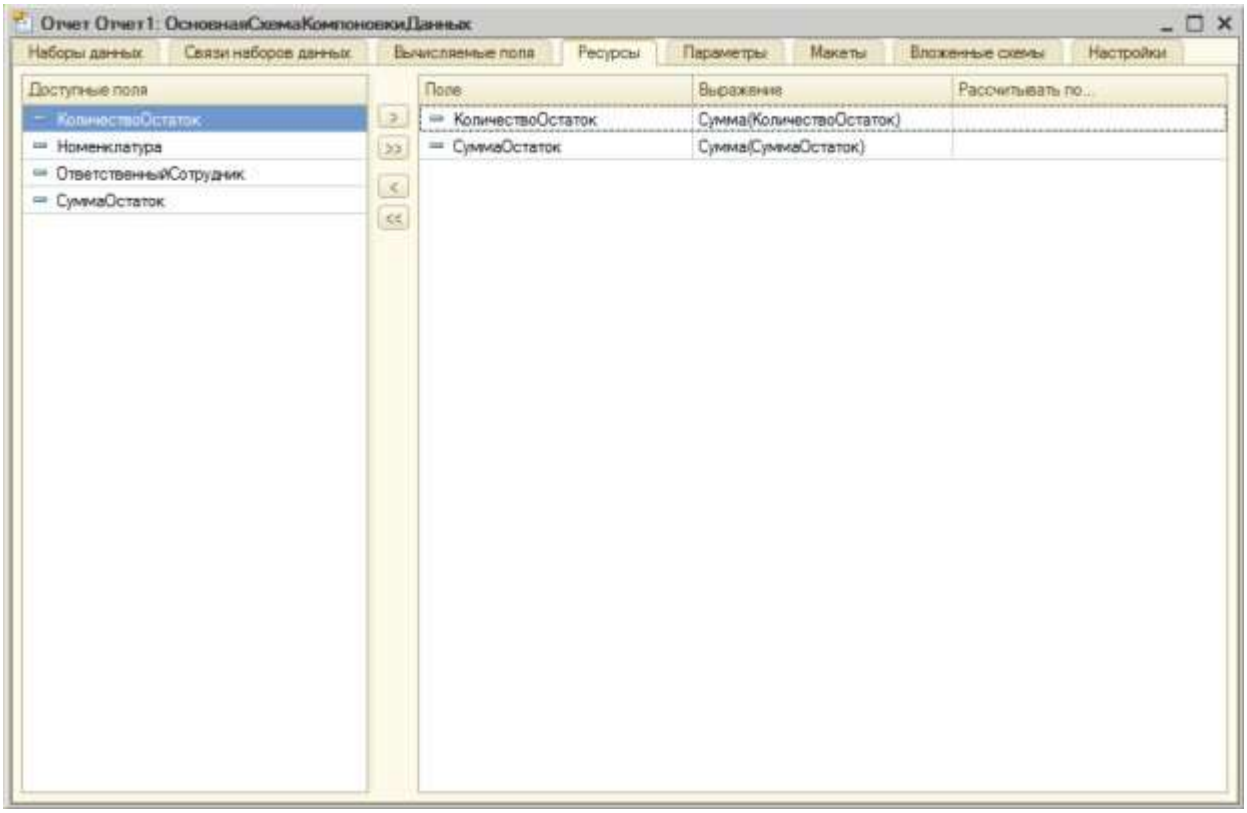


Рисунок 6.20. Настройка состава ресурсов отчета

Теперь займемся настройкой внешнего вида отчета.

Перейдем на закладку **Настройки**, вызовем кнопкой с соответствующим названием **Конструктор настроек** и выберем на его первой странице тип отчета – **таблицу**. Нажмем на кнопку **Далее** и в следующем окне выберем поля, которые будут отображаться в отчете в следующем порядке (Рисунок 6.21.):

- Номенклатура
- ОтветственыйСотрудник
- КоличествоОстаток
- СуммаОстаток

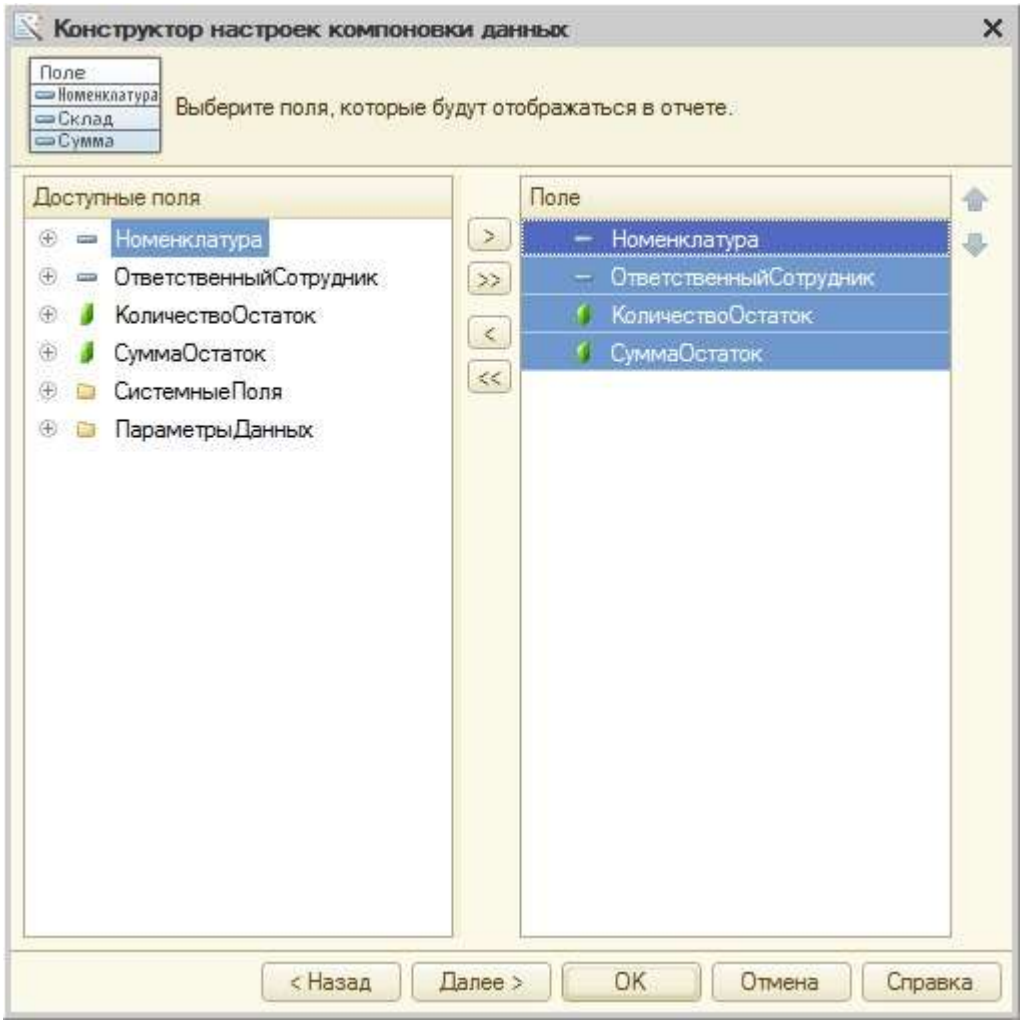


Рисунок 6.21. Выбор полей, которые будут отображаться в отчете

Нажмем кнопку **Далее**, в следующем окне конструктора, служащим для настройки группировки таблиц, в группу **Строки** добавим*поле* **Номенклатура**, в *поле* **Колонки** – **ОтветственныйСотрудник**. Тип группировки оставим в состоянии **Без иерархии**.

В следующем окне конструктора, который позволяет задать упорядочение отчета, зададим упорядочивание *по* полю**Номенклатура**, *по* возрастанию, Рисунок 6.22.

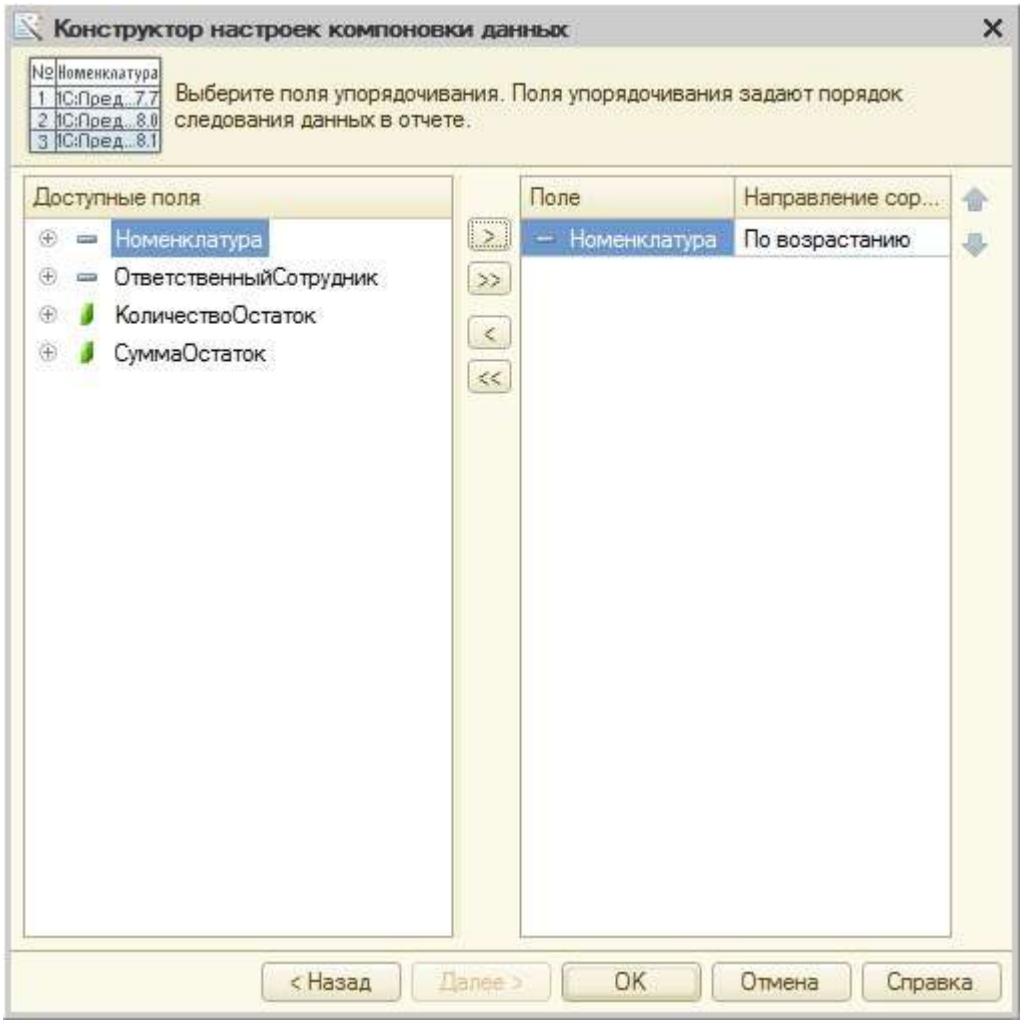


Рисунок 6.22. Настройка упорядочивания отчета

Нажмем **ОК**, в отчет будет добавлена новая *таблица*. В нижней части формы конструктора СКД, на закладке **Параметры**, выделим*параметр* **Период** и нажмем на кнопку **Свойства элемента пользовательских настроек**. В появившемся окне установим флаг**Включать в пользовательские настройки**, режим редактирования оставим в значении **Быстрый доступ**, Рисунок 6.23. Это позволит нам вывести данный *параметр* в форму отчета, позволит пользователю выбирать нужный период перед построением отчета

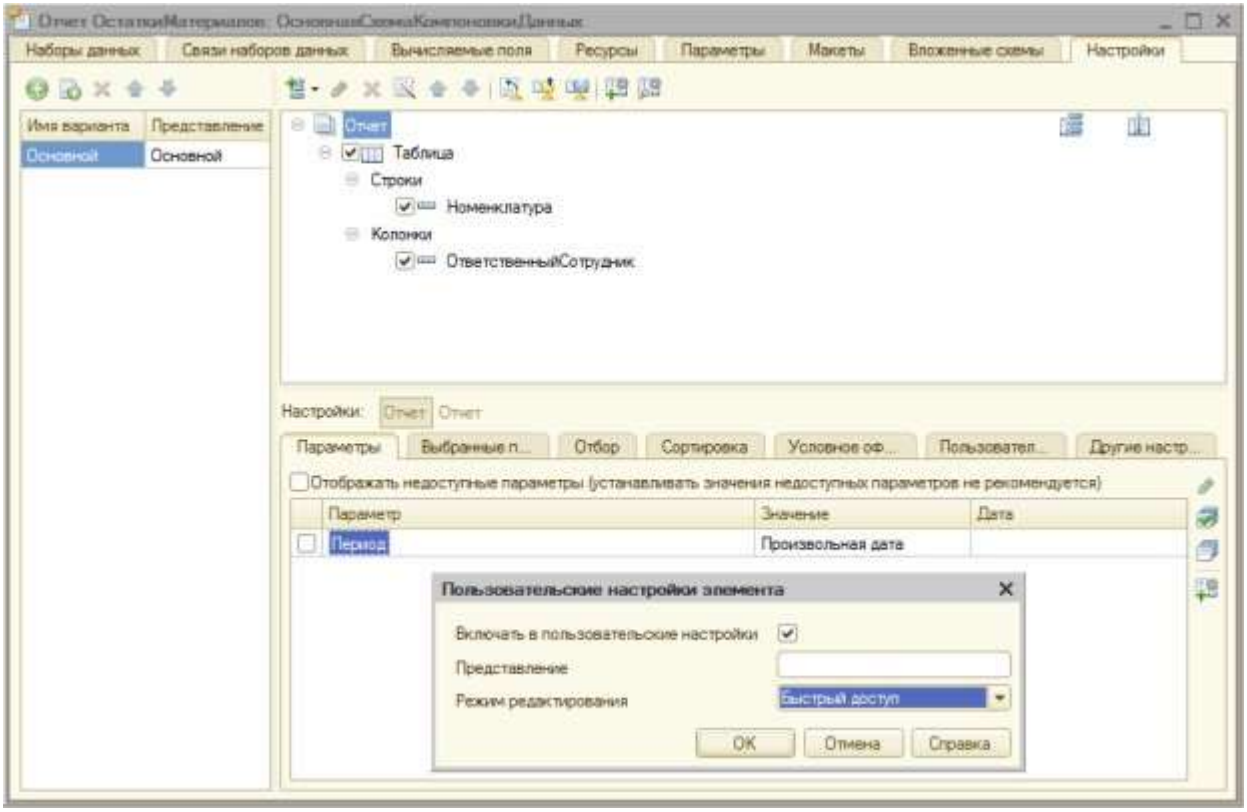
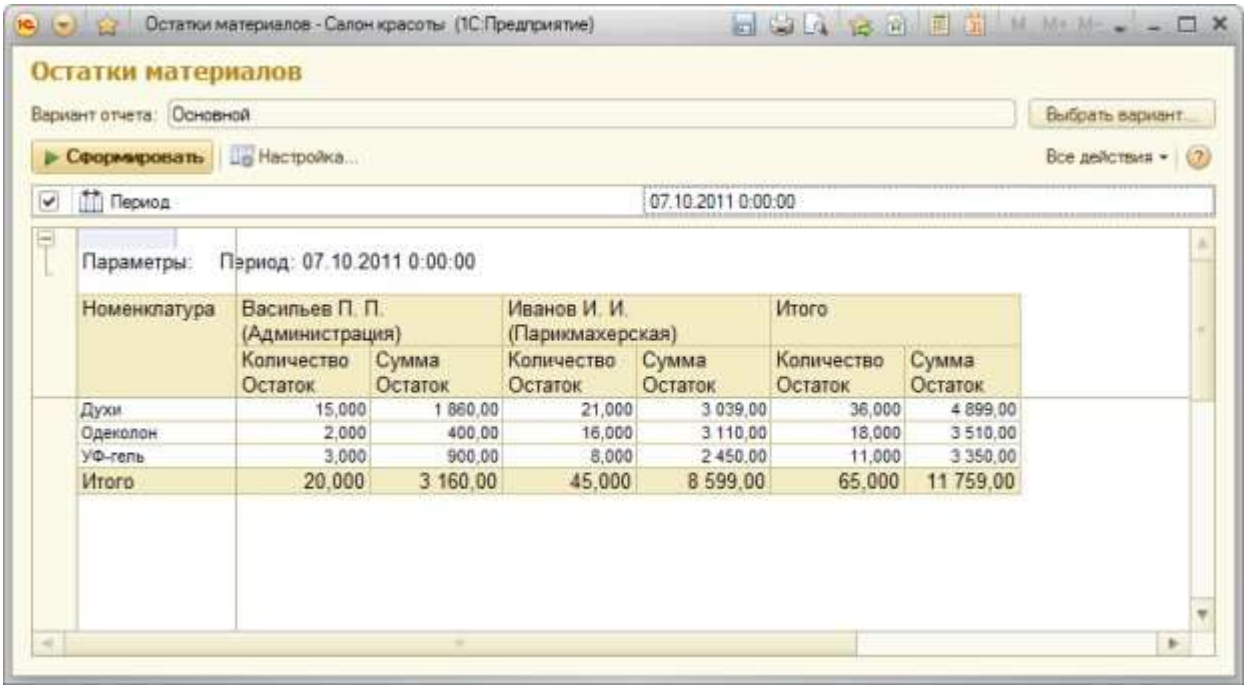


Рисунок 6.23. Настройка вывода параметра

Запустим систему в режиме «1С:Предприятие» и построим отчет, Рисунок 6.24.



Номенклатура	Васильев П. П. (Администрация)		Иванов И. И. (Парикмахерская)		Итого	
	Количество Остаток	Сумма Остаток	Количество Остаток	Сумма Остаток	Количество Остаток	Сумма Остаток
Духи	15,000	1 860,00	21,000	3 039,00	36,000	4 899,00
Одеколон	2,000	400,00	16,000	3 110,00	18,000	3 510,00
УФ-гель	3,000	900,00	8,000	2 450,00	11,000	3 350,00
Итого	20,000	3 160,00	45,000	8 599,00	65,000	11 759,00

Рисунок 6.24. Готовый отчет по количественным и суммовым остаткам материалов

Перед построением отчета, если мы хотим задать *параметр* **Период**, установим флаг перед этим параметром и выберем нужную дату.

Литература:

Официальный сайт интернет-униерситета «Интуит» (электронный ресурс), режим доступа http://www.intuit.ru/studies/courses/2318/618/lecture/17939.

Контрольные вопросы для самопроверки:

1. Какие виды регистров вы знаете?
2. Для чего нужны документы в «1С:Предприятие»?
3. Как автоматизировать работу с документами?

Задание к лабораторной работе

Создать документы и регистры в системе в соответствии с порядком, отраженным в теоретической части

Методические указания и порядок выполнения работы

Работа выполняется в точном порядке, как это указано на рисунках и в теоретической части

Индивидуальное задание

не предусмотрено

Требования к отчету и защите

1. Результатом выполнения лабораторной работы является сформированный в программе файл, содержащий выполненные задания. В ЭИОС результаты работы не выкладываются.

2. Планируется защита работы, где студент комментирует порядок выполнения заданий, а также отвечает на вопросы, представленные выше

ЛИТЕРАТУРА

1. Рудинский, И.Д. Технология проектирования автоматизированных систем обработки информации и управления: учеб. пособие / И. Д. Рудинский . - Москва: Горячая линия, 2011. - 303 с.

2. Советов, Б.Я. Теоретические основы автоматизированного управления: учеб. / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. - Москва: Высшая школа, 2006. - 462 с.

3. Хетагуров, Я.А. Проектирование автоматизированных систем обработки информации и управления (АСОИ и У): учеб. / Я. А. Хетагуров. - Москва: Высшая школа, 2006. - 223 с.

4. Малюк, А.А. Введение в защиту информации в автоматизированных системах: учеб. пособие / А. А. Малюк, С. В. Пазизин, Н. С. Погожин. - 3-е изд., стер. - Москва : Горячая линия-Телеком, 2005. - 146 с.

5. Интеллектуальные системы управления организационно-техническими системами / А. Н. Антамошин [и др.]. - Москва: Горячая линия-Телеком, 2006. - 160 с.