Методические указания

по организации практических занятий и самостоятельной работы

по МДК.02.02 Инструментальные средства разработки программного обеспечения

Специальность

09.02.07 Информационные системы и программирование

Квалификация Программист

2023

АНО СПО «КИТП»

Методические указания по **МДК.02.02 Инструментальные средства разработки программного обеспечения** разработаны с учетом ФГОС среднего профессионального образования специальности 09.02.07 Информационные системы и программирование, предназначены для студентов и преподавателей колледжа.

Методические указания определяют этапы выполнения работы на практическом занятии, содержат рекомендации по выполнению индивидуальных заданий и образцы решения задач, а также список рекомендуемой литературы.

Составитель (автор): преподаватель колледжа

Рассмотрены на заседании предметно-цикловой информационных технологий

Протокол № от « 30 » июня 2023 г

Председатель предметно-цикловой комиссии _____
личная подпись

и одобрены решением учебно-методического совета колледжа.

Протокол № от « 30 » июня 2023 г

Председатель учебно-методического совета колледжа _____
личная подпись

Рекомендованы к практическому применению в образовательном процессе

Рецензенты:

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Методические указания по МДК02.02 Инструментальные средства разработки программного обеспечения разработаны с учетом ФГОС среднего профессионального образования специальности 09.02.07 Информационные системы и программирование, предназначены для студентов и преподавателей колледжа.

Методические указания определяют этапы выполнения работы на практическом занятии, содержат рекомендации по выполнению индивидуальных заданий и образцы решения задач, а также список рекомендуемой литературы.

В предлагаемом пособии на практическом примере рассмотрены особенности разработки структуры проекта, тестирования и отладки проекта программного обеспечения.

Практическая работа№ 1 «Разработка структуры проекта»

Цель работы: научиться разрабатывать структуру проекта

ХОД РАБОТЫ

1. Выполним создание новой информационной базы

1.1 Создадим на своем сетевом диске (например w1924584) новую папку **1СП_Иванов** (фамилия Ваша).

1.2 Сделаем двойной щелчок по значку 1С Предприятие на Рабочем столе или воспользуемся командой Пуск > Все программы > 1С Предприятие 8> 1С:Предприятие.

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, Рисунок 1.1., выберем вариант создания информационной базы без конфигурации.

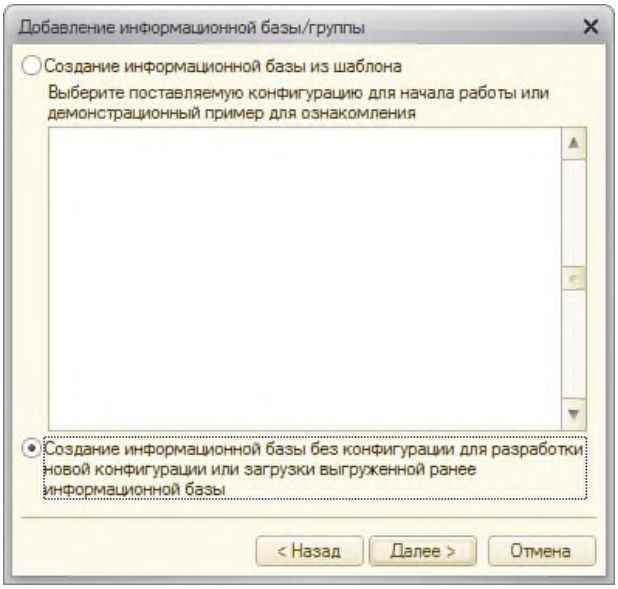


Рисунок 1.1- Создание информационной базы без конфигурации

1.3 Дадим информационной базе имя **1СП_Иванов (фамилия Ваша)**, зададим в качестве папки информационной базы папку с Вашего диска **Y:\ 1СП_Иванов**, остальные параметры оставим по умолчанию.

1.4 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.

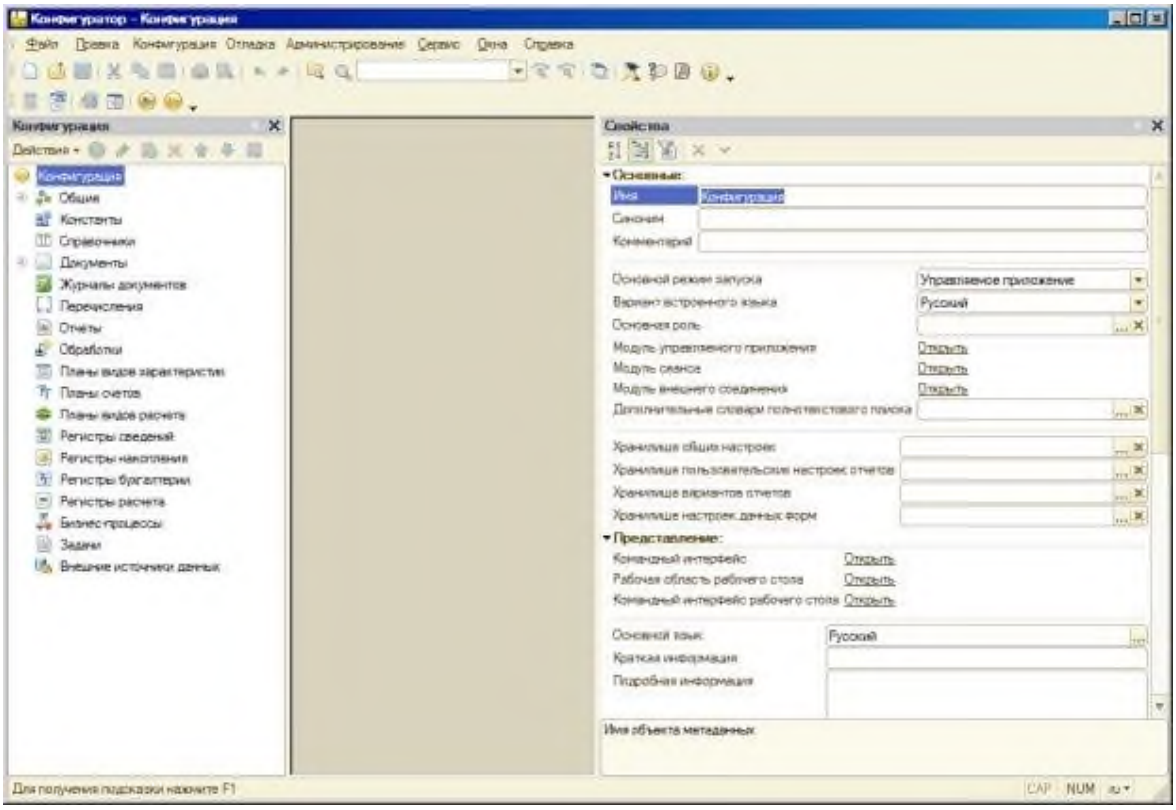


Рисунок 1.2- Свойства новой информационной базы

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 8.2.16 и др.

В качестве имени конфигурации введем **СалонКрасотыИванов** (фамилия Ваша), поле **Синоним** будет автоматически заполнено текстом **Салон красоты Иванов**. ИБД создана.

1.5 В окне дерева конфигурации представлены различные объекты. Кратко перечислим их. 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 8.3 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.

2. Повторный запуск системы, добавление информационной базы, настройка

2.1 Итак, у нас имеется информационная база, которая была создана ранее. Она расположена по адресу **Y:\ 1СП_Иванов**. Подключим эту базу для использования в 1С:Предприятие 8.3.

2.2 Сделаем двойной щелчок по значку 1С Предприятие на Рабочем столе или воспользуемся командой Пуск > Все программы > 1С Предприятие 8> 1С:Предприятие. В появившемся окне интерактивной программы запуска, (см. Рисунок 1.5), нажмем на кнопку **Добавить**.

2.3 Появится окно, которое устроено по принципу пошагового мастера, задающего вопросы пользователю для достижения некоторой цели. Первым вопросом здесь будет выбор между созданием новой информационной базы и добавлением существующей, Рисунок 1.3.

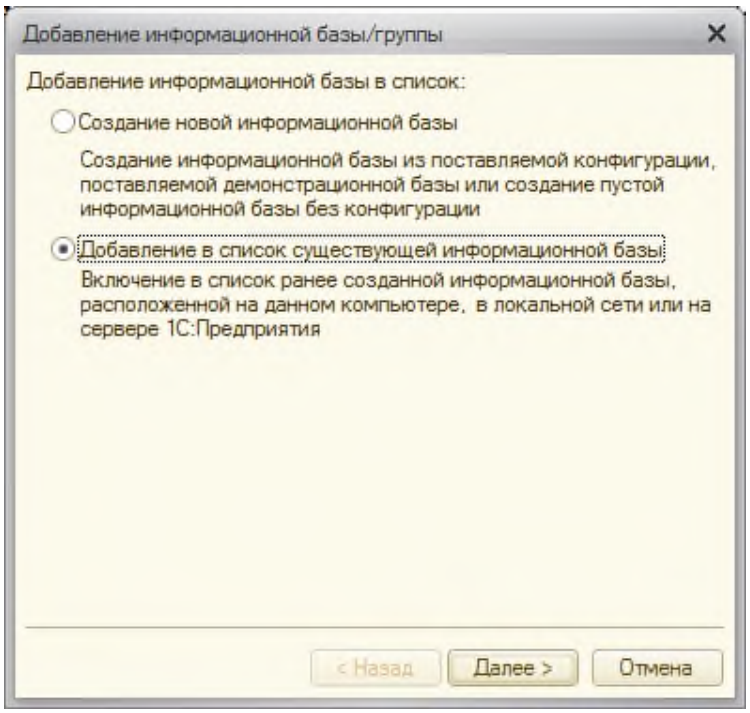


Рисунок 1.3- Добавление новой информационной базы

2.4 Мы собираемся добавить в окно запуска уже существующую информационную базу, выберем соответствующий вариант и нажмем **Далее**. Если нам нужно будет создать новую информационную базу и мы выберем вариант **Создание новой информационной базы**, мы сможем пойти одним из двух путей – либо создать новую пустую базу для разработки собственной конфигурации, либо создать базу из шаблона конфигурации.

2.5 Следующее окно, позволяет задать название и тип расположения базы – введем в поле названия **1СП_Иванов**, в типе расположения оставим значение по умолчанию –**На данном компьютере или на компьютере в локальной сети**.

2.6 Очередное нажатие на кнопку **Далее** приводит нас к окну, где нужно указать путь к информационной базе. Нас интересует папка, где расположен файл **1Cv8.1CD**, в нашем случае это папка **Y:\ 1СП_Иванов**.

2.7 Нажав еще раз на кнопку Далее мы переходим к очень важному этапу настройки информационной базы, Рисунок 1.4.

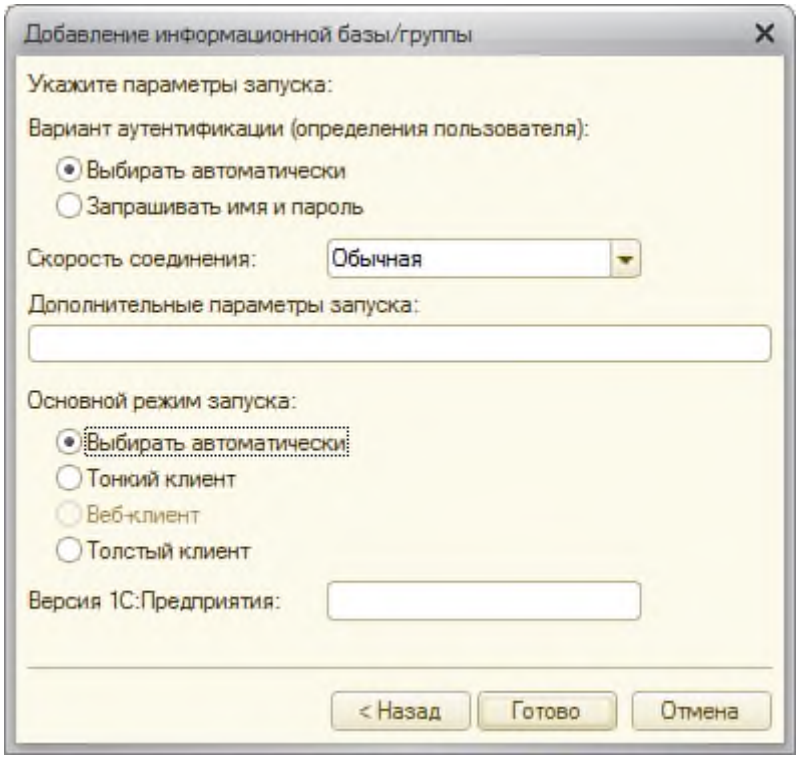


Рисунок 1.4- Настройка параметров запуска информационной базы

Здесь можно, в частности, указать основной режим запуска базы и версию 1С:Предприятия, необходимую для данной базы, введя ее в виде обычного текста в соответствующее поле, задав дополнительные параметры запуска, при необходимости – указав скорость соединения – это актуально для работы в режиме тонкого клиента через Интернет.

Мы оставим здесь параметры, установленные по умолчанию и нажмем на кнопку **Готово**. После этого информационная база появится в списке информационных баз, Рисунок 1.5.

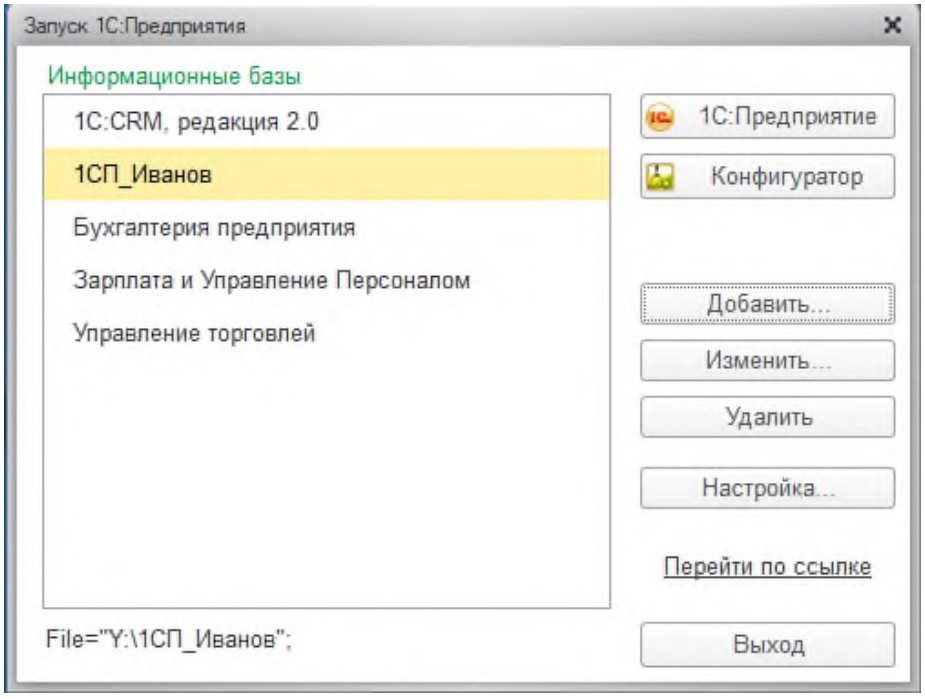


Рисунок 1.5- Информационная база в списке информационных баз

2.8 Нажимаем на кнопку **Конфигуратор**. Нажатие на кнопку **1С:Предприятие** приведет к запуску информационной базы в режиме 1С:Предприятие – в режиме, в котором работают пользователи. Причем, выбор приложения (версии и вида) будет зависеть как от настроек информационной базы, произведенных при ее добавлении или при изменении ее параметров, так и от настроек пользователя, под которым будет осуществлен вход в информационную базу.

Кнопка **Конфигуратор** предназначена для открытия базы в режиме конфигурирования – основном режиме, которым пользуются разработчики. Наша база содержит список пользователей с различными правами на работу с базой – для продолжения нам придется войти в систему под учетной записью с именем **Администратор**, без пароля, Рисунок 1.6.

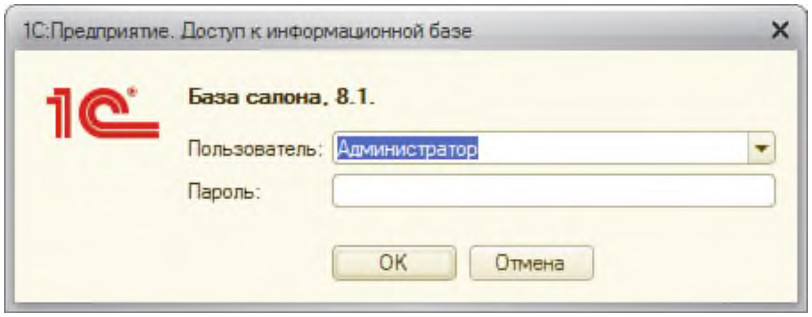


Рисунок 1.6- Вход в информационную базу под учетной записью Администратор

Нажатие кнопки **ОК** в этом окне приведет к открытию окна **Конфигуратора**. Сразу же выполним в этом окне команду меню **Конфигурация > Открыть конфигурацию**, эта команда приведет к открытию дерева конфигурации, окно которого расположено в левой части экрана, Рисунок 1.7.

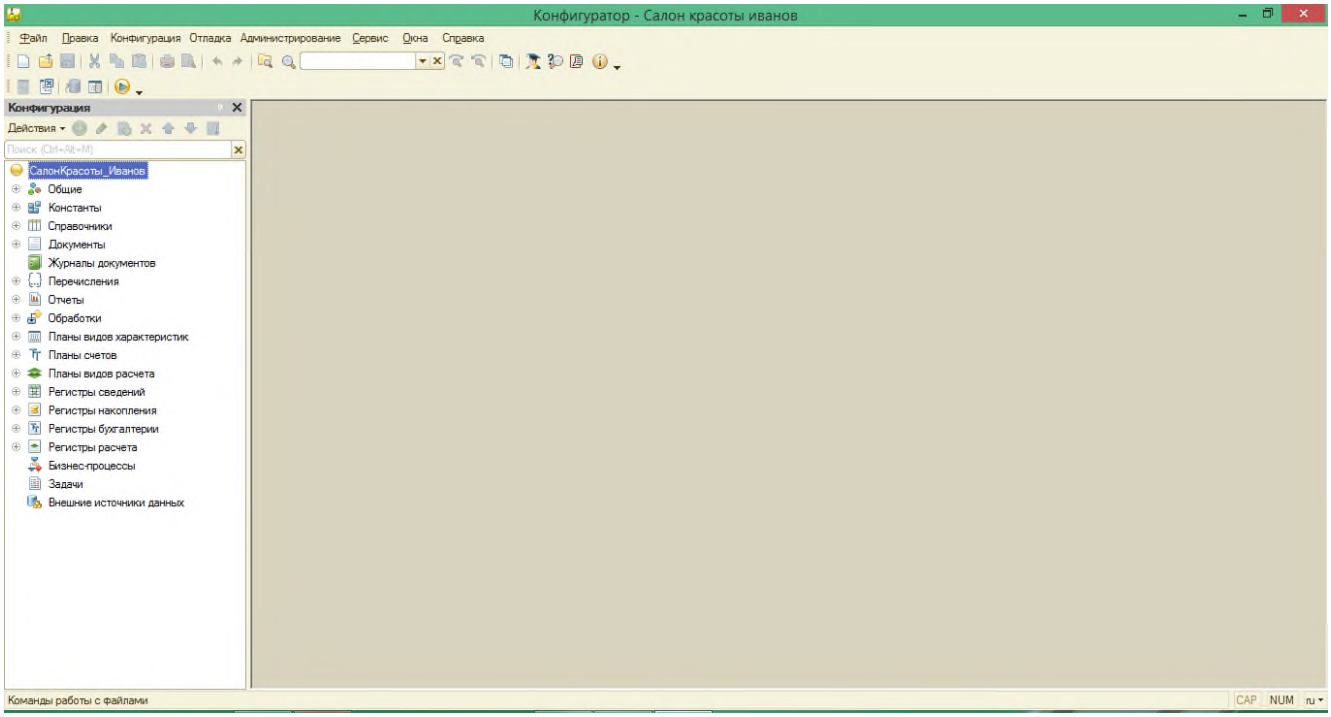


Рисунок 1.7- Информационная база открыта в Конфигураторе

1.9 Нажатием кнопки **1С:Предприятие**, или выполнить в Конфигураторе команду **Сервис > 1С:Предприятие** (так же можно воспользоваться клавиатурным сокращением Ctrl+F5 или

нажать кнопку **1С:Предприятие** на панели инструментов **Конфигурация**). После запуска информационной базы в режиме **1С:Предприятие** она сохраняет ранее существующую функциональность, то есть, позволяет пользователям продолжать работу.

Займемся разработкой новой конфигурации для режима "Управляемое приложение"

.

Практическая работа№ 2 «Разработка модульной структуры проекта»

Цель работы: научиться разработке модульной структуры проекта

ХОД РАБОТЫ

1. Создадим подсистемы – основу командного интерфейса управляемого приложения

1.1 Чтобы просмотреть подсистемы нужно открыть нашу конфигурацию и посмотреть ветвь дерева конфигурации **Общие > Подсистемы**, Рисунок 2.1. Точно так же можно будет обращаться к конфигурации в дальнейшем.

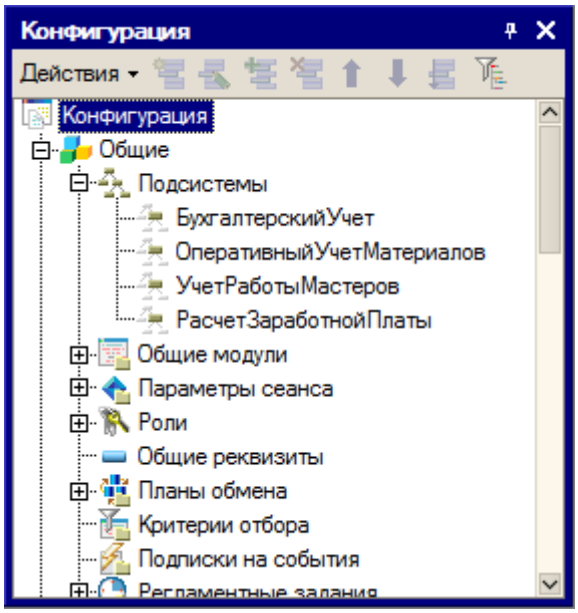


Рисунок 2.1- Подсистемы в информационной базе

Итак, в нашей старой конфигурации должны быть созданы подсистемы:

- «Бухгалтерский учет»;
- «Оперативный учет материалов»;
- «Учет работы мастеров»;
- «Расчет заработной платы».

1.2 Создадим те же подсистемы в новой конфигурации. 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 2.2.

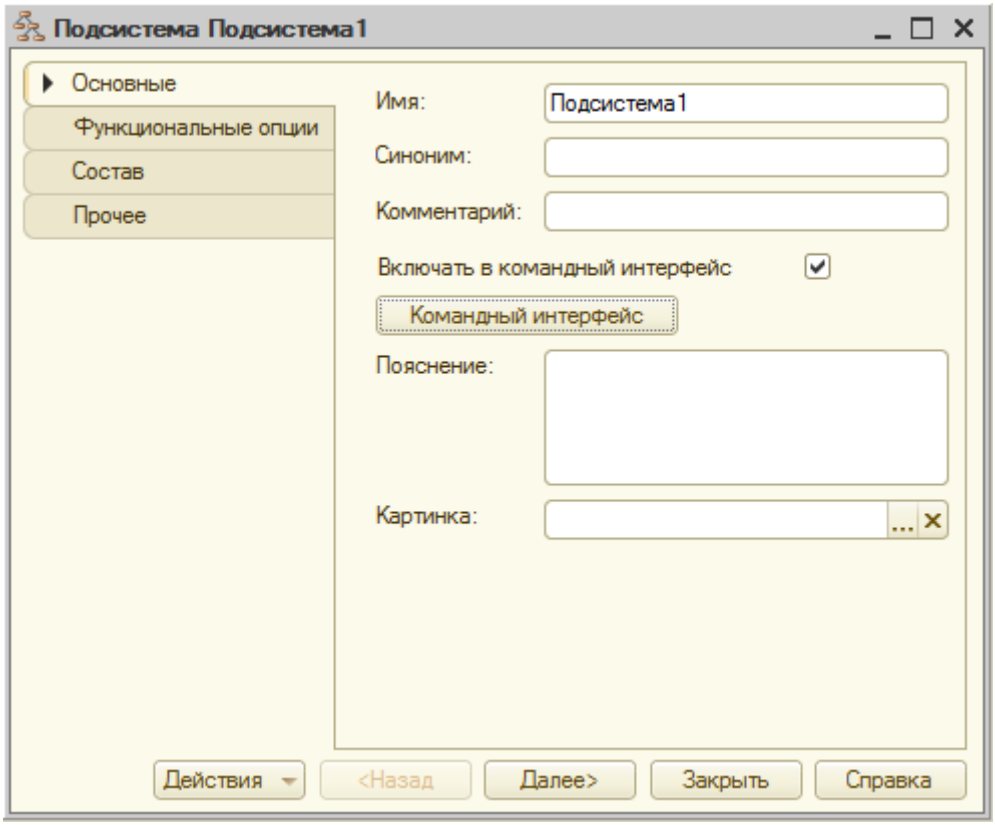


Рисунок 2.2- Окно редактирования объекта

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

1.3 Зададим следующие параметры для нашей новой подсистемы:

Имя: БухгалтерскийУчет

Синоним: Бухгалтерский учет

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.

1.4 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 2.3.

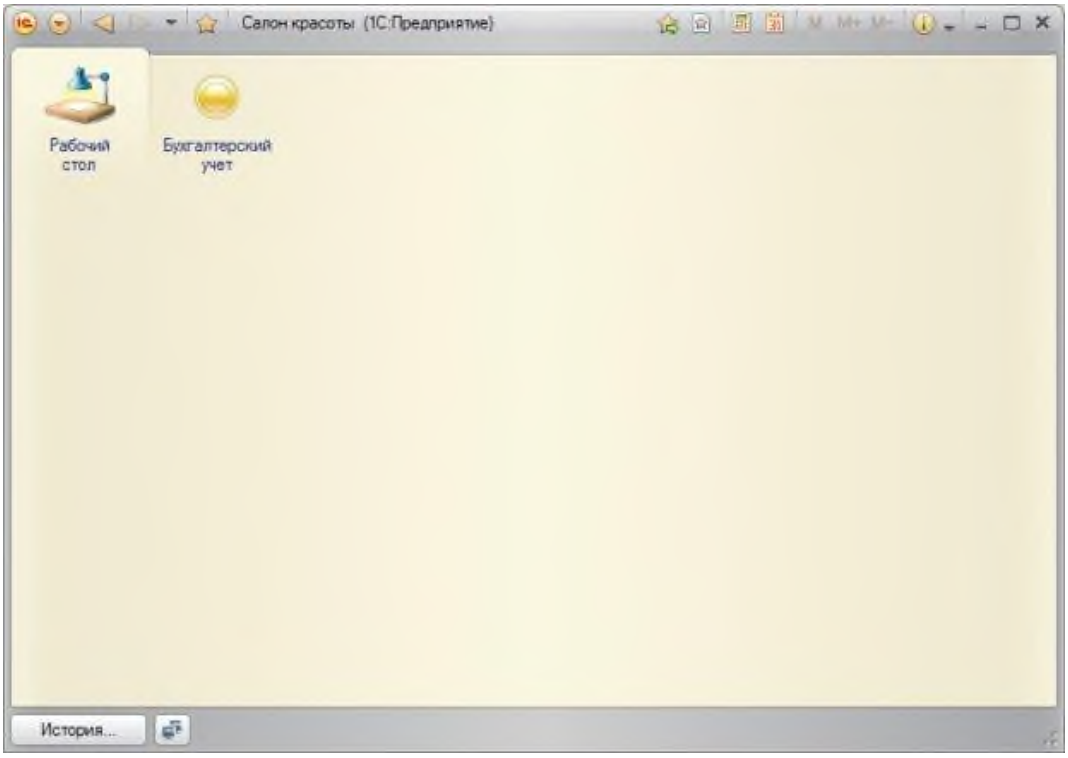


Рисунок 2.3- Разрабатываемая конфигурация в режиме 1С:Предприятие

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.

1.5 Обратите ваше внимание на кнопку **Главное меню**. Она открывает меню, содержащее стандартные для Windows-программ команды, Рисунок 2.4.

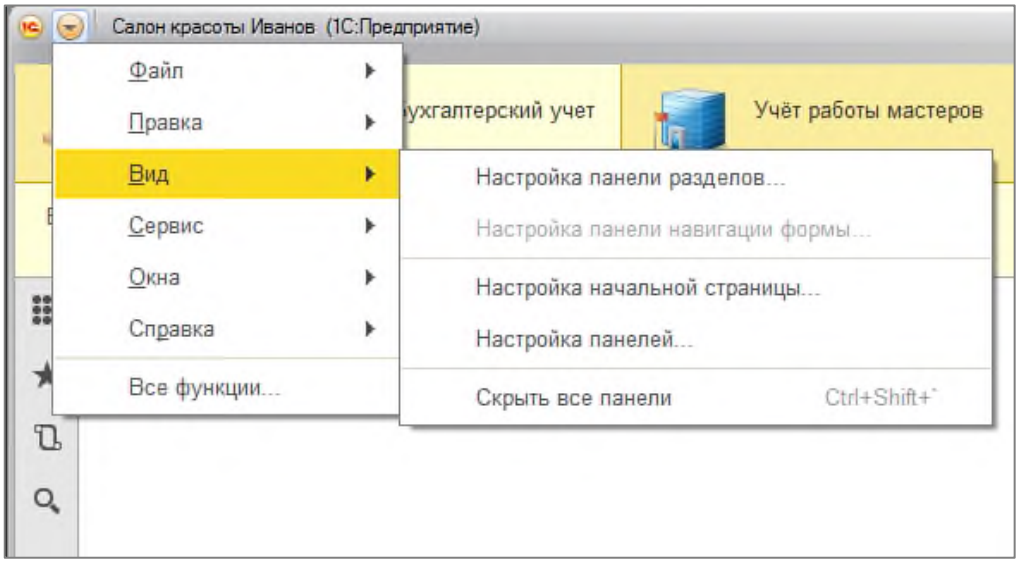


Рисунок 2.4 - Главное меню в режиме 1С:Предприятие

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 2.5., открывает доступ к дереву объектов конфигурации, позволяет использовать некоторые стандартные команды.

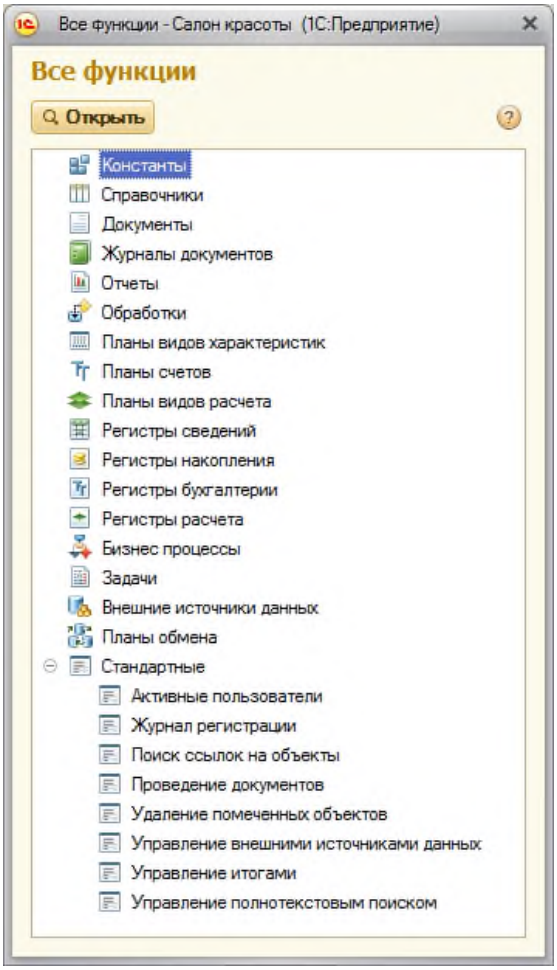


Рисунок 2.5- Окно Все функции

1.6 Вернемся в Конфигуратор, добавим к списку подсистем, которые следует создать, еще одну «**Администрирование**»

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.

Теперь создадим полный набор подсистем, Рисунок 2.6.

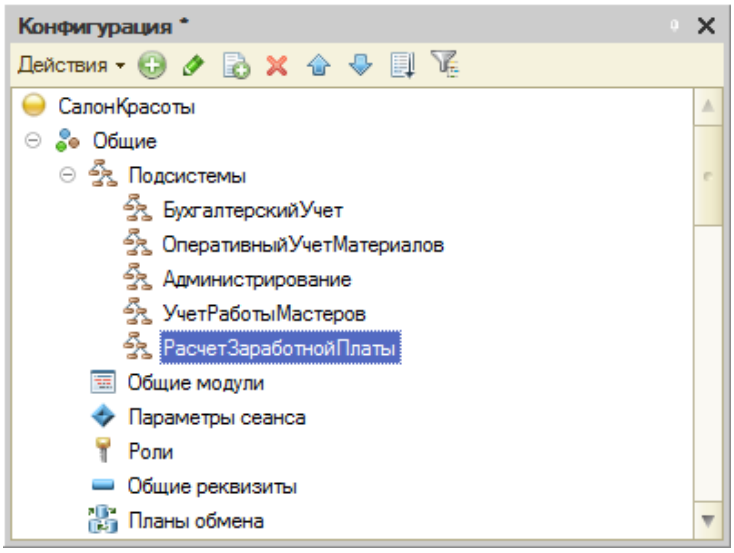


Рисунок 2.6- Набор подсистем конфигурации

Снова откроем конфигурацию в режиме 1С:Предприятие, Рисунок 2.7.

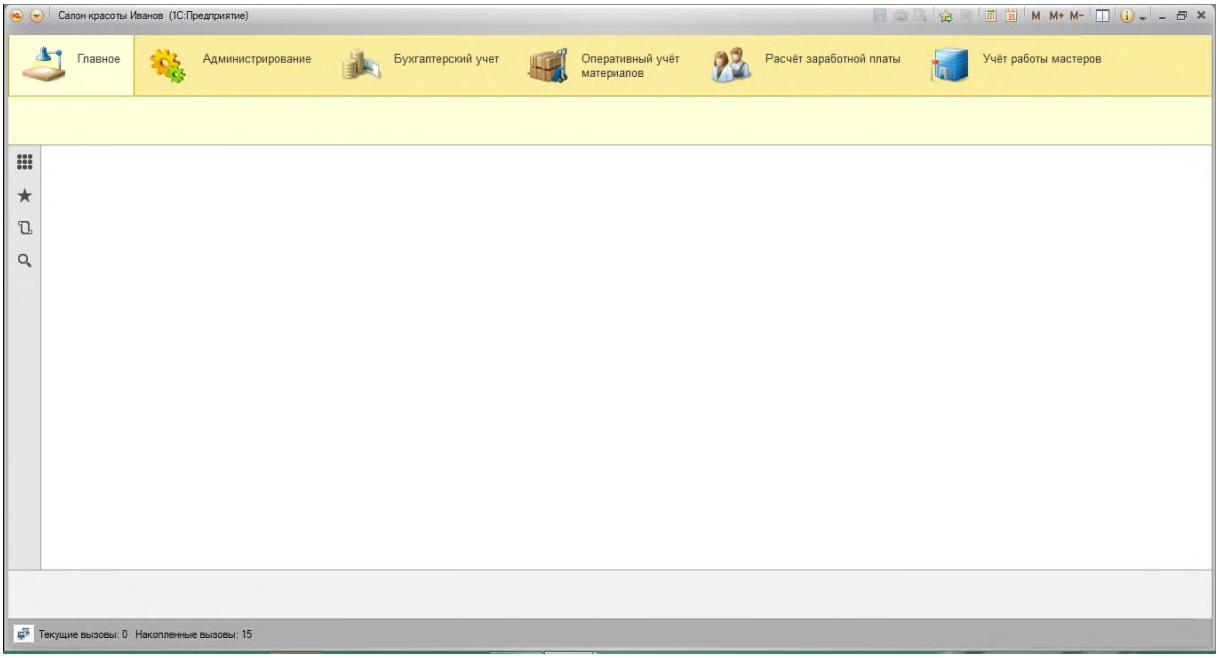


Рисунок 2.7- Панель разделов после добавления подсистем

1.7 Изменим порядок следования подсистем. Логично было бы расположить разделы нашего прикладного решения таким образом, чтобы раздел **Администрирование** оказался в правой части панели. Обычно наиболее часто используемые команды располагают левее и выше других. Можно заметить, что порядок расположения разделов не соответствует порядку расположения объектов **Подсистема** в дереве конфигурации (обратитесь к двум предыдущим рисункам для того, чтобы это увидеть). Для того чтобы изменить порядок следования подсистем в панели разделов нужно воспользоваться командой контекстного меню корневого объекта дерева конфигурации **Открыть командный интерфейс конфигурации**, Рисунок 2.8.

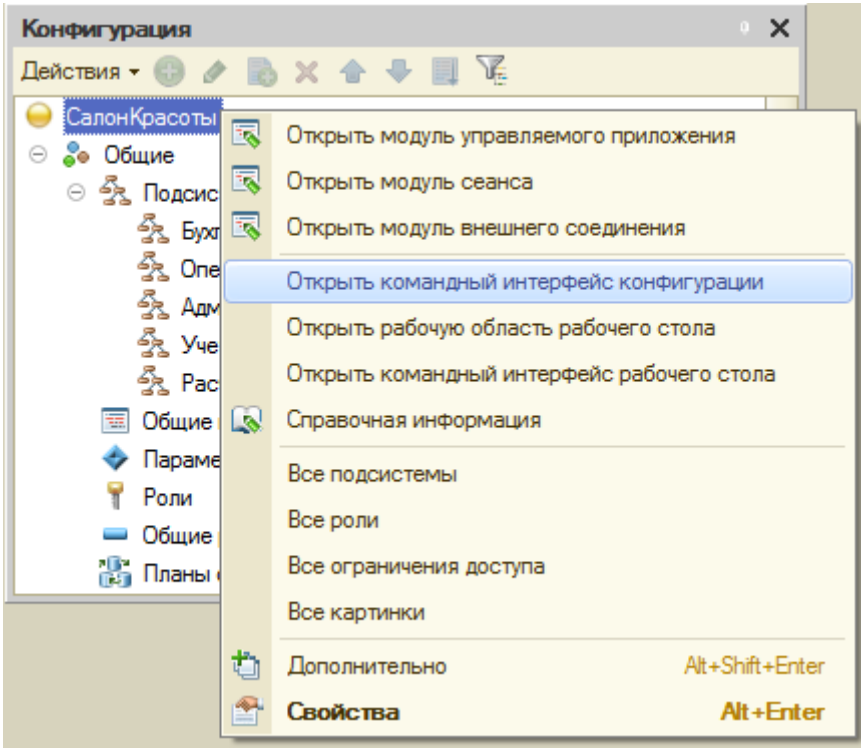


Рисунок 2.8- Открыть командный интерфейс конфигурации

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** так, чтобы они приняли вид, представленный на рисунке 2.9.

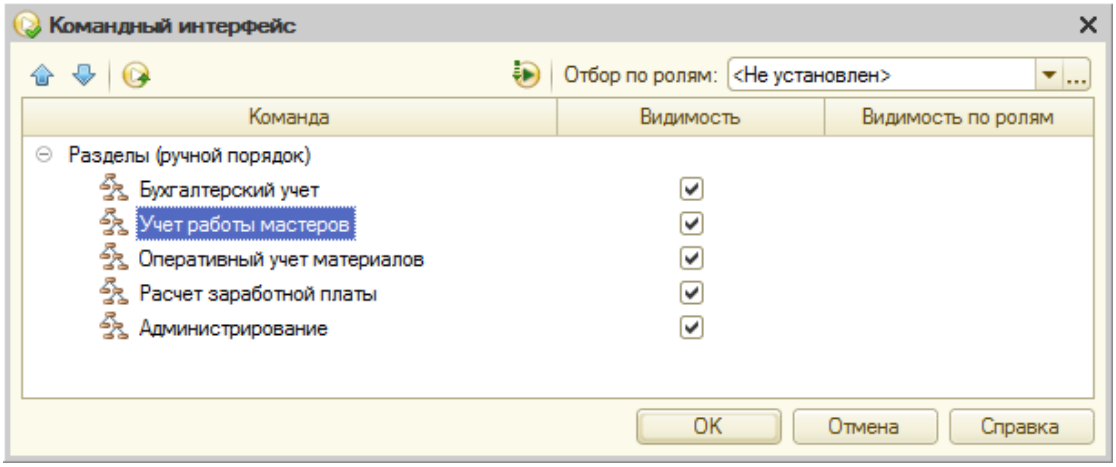


Рисунок 2.9- Настройка командного интерфейса

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

2. Рассмотрим теперь настройку видимости разделов по ролям.

2.1 Для демонстрации настройки видимости подсистем по ролям нам понадобятся два пользователя и две роли. Сначала создадим две роли – **Администратор** и **Сотрудник**. В дереве конфигурации перейдем в ветвь **Общие > Роли**, создадим новую роль, дадим ей имя **Администратор**.

Отметим права на доступ ко всем объектам (можно выполнить команду **Действия > Установить все права**), установим флажок **Устанавливать права для новых объектов**, Рисунок 2.10 Тем самым, при создании новых объектов, права на выполнение различных действий с ними будут добавляться к роли автоматически.

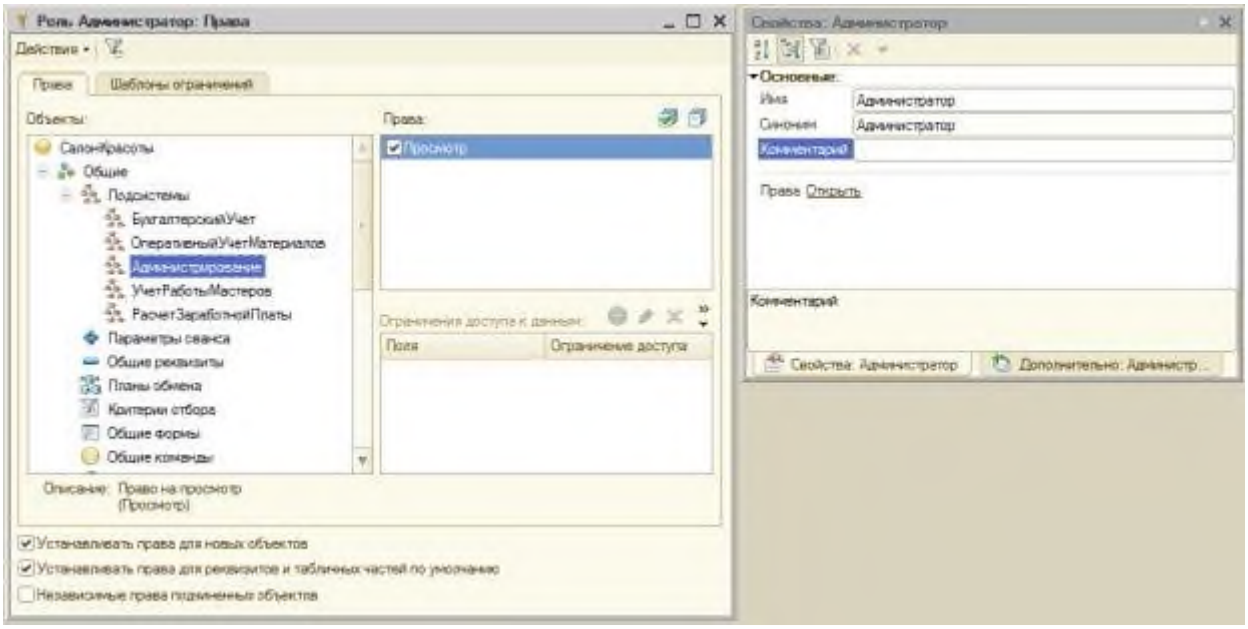


Рисунок 2.10- Настройка роли Администратор

2.2 Вторая роль, которая нас сейчас интересует, отличается от только что созданной названием – назовем ее **Сотрудник,** и тем, что у нее отключено право **Администрирование** у корневого объекта конфигурации. В свое время мы уделим настройке прав доступа к объектам больше внимания, сейчас сосредоточимся на настройках видимости закладок панели разделов.

Выполним уже знакомую вам команду контекстного меню корневого элемента дерева конфигурации **Открыть командный интерфейс конфигурации**, в окне **Командный интерфейс** снимем флаг напротив раздела **Администрирование** у роли **Сотрудник**, остальные флаги должны быть установлены, Рисунок 2.11.

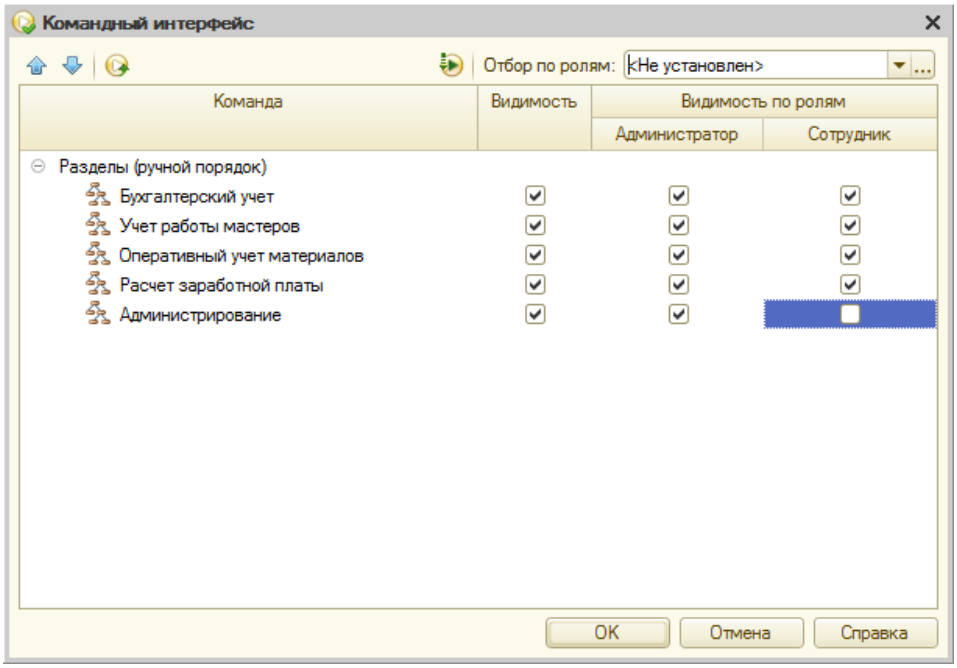


Рисунок 2.11- Настройка видимости разделов по ролям

2.3 Создадим двух пользователей конфигурации. В Конфигураторе выполним команду **Администрирование > Пользователи**, в появившемся окне **Список пользователей** нажмем на кнопку **Добавить**. На вкладке **Основные** окна **Пользователь** дадим первому пользователю имя **Администратор**, реквизит **Полное имя** будет заполнен автоматически.

Остальные параметры оставим в значении по умолчанию, что приведет к тому, что у пользователя будет пустой пароль и он будет отображаться в списке выбора пользователей (Рисунок 2.12.), хотя, в целях безопасности, пользователя с административными правами можно скрыть из списка выбора, да и имя "Администратор" лучше заменить на что-нибудь менее очевидное.

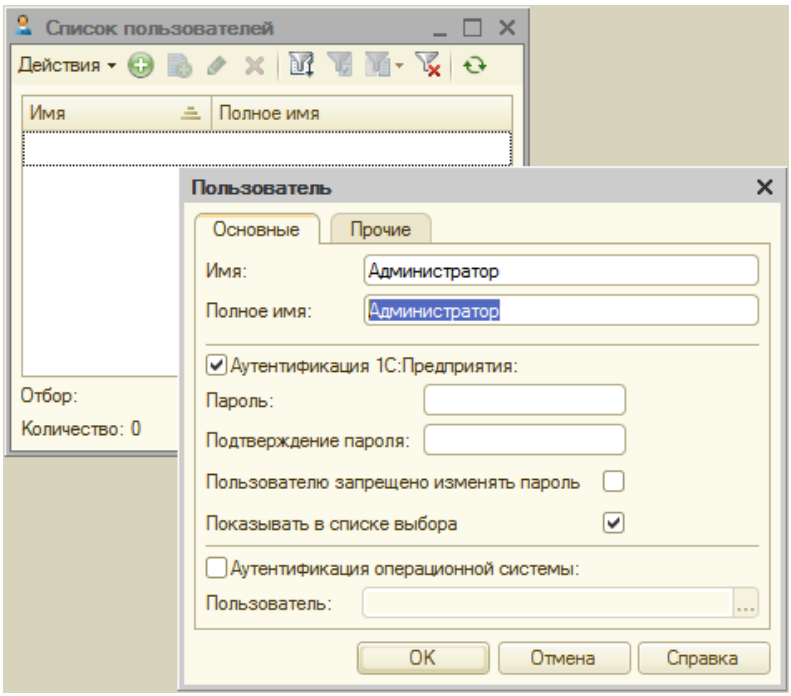


Рисунок 2.12- Создание нового пользователя

Перейдем на вкладку **Прочие** и в списке **Доступные роли** установим роль **Администратор**, Рисунок 2.13.

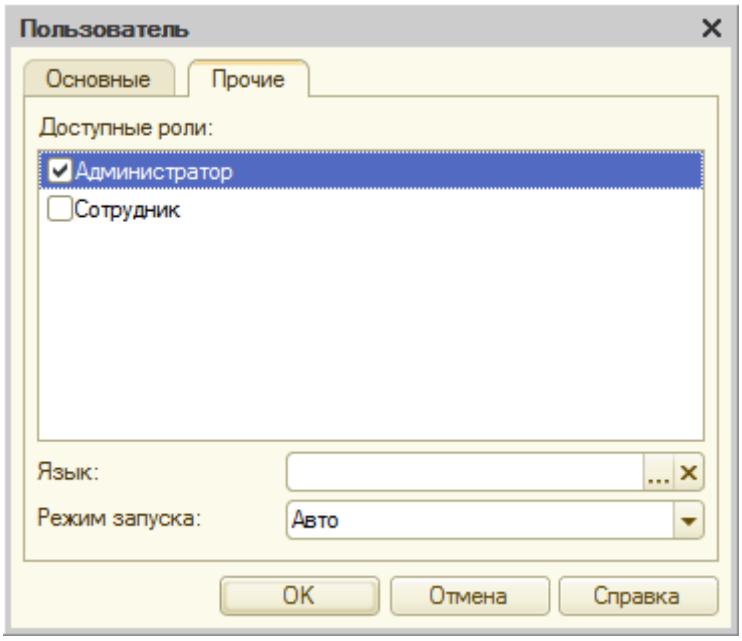


Рисунок 2.13- Настройка доступных ролей

2.4 Добавим в список второго пользователя, дадим ему имя **Директор**, на закладке **Прочие** установим среди доступных ролей роль **Сотрудник**. В итоге окно **Пользователи** примет такой вид, Рисунок 2.14.

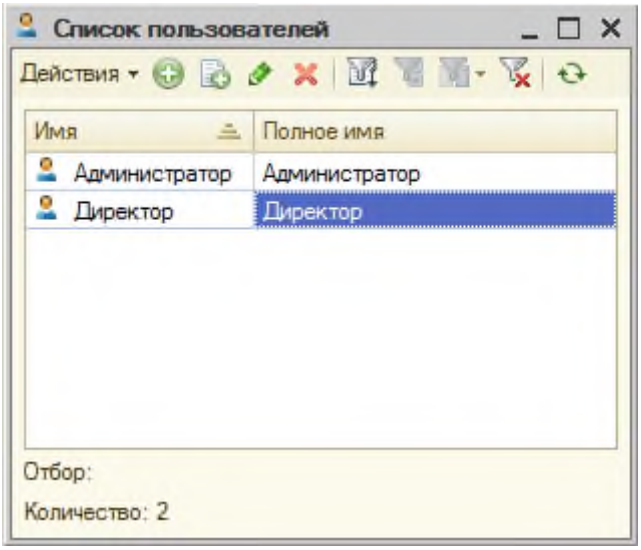


Рисунок 2.14- Список пользователей

2.5 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 2.15.

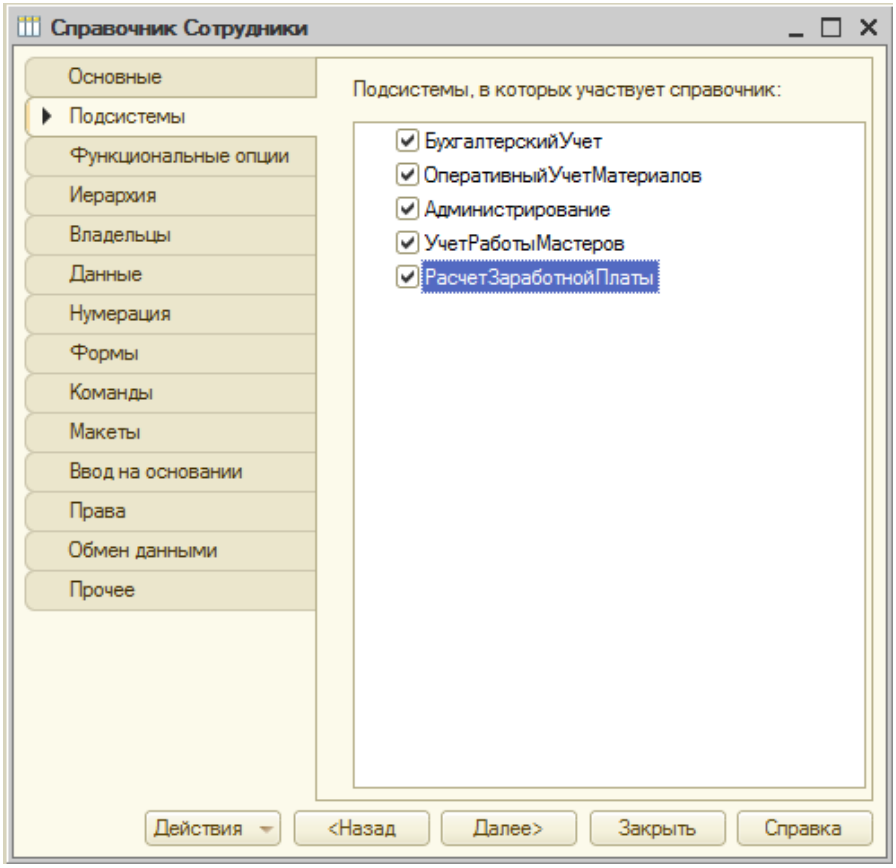


Рисунок 2.15- Новый справочник, добавленный во все подсистемы

2.6 Проверим результаты нашей работы в режиме 1С:Предприятие. После запуска конфигурации появится окно выбора пользователя, Рисунок 2.16.

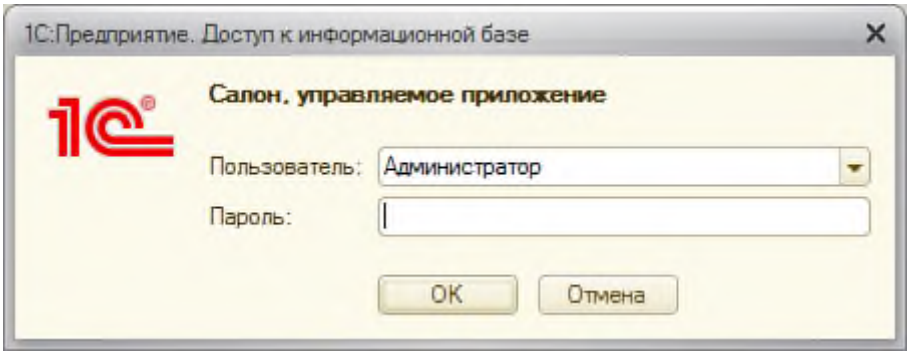


Рисунок 2.16 - Окно выбора пользователя

Выбрав в списке пользователей пользователя **Администратора** отметим, что панель разделов в его командном интерфейсе имеет закладки для каждой из подсистем. Выбрав пользователя **Директор**, можно заметить, что его панель разделов не содержит раздела **Администрирование**.

Вывод

В этой работе мы рассмотрели особенности загрузки 1С:Предприятие 8.3. – теперь на одном компьютере могут существовать ИБД, всеми ими можно пользоваться. Мы ознакомились с процессом создания новой информационной базы, приступили к разработке прикладного

решения, создав подсистемы, которые являются основой командного интерфейса, и настроив видимость разделов интерфейса по ролям для различных пользователей, воспользовавшись объектами Роль и Пользователь.

Практическая работа №3 «Разработка перечня артефактов и протоколов проекта»

Цель работы: научиться разработке перечня артефактов и протоколов проекта

.

ХОД РАБОТЫ

1. Создадим справочник «Организации»

Создание выполняем через кнопку добавить на дереве конфигурации, когда выделен объект «Справочники». 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.

Справочник **Организации** 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 3.1.

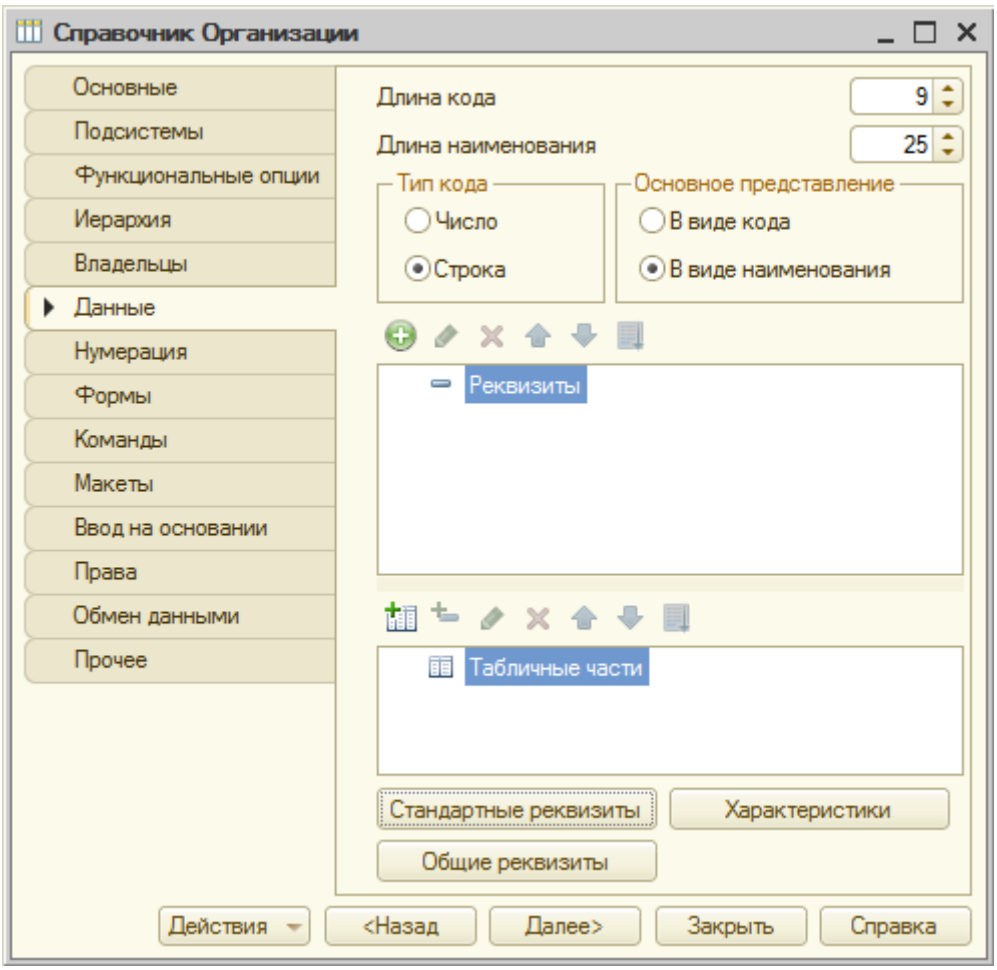


Рисунок 3.1- Настройка справочника

2. Ознакомимся со списком стандартных реквизитов можно, нажав на кнопку **Стандартные** реквизиты – появится окно, содержащее список таких реквизитов, Рисунок 3.2.

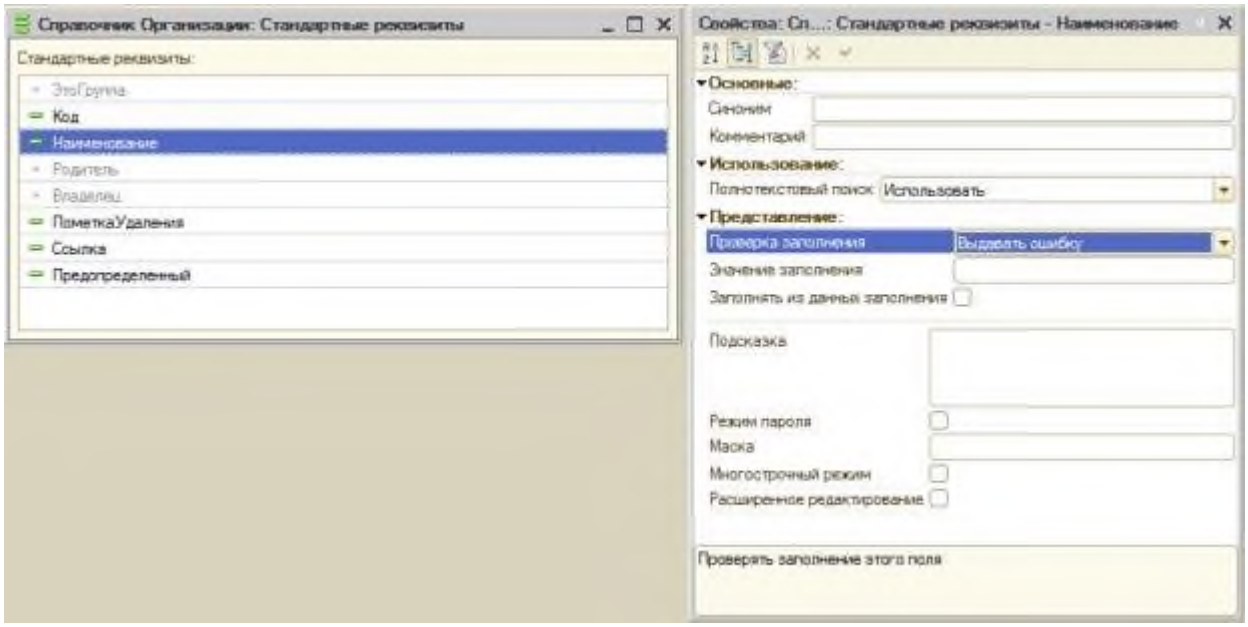


Рисунок 3.2- Стандартные реквизиты справочника и их свойства

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

3. Добавим реквизит в справочник. 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: Полное Наименование

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. Если нам нужны особые алгоритмы проверки содержимого поля перед записью элемента справочника, мы можем реализовать эти алгоритмы самостоятельно.

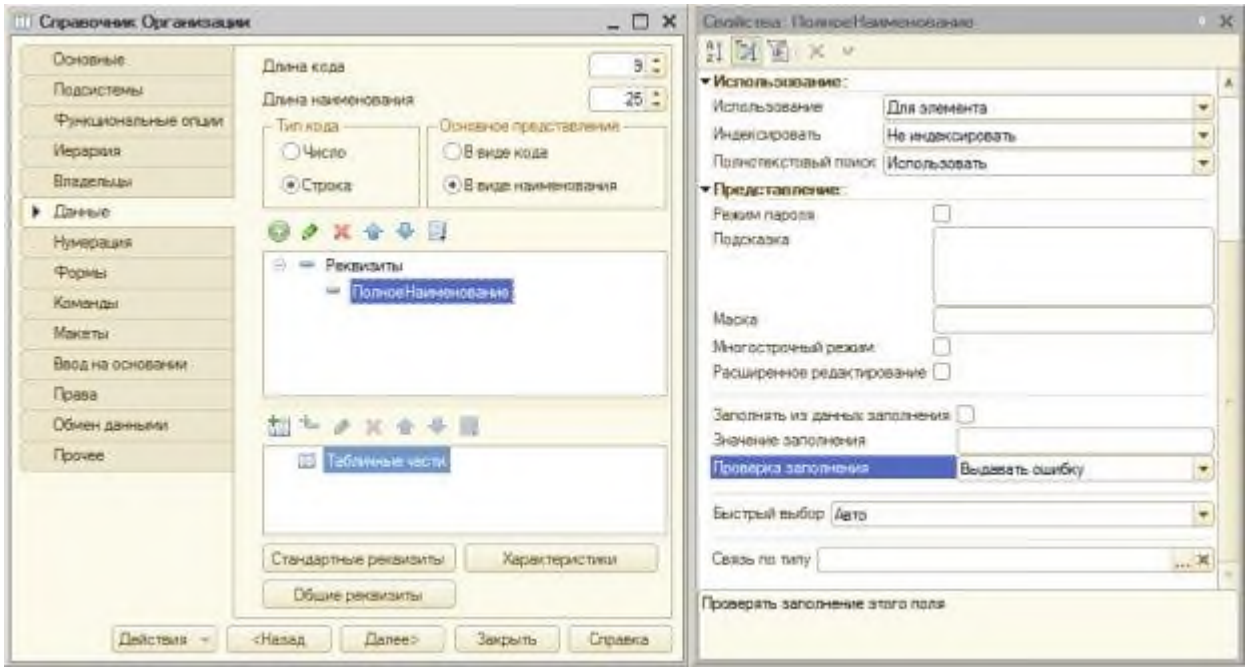


Рисунок 3.3- Настройка нового реквизита справочника

4. 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 3.4.

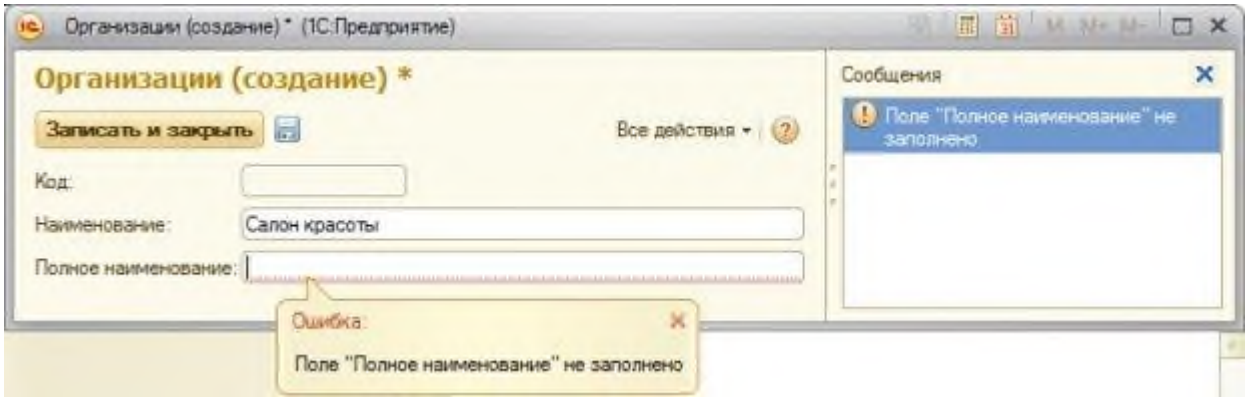


Рисунок 3.4- Сообщение об ошибке при попытке записи элемента справочника

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, Рисунок 3.5.

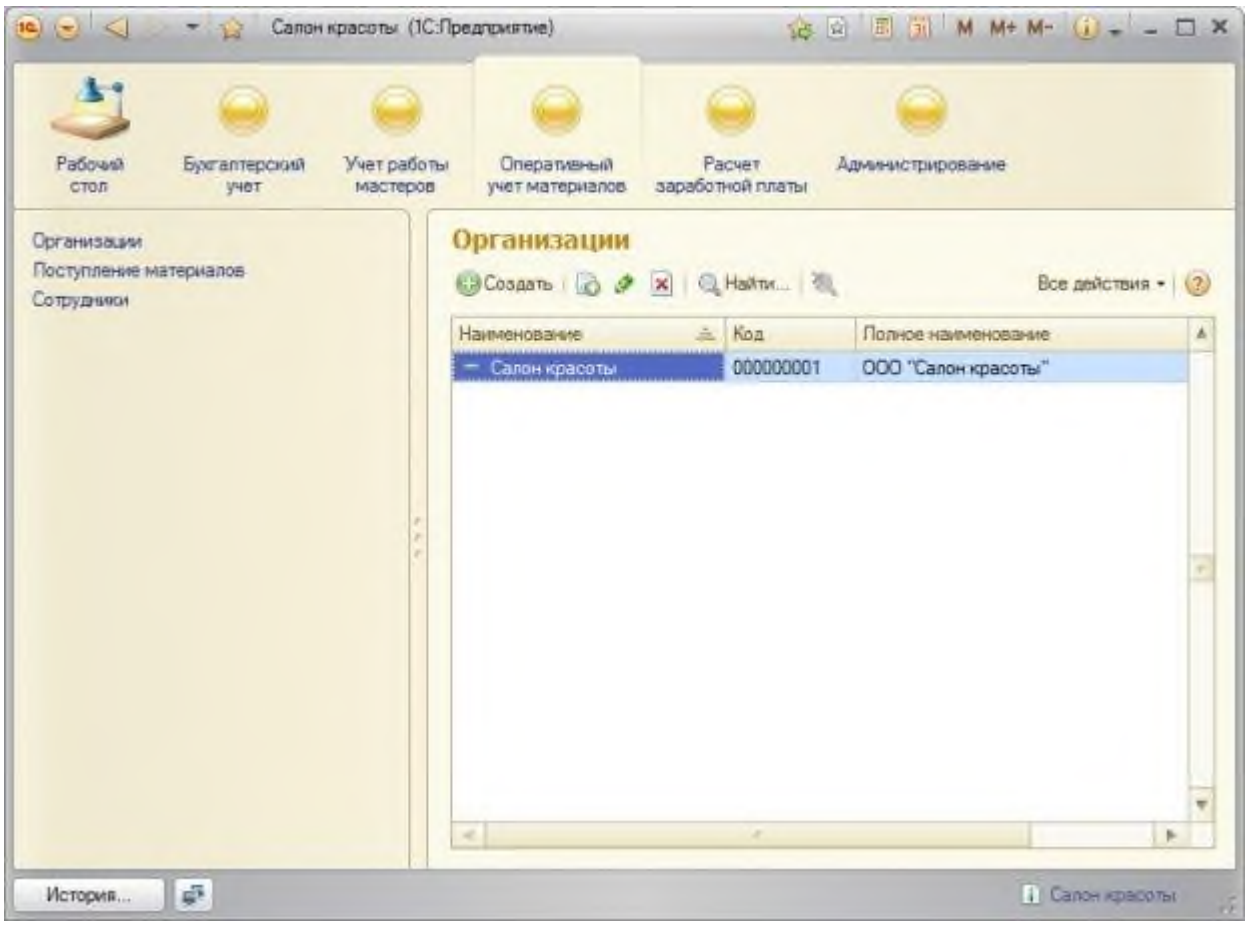


Рисунок 3.5- Новый элемент справочника в списке

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**.

5. Создадим справочник ФизическиеЛица

5.1 Создадим новый справочник, дадим ему имя **ФизическиеЛица**, включим его в состав подсистем **БухгалтерскийУчет**,**УчетРаботыМастеров** и **РасчетЗаработнойПлаты**.

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

Справочник **ФизическиеЛица** 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.

5.2 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 **РасчетЗаработнойПлаты**. На вкладке **Данные** окна редактирования объекта для перечисления задаются значения перечисления. Зададим два значения – **Мужской** и **Женский**, Рисунок 3.6.

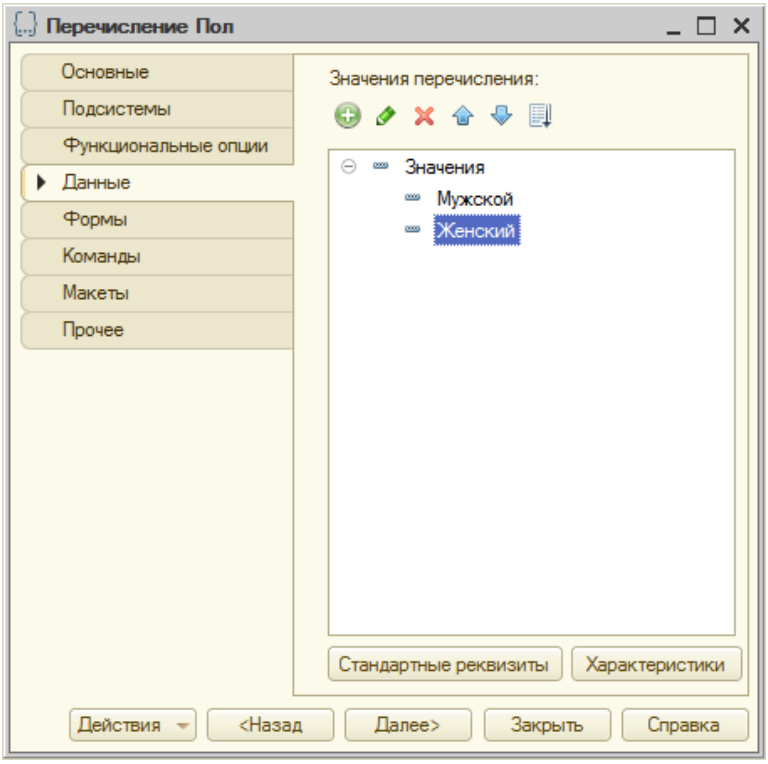


Рисунок 3.6- Создание перечисления Пол

6. Теперь создадим новый справочник – дадим ему имя **Районы**, включим в состав подсистемы **РасчетЗаработнойПлаты**, 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 3.7.

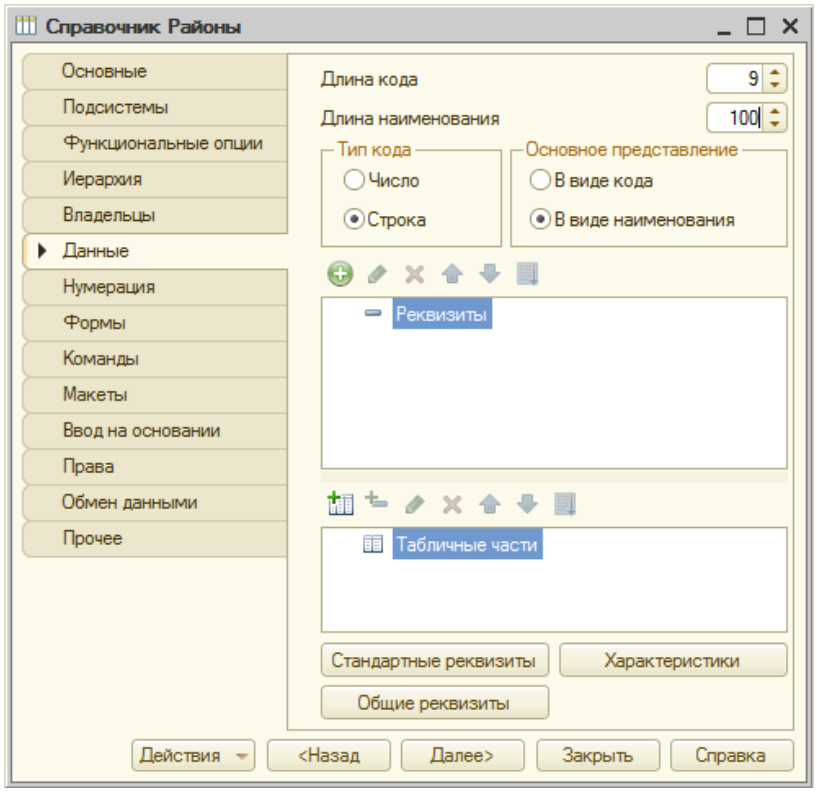


Рисунок 3.7 - Создание справочника Районы

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 3.8.

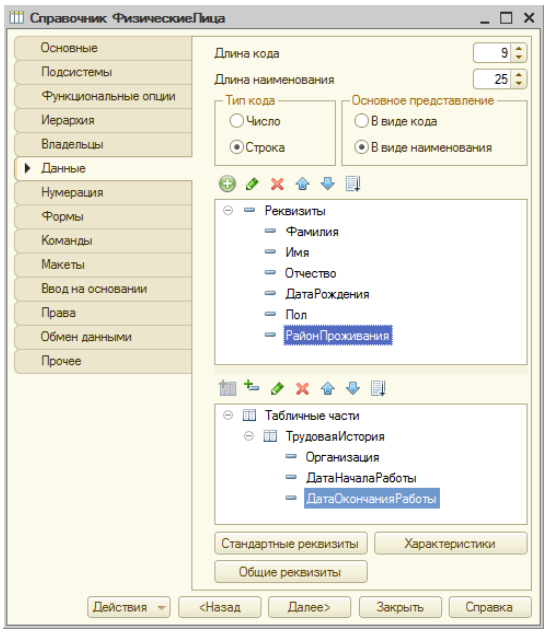


Рисунок 3.8- Состав справочника ФизическиеЛица

7. Данные перечисления **Пол** представлены на рисунке 3.9

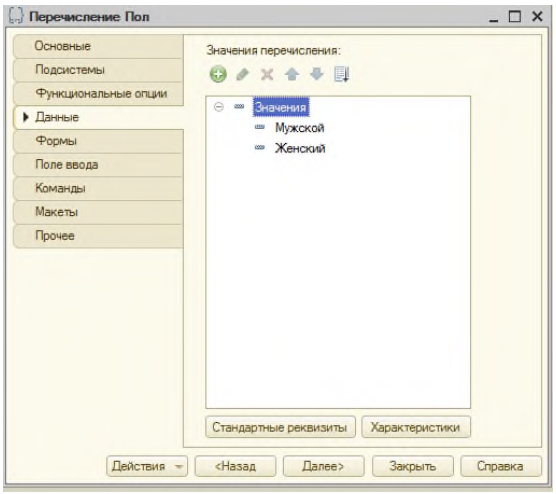


Рисунок 3.9 - Данные перечисления «**Пол**»

8. Просмотрим стандартную форму справочника. Если мы попытаемся открыть справочник в режиме 1С:Предприятие – с ним можно будет работать, так как система автоматически сгенерирует его форму, Рисунок 3.10. – 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.

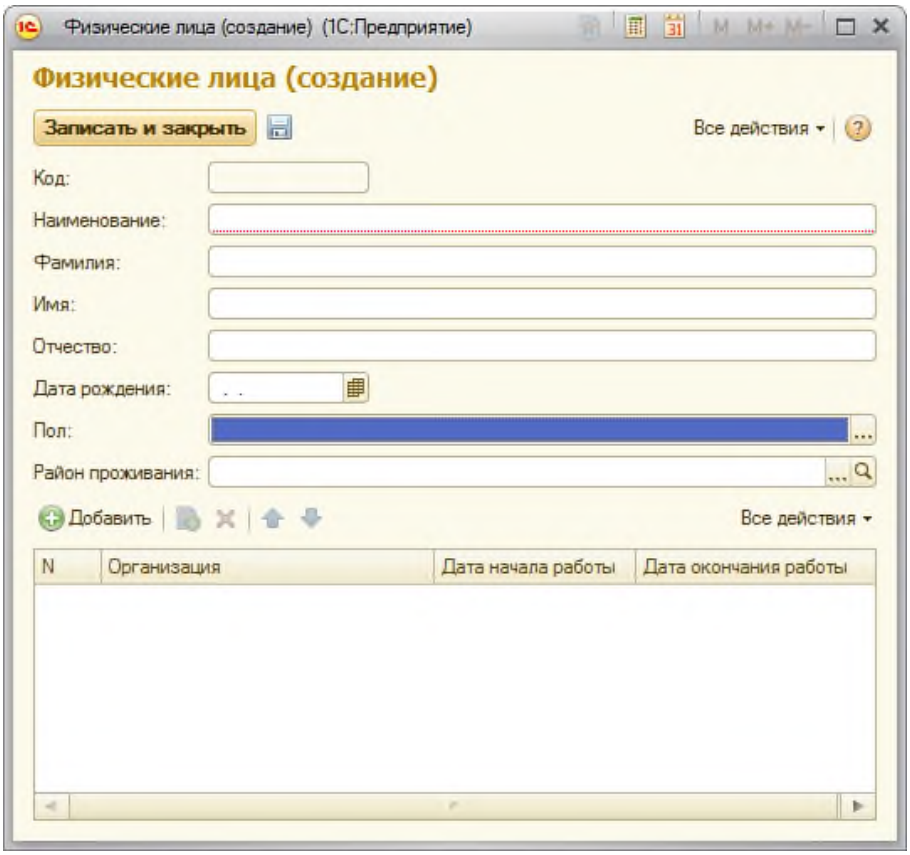


Рисунок 3.10- Форма справочника, сгенерированная автоматически

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.

9. Разработаем форму справочника ФизическиеЛица

9.1 Откроем закладку **Формы** окна редактирования справочника **ФизическиеЛица**. Можно отметить, Рисунок 3.11, что ни одной формы не задано – то есть все они создаются системой автоматически. Нам же нужна собственная форма **элемента** справочника.

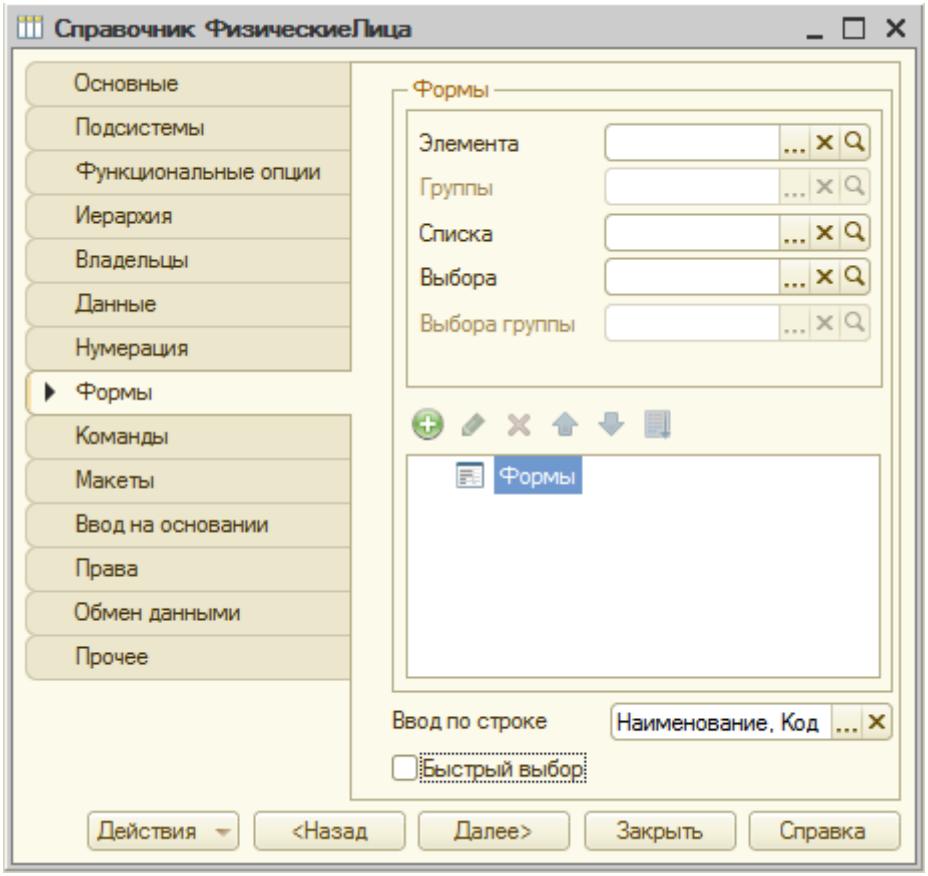


Рисунок 3.11- Вкладка Формы окна редактирования объекта

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 3.12.

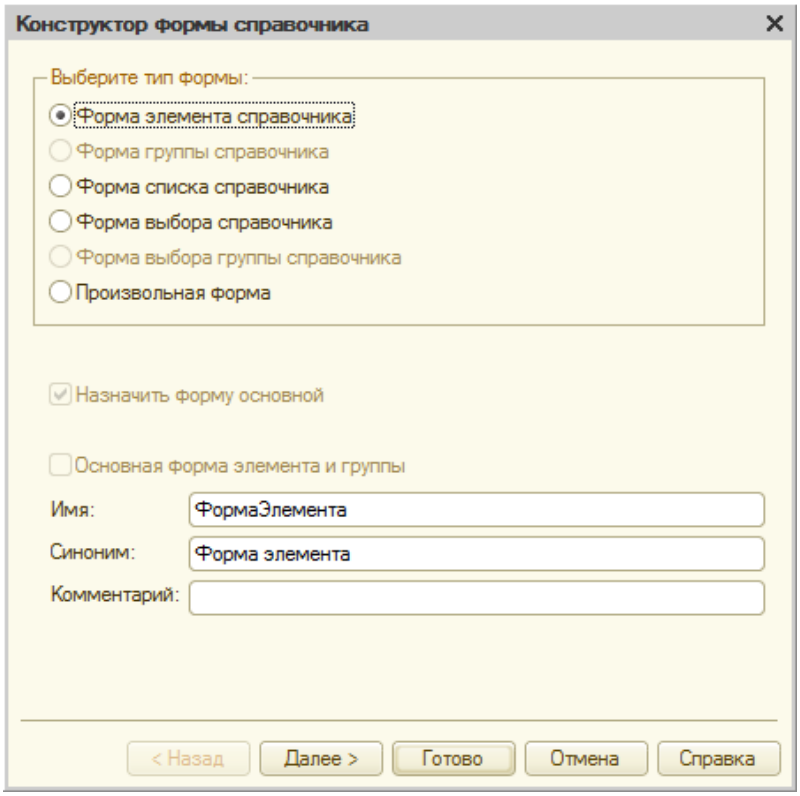


Рисунок 3.12- Первое окно конструктора форм справочника

9.2 В следующем окне, Рисунок 3.13., 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**.

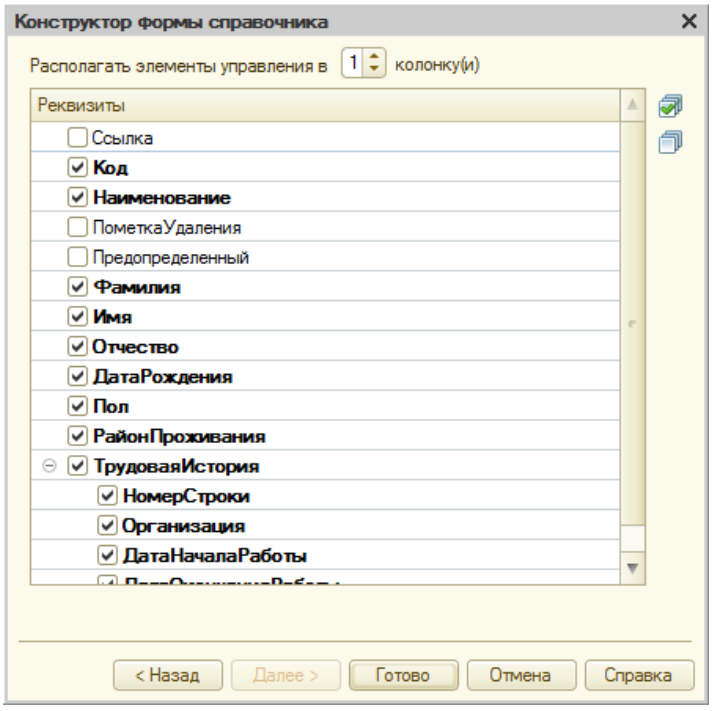


Рисунок 3.13- Второе окно конструктора форм справочника

9.3 После этого нужно открыть окно редактора форм для формы элемента справочника, Рисунок 3.14. Ранее мы уже сталкивались с этим окном, теперь рассмотрим его подробнее.

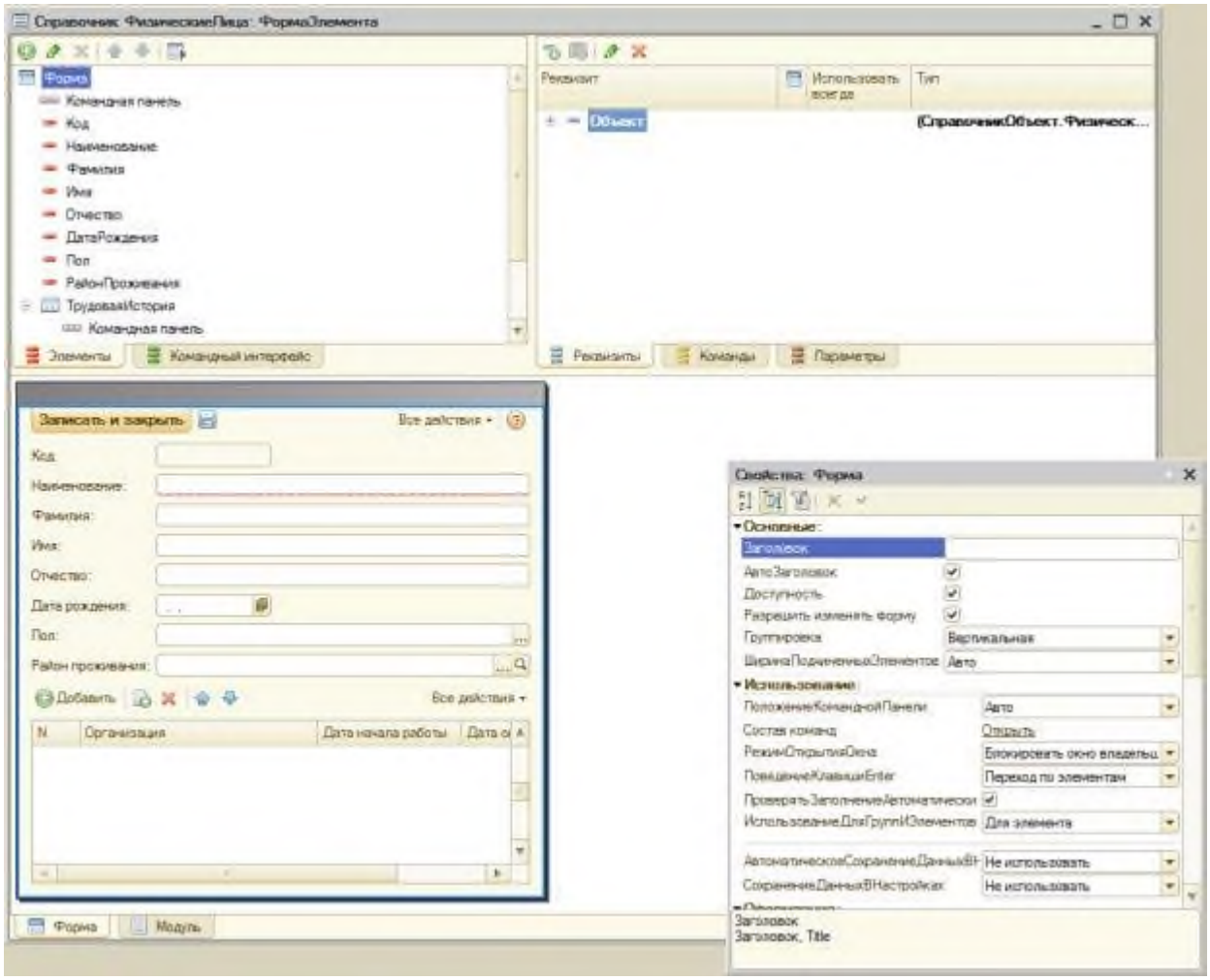


Рисунок 3.14- Окно редактирования формы элемента справочника

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

9.4 Изучим окно формы. 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 3.15, 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).

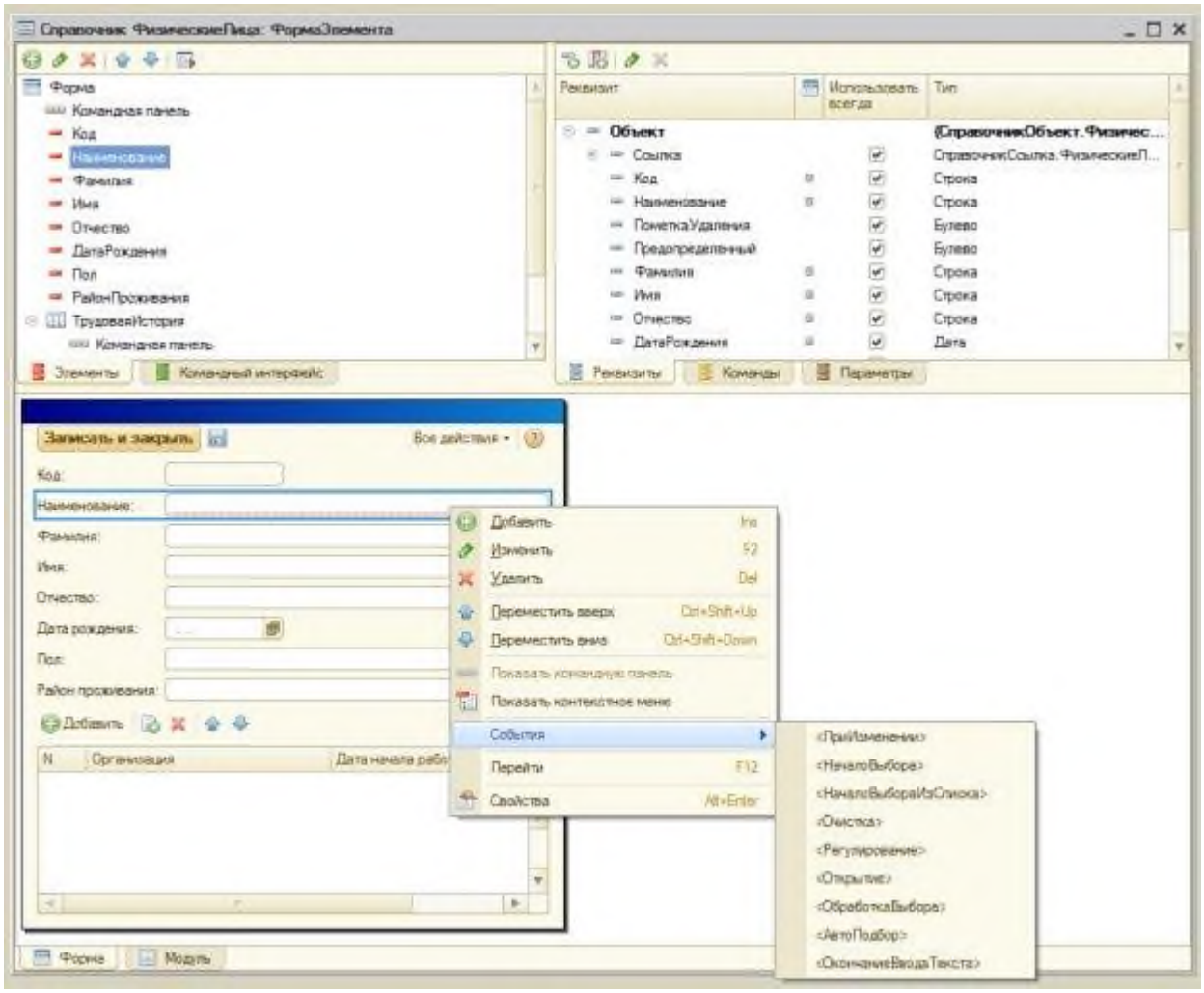


Рисунок 3.15- Работа с элементами формы

Редактор реквизитов представлен вкладкой **Реквизиты** (Рисунок 3.15). 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 3.16.) содержит стандартный набор команд – в нашем случае – стандартный для формы и табличного поля, размещенного на форме.

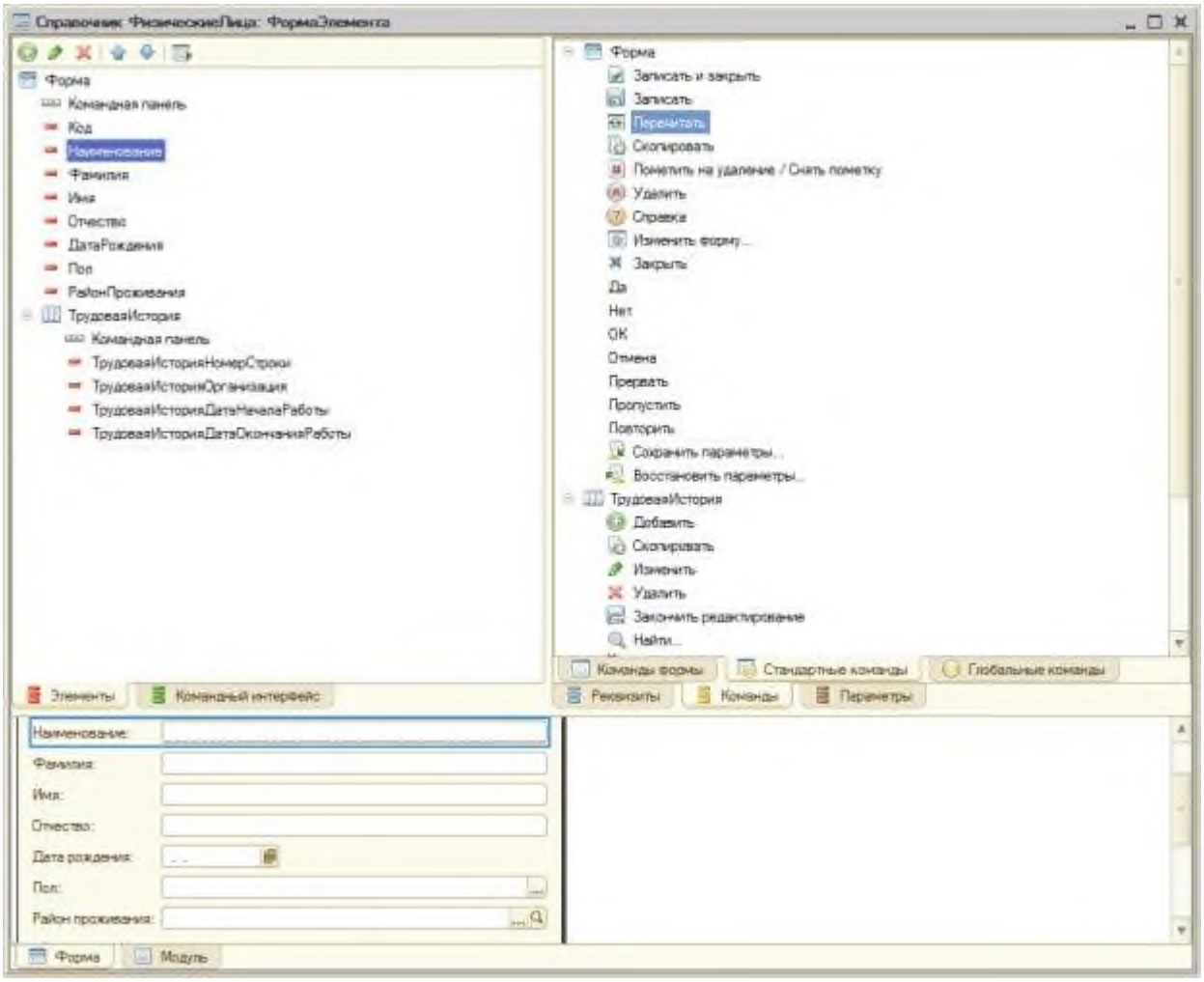


Рисунок 3.16- Стандартные команды

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

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

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

10. 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 3.17. 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**.

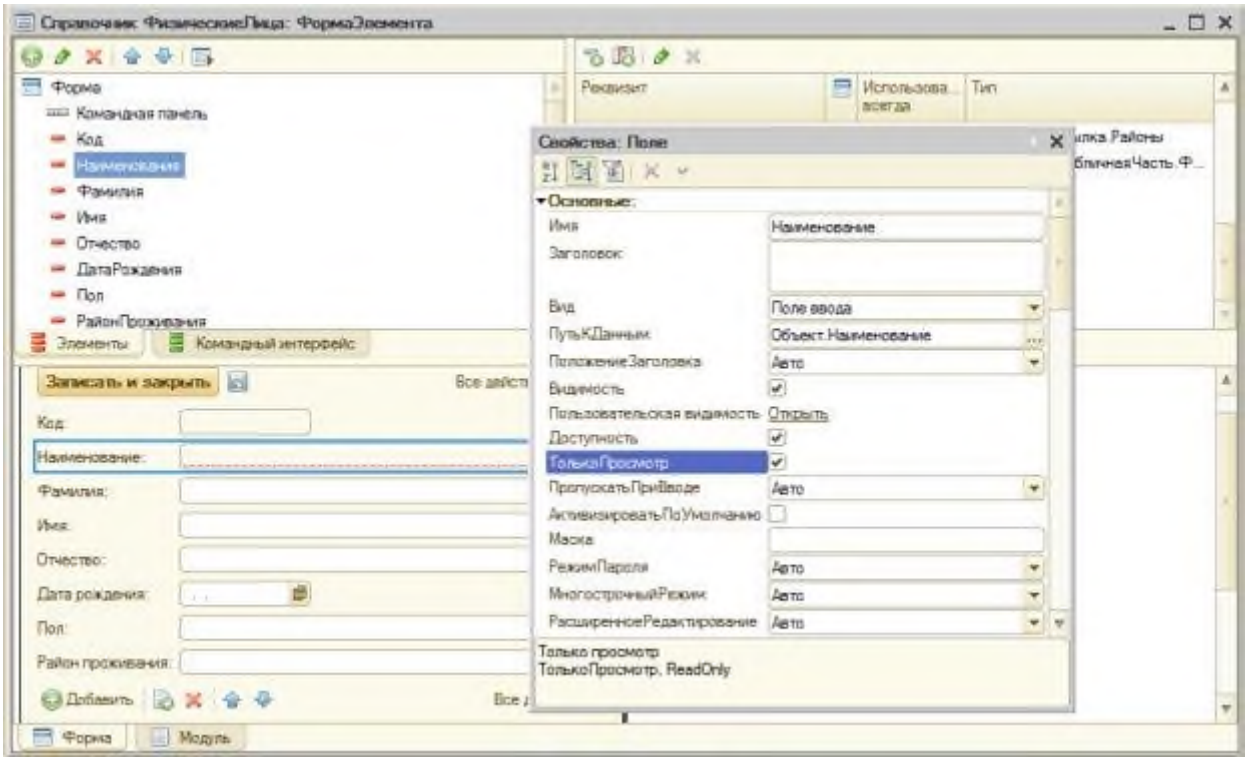


Рисунок 3.17- Настройка элемента Наименование

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

Практическая работа №4 «Настройка работы системы контроля версий»

Цель работы: научиться выполнять настройку работы системы контроля версий

ХОД РАБОТЫ

1. Применим 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.

1.1 Выделим форму в окне Элементы, откроем окно ее свойств и рассмотрим группу свойств **События**, Рисунок 4.1. 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.

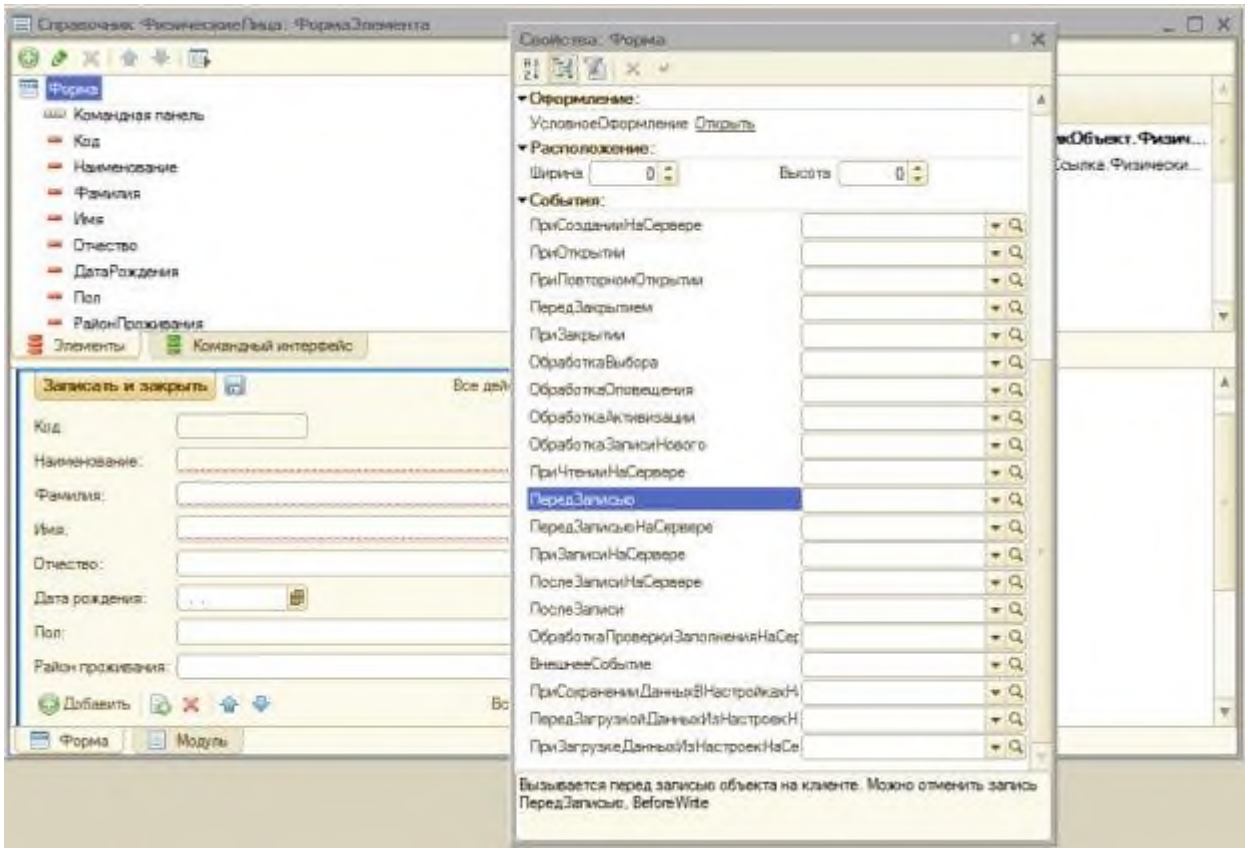


Рисунок 4.1- Выбор события для выполнения запланированных действий

1.2 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**.

1.3 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 4.2. Не будем включать проверку заполнения, выполним ее и еще некоторые проверки самостоятельно.

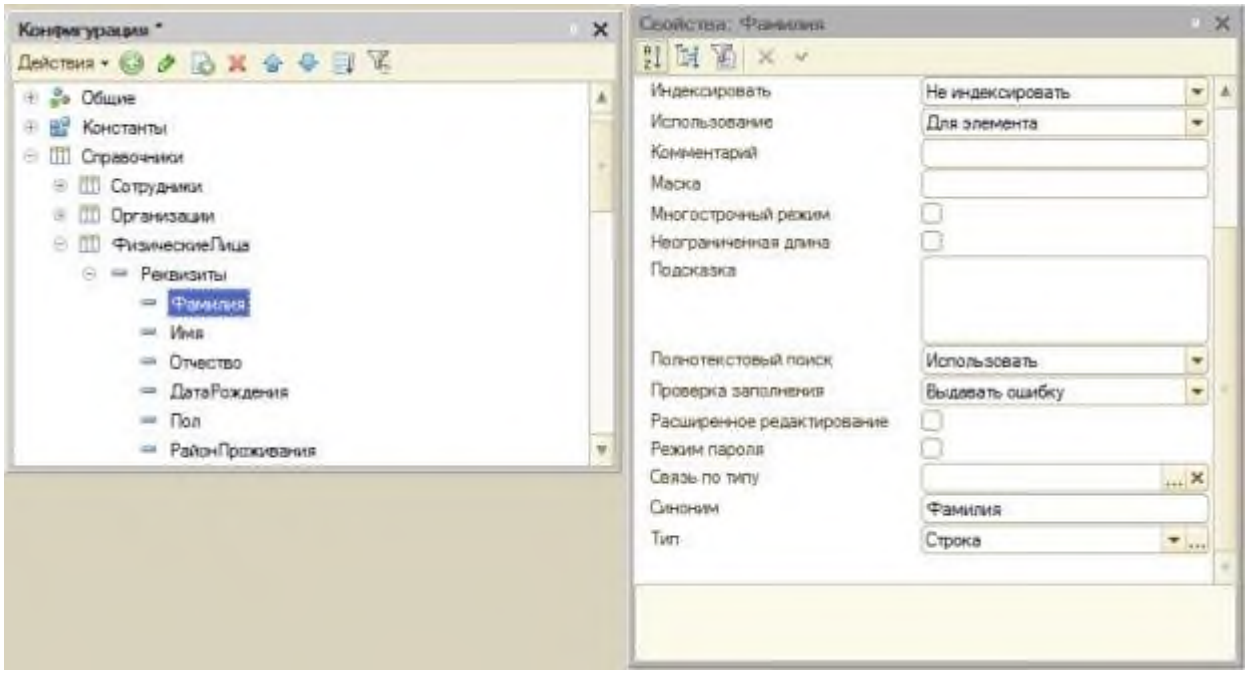


Рисунок 4.2- Настройка проверки заполнения

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**.

1.3 В процедуре **ПередЗаписью** 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
```

1.4 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 4.3.

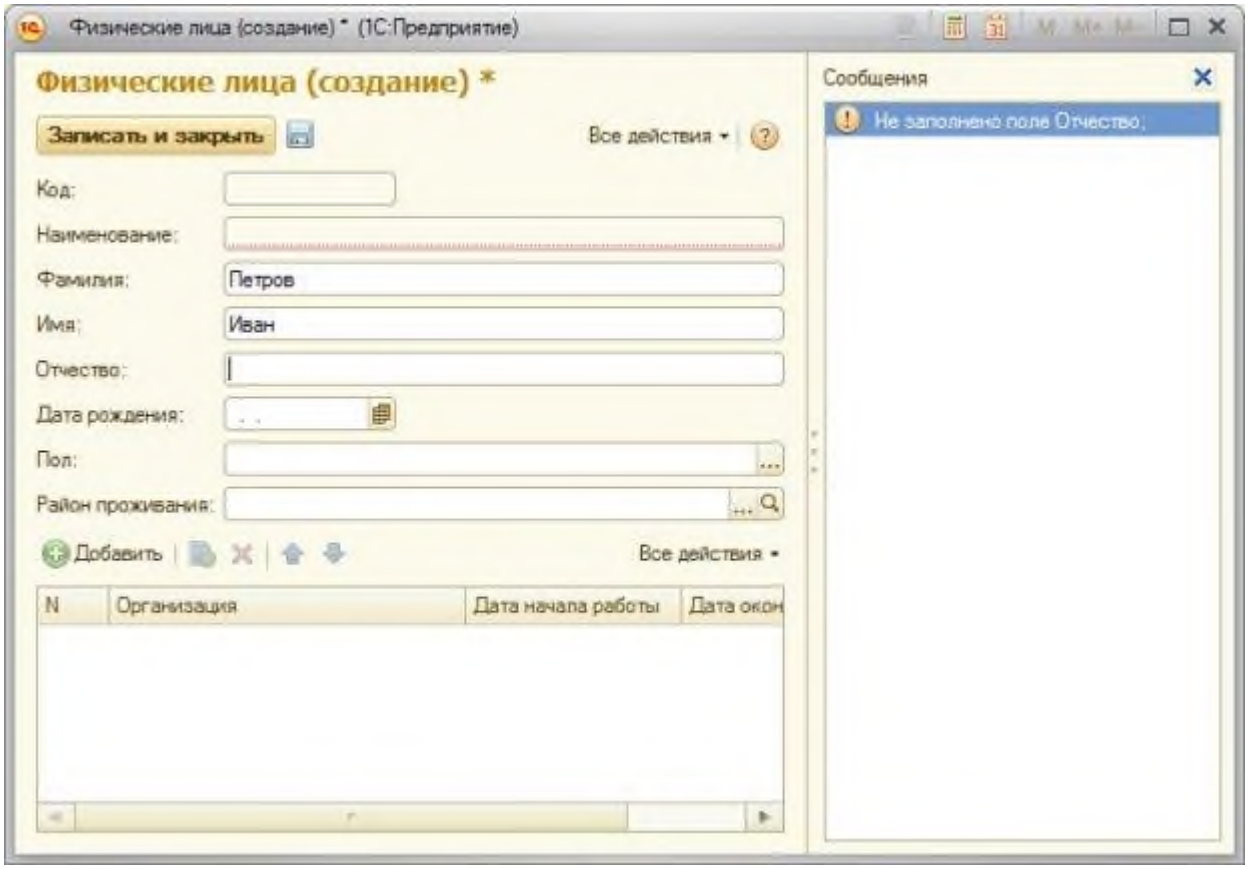


Рисунок 4.3- Сообщение об ошибке

1.5 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 4.4.

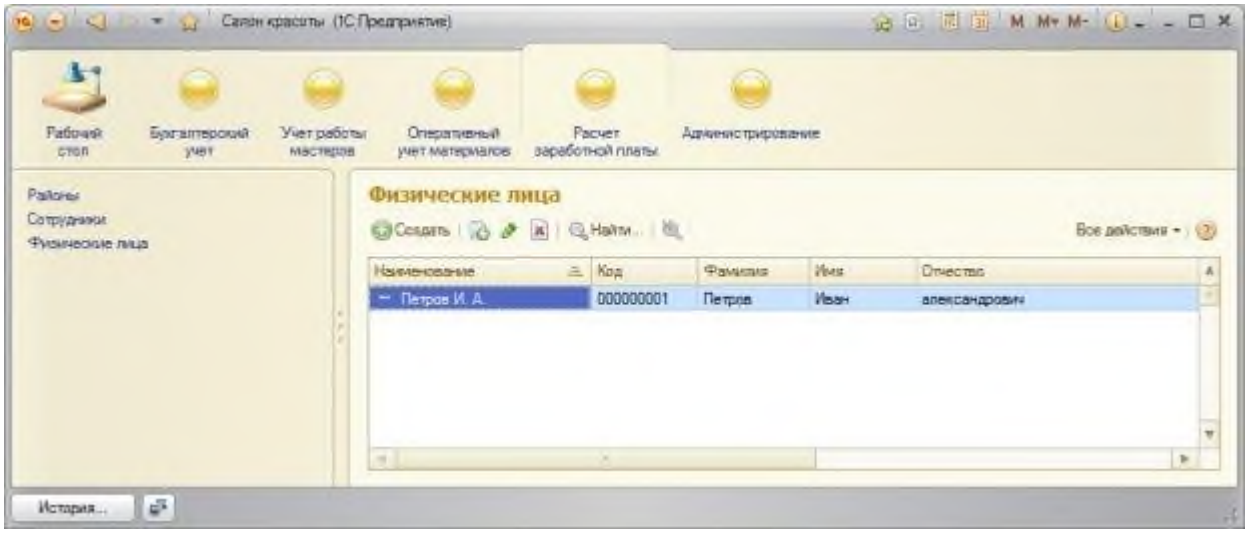


Рисунок 4.4- Новая запись в справочнике Физические лица

1.6 Запрограммируем сообщение пользователю

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
```

1.7 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 4.5, 4.6).

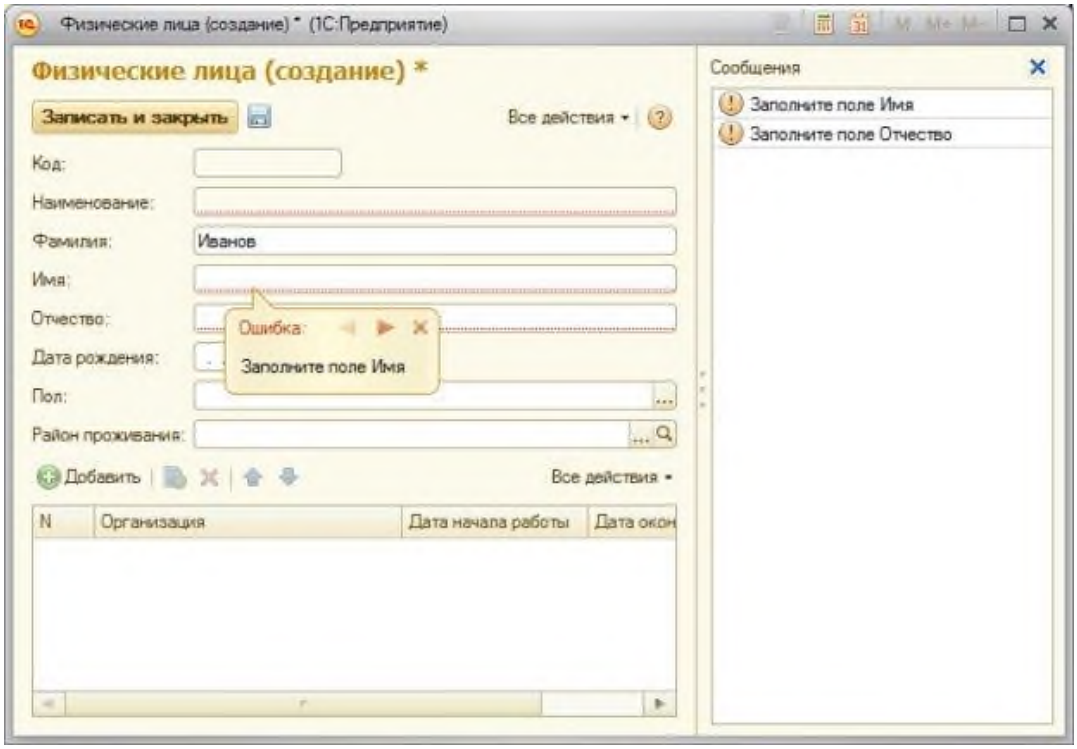


Рисунок 4.5- Сообщение об ошибке, привязанное к полю Имя

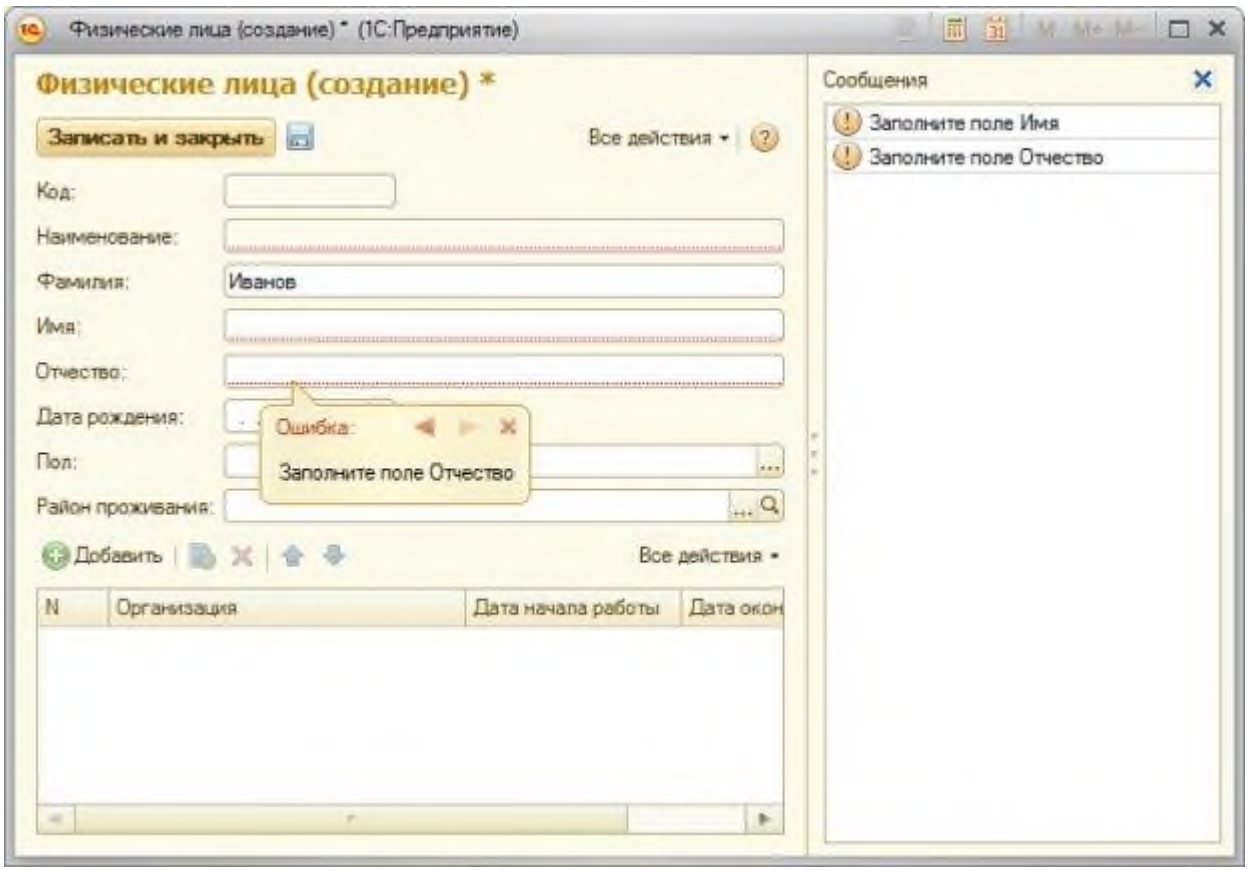


Рисунок 4.6- Сообщение об ошибке, привязанное к полю Отчество

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

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	Владимиров	11.08.1976	Женский	Советский

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**.

2. Выполним настройку командного интерфейса для ускорения доступа к справочнику

2.1 Добавим команду создания нового элемента справочника **ФизическиеЛица** в панель действий раздела **Расчет заработной платы**. 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 4.7.

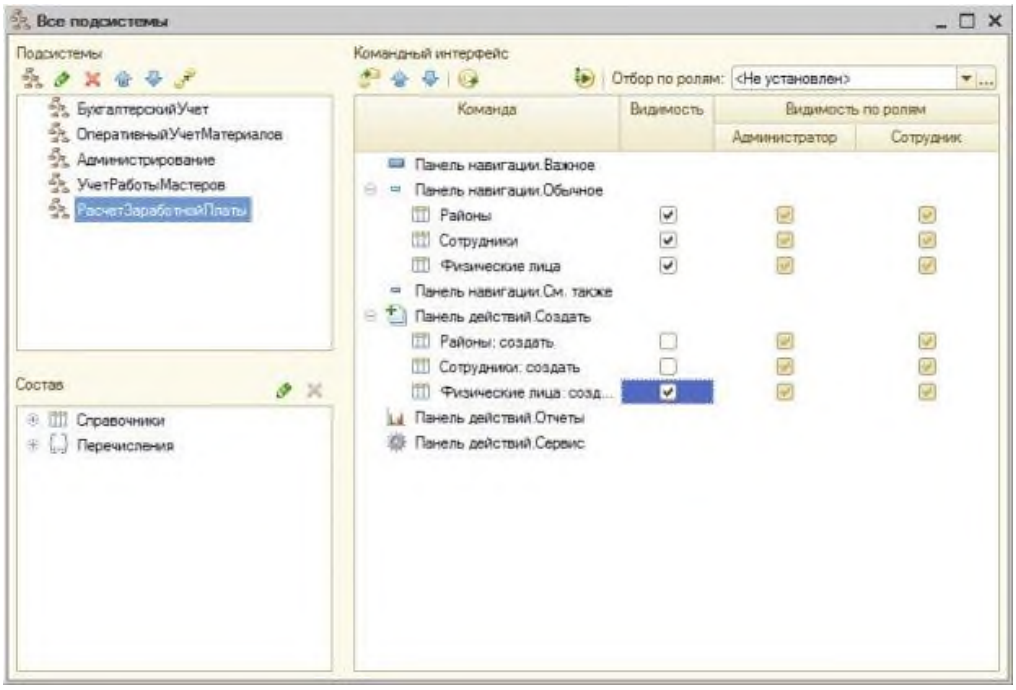


Рисунок 4.7- Настройка панели действий раздела Расчет заработной платы

2.2 Мы можем включить команду добавления нового физического лица в командный интерфейс **Рабочего стола**.

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, рисунок 4.8.

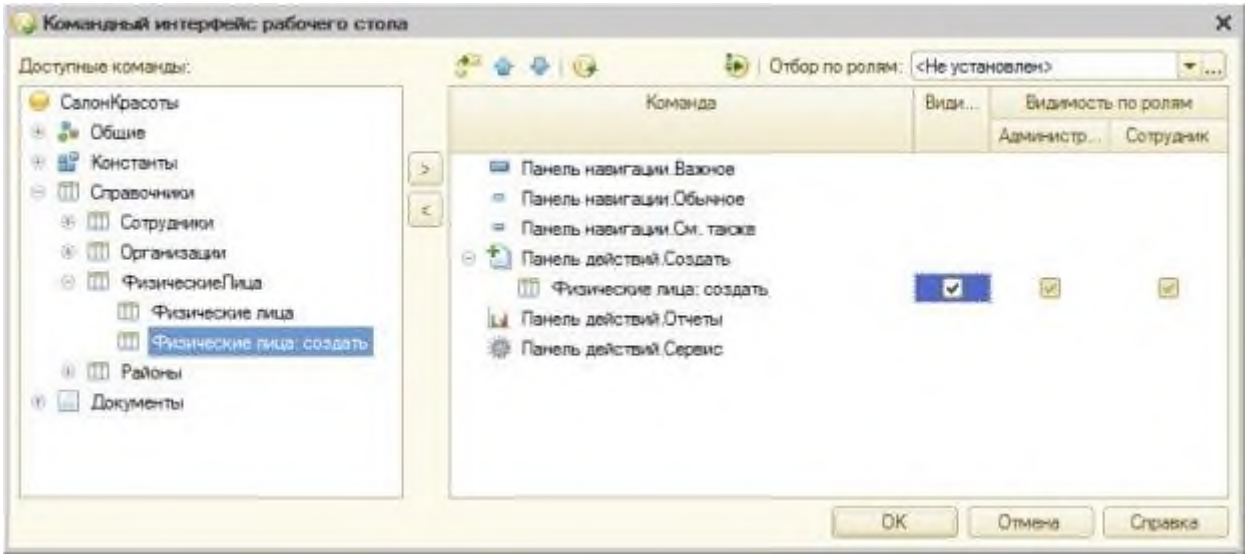


Рисунок 4.8- Настройка панели действий Рабочего стола

Теперь,(Рисунок 4.9.), команда для быстрого создания элементов справочника **ФизическиеЛица** добавлена в панель действий **Рабочего стола**, аналогичная команда появилась в разделе **Расчет заработной платы**.

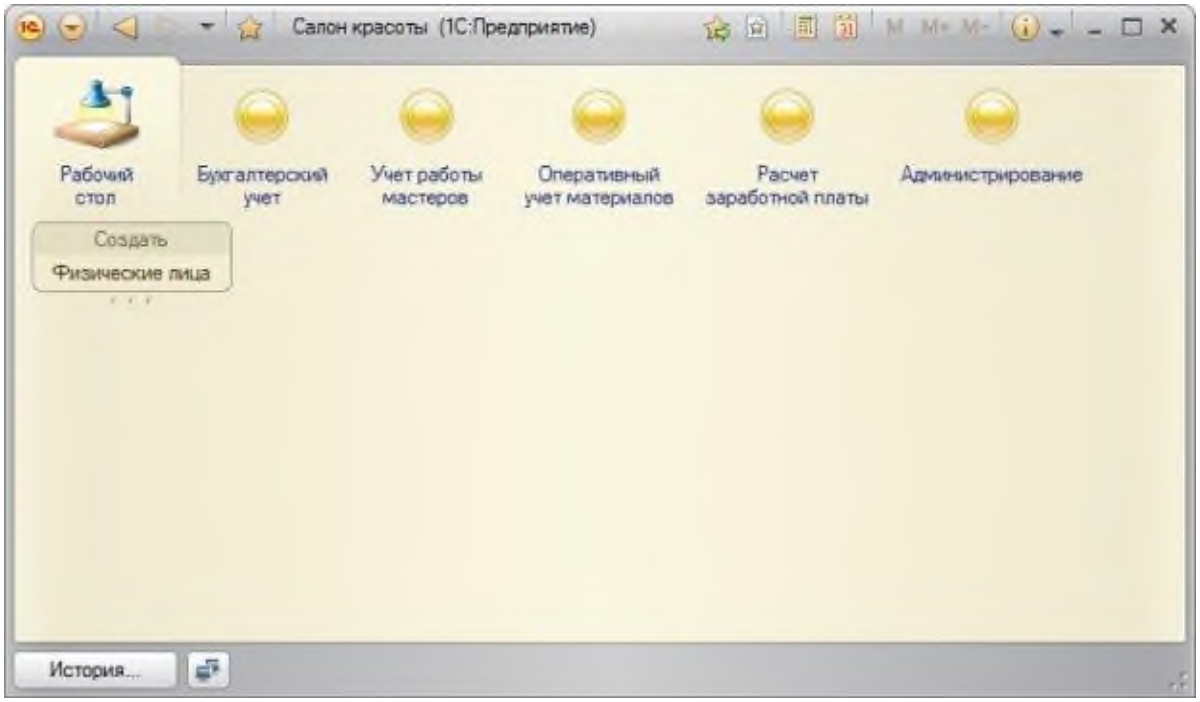


Рисунок 4.9- Новая команда в панели действий рабочего стола

Вывод

В этой работе мы создали справочники **Организации** и **ФизическиеЛица**. Для справочника **ФизическиеЛица** мы реализовали программное заполнение реквизита на основе других реквизитов, познакомились с объектом **СообщениеПользователю**, который позволяет выводить сообщения в привязке к элементам управления. Так же мы рассмотрели основные составные части редактора управляемых форм и настроили командный интерфейс для ускорения доступа пользователя к часто используемой функциональности справочника **ФизическиеЛица**.

Практическая работа №5 «Разработка и интеграция модулей проекта»

Цель работы: научиться разработке и интеграция модулей проекта.

ХОД РАБОТЫ

1. Создадим новую константу (Рисунок 5.1.), заполним ее параметры следующим образом:

Имя: ТекстСообщения

Тип: Строка

Длина: 50

2. Включим константу в состав подсистемы **УчетРаботыМастеров**. Предполагается, что данная константа будет использоваться для показа сообщения пользователям, входящим в систему.

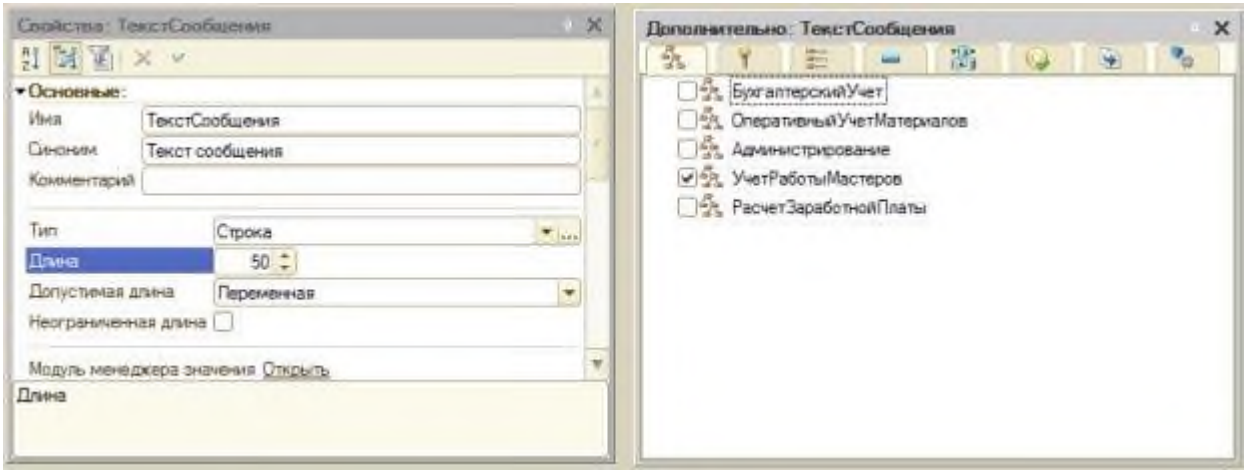


Рисунок 5.1- Настройка параметров новой константы

3. Посмотрим, как включение константы в подсистему **УчетРаботыМастеров**, отразится на интерфейсе нашего приложения в режиме 1С:Предприятие. Видно, Рисунок 5.2, 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.

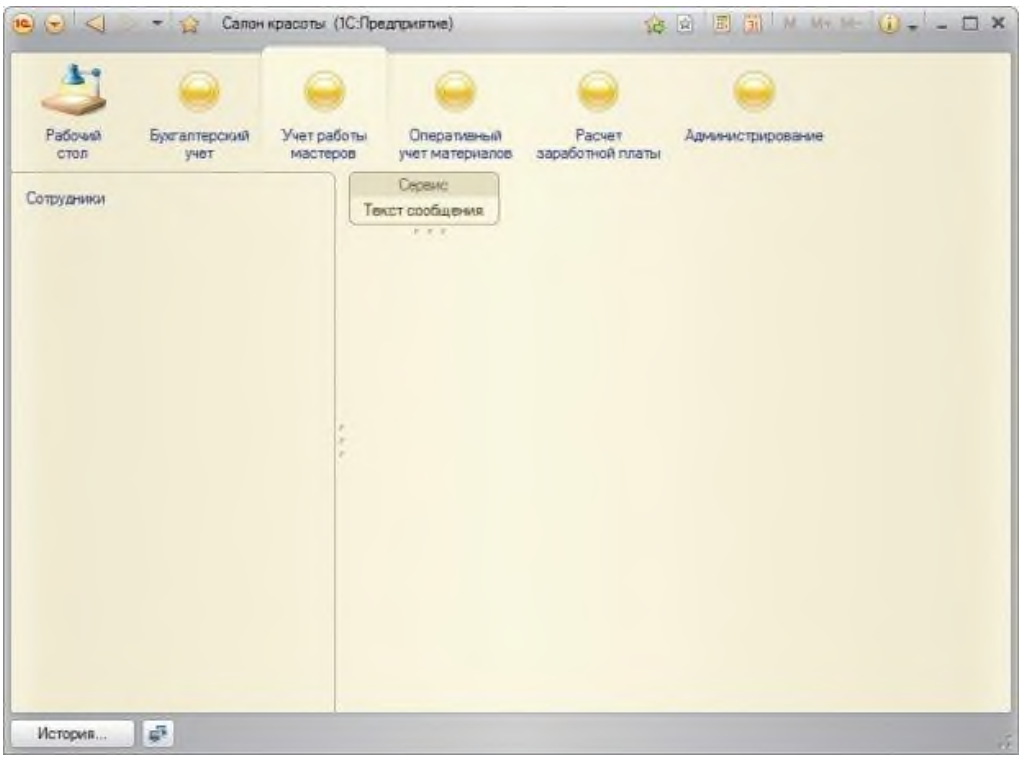


Рисунок 5.2- Константа в панели действий в разделе Учет работы мастеров

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.

4. 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 5.3. и нажмем на кнопку **Записать и закрыть**.

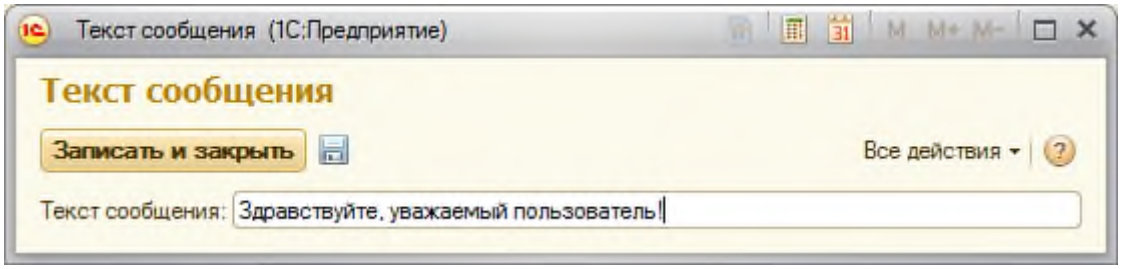


Рисунок 5.3- Форма редактирования константы Текст сообщения

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 5.4.

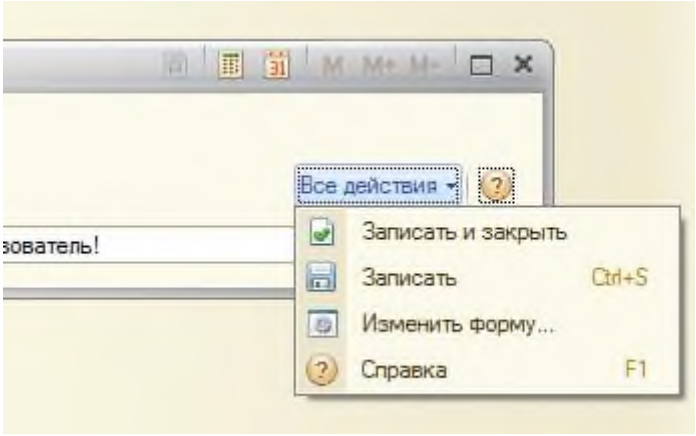


Рисунок 5.4- Меню Все действия

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 5.5.

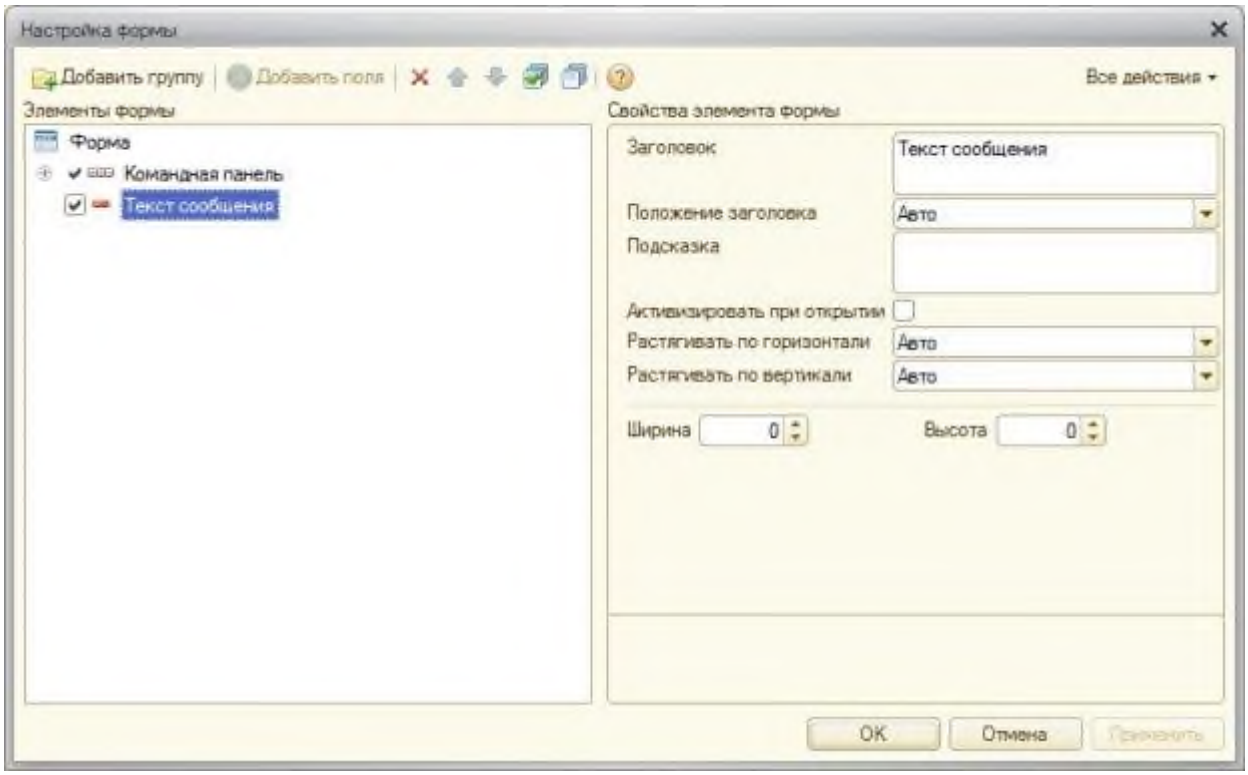


Рисунок 5.5- Окно Настройка формы

Нужно учитывать, что пользователь сможет настраивать внешний вид форм в том случае, если для него установлено право **Сохранение данных пользователя**. Это право можно настраивать, как и другие права, в роли пользователя, Рисунок 5.6. В нашем случае оно установлено.

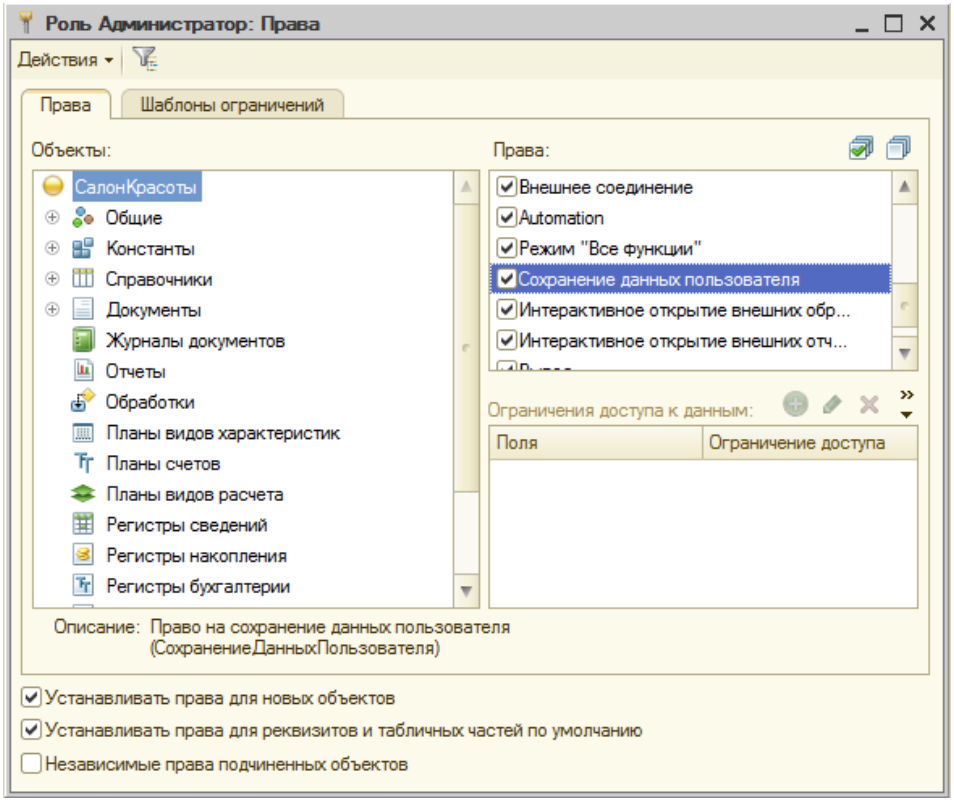


Рисунок 5.6- Право Сохранение данных пользователя

В нашем случае (Рисунок 5.5.) в группе **Элементы формы** выделен элемент **Текст сообщения**, в группе **Свойства элемента формы** мы можем настраивать его свойства.

5. Изменим свойство **Заголовок**, вместо "Текст сообщения" введем "Текст сообщения для пользователей", в итоге форма будет выглядеть так, как показано на Рисунок 5.7.

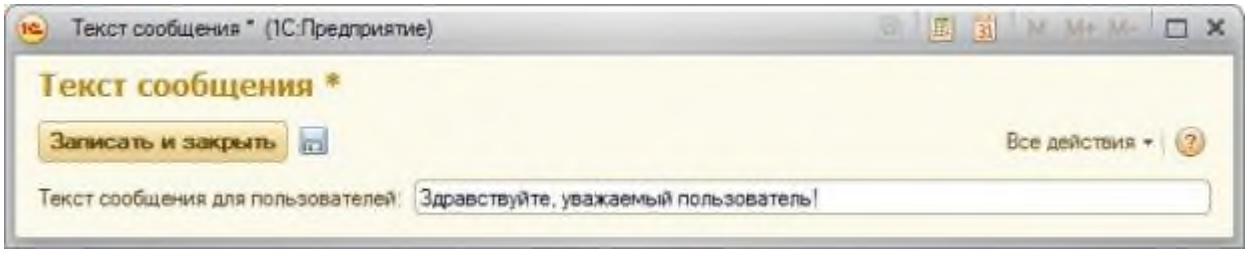


Рисунок 5.7- Отредактированный заголовок объекта

6. 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**.

7. Создадим форму констант

7.1 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 5.8., нужно оставить тип формы в значении **Форма констант**, при необходимости заполнить другие поля и нажать на кнопку Далее.

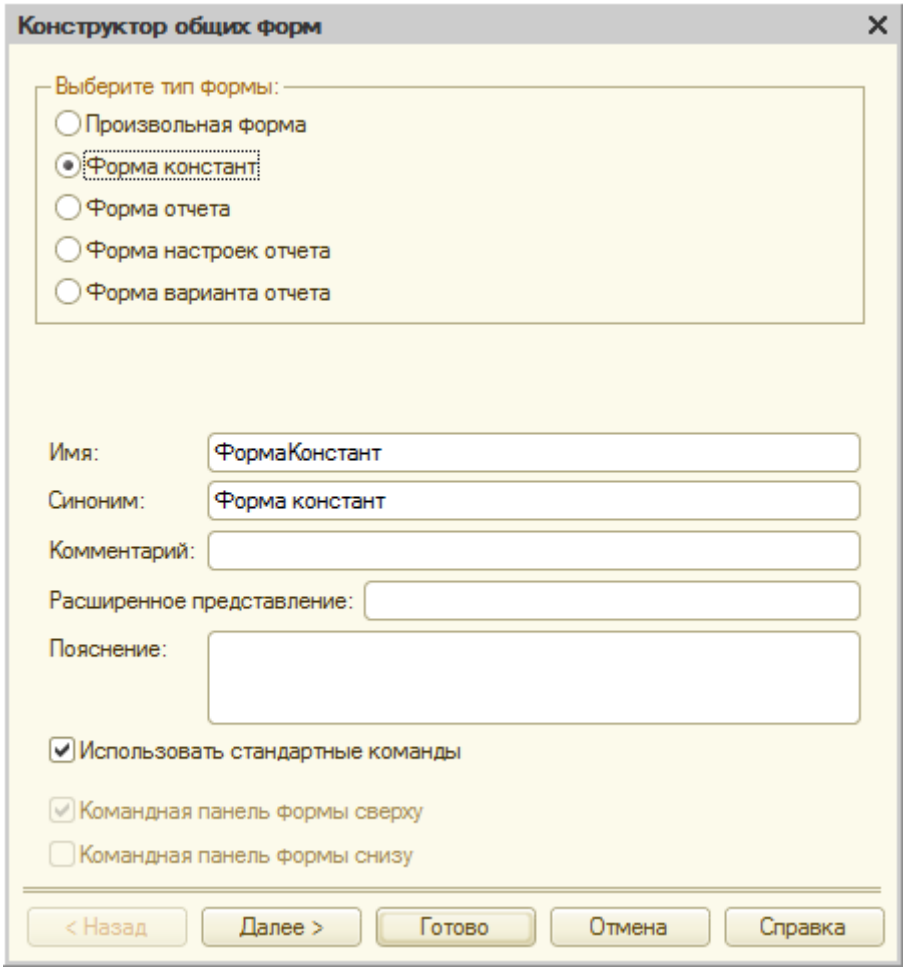


Рисунок 5.8- Конструктор общих форм

7.2 В появившемся окне, Рисунок 5.9. 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**.

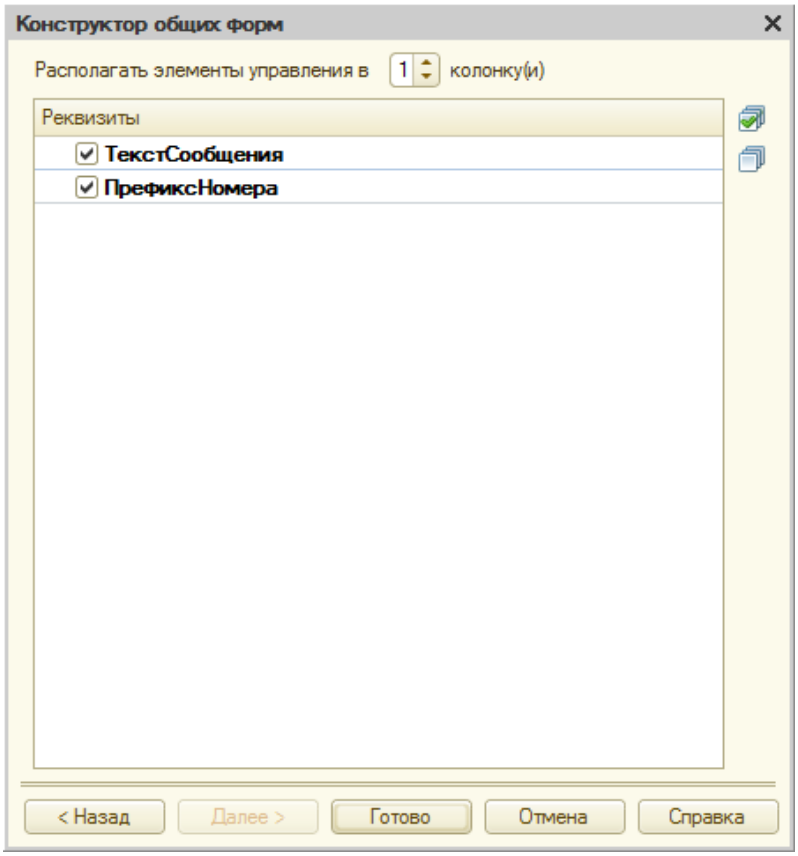


Рисунок 5.9- Конструктор общих форм, состав формы констант

7.3 В ветви **Общие формы** появится новая форма с именем **ФормаКонстант**, будет открыто окно редактирования формы, Рисунок 5.10.

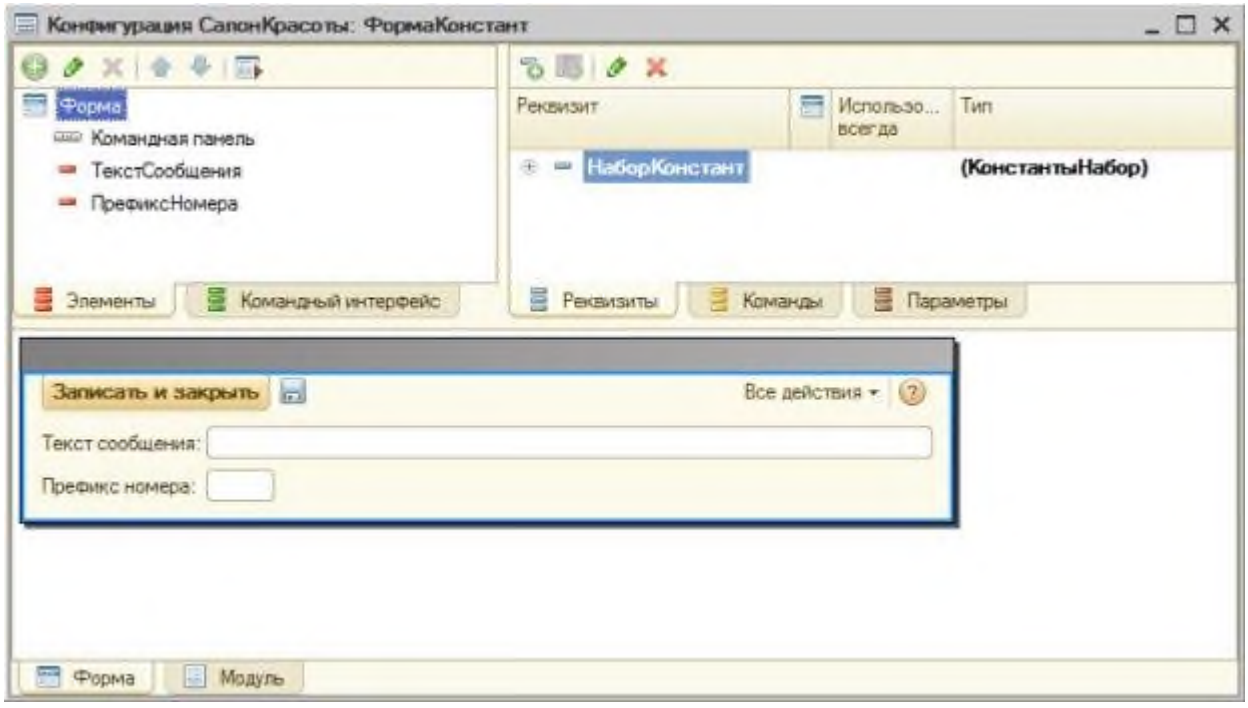


Рисунок 5.10- Окно редактирования формы

7.4 Форму констант так же нужно включить в одну из подсистем. Включим ее в подсистему **Администрирование**, посмотрим, что у нас получилось, Рисунок 5.11.

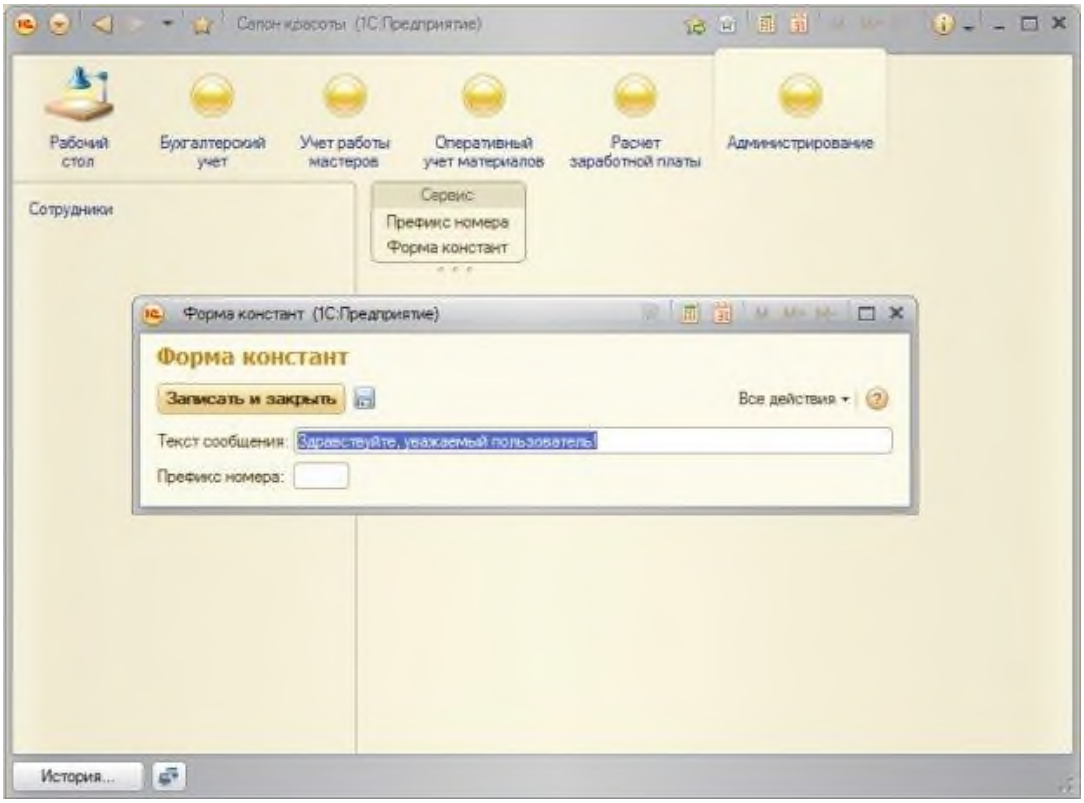


Рисунок 5.11- Окно редактирования формы

7.5 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 5.12., снять флаг **Видимость** для команды **Префикс номера** группы **Сервис** панели действий.

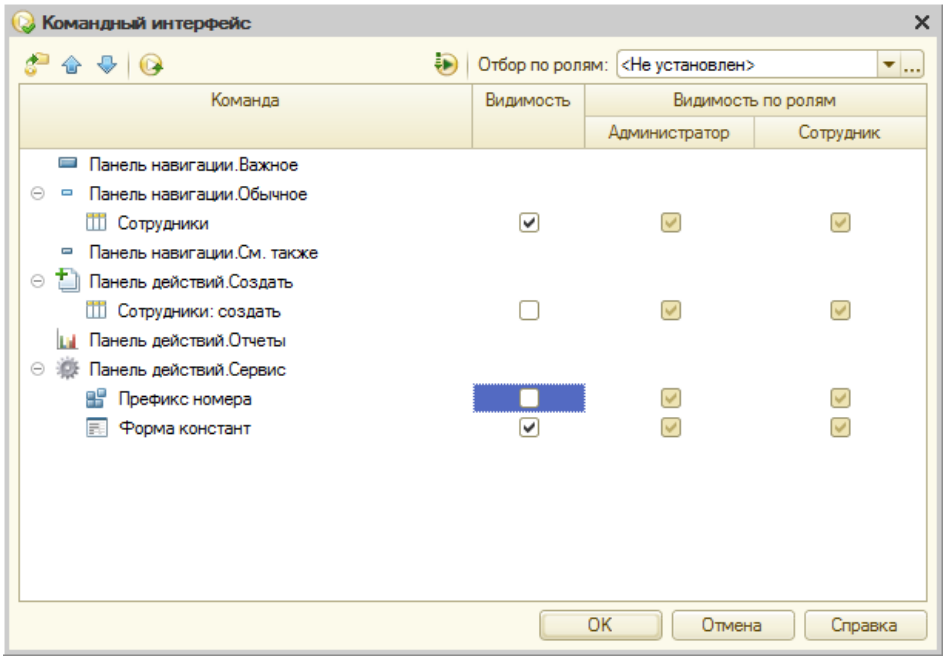


Рисунок 5.12- Настройка панели действий

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

7.6 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 **Открыть модуль управляемого приложения** корневого элемента конфигурации. Для этого модуля предусмотрено несколько стандартных обработчиков событий, которые можно найти в панели инструментов **Модуль**, Рисунок 5.13. Нас интересует обработчик **ПриНачалеРаботыСистемы**.

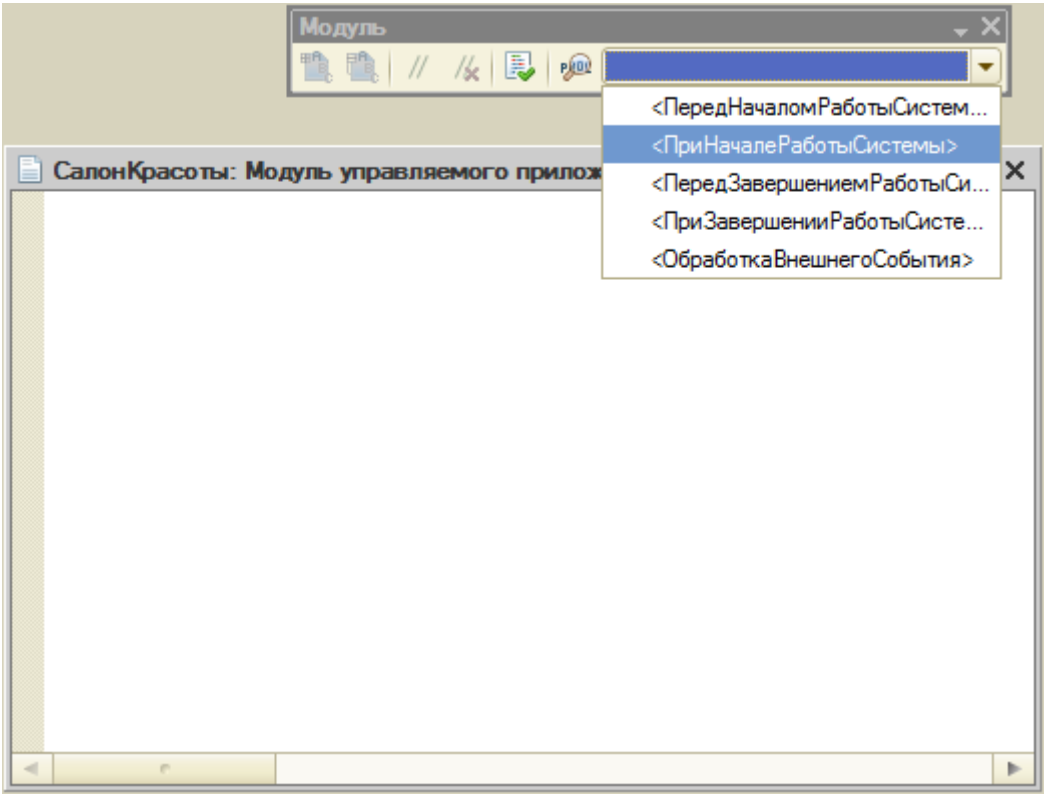


Рисунок 5.13- Выбор обработчика ПриНачалеРаботыСистемы

7.7 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 5.14.

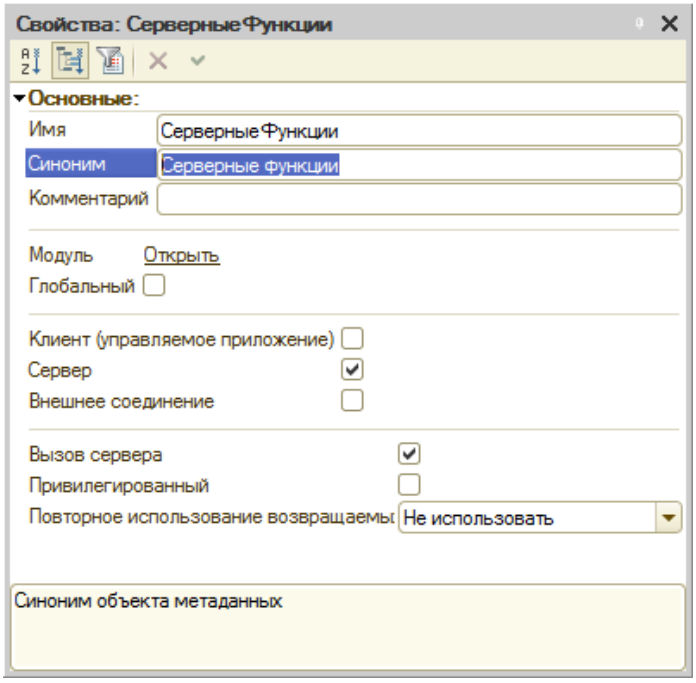


Рисунок 5.14- Общий модуль СерверныеФункции, свойства

7.8 Откроем редактор кода для кода модуля (например, двойным щелчком по модулю в дереве конфигурации) и введем следующий код, Рисунок 5.15:

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

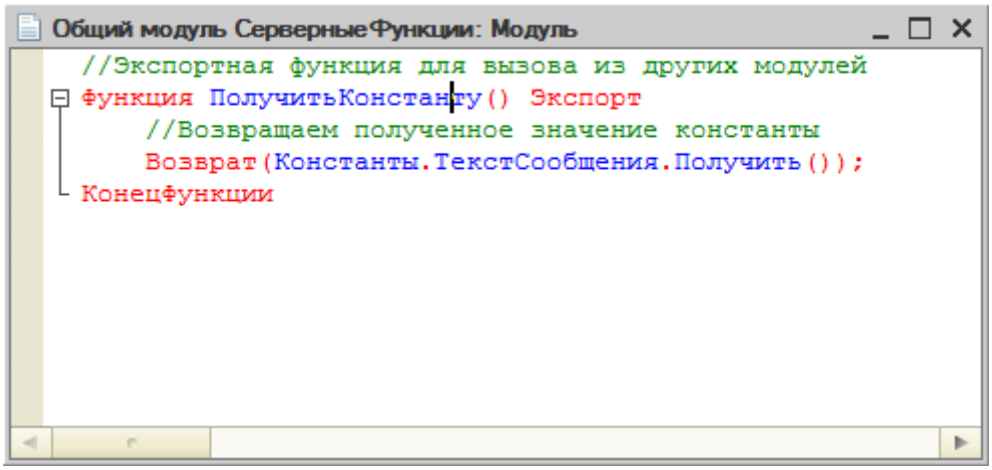


Рисунок 5.15- Общий модуль «СерверныеФункции», код

7.9 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 5.16.

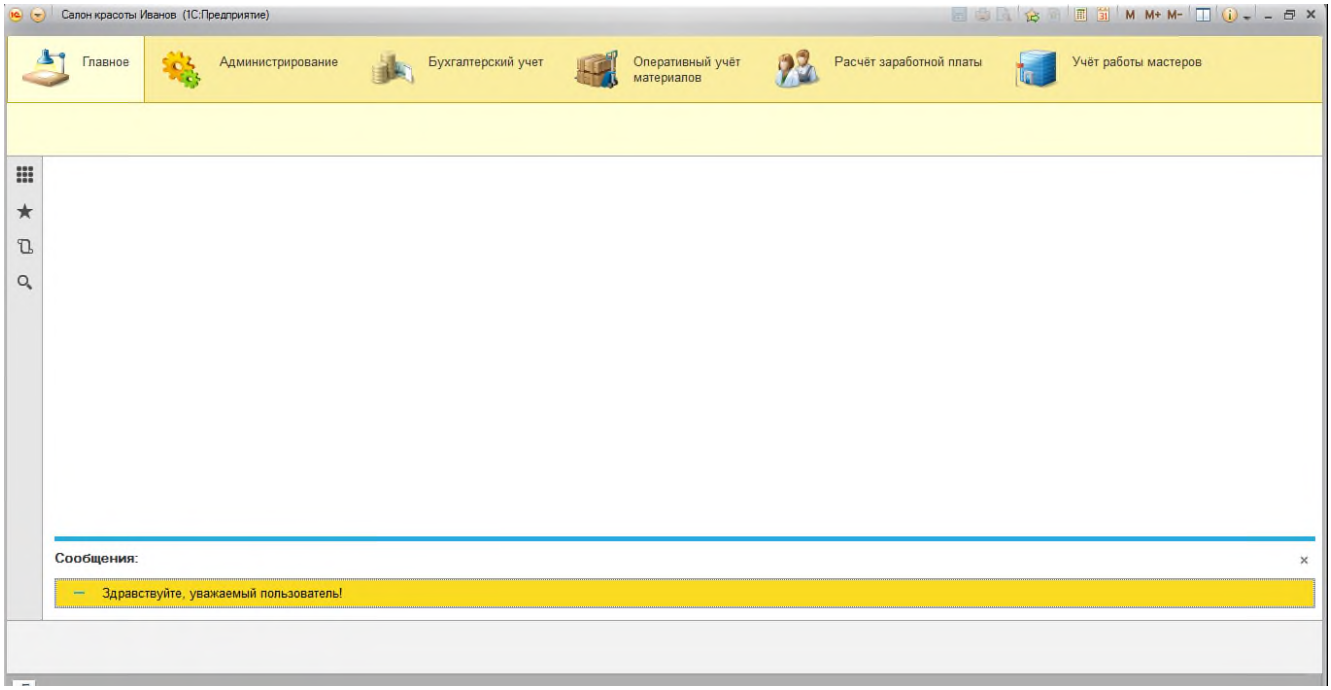


Рисунок 5.16- Вывод сообщения пользователю

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.

7.10 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 5.17:

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

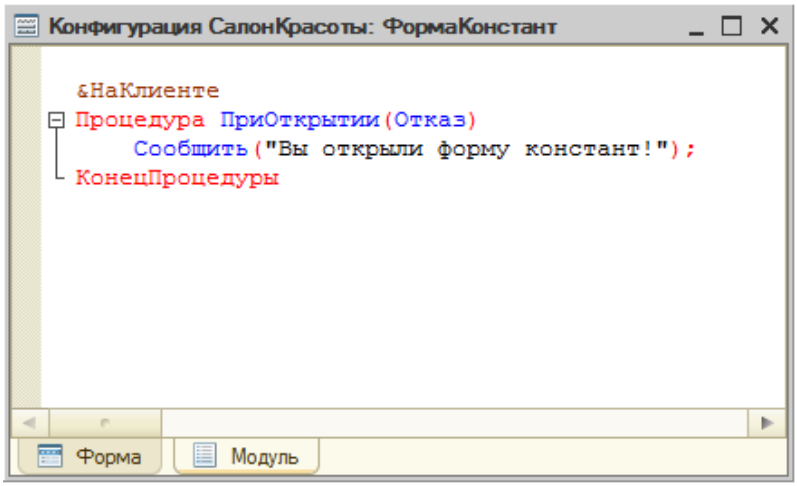


Рисунок 5.17- Вывод сообщения пользователю из модуля формы констант

Благодаря этому коду при открытии формы констант будет появляться следующее сообщение, Рисунок 5.18.

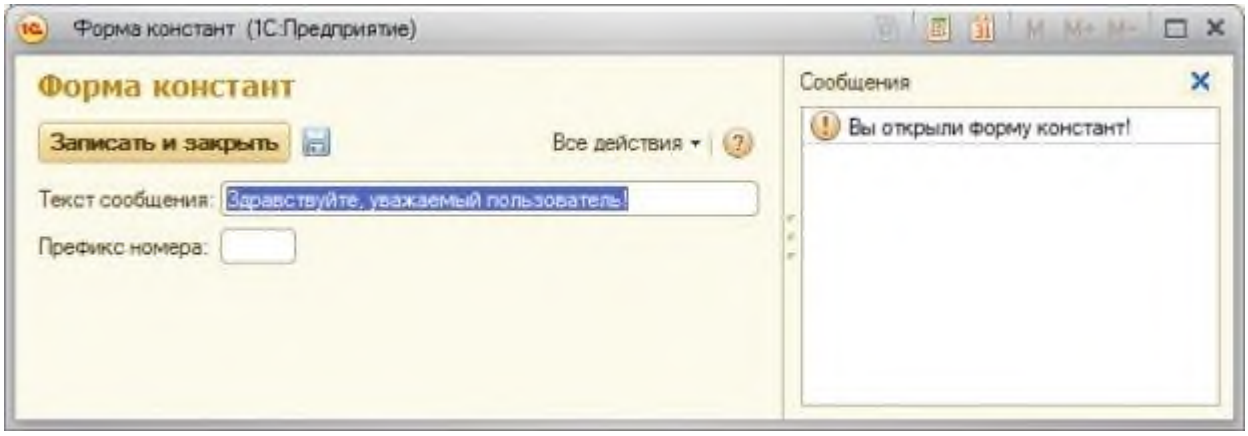


Рисунок 5.18- Вывод сообщения в форму констант

8. 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 рисунке 5.19. вы можете видеть вывод сообщений, выполненный вышеприведенным кодом.

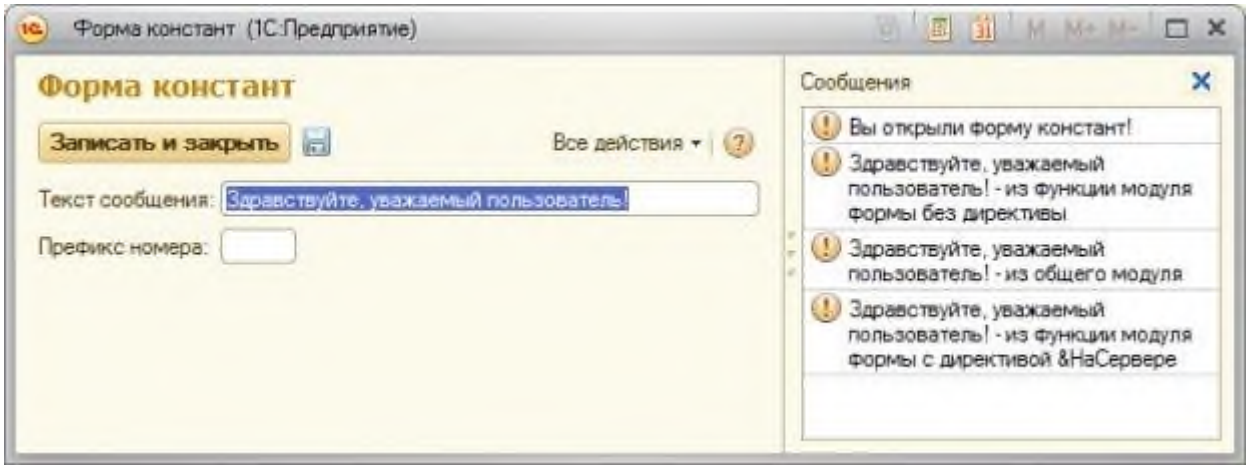


Рисунок 5.19- Вывод сообщения в форму констант, разные варианты работы с серверными данными

9. Создадим общие реквизиты.

9.1 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.

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

9.3 Прежде чем продолжать работу над общими реквизитами, создадим следующий объект конфигурации, не настраивая его дополнительных свойств – документ с именем **ПоступлениеМатериалов**. Включим его в подсистему **ОперативныйУчетМатериалов**.

Создадим новый **общий реквизит** со следующими параметрами, Рисунок 5.20.:

Имя: Комментарий

Тип: Строка, длина 50

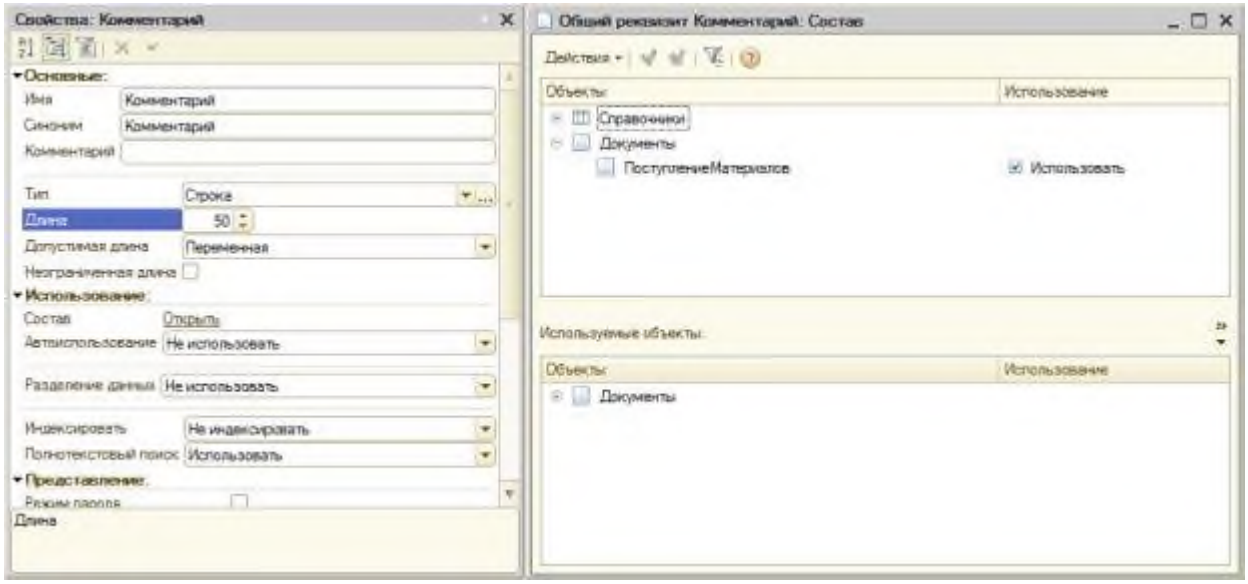


Рисунок 5.20- Настройка общего реквизита

9.4 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.

9.5 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 5.21.

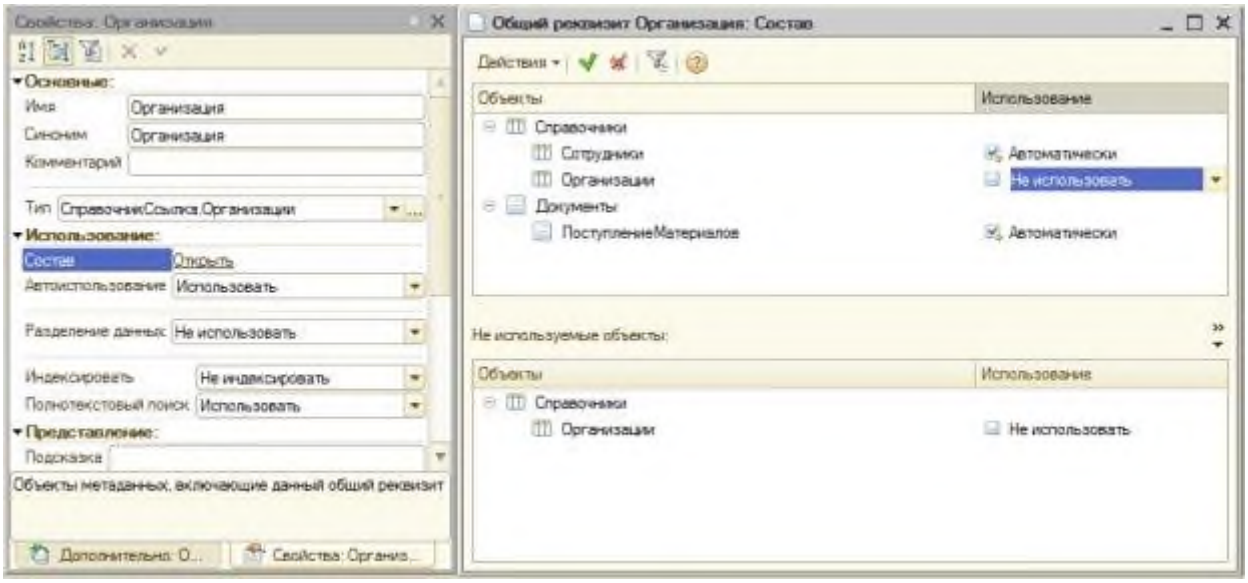


Рисунок 5.21- Настройка общего реквизита

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

9.7 Для начала перейдем на вкладку **Оперативный учет материалов**. Обратите внимание на то, что в панель навигации раздела были автоматически добавлены ссылки для доступа к справочнику **Организации** и к документу **Поступление материалов**. Щелкнем по ссылке **Организации**. В рабочей области окна появится список справочника. Щелкнем по кнопке **Создать**, которая расположена на командной панели списка – появится отдельное окно для заполнения свойств элемента справочника, Рисунок 5.22. Можно отметить, что помимо стандартных реквизитов (**Наименование**, **Код**) данный справочник не содержит ничего другого – это неудивительно, мы исключили его из состава общего реквизита **Организация**.

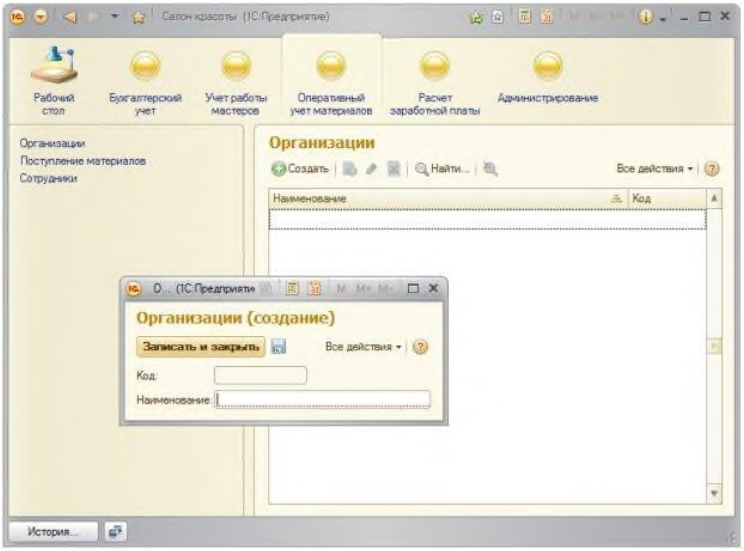


Рисунок 5.22- Справочник «Организации»

9.8 Теперь откроем список справочника **Сотрудники** и нажмем на кнопку **Добавить**. Общий реквизит **Организация** у данного справочника присутствует, Рисунок 5.23.

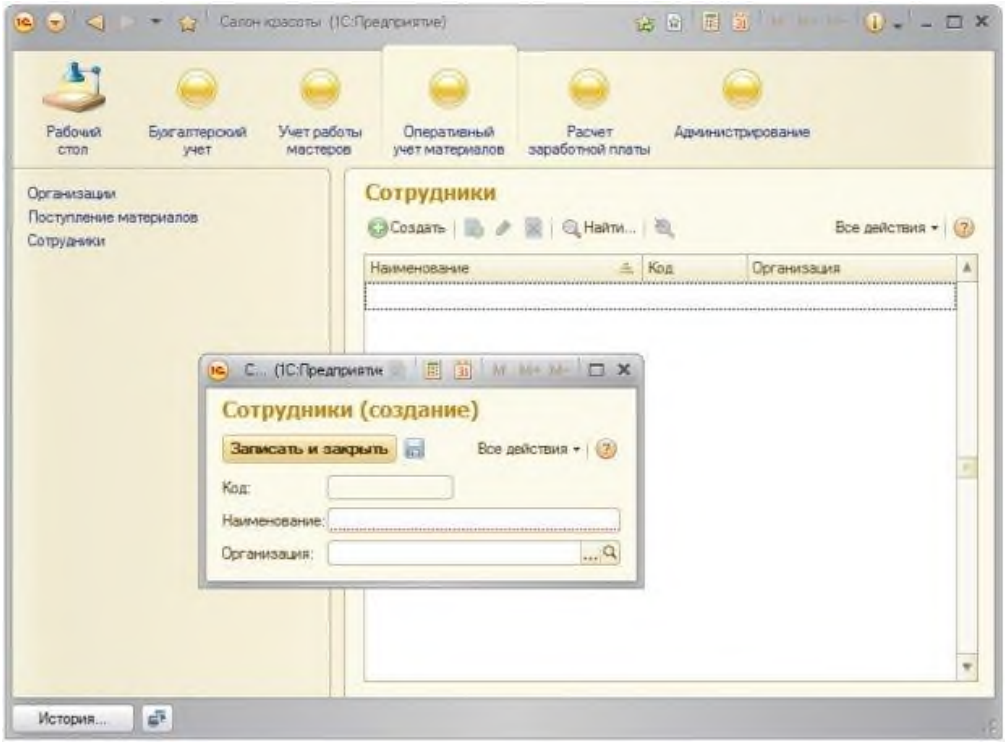


Рисунок 5.23- Справочник «Сотрудники»

9.9 Откроем теперь окно создания документа **ПоступлениеМатериалов**. Здесь мы видим два общих реквизита – **Комментарий** и **Организация**, Рисунок 5.24.

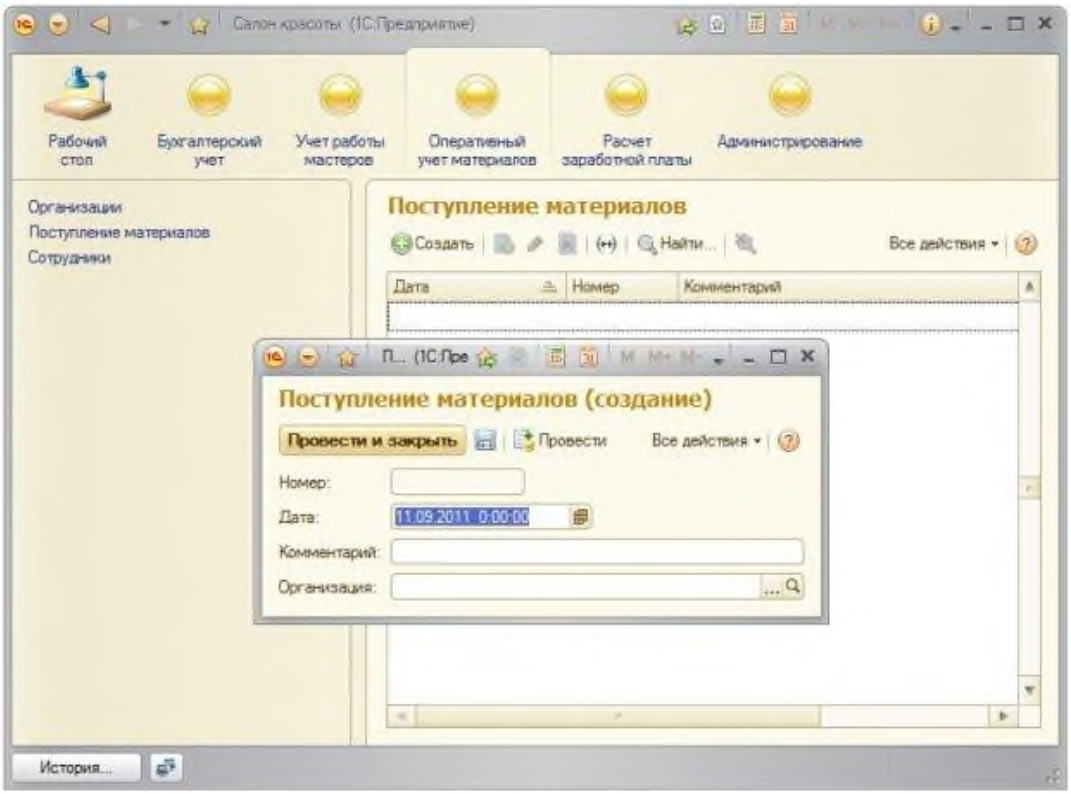


Рисунок 5.24- Документ «ПоступлениеМатериалов»

Вывод

В этой работе мы научились создавать константы и программно работать с ними. Так же здесь мы начали обсуждение вопросов клиент-серверного программирования, в частности, использовали директивы компиляции **&НаСервере, &НаКлиенте**. Мы познакомились с использованием экспортных методов общих модулей, с модулем управляемого приложения, с модулем формы, рассмотрели использование общих реквизитов.

Практическая работа №6 «Отладка отдельных модулей программного проекта»

Цель работы: научиться выполнять отладку отдельных модулей программного проекта.

ХОД РАБОТЫ:

1. Выполним работу с иерархическими справочниками.

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

Имя: ЕдиницыИзмерения

Длина наименования: 100 символов

Подсистемы: БухгалтерскийУчет, ОперативныйУчетМатериалов

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

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

Имя: Номенклатура

Подсистемы: БухгалтерскийУчет, ОперативныйУчетМатериалов

На вкладке окна редактирования объекта **Иерархия**, Рисунок 6.1., установим следующие параметры:

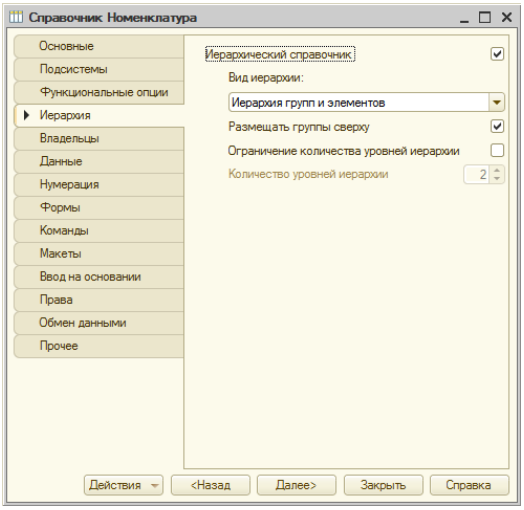


Рисунок 6.1- Настройка иерархического справочника

Иерархический справочник: Установлено

Вид иерархии: Иерархия групп и элементов.

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.

3. На вкладке **Данные**, Рисунок 6.2, 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

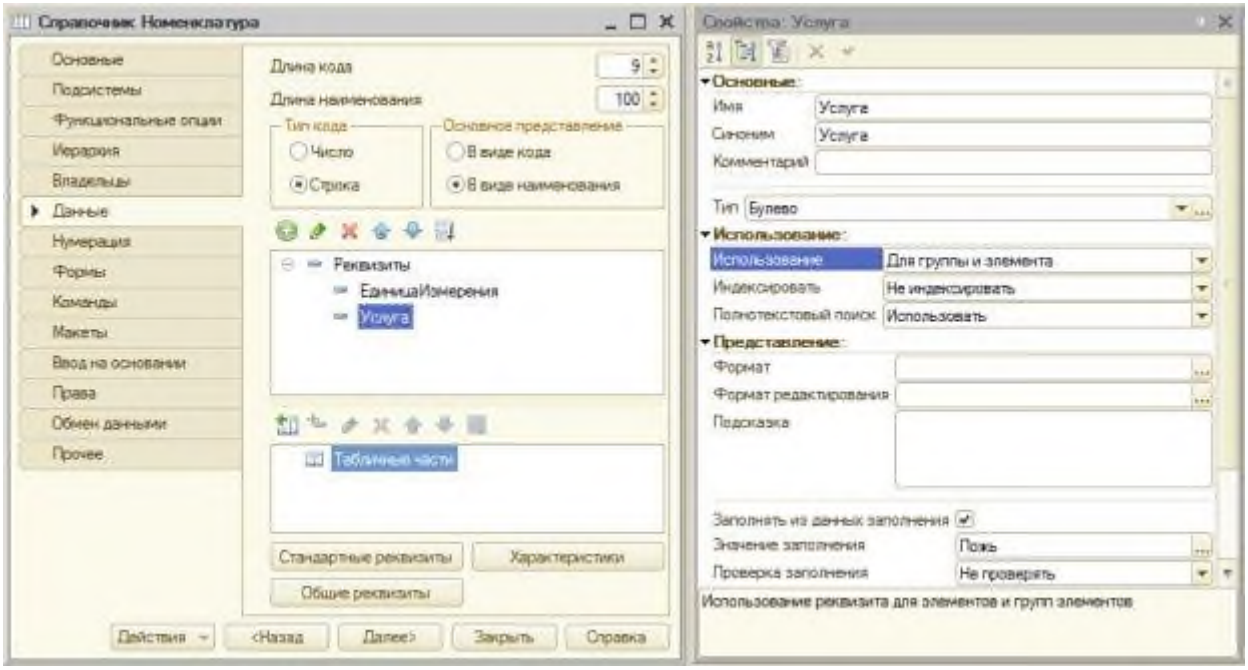


Рисунок 6.2- Состав реквизитов справочника «Номенклатура»

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.

4. 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.

5. Перейдем в модуль объекта (кнопка **Модуль объекта** на закладке **Прочие** окна редактирования объекта), из списка процедур и выберем **ОбработкаЗаполнения**, Рисунок 6.3.

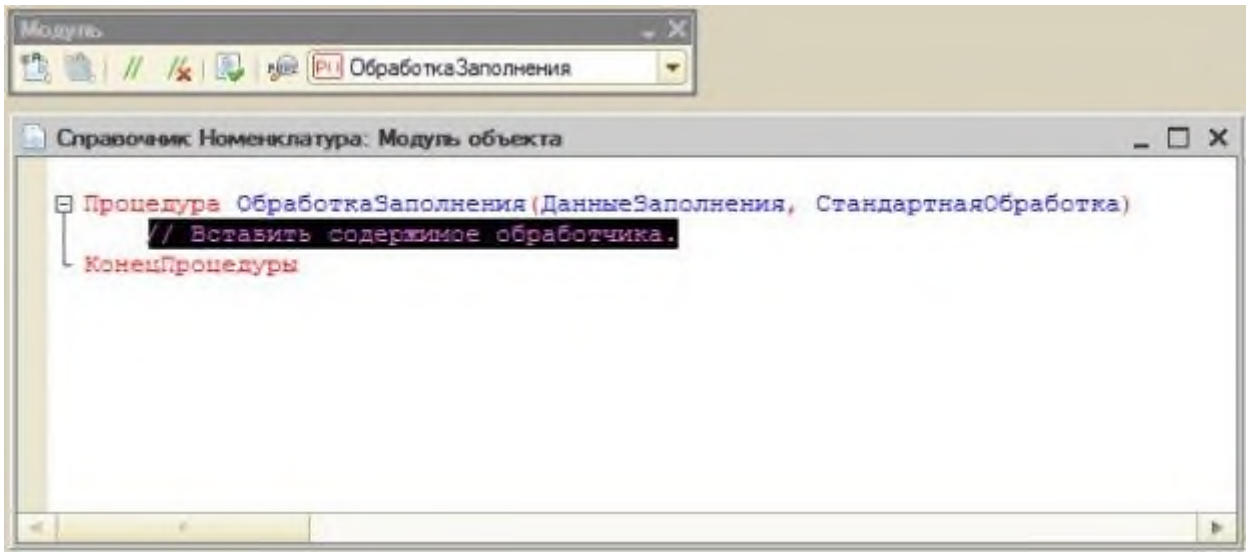


Рисунок 6.3- Процедура обработки заполнения справочника

6. В режиме 1С:Предприятие откроем справочник **Номенклатура**, создадим две группы – **Товары** – флаг **Услуга** для элементов в этой группе не устанавливаем, и **Услуги** – флаг установлен, Рисунок 6.4.

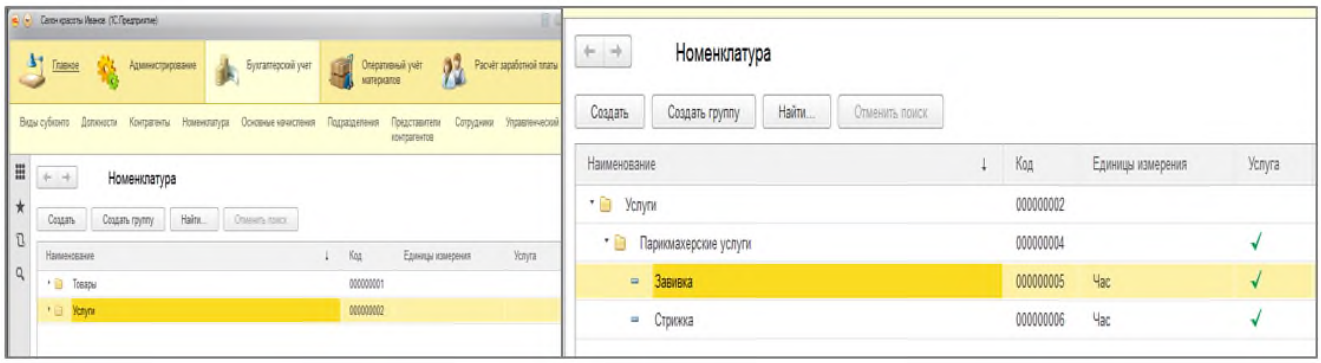


Рисунок 6.4- Группы в справочнике «Номенклатура»

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, как на рисунке 6.7.

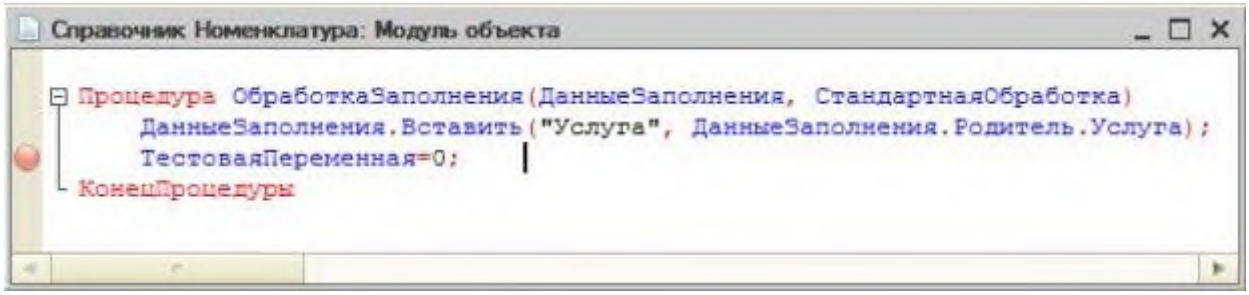


Рисунок 6.7 - Заполнение реквизита Услуга на основании параметров элемента родителя

7. 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 6.8.

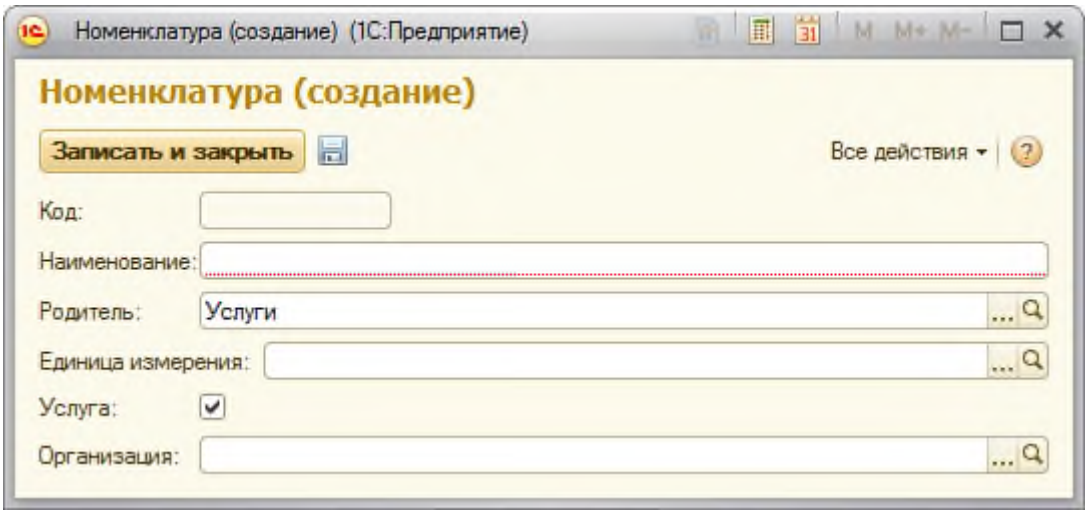


Рисунок 6.8- Результат заполнения реквизита Услуга на основании параметров элемента родителя

8. 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**.

.9. Выполним работу с подчиненными справочниками

9.1 Создадим новый справочник, назовем его **Контрагенты**.

Добавим его в подсистемы **БухгалтерскийУчет** и **ОперативныйУчетМатериалов**.

Справочник будет **иерархическим**, с иерархией групп и элементов.

В состав реквизитов справочника добавим следующие (Рисунок 6.9.):

Имя: ПолноеНаименование, тип – Строка, длина 100.

Имя: КонтактныеСведения, тип – Строка, длина 100

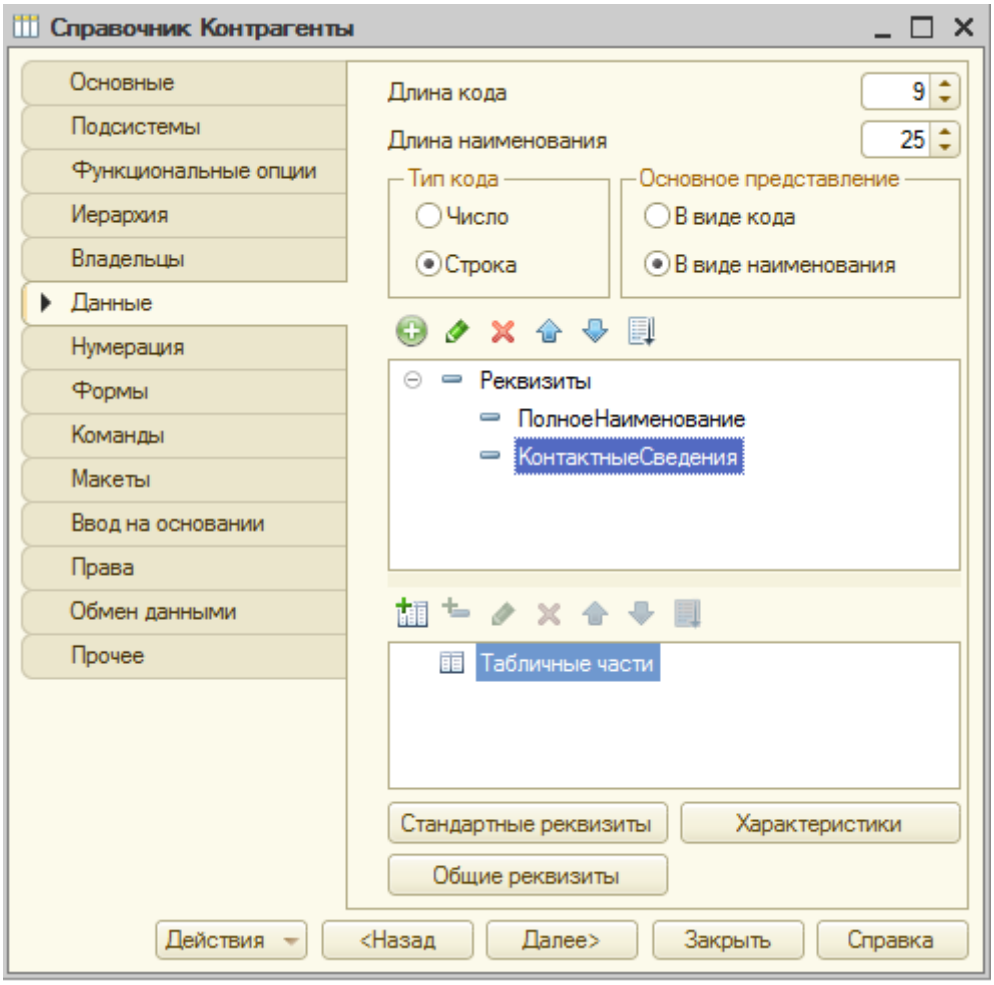


Рисунок 6.9- Справочник «Контрагенты»

9.2 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:

Мы укажем в параметре **Использование подчинения** вариант **Элементам**, Рисунок 6.10.

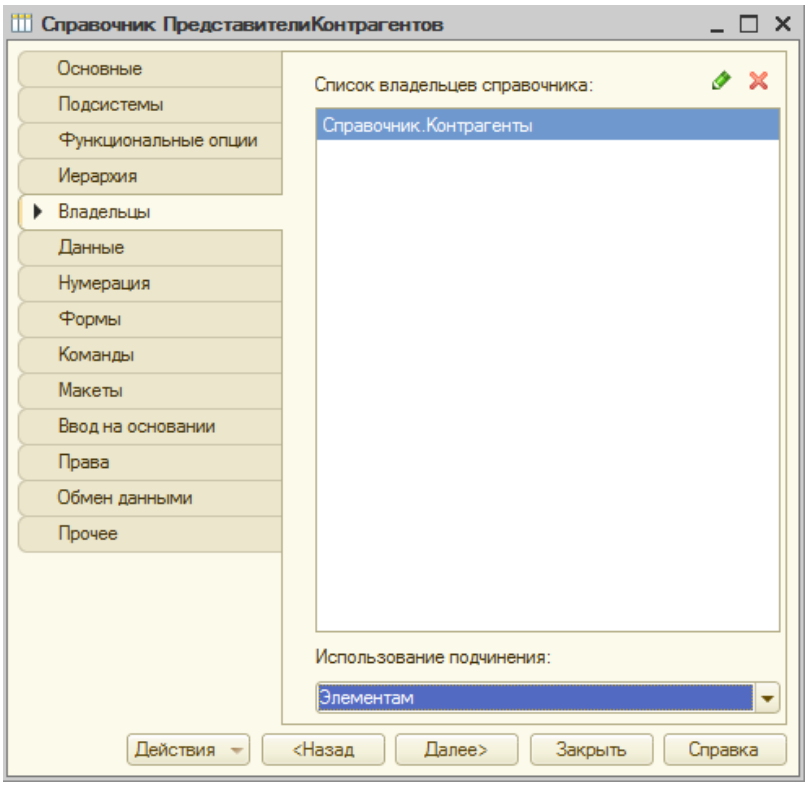


Рисунок 6.10- Настройка подчинения

Добавим справочник в состав подсистем **БухгалтерскийУчет** и **ОперативныйУчетМатериалов**.

В состав реквизитов справочника добавим следующие:

Имя: ФИО, тип – Строка, длина 100

Имя: КонтактныеСведения, тип – Строка, длина 100.

Имя: ПредставительРаботает, тип – Булево.

9.3 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 6.11.

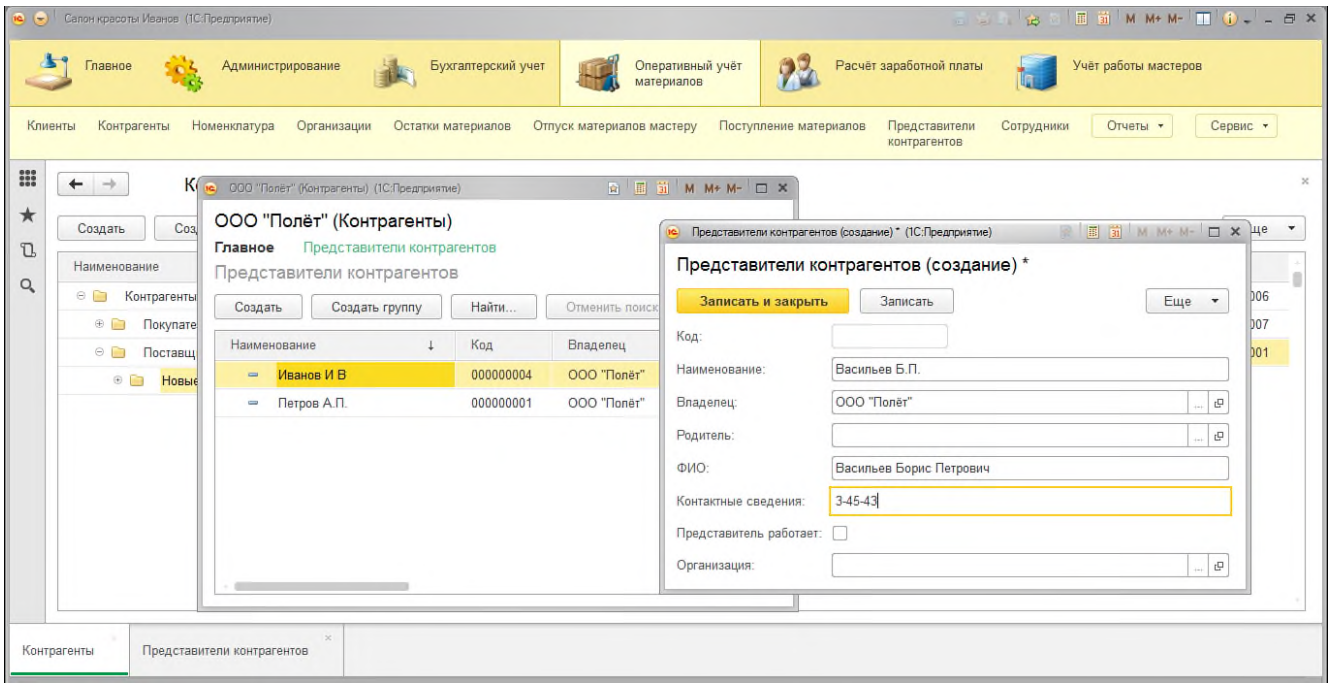


Рисунок 6.11- Форма элемента справочника Контрагенты

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 6.12.

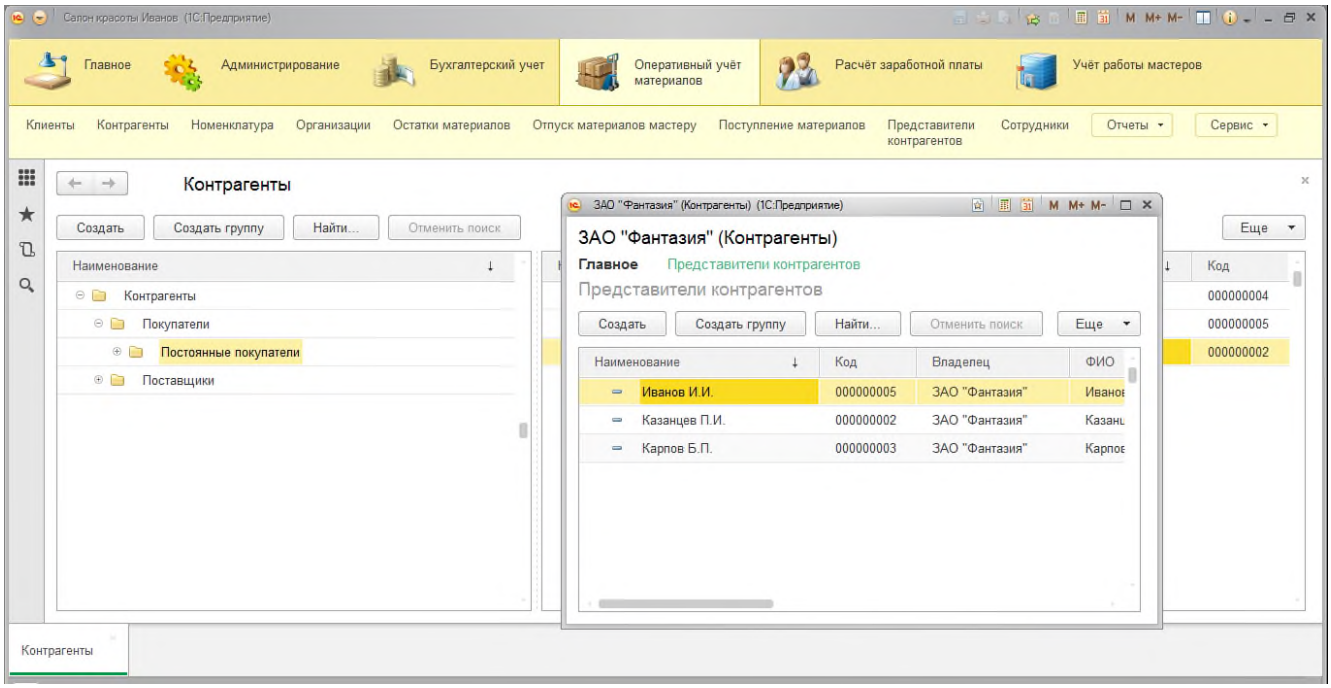


Рисунок 6.12- Формы списка и элемента справочника ПредставителиКонтрагентов

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 6.13.

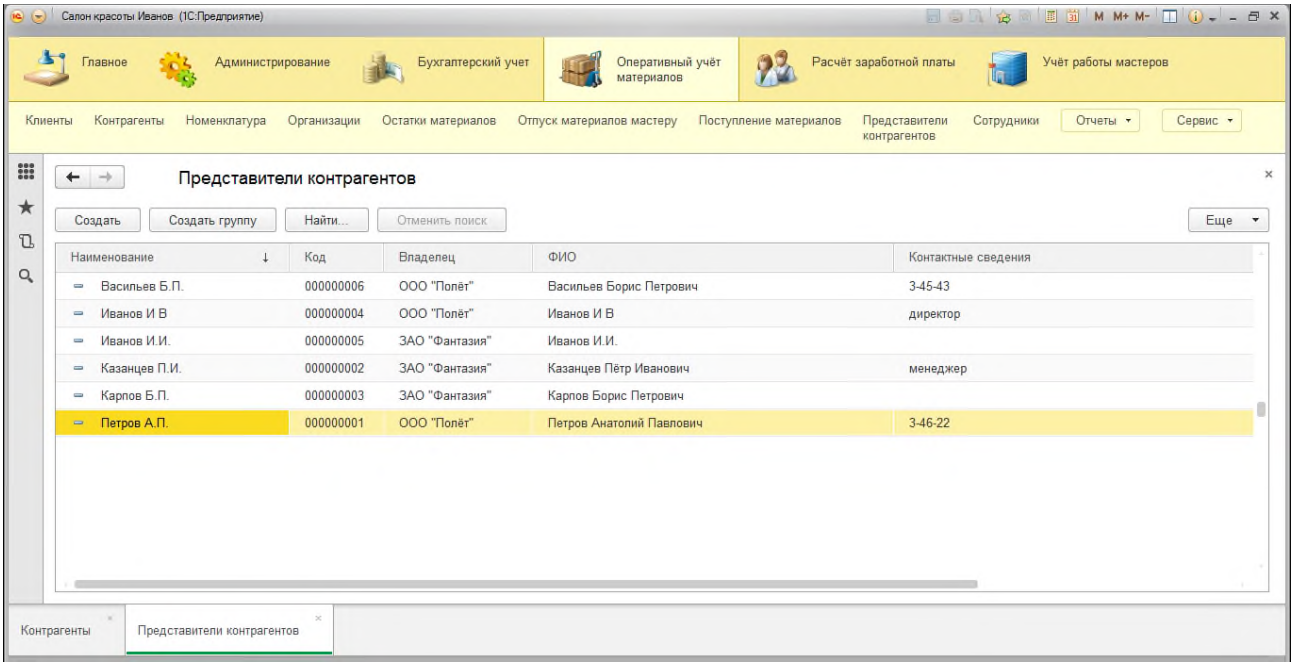


Рисунок 6.13- Просмотр формы списка справочника ПредставителиКонтрагентов

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 6.14.

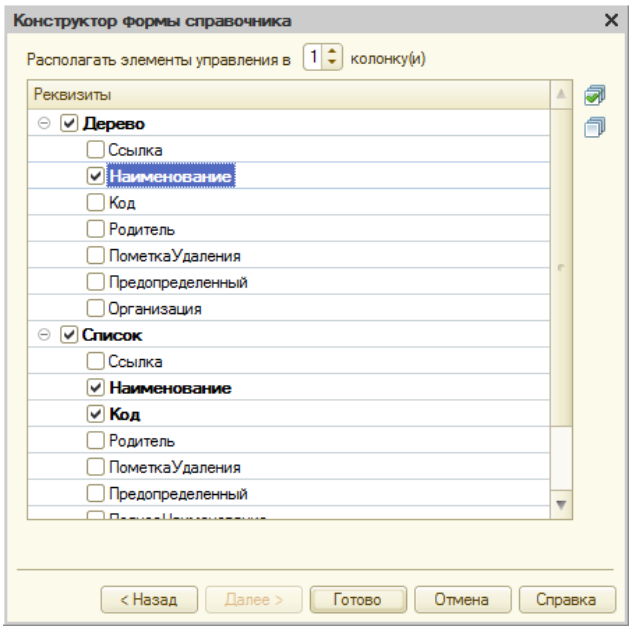


Рисунок 6.14- Конструктор формы справочника Контрагенты

Вот как будет выглядеть форма списка справочника в режиме 1С:Предприятие, рисунок 6.15

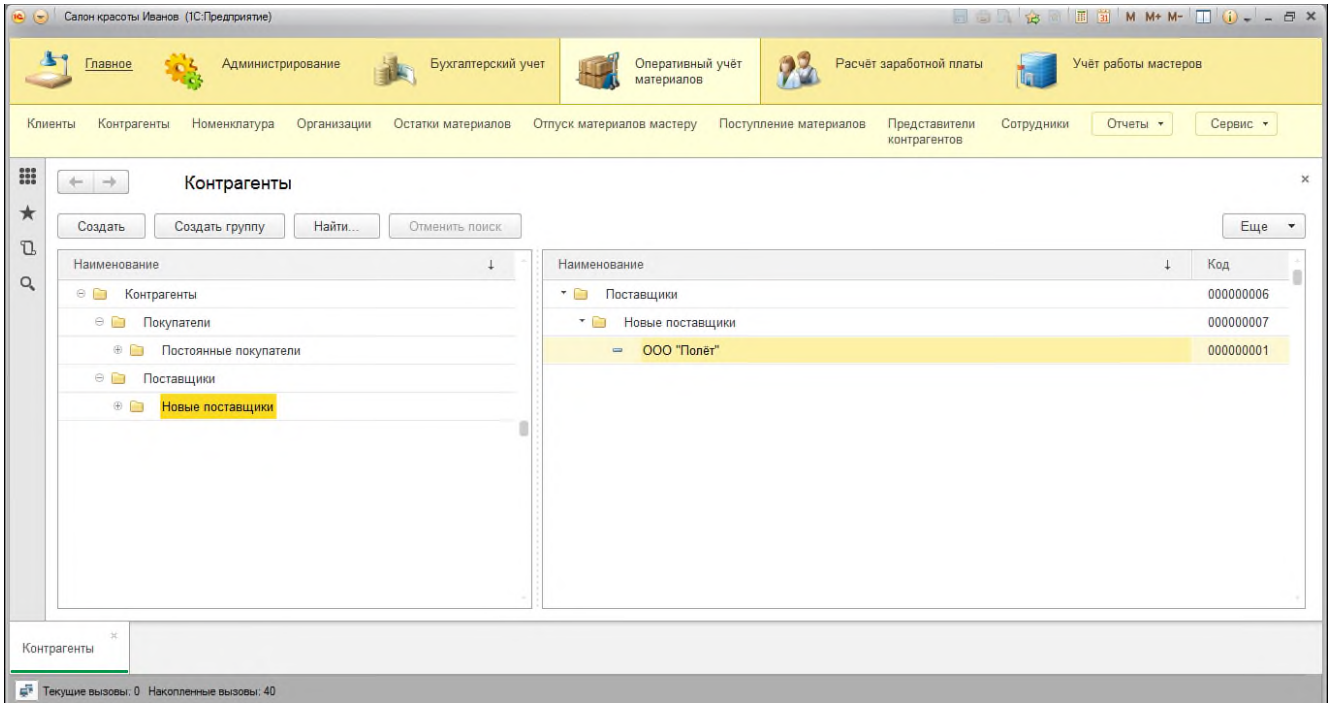


Рисунок 6.15 - Форма справочника со списком и деревом элементов

Практическая работа №7 «Организация обработки исключений»

Цель: научиться организации обработки исключений

ХОД РАБОТЫ:

1. 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 7.1.

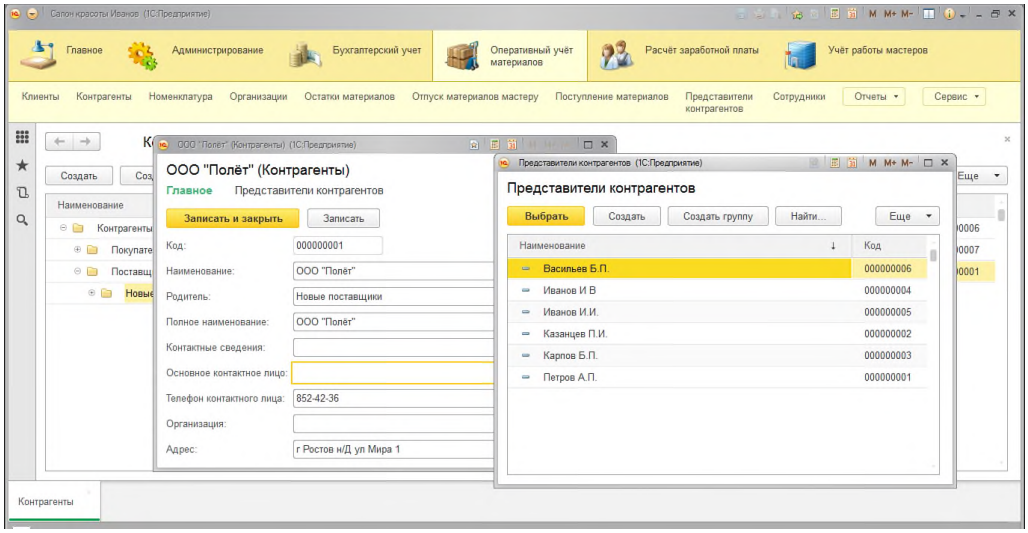


Рисунок 7.1- Попытка заполнения реквизита Основное контактное лицо

2. 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.

3. 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 7.2.

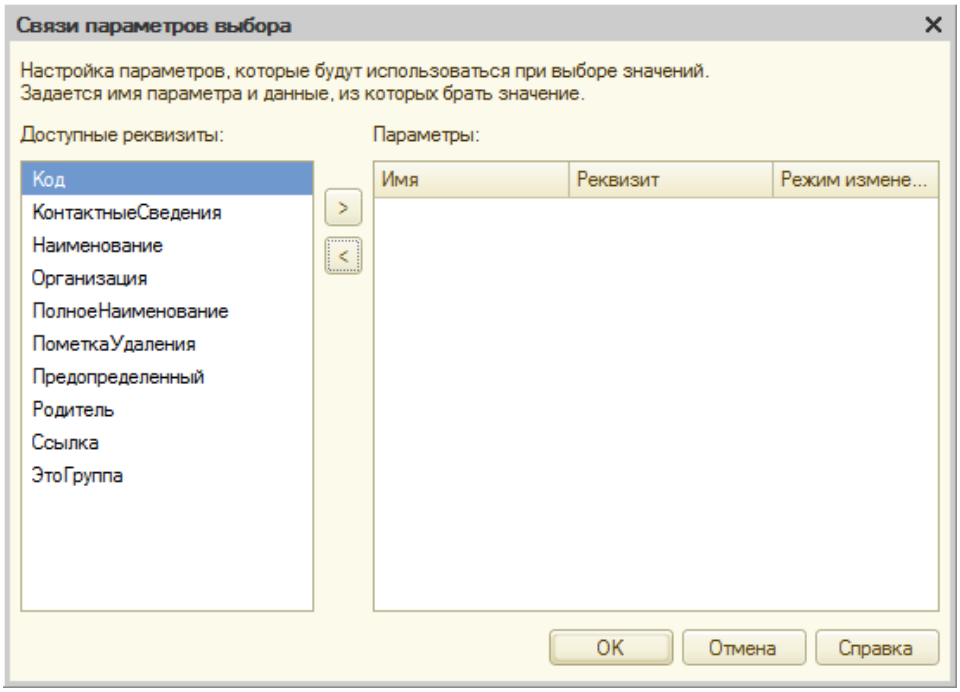


Рисунок 7.2- Окно Связи параметров выбора

4. 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 7.3.

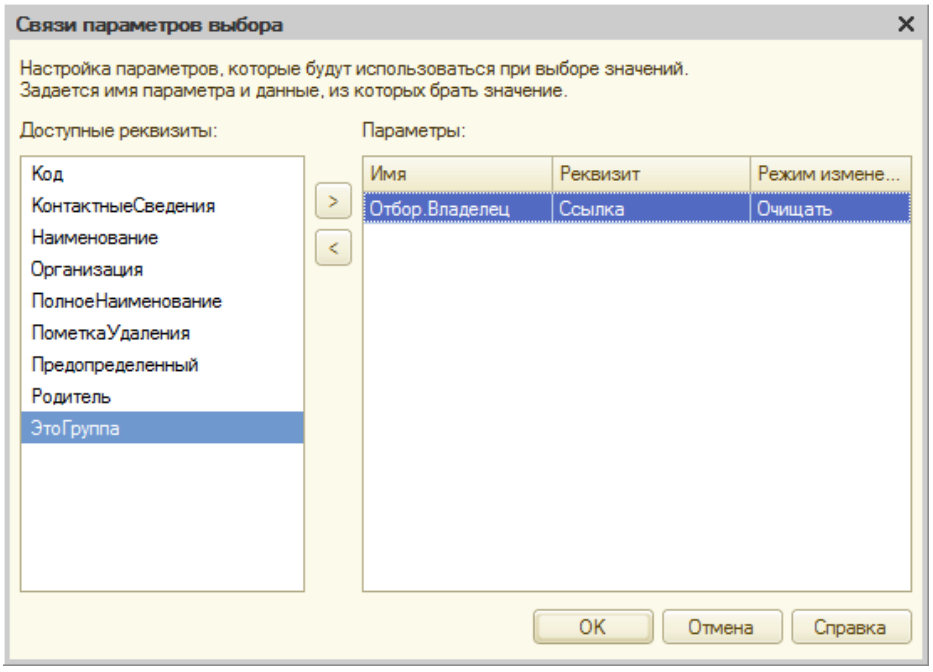


Рисунок 7.3- Окно Связи параметров выбора с настроенным параметром

5. 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 7.4.

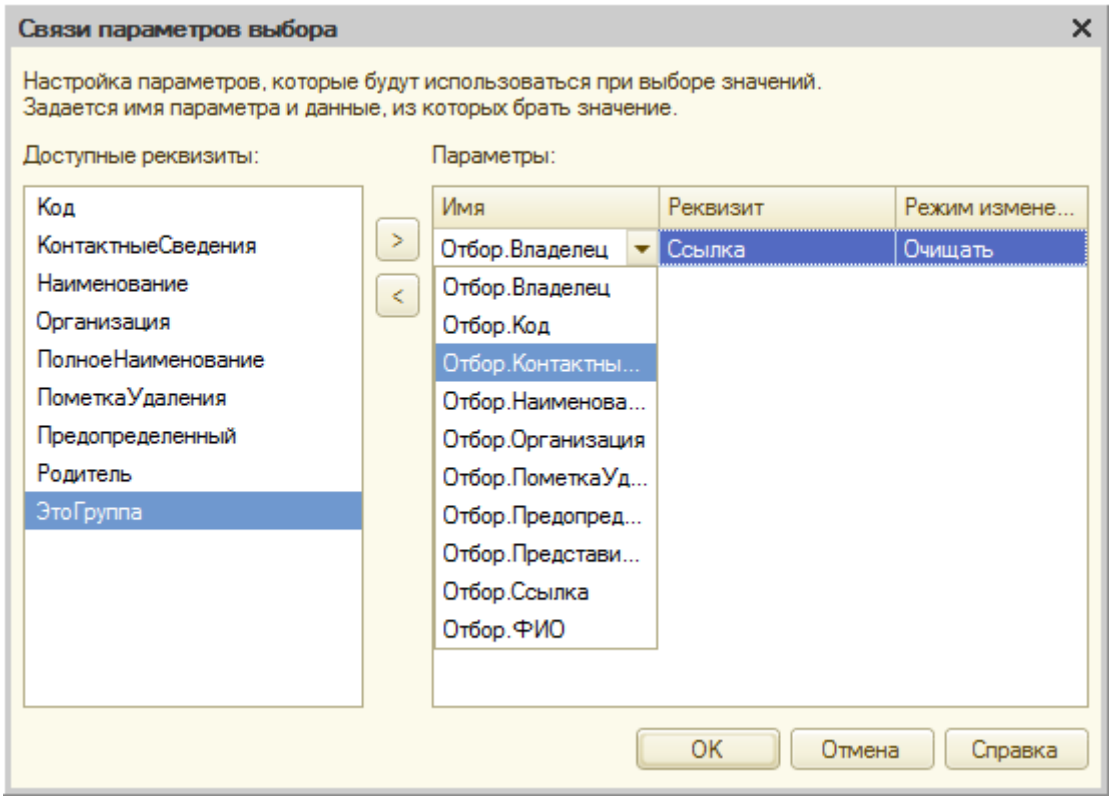


Рисунок 7.4- Возможные настройки в окне Связи параметров выбора

6. 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.

7. Добавим в справочник **Контрагенты** еще один реквизит – **ТелефонКонтактногоЛица**. Зададим тип – **Строка**, длина – **100**. Этот реквизит соответствует реквизиту**КонтактныеСведения** справочника **ПредставителиКонтрагентов**. Он нужен нам исключительно для удобства – для того, чтобы, когда в форме контрагента указано основное контактное лицо, пользователю не пришлось бы, для поиска телефона контактного лица, заглядывать в его карточку.

8. После добавления реквизита в справочник **Контрагенты**, запустим режим 1С:Предприятие и откроем форму одного из элементов этого справочника. Если присмотреться к этой форме на данном этапе работы, окажется, что реквизита **ТелефонКонтактногоЛица** на ней не наблюдается. Все дело в том, что, создав собственную форму элемента для справочника, мы отказываемся от автоматического механизма создания форм, который, если бы не наша, самостоятельно созданная ранее форма, автоматически построил бы форму с новым реквизитом.

9. Добавим элемент управления для реквизита **ТелефонКонтактногоЛица** на форму. Откроем форму элемента справочника Контрагенты для редактирования и перетащим реквизит **ТелефонКонтактногоЛица** с вкладки **Реквизиты** на вкладку **Элементы**, Рисунок 7.5.

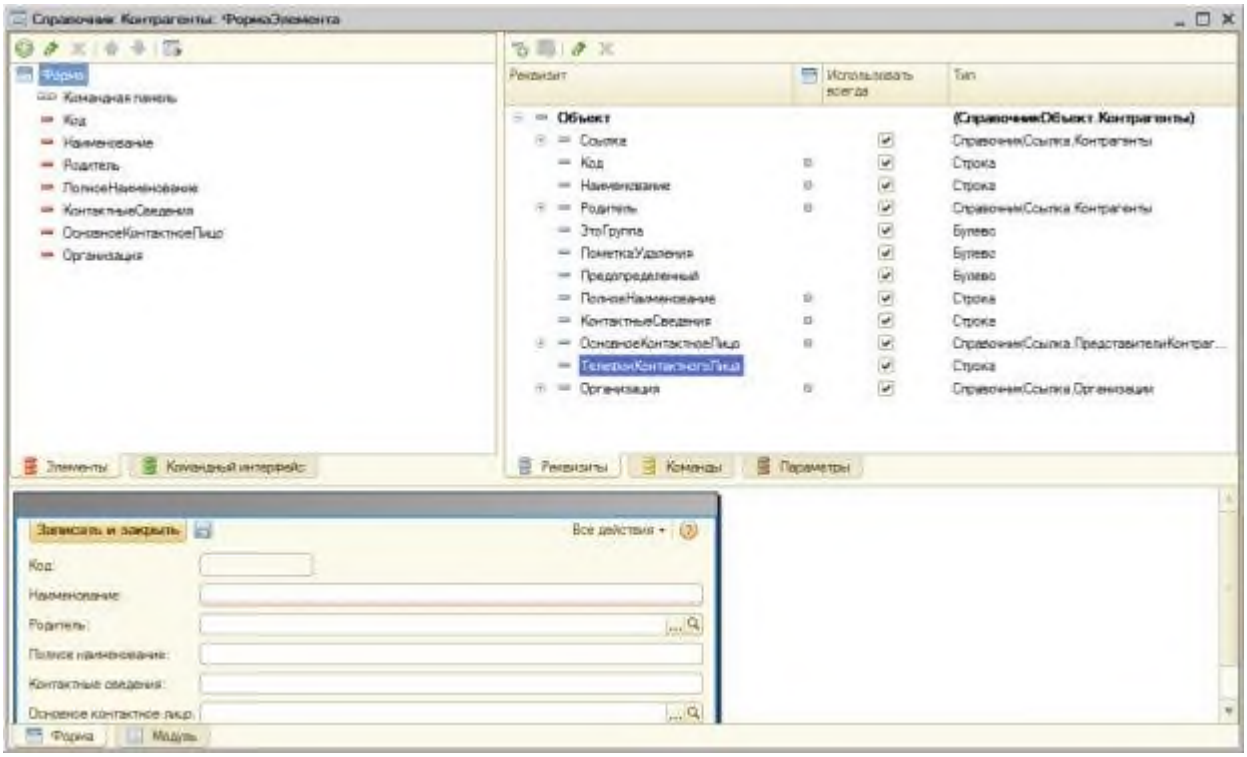


Рисунок 7.6- Реквизит «ТелефонКонтактногоЛица» нужно переместить с вкладки «Реквизиты» на вкладку «Элементы»

10. В справочнике **ПредставителиКонтрагентов** есть реквизит, который указывает на то, что представитель контрагента работает в организации-контрагенте – это реквизит логического типа **ПредставительРаботает**. Нам нужно реализовать следующий функционал. Если пользователь, заполняя карточку элемента справочника **Контрагенты** выбирает в качестве реквизита **ОсновноеКонтактноеЛицо** сотрудника, у которого флаг **ПредставительРаботает** не установлен – мы предупреждаем пользователя об этом, выводя сообщение.

Для этого нам понадобится перехватить событие изменения поля **ОсновноеКонтактноеЛицо**, после чего проверить, установлен ли у выбранного контактного лица флаг **ПредставительРаботает**, и, если такой флаг не установлен – вывести сообщение пользователю.

11. Из контекстного меню элемента формы **ОсновноеКонтактноеЛицо** выберем событие **ПриИзменении**, откроется редактор кода, в котором уже будет создана пустая клиентская процедура для перехвата этого события. К этому моменту реквизит **ОсновноеКонтактноеЛицо** уже будет содержать выбранного представителя контрагента. Нам понадобится серверная процедура, которая обратится к реквизиту этого представителя **ПредставительРаботает** и вернет нам его значение. После того, как мы получим с сервера сведения о том, работает ли представитель, мы примем решение – выводить ли пользователю сообщение или нет.

12. Все это реализовано с помощью нижеприведенного кода:

```
&НаКлиенте
Процедура ОсновноеКонтактноеЛицоПриИзменении(Элемент)
Если НЕ ПроверитьЗаполнениеРеквизита() Тогда
```

```
        Сообщить("Выбранное контактное лицо,
"+Объект.ОсновноеКонтактноеЛицо+", не работает у контрагента.");
        КонецЕсли;
        КонецПроцедуры

        &НаСервере
        Функция ПроверитьЗаполнениеРеквизита()
         Возврат (Объект.ОсновноеКонтактноеЛицо.ПредставительРаботает);
        КонецФункции
```

13. Кроме того, зададим автоматический механизм переноса в реквизит **ТелефонКонтактногоЛица** сведений из элемента справочника **ПредставителиКонтрагентов**, который указан в поле **ОсновноеКонтактноеЛицо**. В частности, дополним обработчик события для поля **ОсновноеКонтактноеЛицо ПриИзменении():**

14. Подготовим серверную процедуру, которая будет работать с реквизитами объекта, она будет иметь следующий вид:

```
        &НаСервере
        Процедура УстановитьНомерПредставителя()

Объект.ТелефонКонтактногоЛица=Объект.ОсновноеКонтактноеЛицо.Контактн
ыеСведения;
        КонецПроцедуры
```

15. Вызовем эту процедуру в уже существующем обработчике **ПриИзменении**, код модуля приобретет вид, показанный на Рисунок 7.7.

Кроме того, в окне его свойств отредактируем свойство **Вид** – выберем его значение **Поле надписи**. Пользователь не будет ничего в это поле вводить самостоятельно, поэтому поле надписи нас вполне устроит.

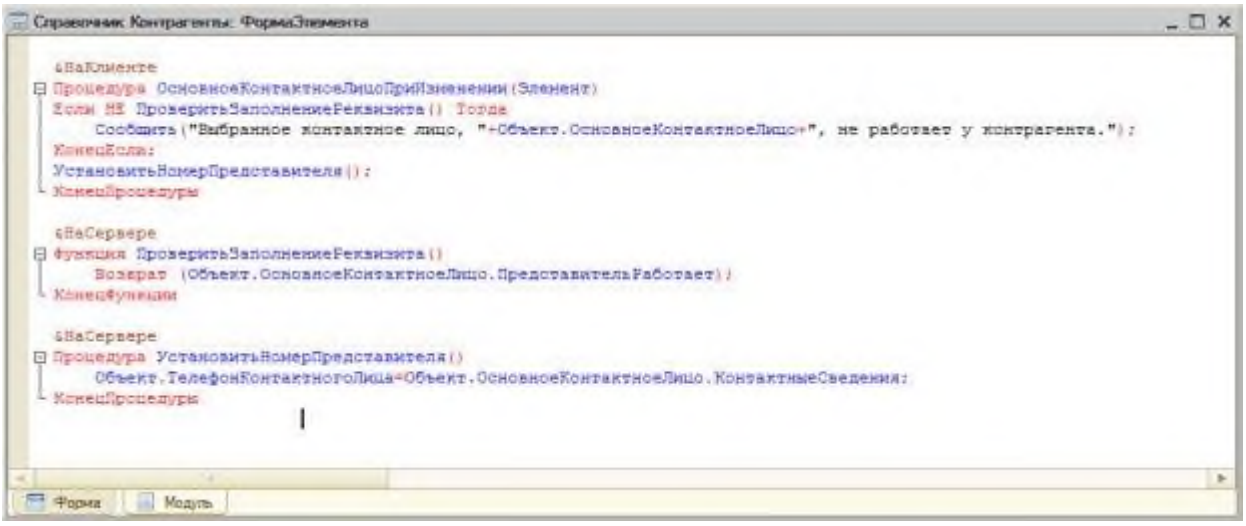


Рисунок 7.7- Код модуля формы элемента справочника Контрагенты

16. При выборе "неподходящего" представителя окно элемента справочника Контрагенты примет следующий вид, Рисунок 7.8.

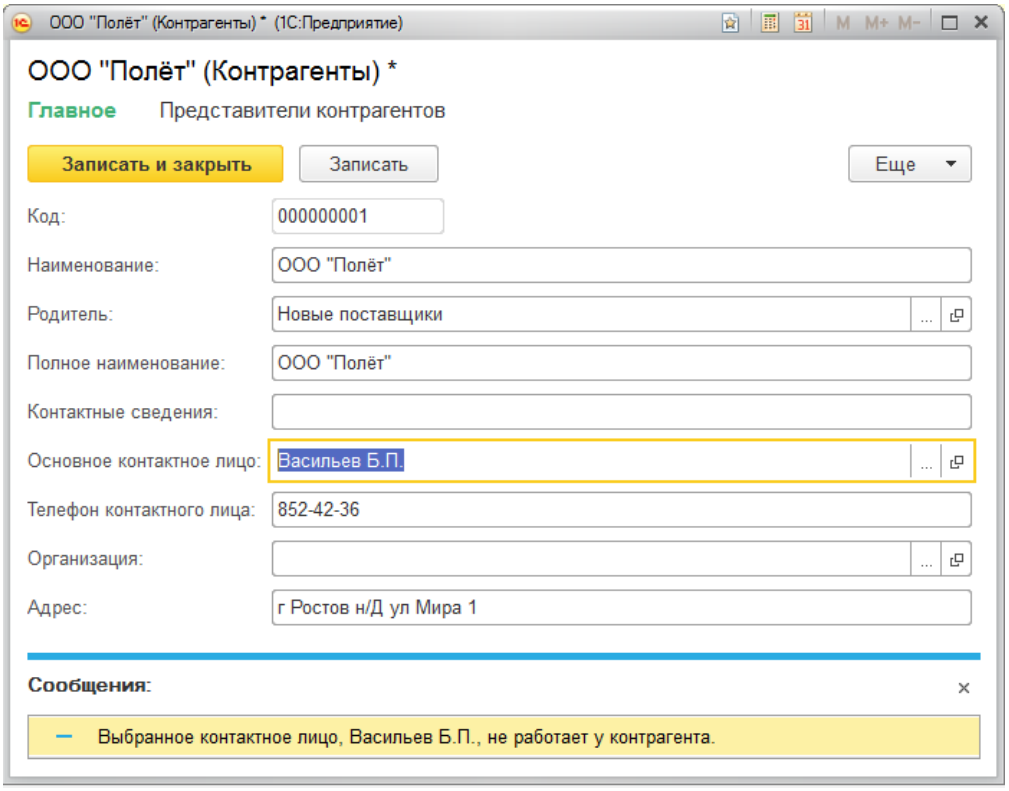


Рисунок 7.8- Сообщение о выборе неподходящего контактного лица

17. Объявим процедуру **ОбработкаПроверкиЗаполнения ()**, открыв модуль объекта (закладка **Прочее** окна редактирования свойств объекта, кнопка **Модуль объекта**) выбором из списка **Процедуры и функции панели инструментов** модуль процедуры ОбработкаПроверкиЗаполнения().

Эта процедура работает на сервере, мы можем напрямую обращаться к реквизитам объекта.

```
Процедура ОбработкаПроверкиЗаполнения(Отказ, ПроверяемыеРеквизиты)
 Если СтрДлина(ПолноеНаименование) < 5 И НЕ ЭтотОбъект.ЭтоГруппа Тогда
    Отказ=Истина;
    Сообщить("Полное наименование организации должно быть не короче 5-ти символов");
 КонецЕсли;
КонецПроцедуры
```

Передаваемый в процедуру параметр **Отказ** можно установить в значение **Истина** для того, чтобы показать, что проверка не пройдена. **ПроверяемыеРеквизиты** – это массив, он содержит реквизиты для автоматической проверки, в частности, там находятся те реквизиты, для которых включена автоматическая проверка заполнения. По умолчанию это – реквизит **Наименование**. В этом можно убедиться, просмотрев содержимое переменной в отладчике.

Практическая работа №8 «Применение отладочных классов в проекте»

Цель работы: научиться применять отладочные классы в проекте.

ХОД РАБОТЫ:

1. Начнем с создания обработки, которая выводит имена всех справочников, имеющихся в системе. Для этого добавим новую обработку в ветви **Обработки** дерева конфигурации. Назовем ее **РаботаСоСправочниками**, Рисунок 8.1.

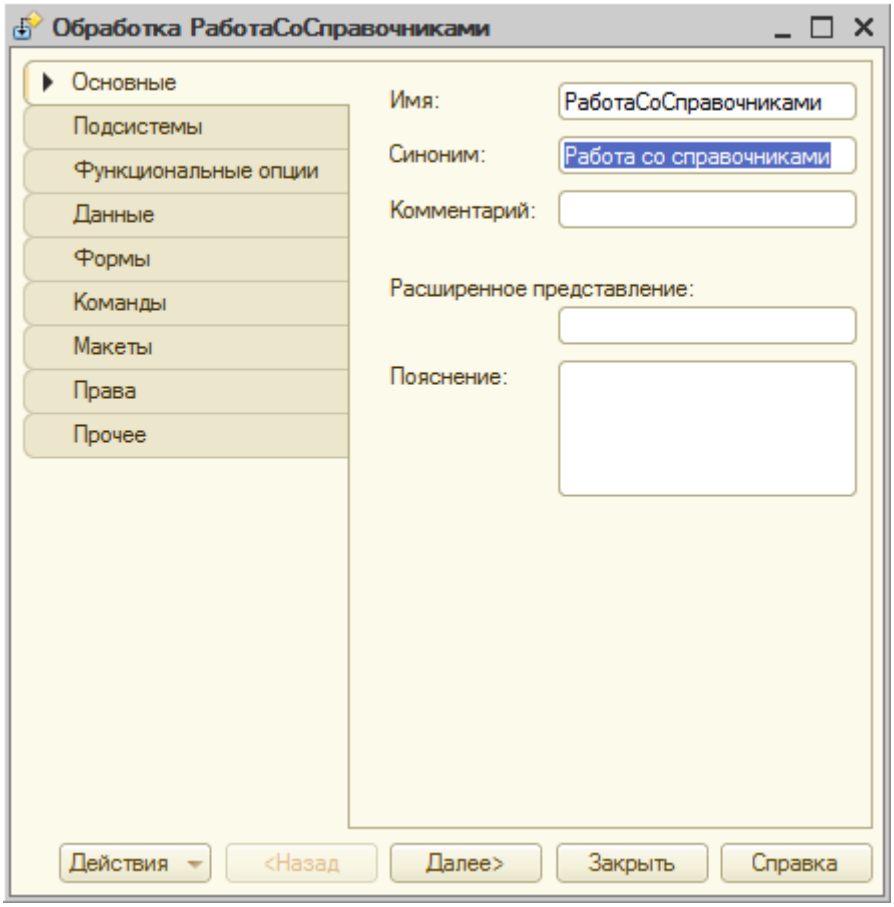


Рисунок 8.1- Создание обработки

2. Включим новую обработку в состав подсистемы **Администрирование** на закладке **Подсистемы**. Перейдем на закладку **Формы** и создадим форму обработки. Наша обработка не имеет реквизитов – сразу после запуска конструктора формы обработки, мы можем нажать на кнопку **Готово** и увидим пустую форму обработки, Рисунок 8.2.

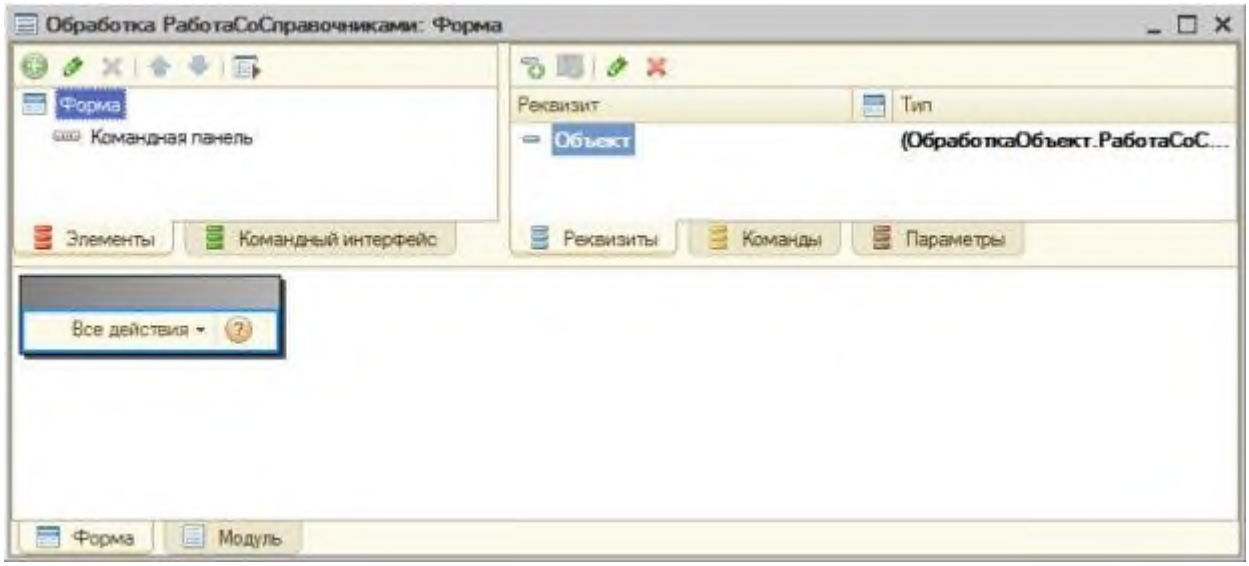


Рисунок 8.2- Форма обработки

3. Перейдем на вкладку **Команды** в окне редактора форм. После этого нам будут доступны еще несколько вкладок, нас интересует первая из них – **Команды формы**, Рисунок 8.3.

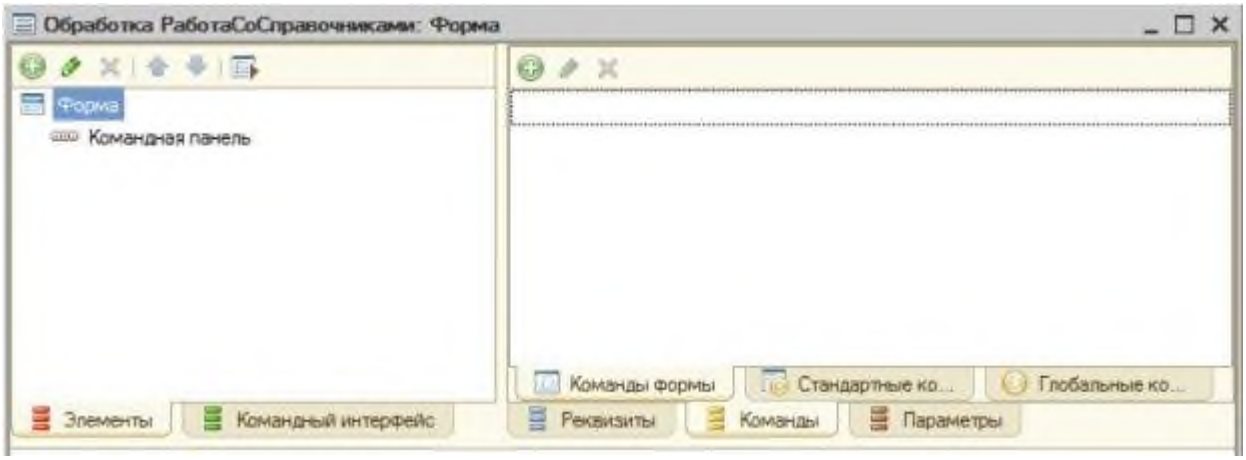


Рисунок 8.3- Переход к командам формы обработки

4. Список команд формы пуст – нам нужно создать собственную команду. Нажмем на кнопку **Добавить** в верхней части панели **Команды формы**, назовем ее **ВывестиСписокСправочников**, в окне свойств команды нажмем на кнопку с увеличительным стеклом в поле свойства **Действие** – в модуле формы будет создана процедура для этой команды, Рисунок 8.4.

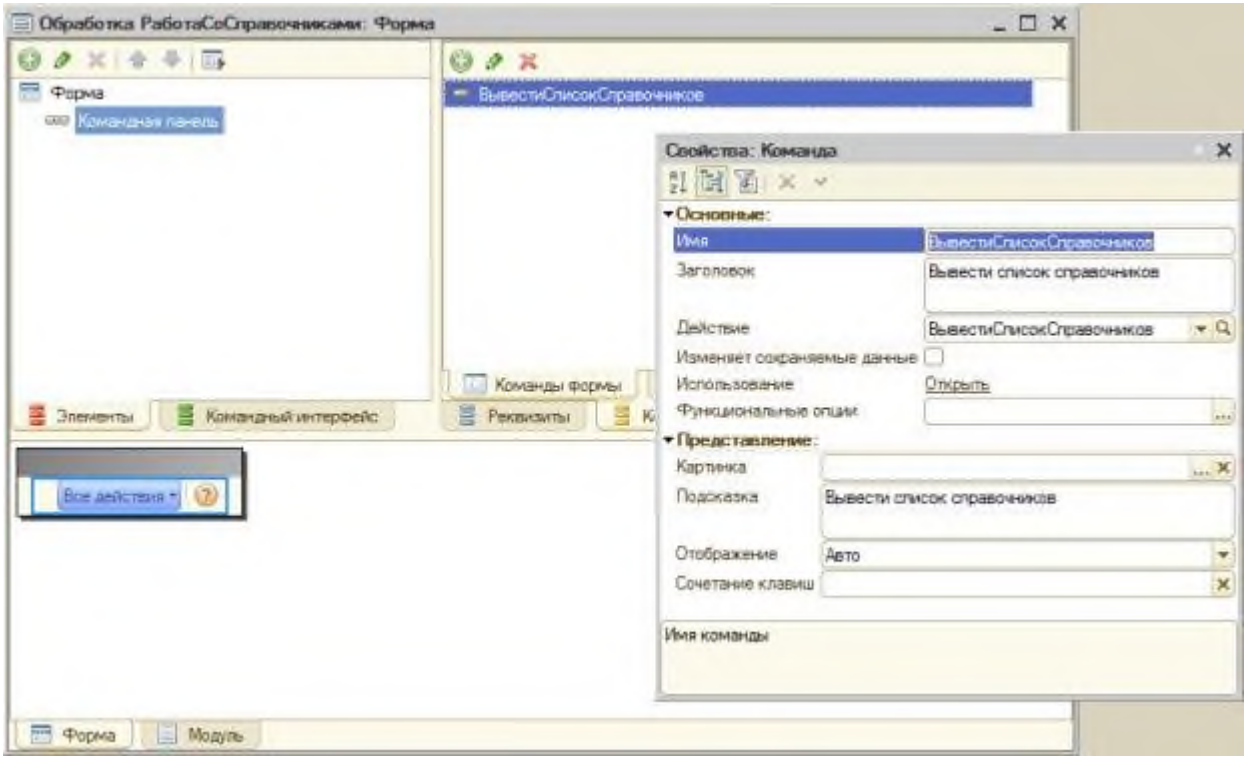


Рисунок 8.4- Настройка команды

В модуль формы был добавлен такой код:

```
&НаКлиенте
Процедура ВывестиСписокСправочников(Команда)
  // Вставить содержимое обработчика.
КонецПроцедуры
```

То, что мы добавили в обработку команду, еще не означает автоматическое добавление на форму команды, например, кнопки, нажатие которой приведет к выполнению команды. Добавить такую кнопку на форму можно несколькими способами. Во-первых, мы можем просто перетащить команду из панели **Команды формы** на панель **Элементы** – на форме появится кнопка **Вывести список справочников**, а напротив команды – серый квадратик, говорящий о присутствии элемента управления, связанного с командой, на форме.

Во-вторых, в список элементов формы можно добавить кнопку (кнопка **Добавить** в командной панели закладки **Элементы**) и задать свойства кнопки, в частности, в свойстве **ИмяКоманды** выбрать нужную команду. После добавления кнопки и настройки ее связи с командой, редактор форм приобрел вид, показанный на Рисунок 8.5.

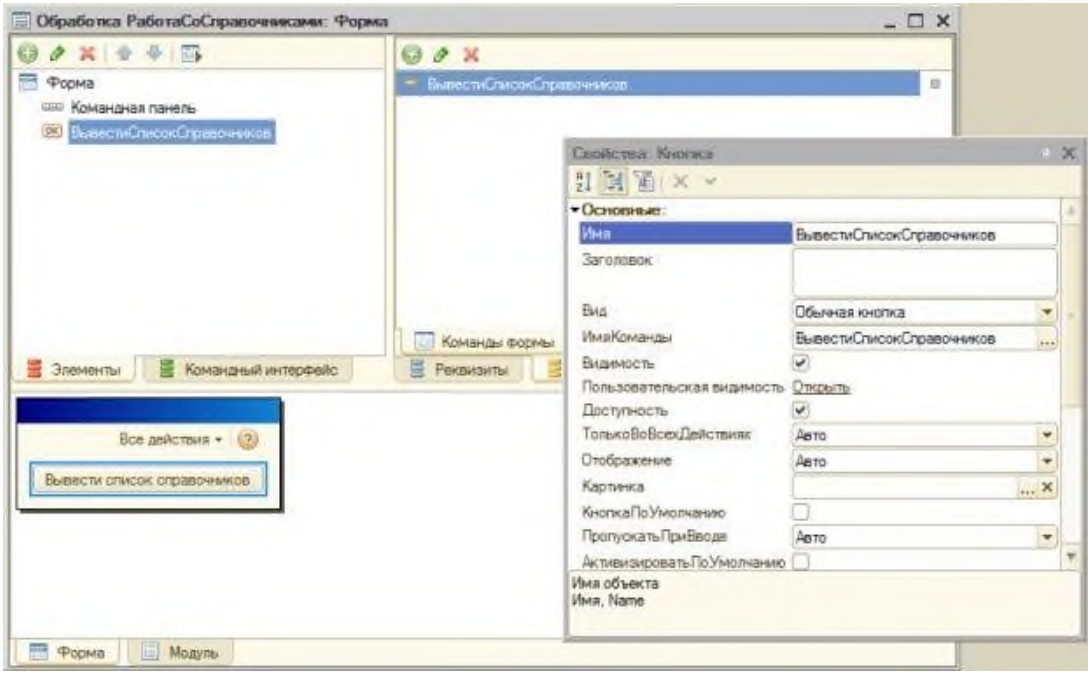


Рисунок 8.5- Настройка кнопки

5. Теперь приступим к редактированию кода. Код команды выполняется на клиенте, нам же нужно работать с базой данных, то есть – объявить серверную процедуру или функцию. В итоге у нас получился следующий код:

```
&НаКлиенте
Процедура ВывестиСписокСправочников(Команда)
  ВывестиИменаСправочников();
КонецПроцедуры

Процедура ВывестиИменаСправочников()
  Для каждого Справочник из Метаданные.Справочники Цикл
        Сообщить (Справочник.Имя);
  КонецЦикла;
КонецПроцедуры
```

Мы получаем имена справочников и выводим их в окно сообщений, Рисунок 8.6.

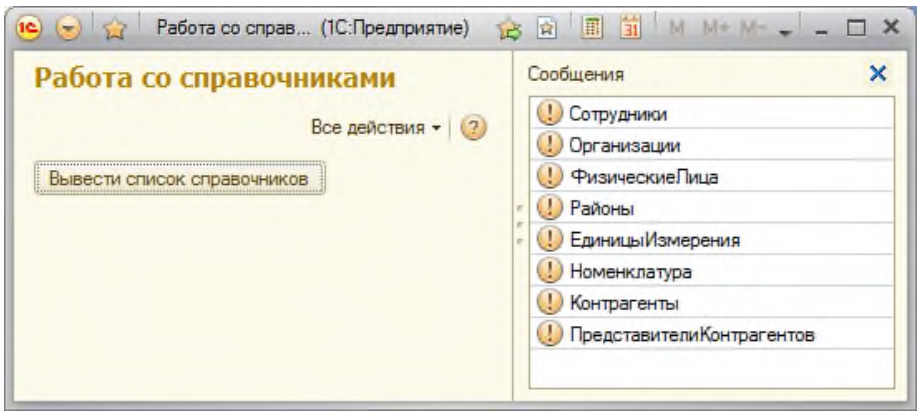


Рисунок 8.6- Вывод списка справочников

6. Теперь нужно программно создать элемент справочника с заданными параметрами. На верхнем уровне типов данных, которые имеют отношение к справочникам, находится объект **Справочники**, имеющий тип **СправочникиМенеджер**. С его помощью можно обращаться к отдельным справочникам, через их **объектыСправочникМенеджер**. При работе с объектом типа **СправочникиМенеджер** используется свойство глобального контекста **Справочники**.

Обращение к объектам **СправочникМенеджер** возможно по имени справочника, заданному в конфигурации. Мы собираемся программно создать элемент с наименованием, которое задаст пользователь в форме обработки. Для этого добавим в список команд формы новую – назовем ее **СоздатьЭлементСправочника**, создадим ее процедуру, добавим ее на форму. Добавим новый реквизит в список реквизитов, назовем его **НаименованиеЭлемента**, зададим тип – Строка, длина 25, так же переместим реквизит в область **Элементы** – там он будет представлен в виде текстового поля, Рисунок 8.7.

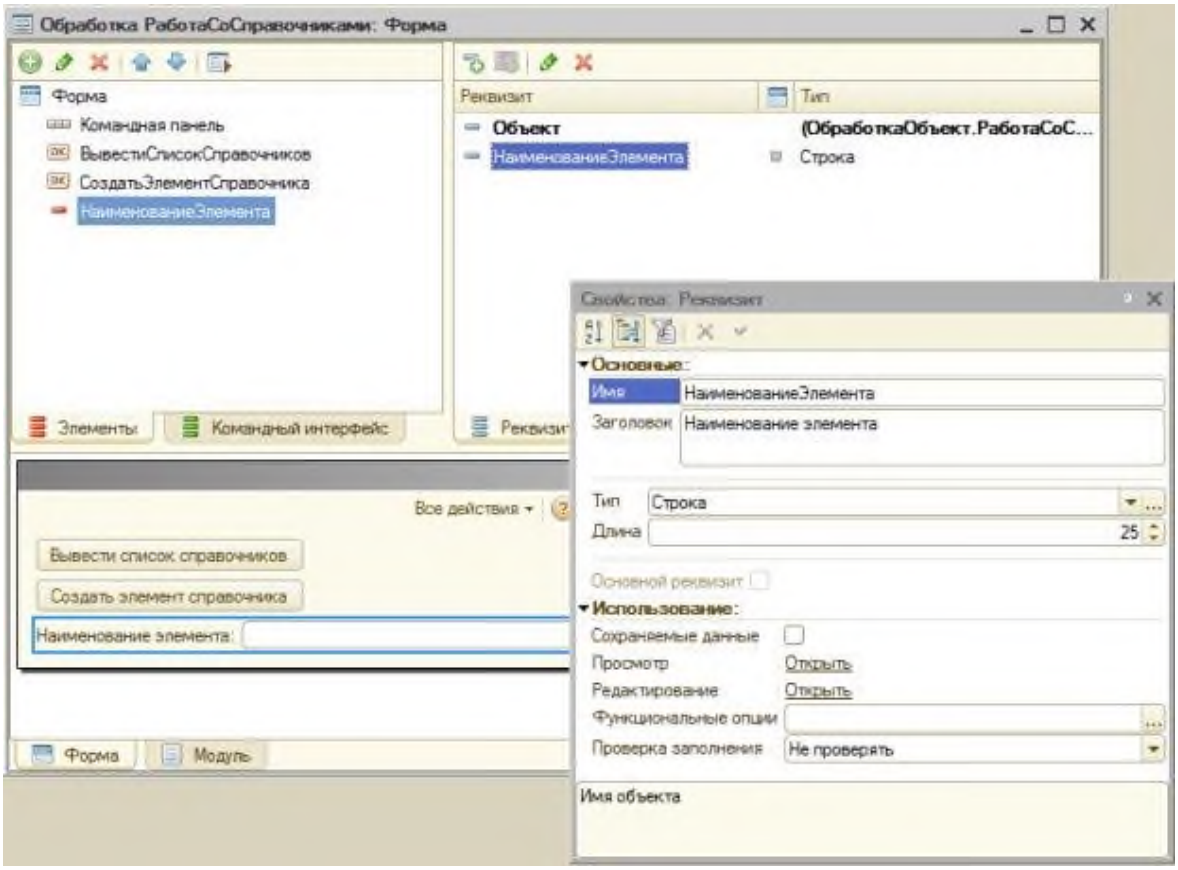


Рисунок 8.7- Настройка нового реквизита формы

7. Добавим еще один реквизит – назовем его **ИмяСправочника**, тип Строка, длина – 100. Сюда пользователь будет вводить имя справочника, в котором он хочет создать новый элемент. На нашей форме теперь имеются три логически связанных элемента. Удобно объединить их в одну группу, чтобы пользователь сразу мог понять, что они работают вместе. Для этого можно сгруппировать элементы. В командной панели вкладки Элементы нажмем на кнопку Добавить, появится окно – Тип элемента (Рисунок 8.8.), среди списка элементов, представленных в котором, можно найти несколько видов групп.

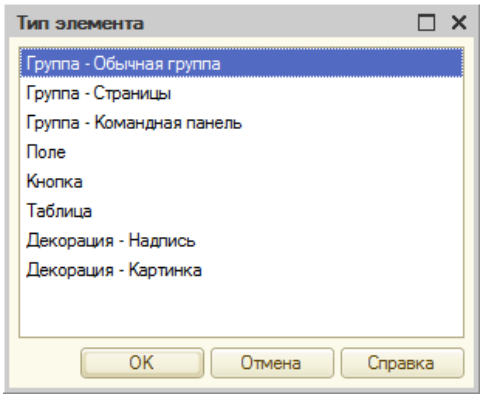


Рисунок 8.8- Добавление новой группы на форму

8. Добавим на форму новую группу, назовем ее **СозданиеЭлементаСправочника**, перетащим в нее элементы управления, относящиеся к этой группе. Результат реорганизации элементов показан на Рисунок 8.9.

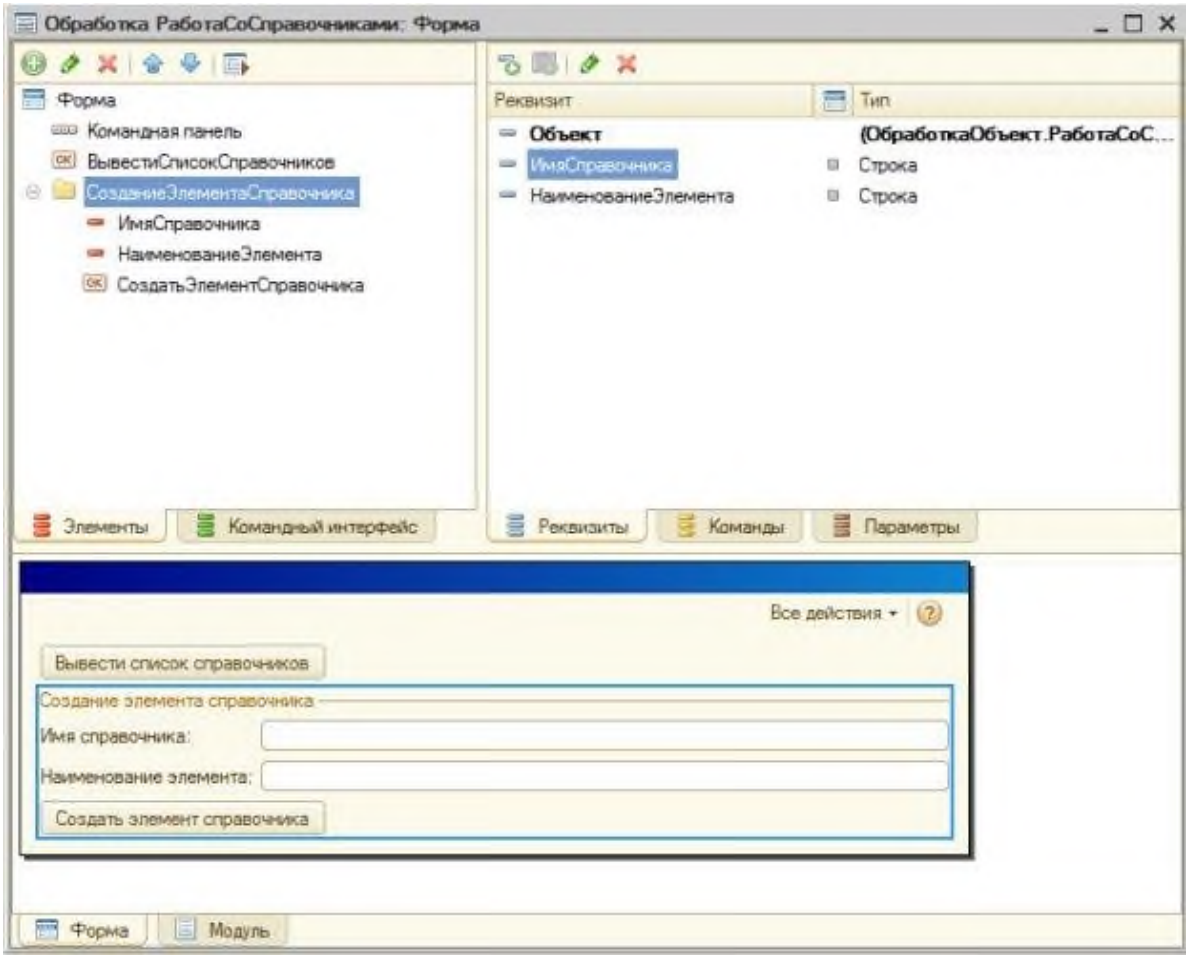


Рисунок 8.9- Добавление новой группы на форму

9. Теперь займемся кодом. Нам, в дополнение к клиентской процедуре команды **СоздатьЭлементСправочника**, понадобится серверная процедура или функция, которая и занимается созданием элемента. Обратиться к объекту **СправочникМенеджер** для конкретного справочника можно различными способами. Предположим, мы заранее знаем, с каким справочником нам нужно работать (например, это – справочник Номенклатура). Для того, чтобы вызвать метод этого справочника СоздатьЭлемент, нам понадобится такая конструкция:

```
НовыйЭлемент=Справочники.Номенклатура.СоздатьЭлемент();
```

10. После того, как мы получили переменную типа **СправочникОбъект**, мы можем настроить необходимые свойства конкретного элемента справочника (в нашем случае – наименование) и записать элемент. Вот, как выглядит результирующий код:

```
&НаКлиенте
Процедура СоздатьЭлементСправочника(Команда)
  КодНовогоЭлемента=СоздатьЭлементСправочникаНаСервере();
  Сообщить("В справочнике "+ИмяСправочника+" создан элемент
   "+НаименованиеЭлемента + " с автоматически присвоенным кодом:
     "+КодНовогоЭлемента);
КонецПроцедуры

Функция СоздатьЭлементСправочникаНаСервере()
  НовыйЭлемент = Справочники[ИмяСправочника].СоздатьЭлемент();
  НовыйЭлемент.Наименование=НаименованиеЭлемента;
  НовыйЭлемент.Записать();
  Возврат (НовыйЭлемент.Код);
КонецФункции
```

Вот, каковы результаты работы этого кода, Рисунок 8.10.

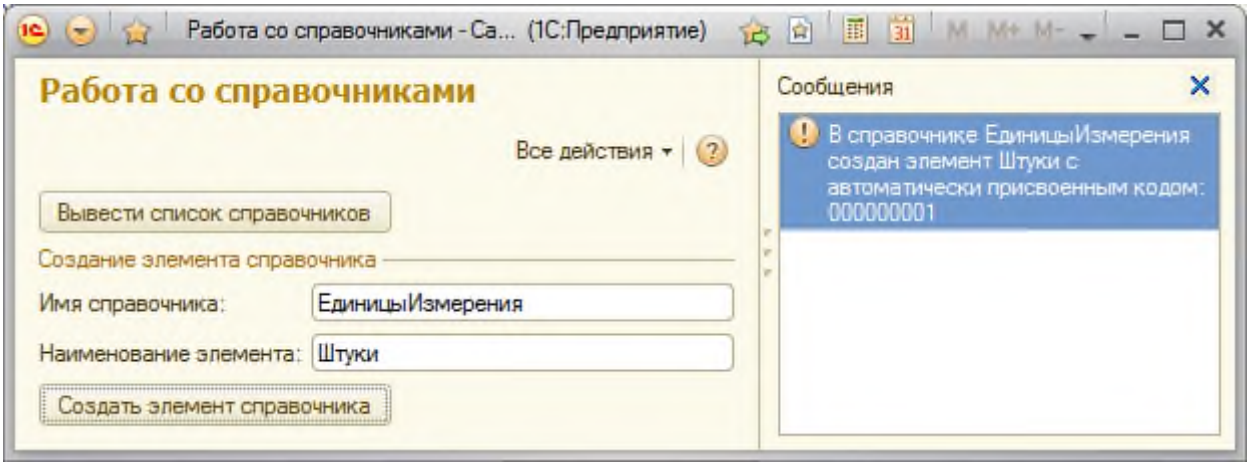


Рисунок 8.10- Создание нового элемента справочника

11. После создания процедуры, связанной с этой командой и серверной процедуры, выполняющей работу с базой. У нас получился такой код:

```
&НаКлиенте
Процедура ПометитьНаУдалениеВсеЭлементыСправочника(Команда)
  ПометитьНаУдаление();
КонецПроцедуры

Процедура ПометитьНаУдаление()
  СчетчикПомеченных = 0;
  Выборка = Справочники[ИмяСправочника].Выбрать();
 Пока Выборка.Следующий() Цикл
```

```
            Элемент=Выборка.ПолучитьОбъект();
            Если НЕ Элемент.ЭтоГруппа Тогда
              Элемент.УстановитьПометкуУдаления(Истина);
              СчетчикПомеченных=СчетчикПомеченных+1;
            КонецЕсли;
          КонецЦикла;
  Сообщить("В справочнике "+ИмяСправочника+" помечено на удаление"+СчетчикПомеченных+" элементов");
КонецПроцедуры
```

.

12. Нужно в заданном справочнике нужно найти элемент с заданным наименованием (или сообщить, что элемента с таким наименованием в справочнике нет), изменить регистр символов в наименовании таким образом, чтобы все буквы были прописными, и сообщить пользователю его код с указанием старого и нового наименования.

Обычным образом добавим в форму обработки новую команду, для указания имени справочника и наименования искомого элемента используем те же реквизитыИмяСправочника и НаименованиеЭлемента, реорганизуем элементы управления на форме, Рисунок 8.11.

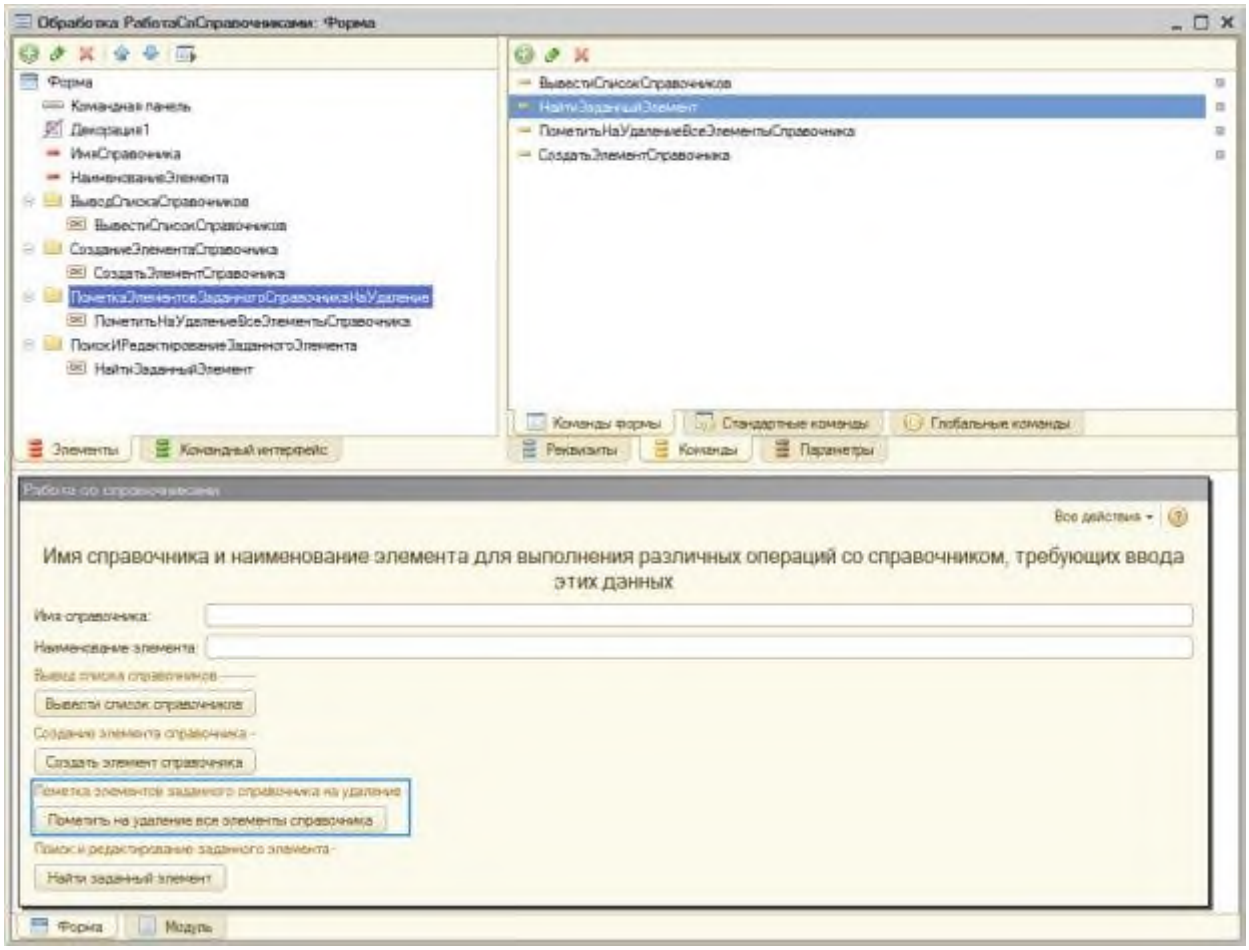


Рисунок 8.11- Переработанная форма

Поиск, редактирование заданного элемента и вывод необходимых сообщений реализуется с помощью следующего кода:

```
&НаКлиенте
Процедура НайтиЗаданныйЭлемент(Команда)
```

```
        НайтиЗаданныйЭлементНаСервере();
    КонецПроцедуры

    Процедура НайтиЗаданныйЭлементНаСервере()

ССылкаНаЭлемент=Справочники[ИмяСправочника].НайтиПоНаименованию(Наим
енованиеЭлемента);
        Если сылкаНаЭлемент=Справочники[ИмяСправочника].ПустаяСсылка()
Тогда
          Сообщить ("В справочнике "+ИмяСправочника+
    " нет элемента "+НаименованиеЭлемента);
        Иначе
          Элемент=ССылкаНаЭлемент.ПолучитьОбъект();
          СтароеНаименование=Элемент.Наименование;
          Элемент.Наименование=ВРег(Элемент.Наименование);
          Элемент.Записать();
          Сообщить("Элемент справочника "+ИмяСправочника+"
            с кодом "+Элемент.Код+" найден, наименование изменено
              с "+СтароеНаименование+" на "+Элемент.Наименование);
        КонецЕсли;
    КонецПроцедуры
```

Вот как выглядит работа этого кода, Рисунок 8.12.

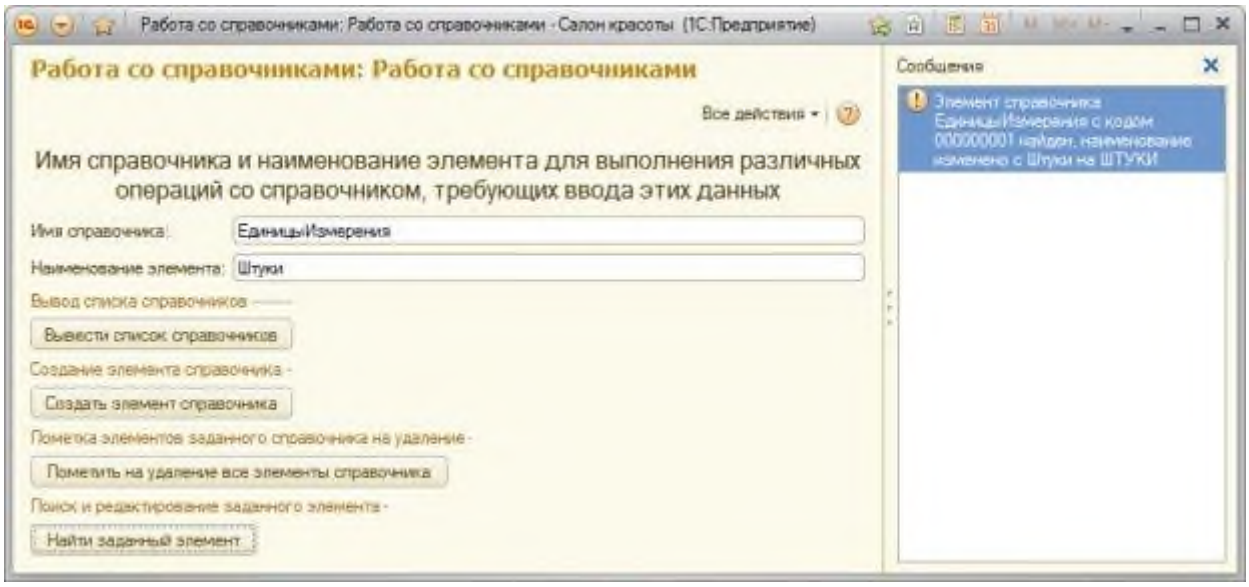


Рисунок 8.12 - Результат работы кода по поиску и редактированию элемента справочника

13. Разберем **обращение к табличным частям справочника**. В форму списка справочника **Физические лица** вставить кнопку «**Трудовая история**», при нажатии на которую в окно сообщений вывести список организаций, в которых работал тот сотрудник, на котором стоит курсор в этой форме списка.

13.1 Откроем в конфигураторе объект **Справочник Физические лица**, закладку **Формы**, создадим новую форму списка (рисунок 8.13).

Создаем новую кнопку «**ТрудоваяИстория**» в командах формы, рисунок 8.14.

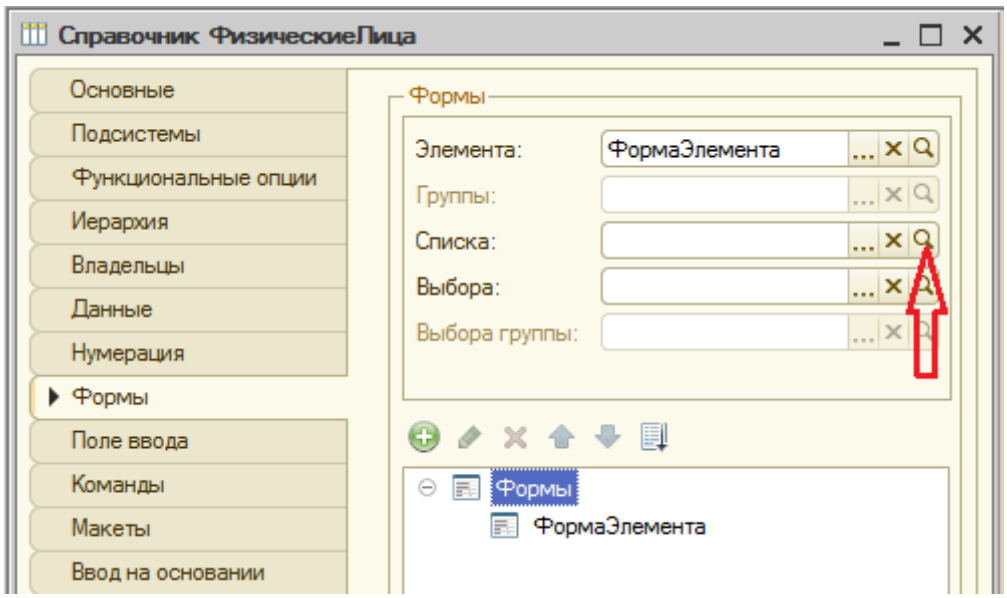


Рисунок 8.13 – Создание формы списка справочника «Физические лица»

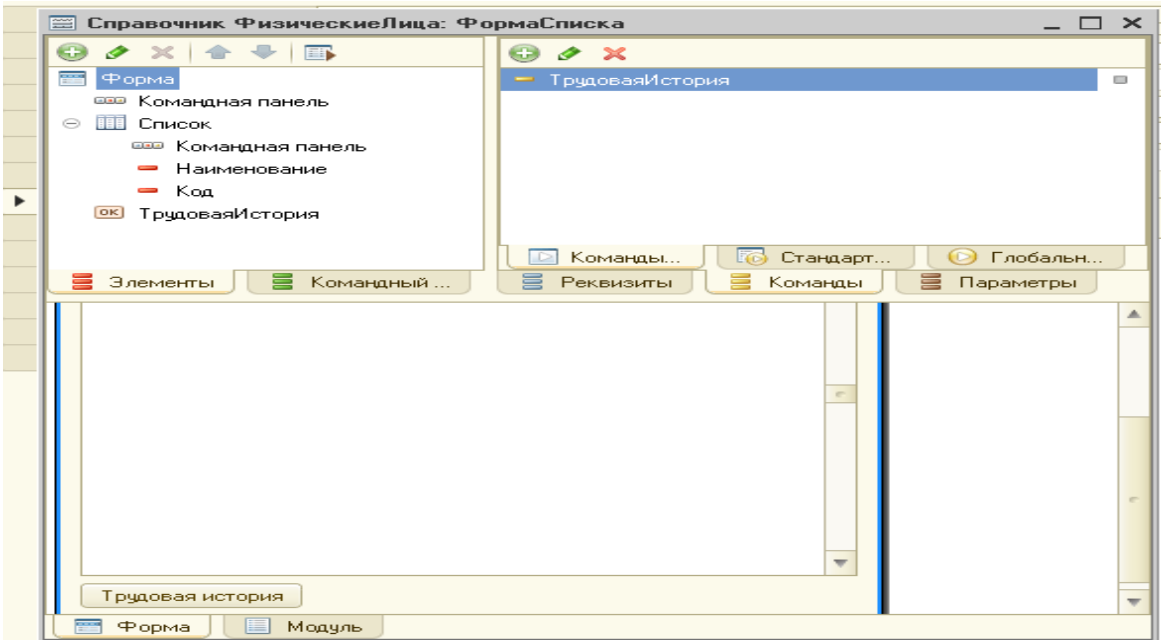


Рисунок 8.14 - Настройка кнопки в форме списка справочника ФизЛица

13.2 Пропишем код в модуле формы списка:

```
&НаКлиенте
Процедура ТрудоваяИстория(Команда)
      ФизЛицо1=Элементы.Список.ТекущаяСтрока;
      ПолучитьИсторию(ФизЛицо1);
КонецПроцедуры
```

```
Процедура ПолучитьИсторию(ФизЛицо)
      Для каждого стр из ФизЛицо.ТрудоваяИстория  Цикл
            Сообщить(Стр.Организация);
      КонецЦикла;
      КонецПроцедуры
```

Табличная часть справочника – это коллекция и обрабатывается циклом «Для каждого…».

13.3 Просмотрите работу модуля в отладке, в режиме пользователя.

14. Выполним обращение к данным подчиненного справочника.

Вывести список всех подразделений у выбранной в форме списка организации.

14.1 Аналогичным образом создадим форму списка и кнопку для справочника **Организации**.

(рисунок 8.15).

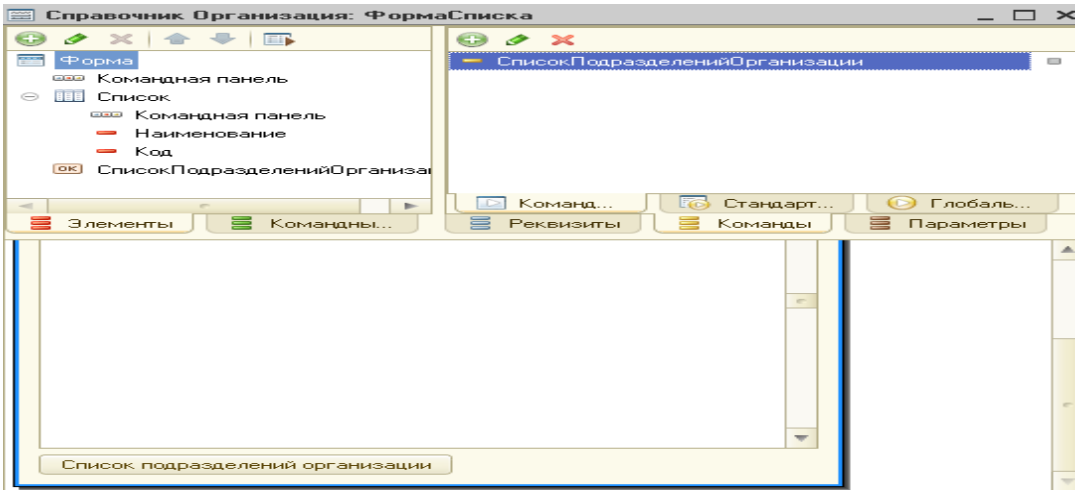


Рисунок 8.15 - Настройка кнопки в форме списка справочника Организации

14.2 Пропишем код в модуле формы списка:

```
&НаКлиенте
Процедура СписокПодразделенийОрганизации(Команда)
      выбОрганизация=Элементы.Список.ТекущаяСтрока;
      ПолучитьСписок(выбОрганизация);
КонецПроцедуры

Процедура ПолучитьСписок(Организация)
Спр=Справочники.ПодразделенияОрганизации.Выбрать(,Организация.ссылка);
  Пока Спр.Следующий() Цикл Сообщить(Спр.Наименование);
  КонецЦикла; КонецПроцедуры
```

Здесь метод **Выбрать()** в качестве параметра содержит ссылку на элемент справочника-владельца.

14.3 Просмотрите работу модуля в отладке, в режиме пользователя

Практическая работа №9 «Отладка проекта»

Цель работы: научиться выполнять отладку проекта

ХОД РАБОТЫ:

1. Создадим в ветви дерева конфигурации **Отчеты** новый отчет, дадим ему имя **СписокКонтрагентов**, Рисунок 9.1.

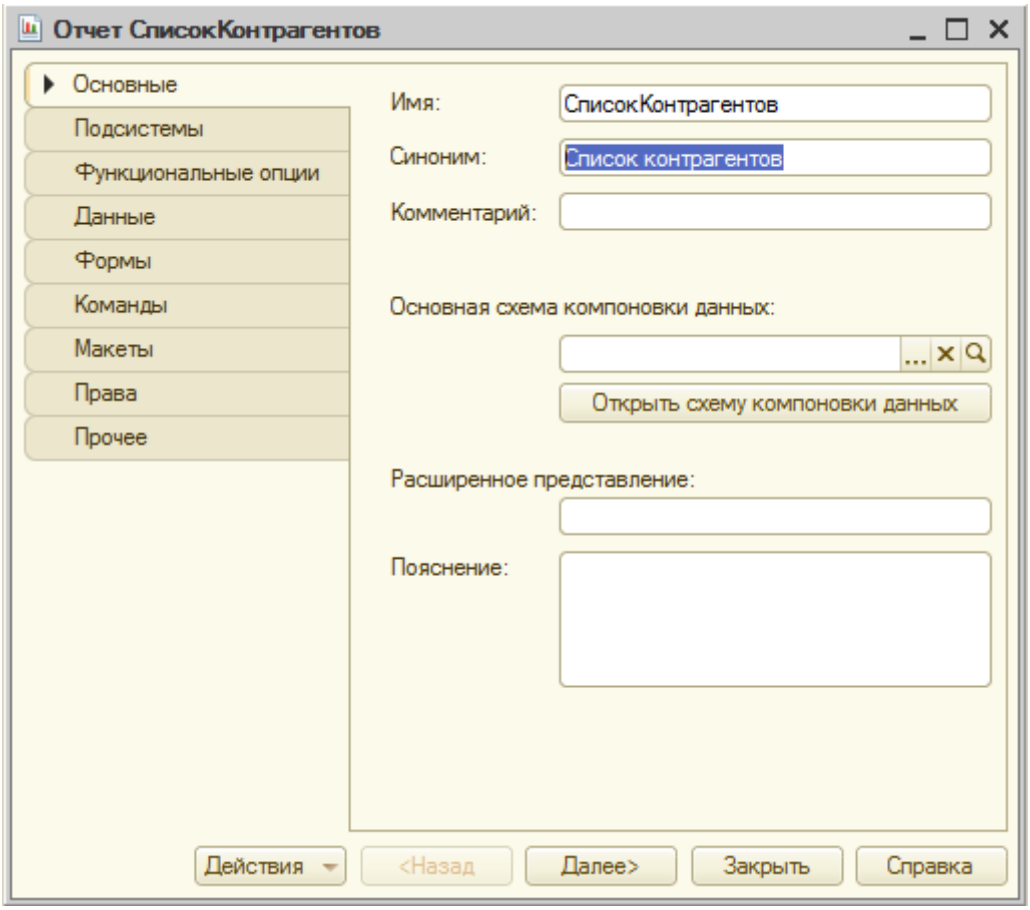


Рисунок 9.1- Создание нового отчета

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

3. 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 9.2.

Конструктор печати предназначен для создания макета печатной формы объекта прикладного решения и процедуры на встроенном языке, которая будет формировать печатную форму на основании этого макета. Конструктор печати может быть вызван, например, из окна редактирования справочника, или отчета.

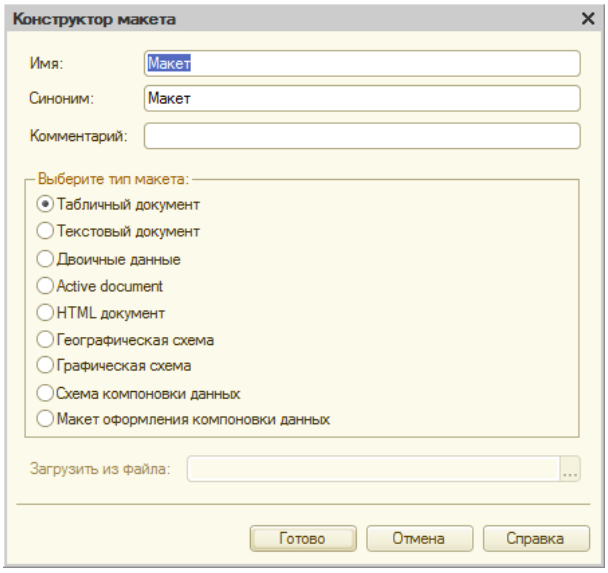


Рисунок 9.2- Создание макета для отчета

4. После нажатия на кнопку **Готово**, мы видим табличный редактор, Рисунок 9.3., 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.

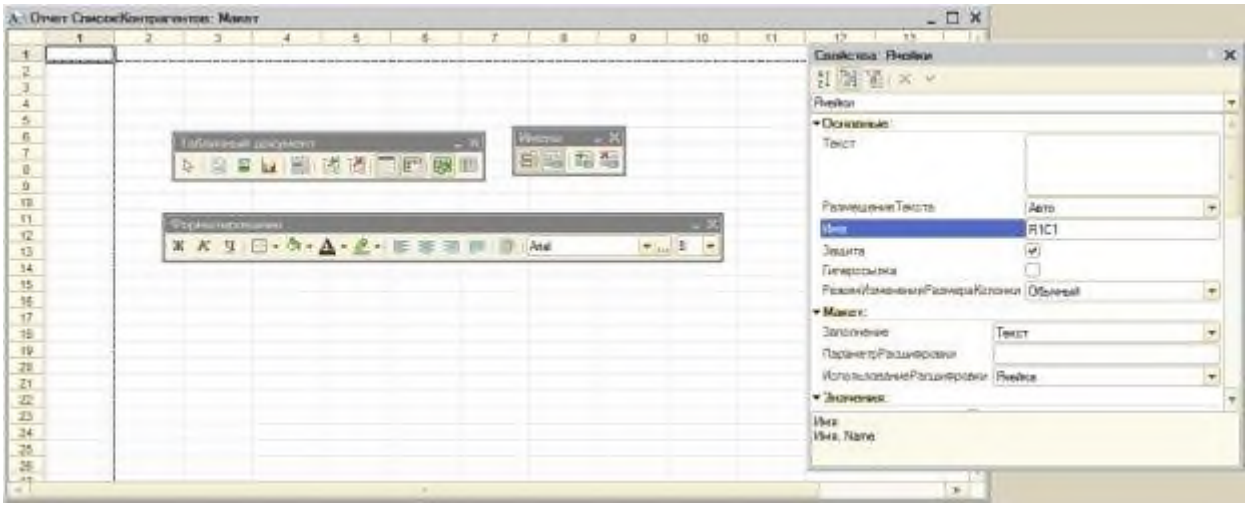


Рисунок 9.3 - Средства редактирования макета отчета

5. На Рисунок 9.4 показан готовый макет.

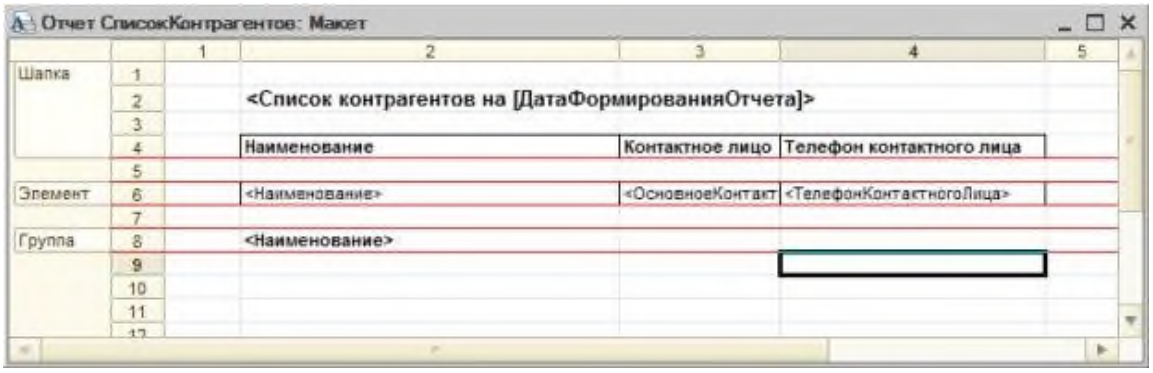


Рисунок 9.4- Готовый макет отчета

6. 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 9.5.

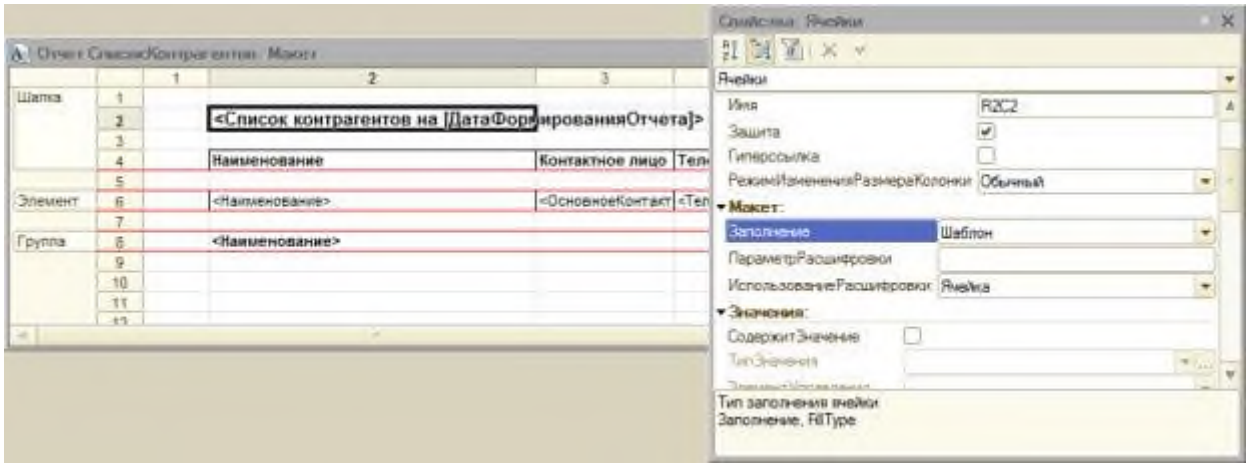


Рисунок 9.5 - Настройка ячейки, содержащей шаблон

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

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

9. 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**.

10. Область **Элемент** содержит три параметра – **Наименование**, **ОсновноеКонтактноеЛицо** и **ТелефонКонтактногоЛица**. После ввода в каждую из ячейку 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.

11. 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.

12. 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 9.6.

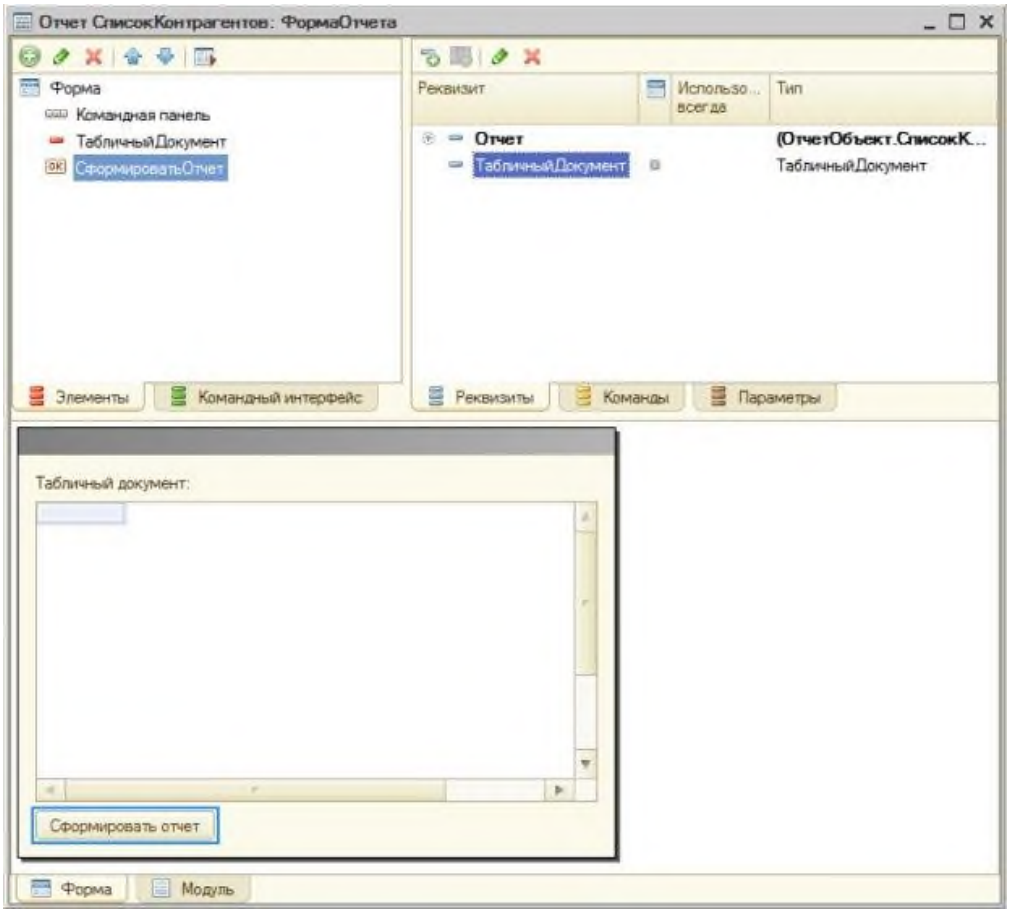


Рисунок 9.6- Настройка формы отчета

13. 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.

14. 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**.

15. Реализуем метод в виде функции. Готовый код формирования отчета (Рисунок 9.7) будет выглядеть следующим образом:

Рисунок 9.7- 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
```

16. 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 9.8, где в поле **Выводить в** можно выбрать либо **Табличный документ** (его обычно и используют), либо – **Текстовый документ.**

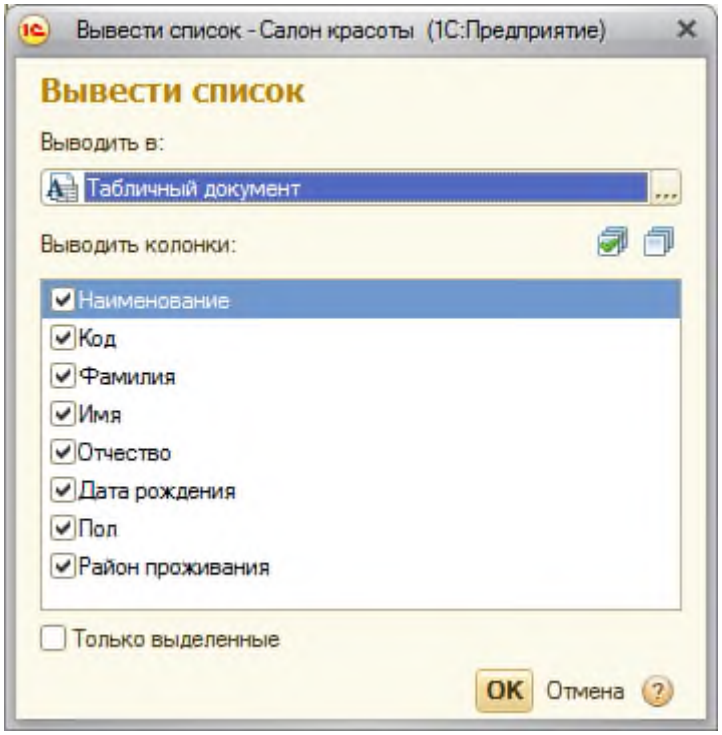


Рисунок 9.8 - Окно Настройка списка

17. 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 9.9., этот документ можно сохранить в нужном формате для дальнейшей обработки в других приложениях.

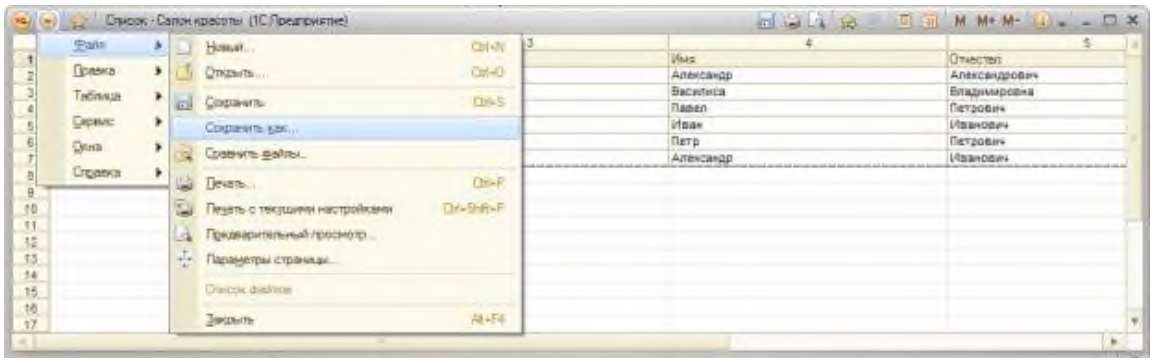


Рисунок 9.9 - Вывод данных в табличный документ

Практическая работа №10 «Инспекция кода модулей проекта»

Цель работы: научиться выполнять инспекцию кода модулей проекта

ХОД РАБОТЫ:

1. Создадим отчет по данным справочника «Сотрудники».

Табличный документ поддерживает возможность группировки строк и столбцов. Это позволяет группировать данные в отчетах, используя произвольное количество вложенных группировок.

Бывают горизонтальные и вертикальные группировки, причем у разработчика есть возможность управлять расположением итогов в группировке: для вертикальных группировок они могут быть расположены сверху или снизу, а для горизонтальных группировок - справа или слева.

Поддерживается отображение уровней группировок, и нажатием цифр в заголовках можно развернуть сразу все группировки данного уровня и свернуть более детальные группировки.

Отступ уровней иерархии при использовании группировок формируется системой автоматически.

1.2 Добавим реквизиты и табличную часть в справочник Сотрудники:

Имя реквизита	**Тип**	**Длина**
ФИО	Справочники.Ссылка.ФизическиеЛица	--
Образование	Строка	20
Оклад	Число	10, точность 2
Адрес	Строка	100
Расчетный счет	Строка	20
Табличная часть «Дети»		
Имя	Строка	20
Тип	Строка	10
ДатаРождения	Дата	--

Укажем, что справочник Сотрудники иерархический.

Получим новый вид справочника Сотрудники (рисунок 9.1.1).

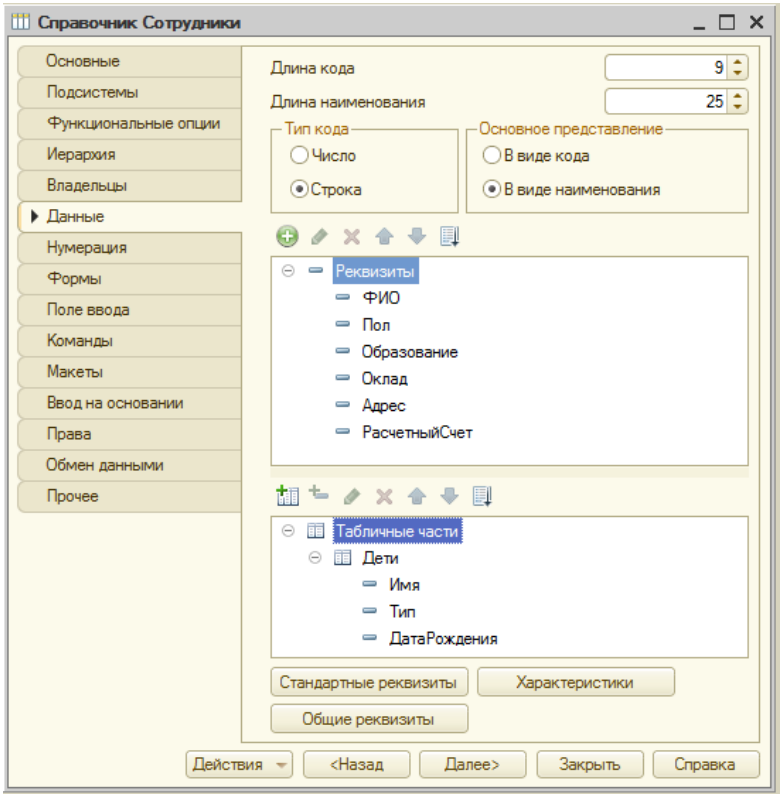


Рисунок 9.1.1 – Справочник Сотрудники

1.3 Создадим новый отчет «**СписокСотрудников**». В отчет добавим реквизит ТабличныйДокумент с одноименным типом данных и команду кнопки **ПечатьСправочника**.

Макет отчета с именем «**МакетПечатиСотрудников**» разместить в справочнике Сотрудники, вид макета представлен на рисунке 9.1.2.

		1	2	3	4
	1				
Шапка	2	***Справочник сотрудников***			
	3				
	4	**ФИО**	**Пол**	**Образование**	**Оклад**
	5				
Строка	6	**<ФИО>**	**<Пол>**	**<Образование>**	**<Оклад>**
	7				
	8				
Подвал	9	Главный бухгалтер	<ГлавныйБухгалтер>		
	10				
	11				

Рисунок 9.1.2 - Макет отчета

1.4 Введем в модуль формы отчета программный код:

```
&НаКлиенте
Процедура ПечатьСправочника(Команда)

ТабличныйДокумент=печать();

КонецПроцедуры
```

```
    Функция Печать()

    Таб = Новый ТабличныйДокумент();
Макет=Справочники.Сотрудники.ПолучитьМакет("МакетПечатиСотрудников");
    Область = Макет.ПолучитьОбласть("Шапка");
    Таб.Вывести(Область);
    Таб.НачатьАвтогруппировкуСтрок();
    Выб=Справочники.Сотрудники.ВыбратьИерархически();
    Пока Выб.Следующий() Цикл
        Если Выб.ЭтоГруппа Тогда
        Область = Макет.ПолучитьОбласть("Группа");
        Область.Параметры.НаименованиеГруппы=Выб.Наименование;
        Иначе
         Область = Макет.ПолучитьОбласть("Строка");
          Область.Параметры.ФИО=Выб.Наименование;
          Область.Параметры.Пол=Выб.Пол;
          Область.Параметры.Образование=Выб.Образование;
          Область.Параметры.Оклад=Выб.Оклад;
          Область.Параметры.сотрудник=Выб.Ссылка;
        КонецЕсли;
        Таб.Вывести(Область,Выб.УровеньВВыборке(),,Истина);
    КонецЦикла;
    Таб.ЗакончитьАвтогруппировкуСтрок();

    Возврат Таб;

    КонецФункции
```

1.5 Просмотрим результат формирования отчета в режиме отладки (пользователя). Результат формирования отчета в режиме пользователя представлен на рисунке 9.1.3.

Справочник сотрудников

ФИО	Пол	Образование	Оклад
Работающие			
Иванов В.С.	Мужской	Высшее	300 000
Сидоров И.И.	Мужской	Среднее	20 000
Уволенные			
Кондратьева А.А.	Женский	Среднее	30 000

Рисунок 9.1.3 - Печатная форма справочника сотрудников

1.6 Выполним стандартную расшифровку отчета.

В системе поддерживается механизм расшифровок, который позволяет пользователю получить детальный или дополнительный отчет, щелкнув мышью на строке или ячейке табличного документа. Платформа поддерживает возможность обработки нажатий клавиши мыши в ячейках табличного документа. При этом система может выполнять как стандартные действия, так и алгоритмы, заданные разработчиком.

Стандартные действия при расшифровке выполняются, например, если щелкнуть мышью на документе или элементе справочника. В этом случае система откроет этот объект для просмотра (если иное поведение не предусмотрено разработчиком).

Для обеспечения стандартной расшифровки в макет необходимо внести добавления, указав параметр расшифровки

1.6.1 Укажем параметр расшифровки для ячейки макета отчета, пример параметра расшифровки представлен на рисунке 9.1.4.

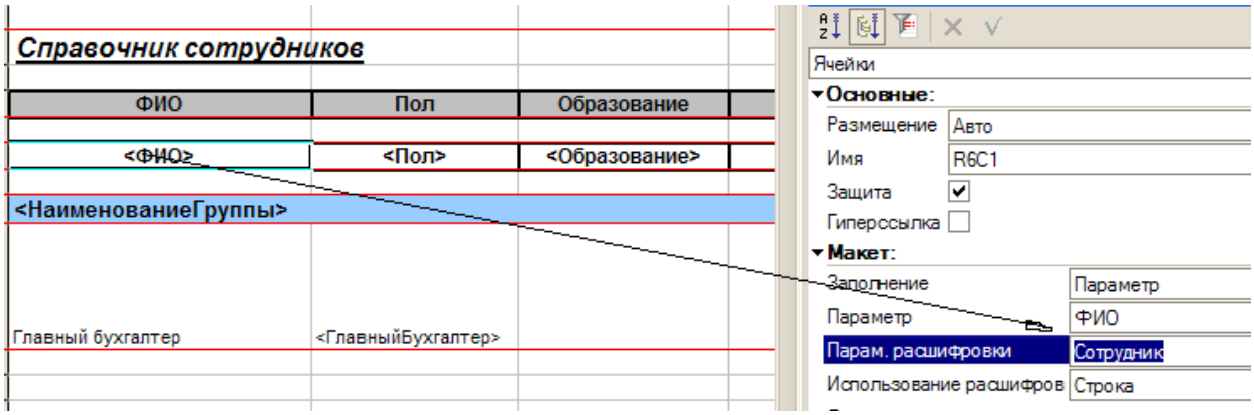


Рисунок 9.1.4 - Макет отчета с подготовкой стандартной расшифровки

1.6.2 В программном коде эту переменную необходимо означить перед выводом области. В предыдущем коде такая строка уже есть:

```
.....
Область.Параметры.Сотрудник=Выб.Ссылка;
.....
```

Результат вывода отчета на рисунке 9.1.5.

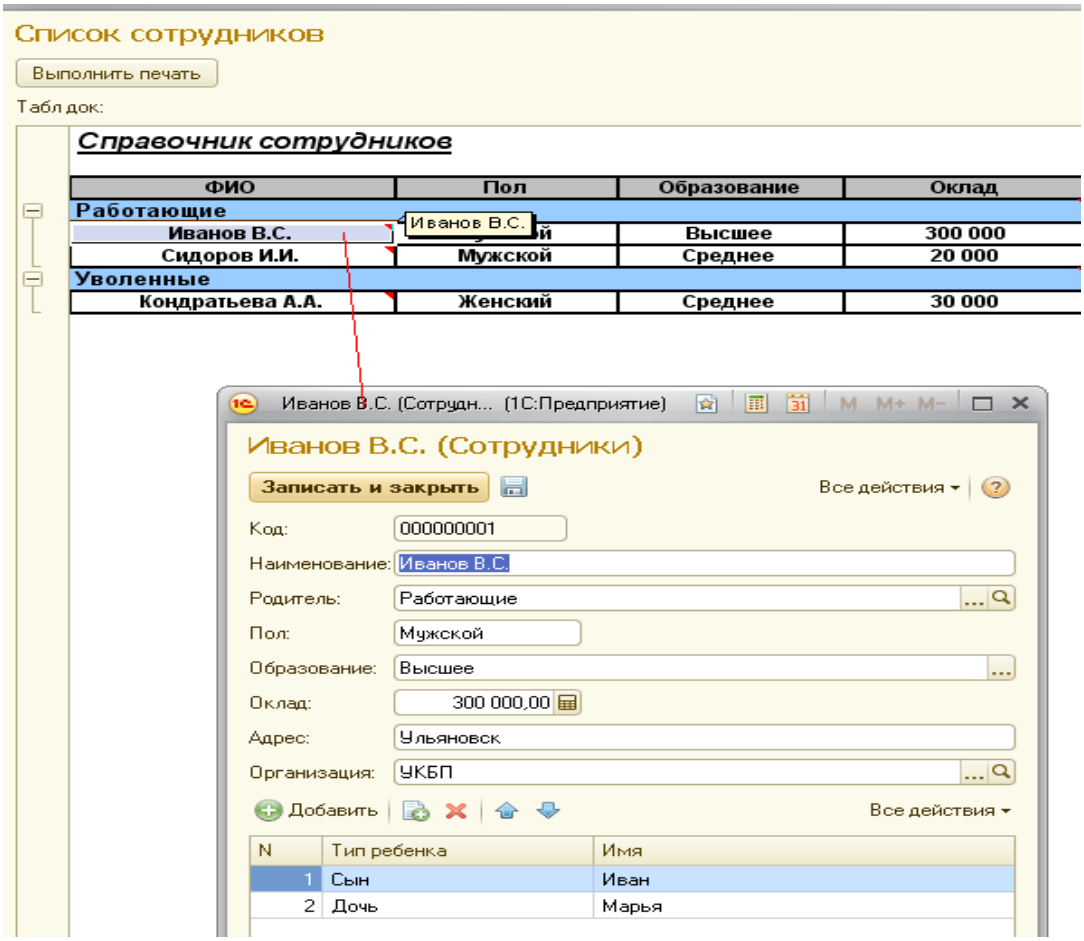


Рисунок 9.1.5 - Печатная форма справочника сотрудников со стандартной расшифровкой

1.7 Выполним нестандартную расшифровку отчета.

Обработка нестандартных расшифровок производится средствами встроенного языка. Например, разработчик может задать алгоритм получения детального отчета, путем переформирования существующего с использованием дополнительных условий отбора ("показать продажи только по этому контрагенту"). Или же, используя расшифровку, пользователь может получить совершенно новый отчет (например "показать расходные накладные, которые сделали вклад в объем продаж по данному контрагенту").

1.7.1 В качестве нестандартной расшифровки предыдущего отчета будем выводить список детей выбранного в основном отчете сотрудника

1.7.2 Для начала выполним в макете основного отчета в свойстве ячейки «ФИО» установлен параметр расшифровки (см. рисунок 9.1.4.)

1.7.3 Дополнительно, необходимо в свойствах элемента ТаблДок на событие «Обработка расшифровки» назначить процедуру – обработку с произвольным именем:

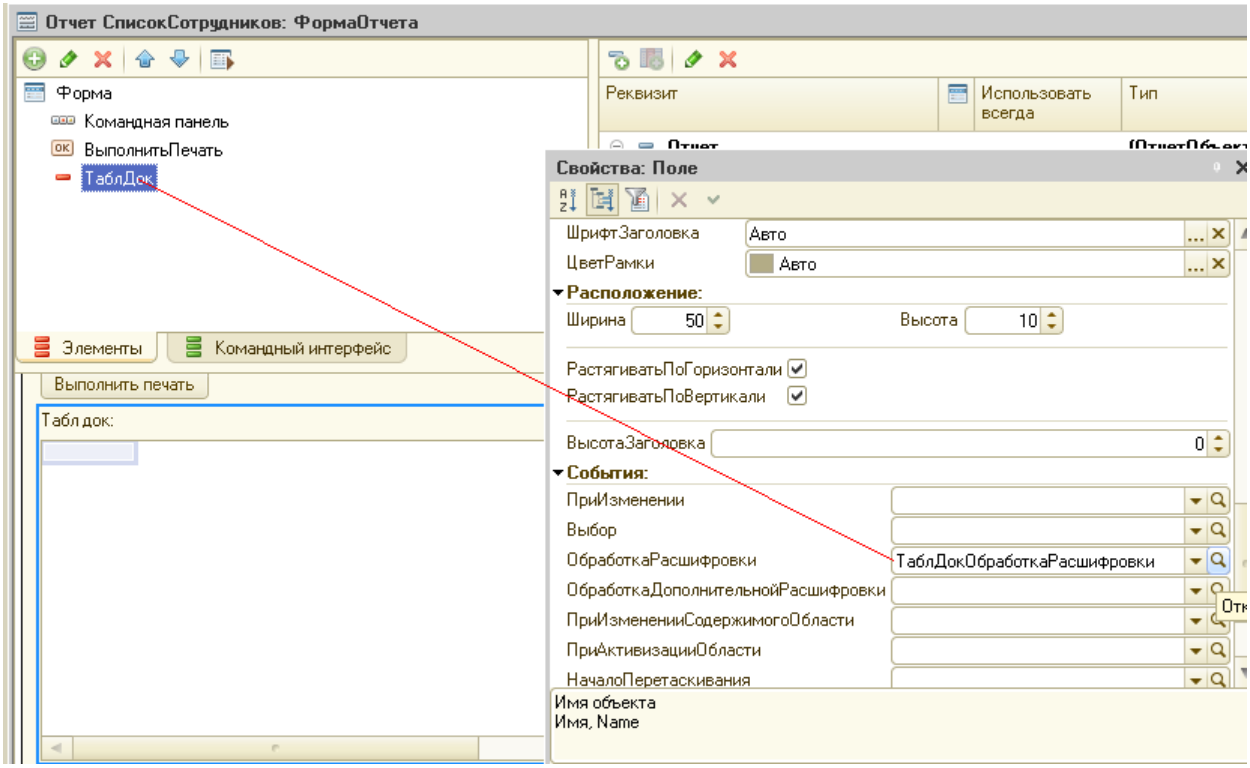


Рисунок 9.1.6 - Настройка табличного документа на нестандартную расшифровку

1.7.4 Сформировать макет для конкретизирующего отчета - расшифровки:

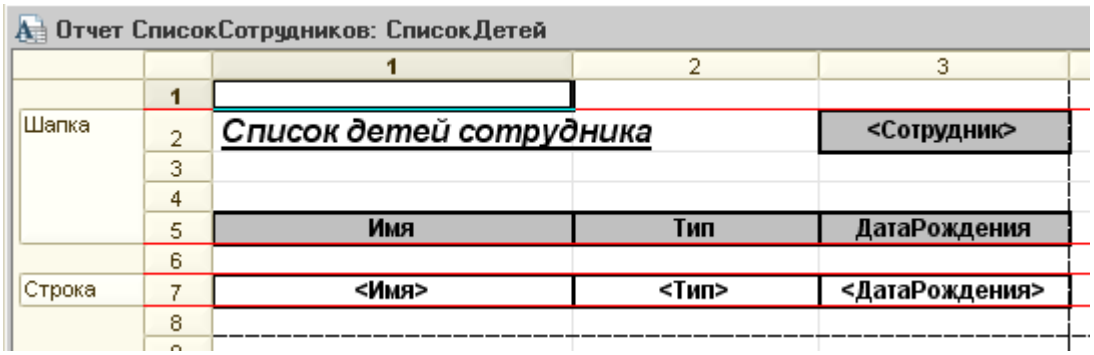


Рисунок 9.1.7 - Макет конкретизирующего отчета

1.7.5 В модуль формы отчета добавить программный код:

```
&НаКлиенте
Процедура ТаблДокОбработкаРасшифровки(Элемент, Расшифровка,
СтандартнаяОбработка)
СтандартнаяОбработка=ложь;
Сотрудник=Расшифровка;
ТабДетей=Новый ТабличныйДокумент();
ТабДетей=ВыполнитьРасшифровку(Сотрудник);
ТабДетей.ТолькоПросмотр=Истина;
ТабДетей.Показать("");
```

```
КонецПроцедуры

Функция ВыполнитьРасшифровку(Сотрудник)
ТабДетей=Новый ТабличныйДокумент();
МакетДетей=
  Отчеты.СписокСотрудников.ПолучитьМакет("СписокДетей");
Область = МакетДетей.ПолучитьОбласть("Шапка");
  Область.Параметры.Сотрудник=Сотрудник;
  ТабДетей.Вывести(Область);

  Для каждого Стр из Сотрудник.Дети Цикл
     Область = МакетДетей.ПолучитьОбласть("Строка");
     Область.Параметры.Имя=Стр.имя;
     Область.Параметры.Тип=Стр.ТипРебенка;
     Область.Параметры.ДатаРождения=Стр.ДатаРождения;
     ТабДетей.Вывести(Область);
  КонецЦикла;
  Возврат ТабДетей;
КонецФункции
```

1.7.6 Просмотрим результат формирования отчета в режиме отладки (пользовательском):

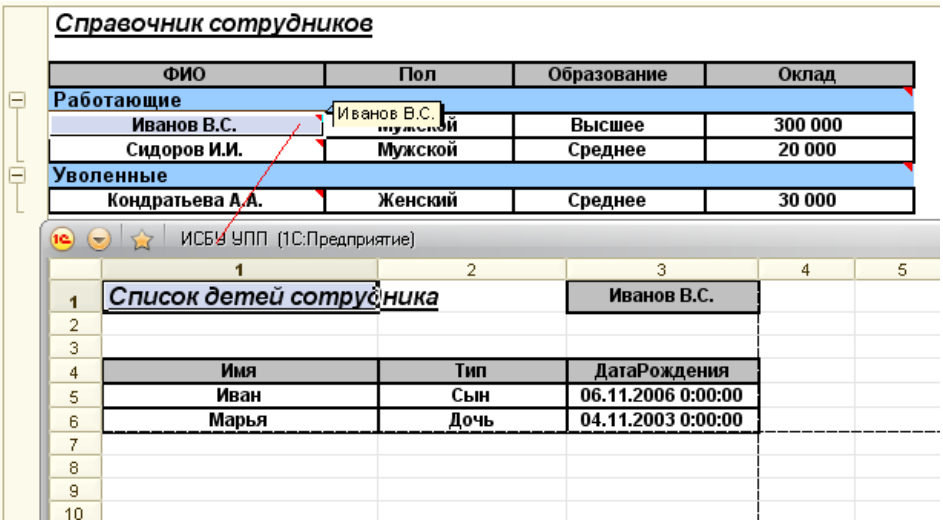


Рисунок 9.1.8 - Печатная форма справочника сотрудников с нестандартной расшифровкой

1.8 Создадим примечание в отчете.

Разработчик имеет возможность задавать примечания для отдельных ячеек или групп ячеек документа. Ячейка с примечанием имеет маленький треугольник в правом верхнем углу. При наведении курсора на ячейку, примечание отображается во всплывающем окне. С помощью примечаний можно добавлять в табличные документы дополнительную (справочную) информацию, которая не отображается на экране (в обычном режиме), но может быть легко просмотрена, если подвести курсор мыши к нужной ячейке.

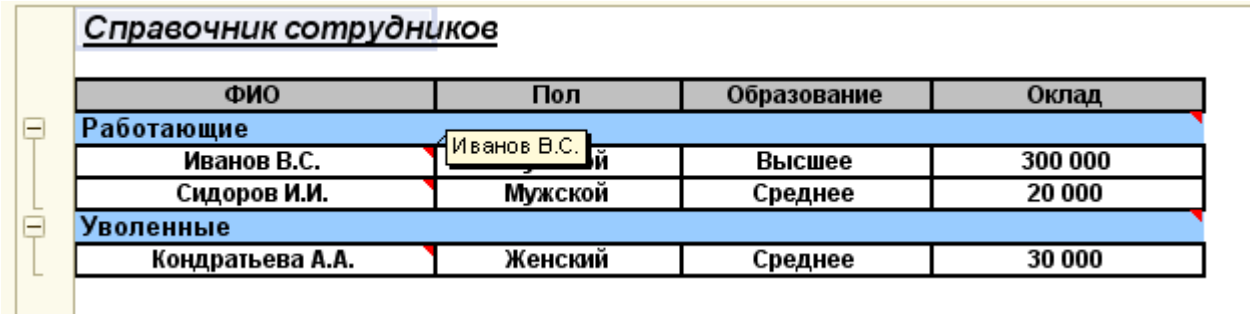
1.8.1 Для добавления примечания в предыдущем примере необходимо внести добавления в программный код отчета:

```
&НаКлиенте
Процедура ПечатьСправочника(Команда)
     ТаблДокум=печать();
КонецПроцедуры


Функция Печать()
Таб = Новый ТабличныйДокумент();
Макет=
Справочники.Сотрудники.ПолучитьМакет("МакетПечатиСотрудников");
Область = Макет.ПолучитьОбласть("Шапка");
Таб.Вывести(Область);
ОбластьПримечанияГр=Макет.Область("R8C1:R8C4");
ОбластьПримечанияЭл=Макет.Область("R6C1");
Таб.НачатьАвтогруппировкуСтрок();
Выб=Справочники.Сотрудники.Выбратьиерархически();

Пока выб.Следующий() Цикл
          Если Выб.ЭтоГруппа Тогда
ОбластьПримечанияГр.Примечание.Текст = Выб.Наименование;
          Область = Макет.ПолучитьОбласть("Группа");
          Область.Параметры.НаименованиеГруппы=Выб.Наименование;
             Иначе
               ОбластьПримечанияЭл.Примечание.Текст=
               Выб.Наименование;
        Область = Макет.ПолучитьОбласть("Строка");
        Область.Параметры.ФИО=Выб.Наименование;
        Область.Параметры.Пол=Выб.Пол;
        Область.Параметры.Образование=Выб.Образование;
        Область.Параметры.Оклад=Выб.Оклад;
        Область.Параметры.сотрудник=Выб.Ссылка;
        КонецЕсли;
        Таб.Вывести(Область,Выб.УровеньВВыборке(),,Истина);
КонецЦикла;
Таб.ЗакончитьАвтогруппировкуСтрок();
Возврат Таб;
КонецФункции
```

1.8.2 Просмотрим результат формирования отчета в режиме отладки (пользовательском):



Справочник сотрудников

ФИО	Пол	Образование	Оклад
Работающие			
Иванов В.С.	Иванов В.С.	Высшее	300 000
Сидоров И.И.	Мужской	Среднее	20 000
Уволенные			
Кондратьева А.А.	Женский	Среднее	30 000

Рисунок 9.1.9 - Печатная форма справочника сотрудников с примечаниями

1.9 Сохраните копию отчета

Поскольку табличный документ, чаще всего, используется для формирования выходных документов, он может быть сохранен в файл на диске для последующего использования или переноса на другие компьютеры. Табличный документ может быть сохранен как в собственном формате, так и экспортирован в другие форматы хранения данных, в том числе в формат документов Microsoft Office 2013 (*.xlsx, *.docx) или в формат электронных документов Adobe (*.pdf):

Практическая работа №11 «Тестирование интерфейса пользователя средствами инструментальной среды разработки»

Цель работы: научиться выполнять тестирование интерфейса пользователя средствами инструментальной среды разработки

ХОД РАБОТЫ:

1. 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 10.1.

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.

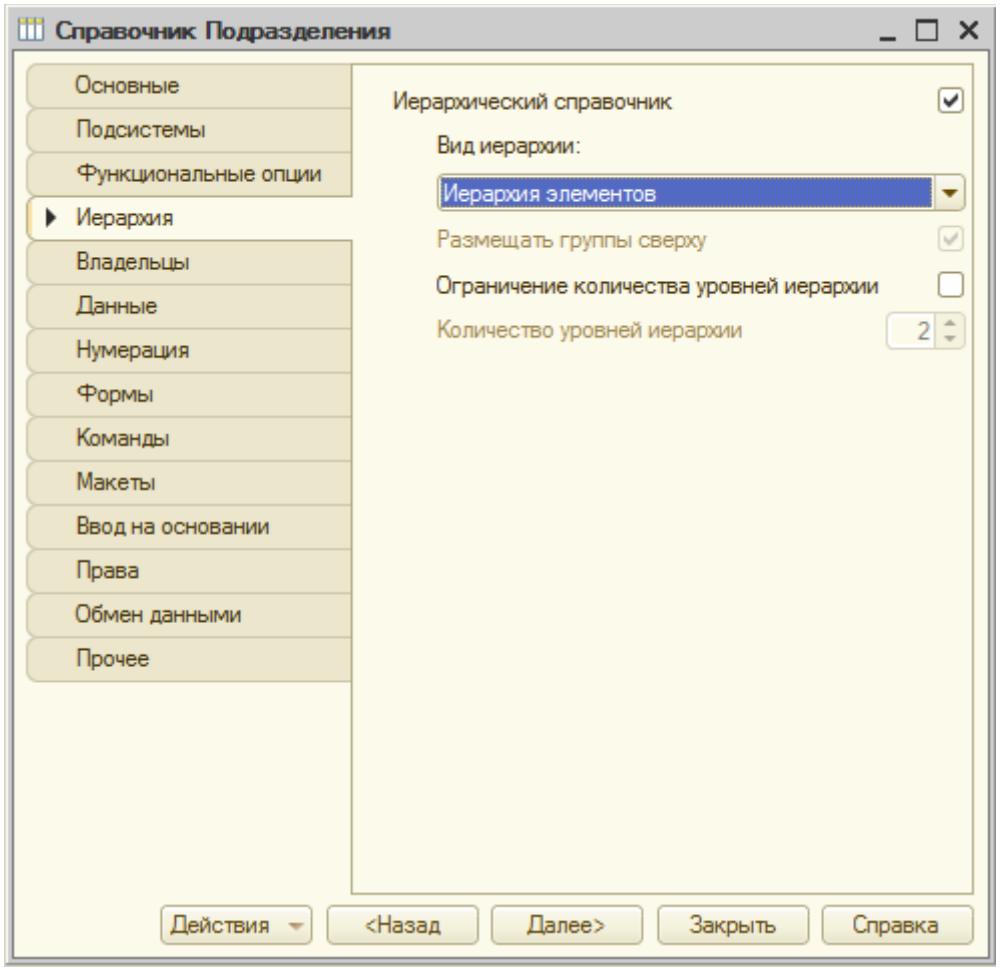


Рисунок 10.1- Настройка иерархии справочника Подразделения

2. 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 10.2.):

- Администрация
- Бухгалтерия
- Парикмахерская

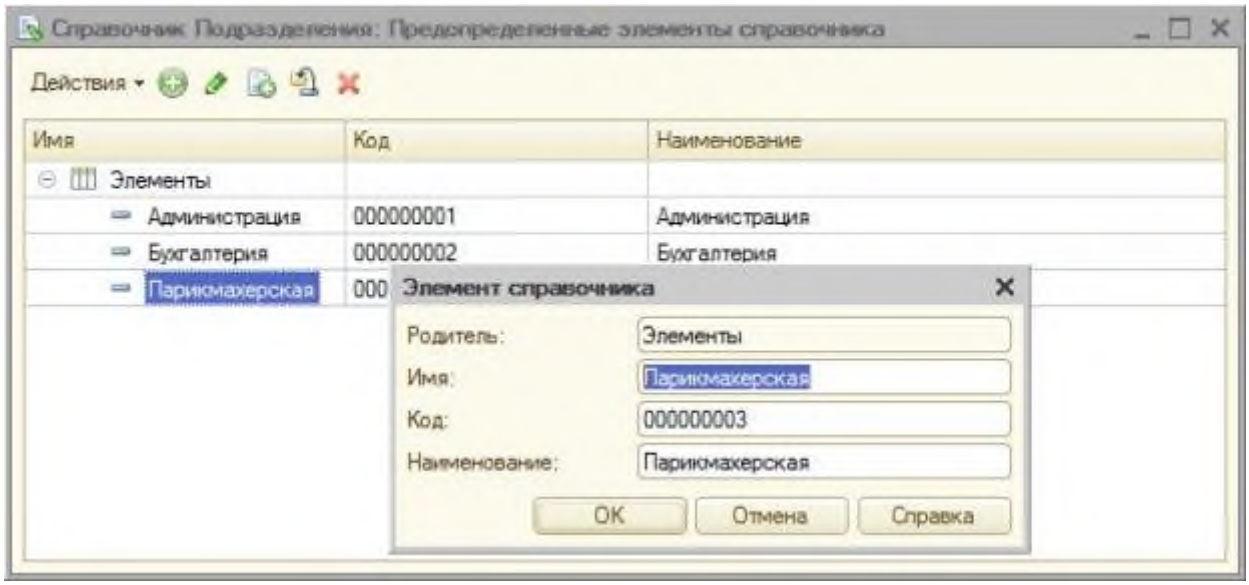


Рисунок 10.2- Создание предопределенных элементов справочника Подразделения

3. Доработаем еще раз справочник **Сотрудники**. Снабдим его следующими реквизитами, Рисунок 10.3:

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

Имя: Расчетчик, **Тип**: Булево

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

Увеличим длину **наименования** до 50 символов.

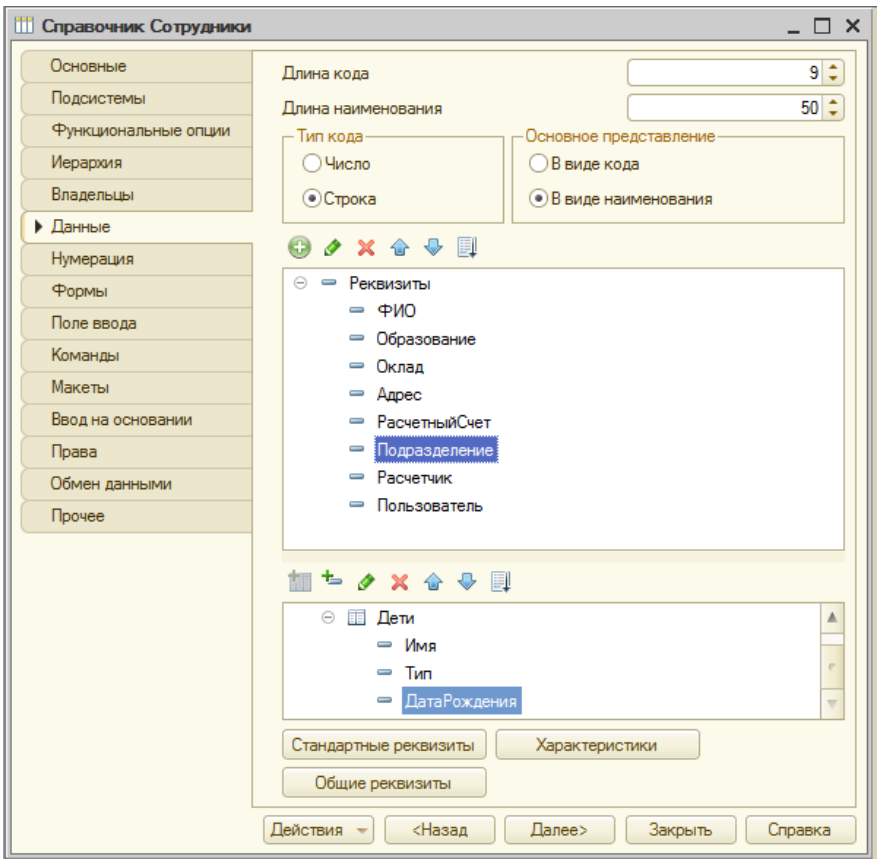


Рисунок 10.3. Реквизиты справочника Сотрудники

4. 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 10.4.

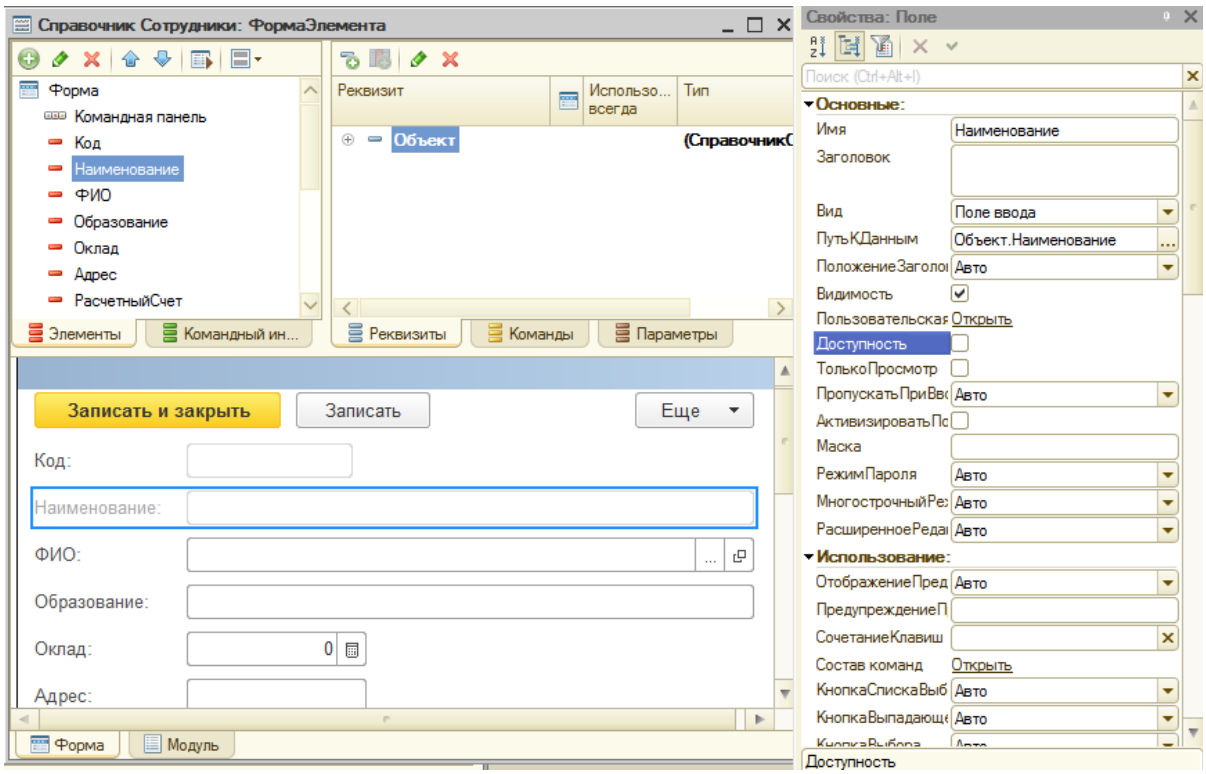


Рисунок 10.4 - Настройка формы элемента справочника Сотрудники

5. Теперь подумаем над тем, как автоматически заполнить поле **Наименование** на основе данных полей **ФИО** и **Подразделение**. Сделать это можно различными способами, мы реализуем следующую функциональность: перехватим события изменения полей **ФИО** и **Подразделение** и вызовем в обработчике каждого из этих событий процедуру, заполняющую поле **Наименование** (рисунок 10.5). Так пользователь, заполняющий элемент справочника, сможет сразу же увидеть результаты формирования наименования.

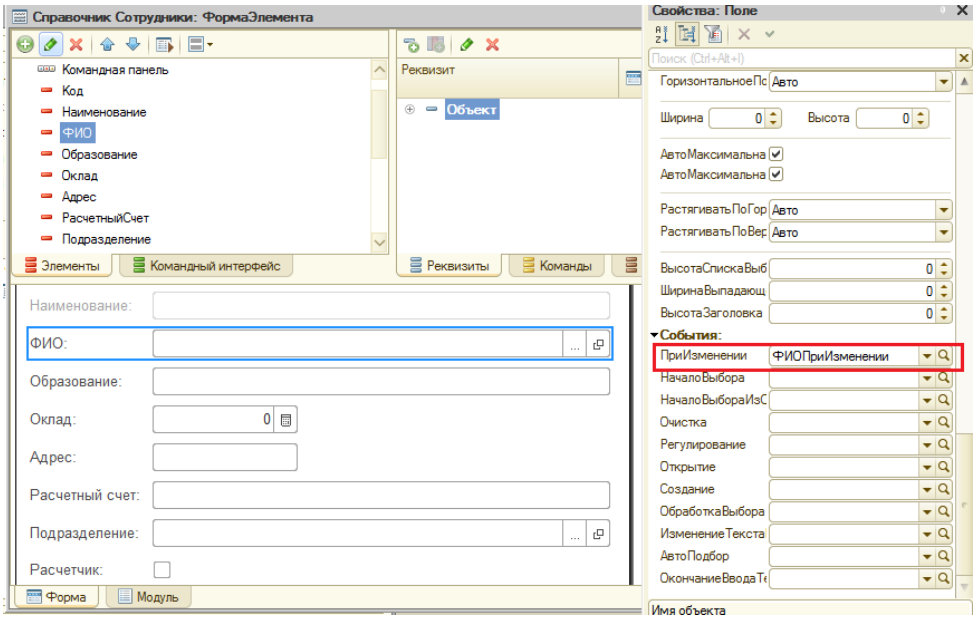


Рисунок 10.5 - Выбор события формы для поля ФИО

Нашей задаче отвечает следующий код:

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

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

Процедура СформироватьНаименование()
  Объект.Наименование=Объект.ФИО.Наименование +" ("+
Объект.Подразделение.Наименование+") ";

КонецПроцедуры
```

Результаты работы созданного нами механизма показаны на Рисунок 10.6.

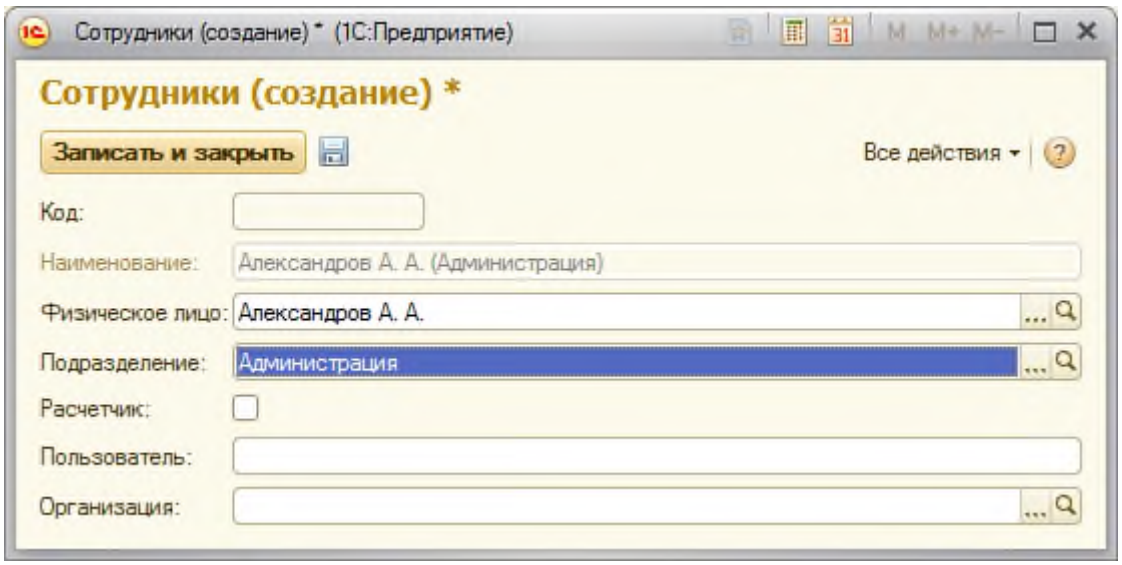


Рисунок 10.6- Настройка формы элемента справочника Сотрудники

В этой работе мы познакомились с созданием обработок и простых отчетов. Так же мы подробно рассмотрели объектную модель 1С:Предприятие 8.3., предназначенную для работы со справочниками и создали справочник с иерархией элементов.

Практическая работа №12 «Разработка тестовых модулей проекта для тестирования отдельных модулей»

Цель работы: научиться разработке тестовых модулей проекта для тестирования отдельных модулей

ХОД РАБОТЫ:

1. 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 11.1.), 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

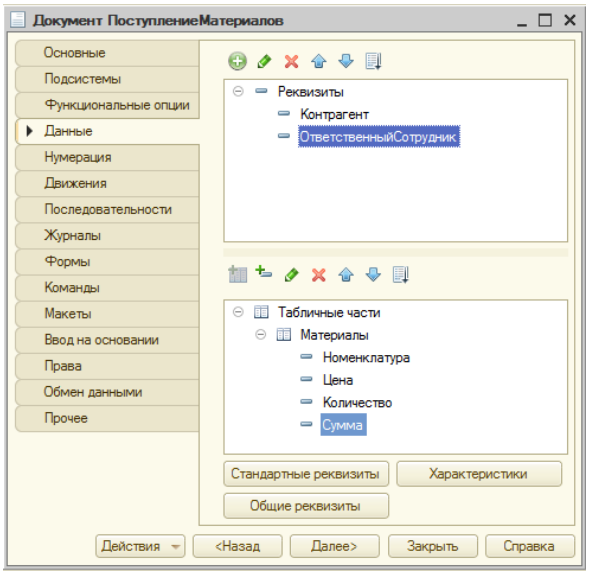


Рисунок 11.1- Настройка состава реквизитов документа

2. На закладке **Нумерация**, Рисунок 11.2., можно задать параметры нумерации документов.

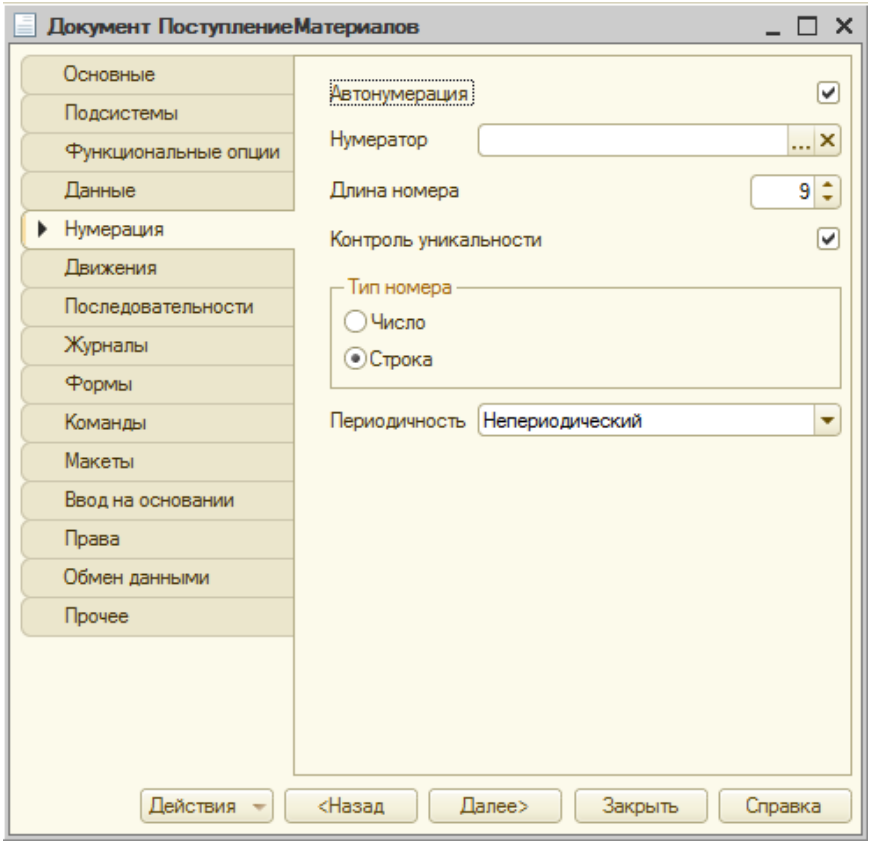


Рисунок 11.2- Настройка параметров нумерации документа

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

3. Закладка **Движения**, Рисунок 11.3., позволяет управлять проведением документа.

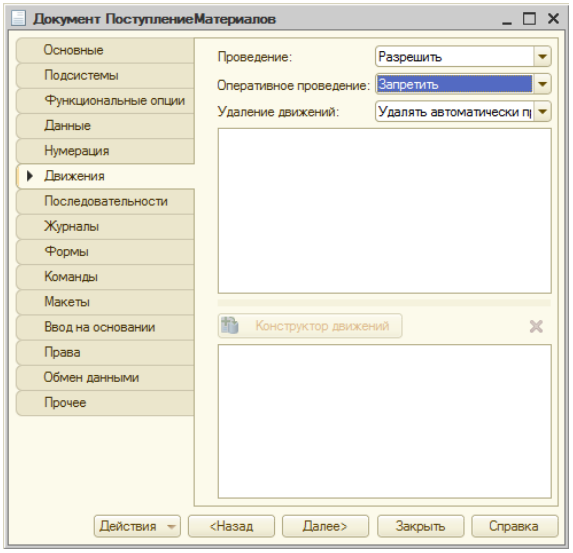


Рисунок 11.3- Настройка параметров проведения документа

4. 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 11.4., то окажется, что эта вкладка позволяет настроить ввод одного документа на основании другого.

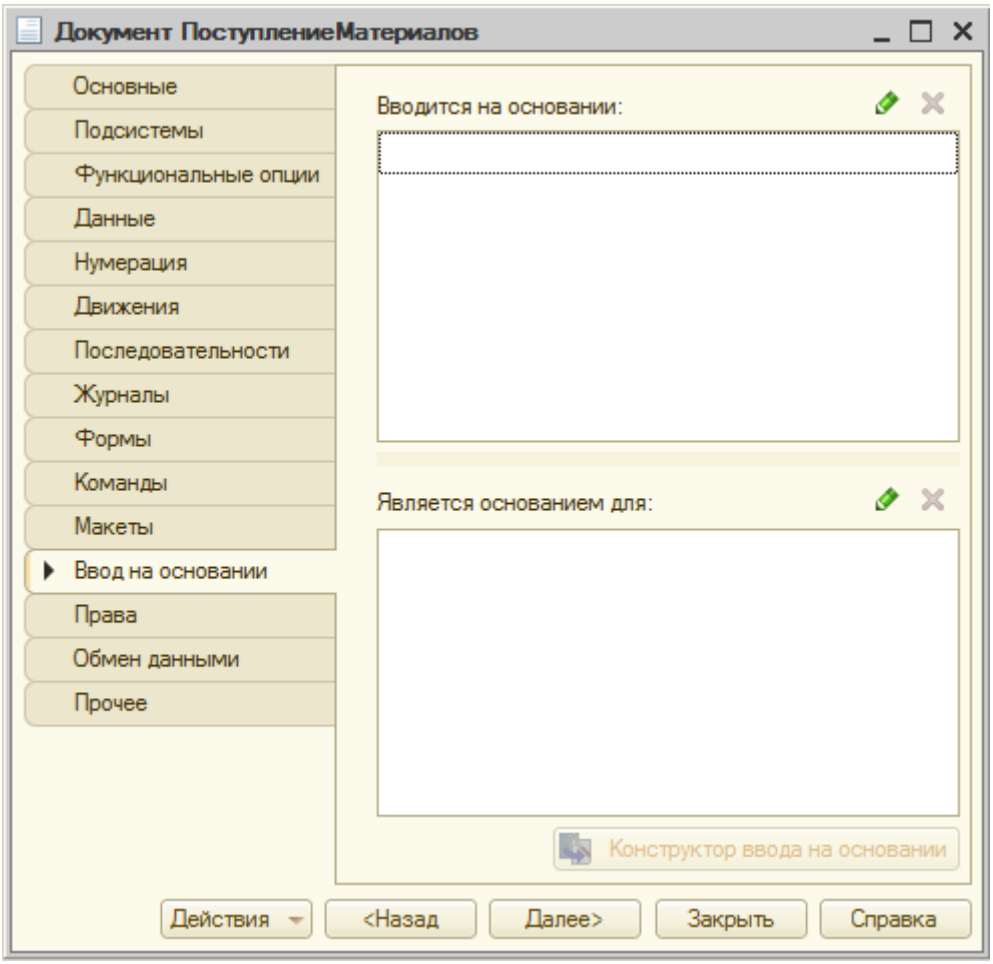


Рисунок 11.4 - Настройка параметров ввода на основании

5. 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.

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

7. Прежде чем продолжать работу с документом **ПоступлениеМатериалов**, приведите данные справочника **Номенклатура** в вашей информационной базе к виду, показанному в таблице 11.1.

Таблица 11.1. 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	

8. На рисунке 11.5. вы можете видеть форму документа после ввода в нее некоторых данных.

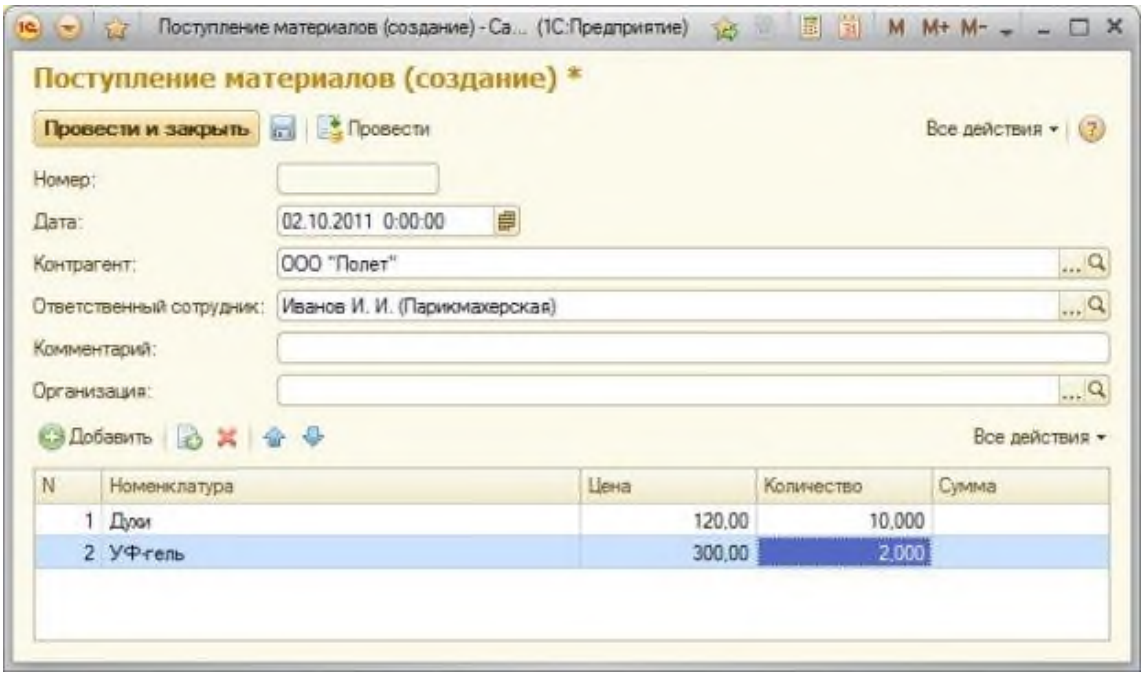


Рисунок 11.5 - Заполнение документа Поступление товаров

9. 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 11.6.). 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.

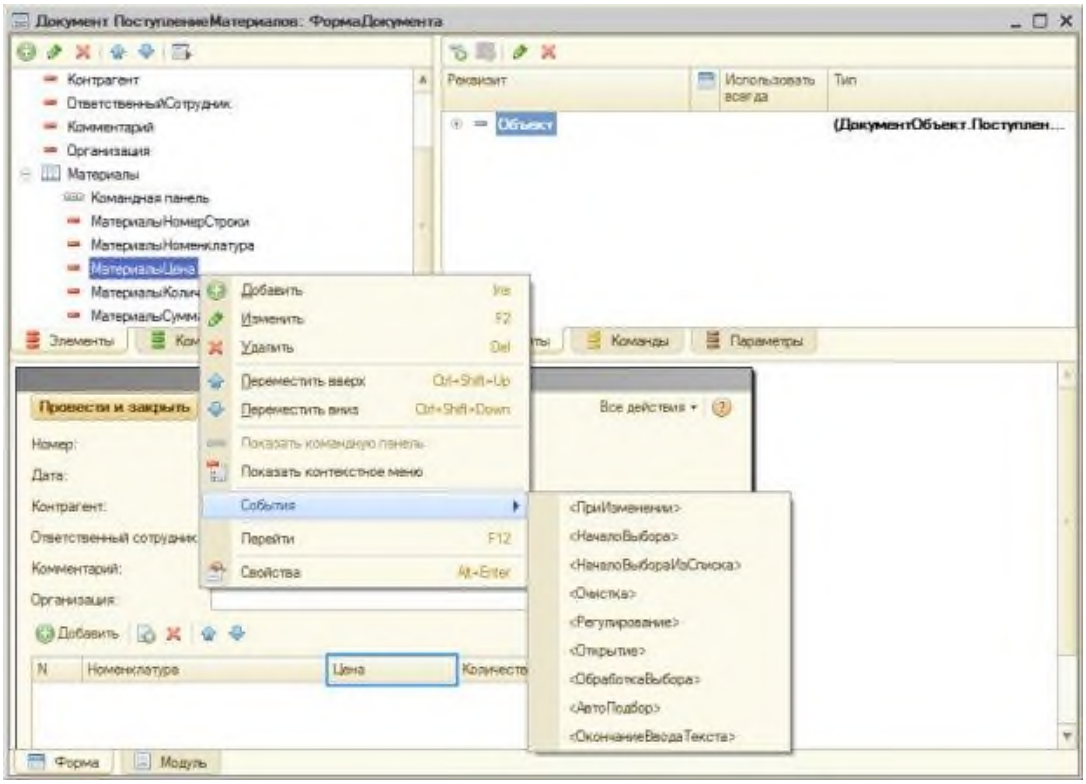


Рисунок 11.6- Назначение обработчика полю табличной части

10. 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 11.7.

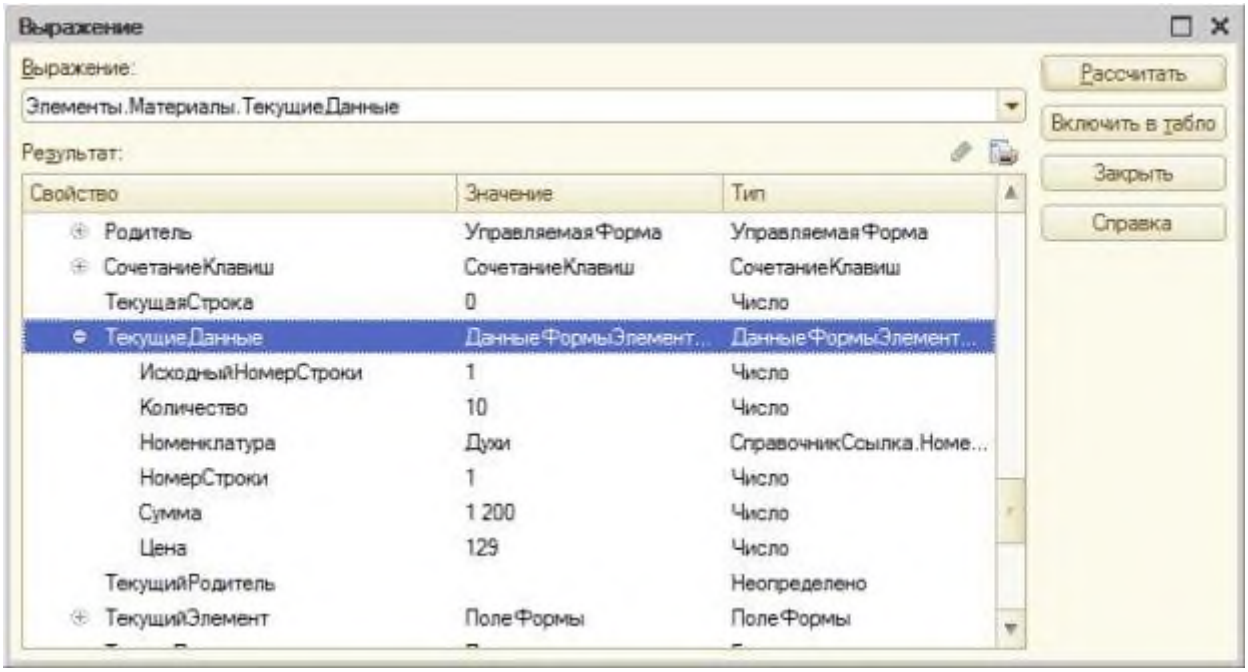


Рисунок 11.7- Просмотр свойства «ТекущиеДанные» в окне «Выражение» при отладке кода

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**.

11. 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 11.8.

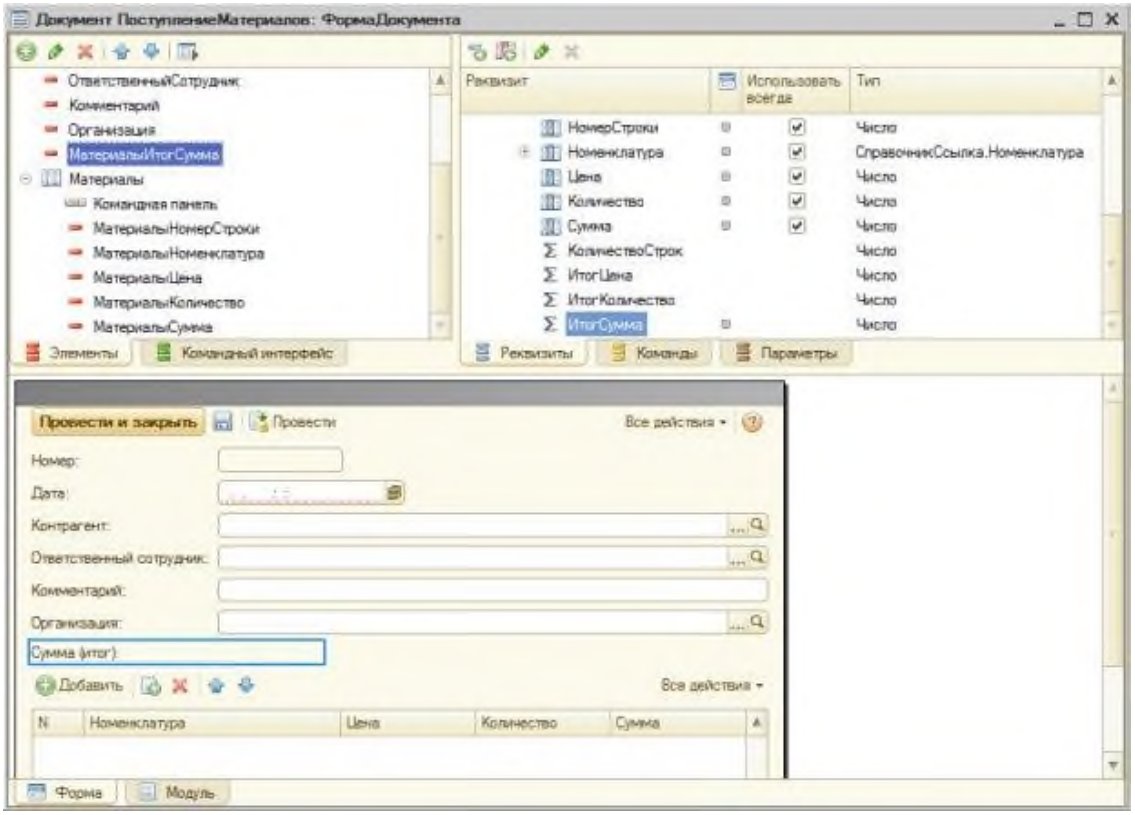


Рисунок 11.8 - Вывод итогового показателя для поля Сумма на форму

Этот реквизит – **ИтогСумма** – нужно перетащить на вкладку **Элементы**. Он будет отображаться на форме, изменяясь при изменениях суммы в строках табличной части, Рисунок 11.9.

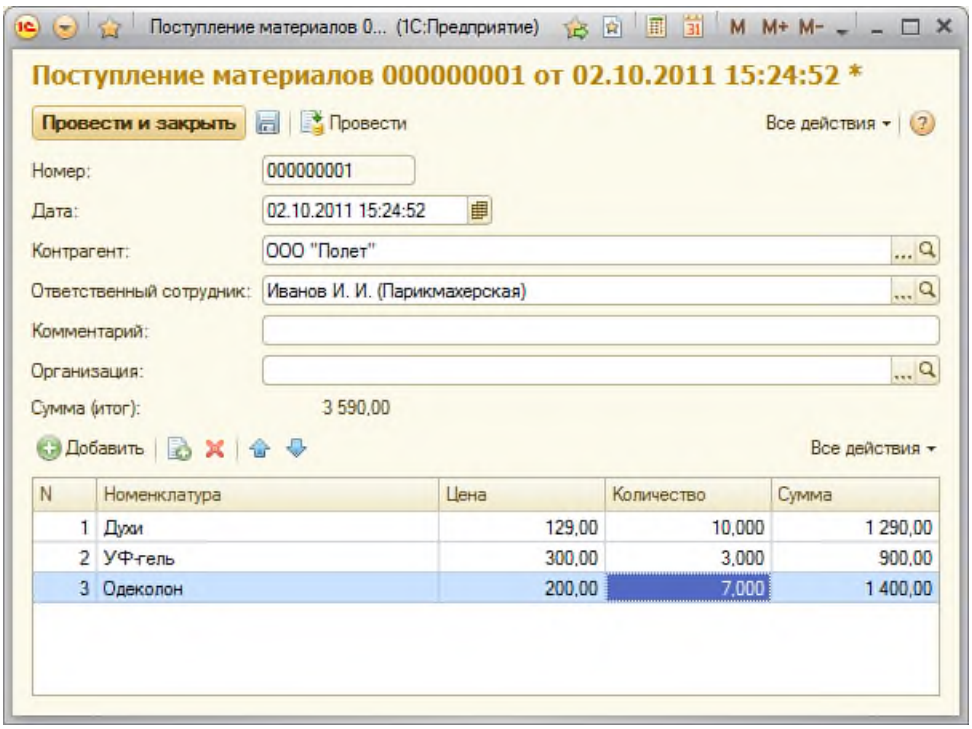


Рисунок 11.9- Форма после модификации

12. 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 11.10.

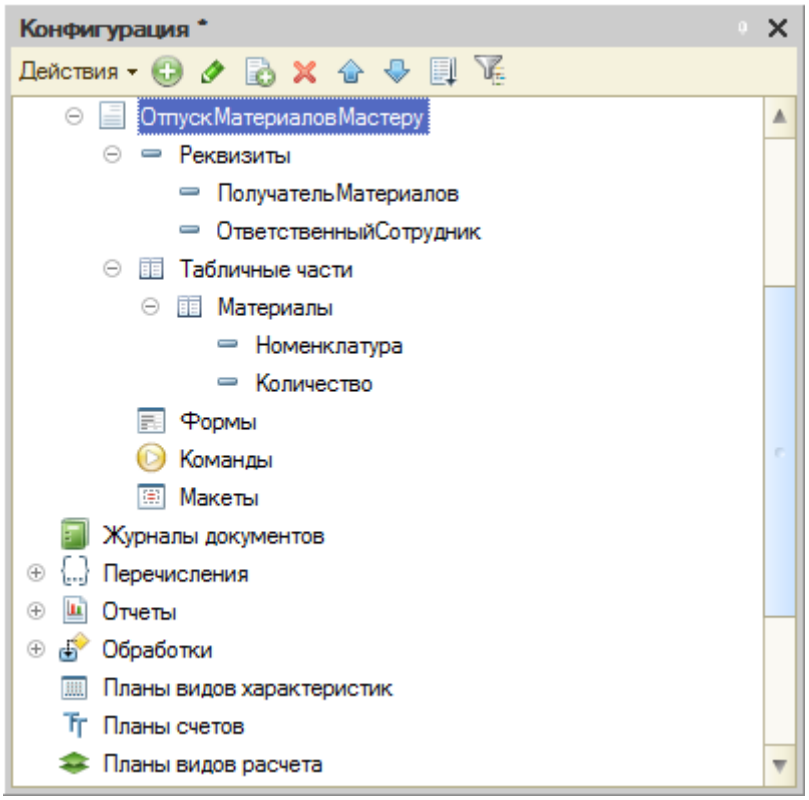


Рисунок 11.10 - Создание документа ОтпускМатериаловМастеру

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

13. 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 11.11.

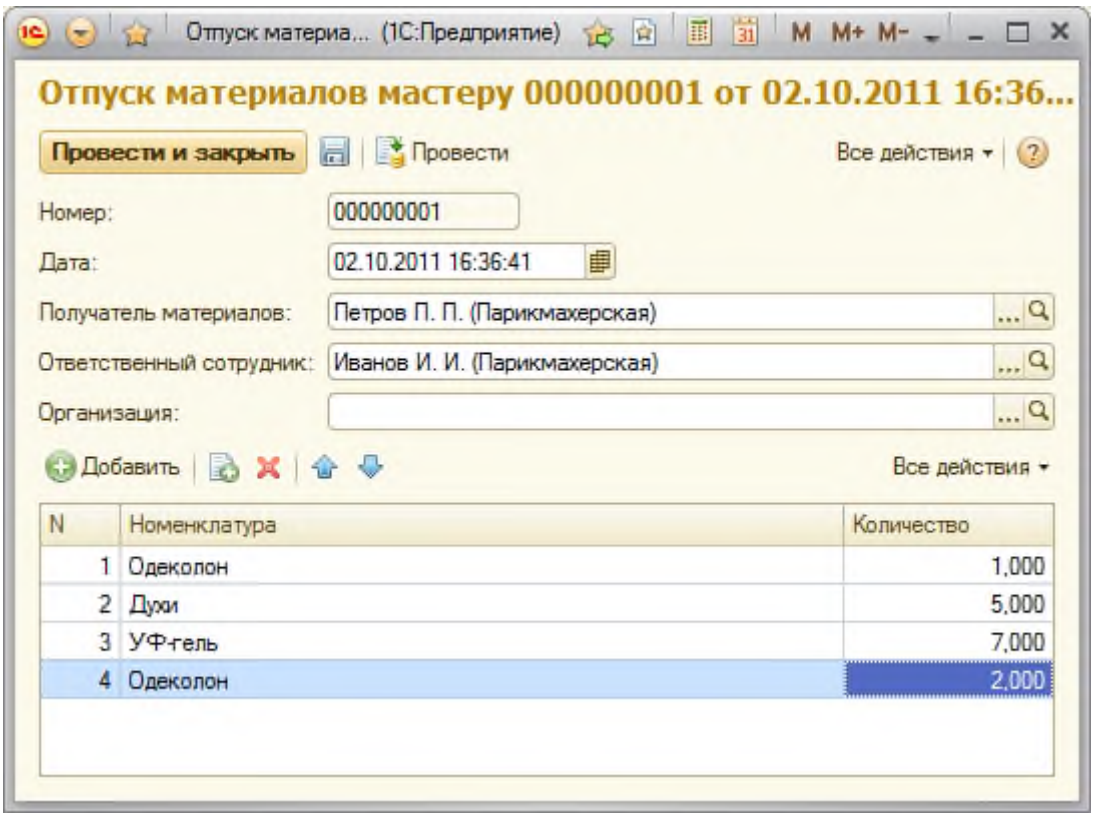


Рисунок 11.11- Документ ОтпускМатериаловМастеру в работе

Практическая работа №13 «Выполнение функционального тестирования»

Цель работы: научиться выполнять функциональное тестирования

ХОД РАБОТЫ:

1. 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.

2. Создадим новый регистр накопления, назовем его **ОстаткиМатериалов**, параметр **Вид регистра** оставим в значении **Остатки**, Рисунок 12.1.

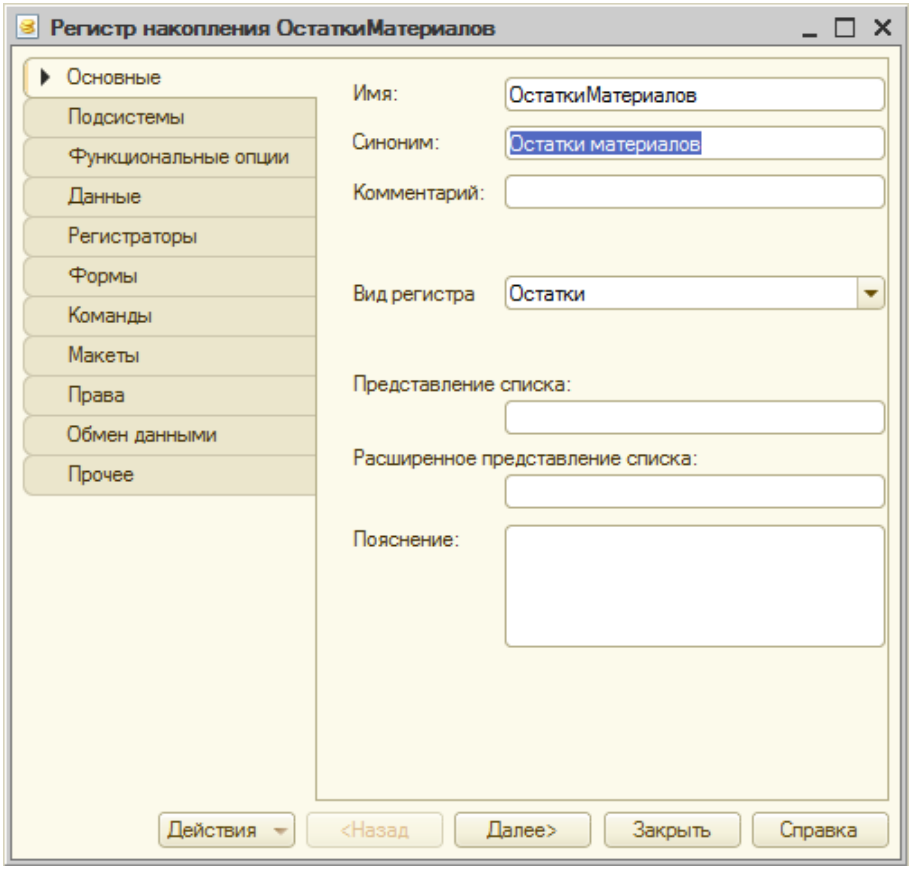


Рисунок 12.1- Регистр накопления ОстаткиМатериалов

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

4. 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 12.2.) – 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.

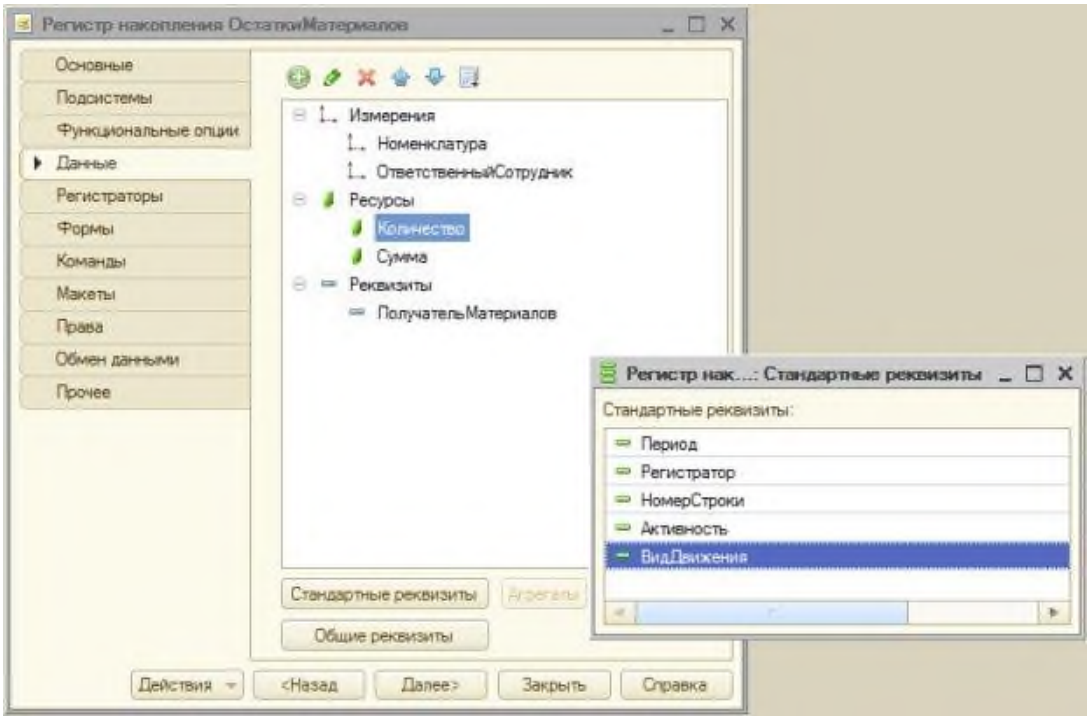


Рисунок 12.2- Регистр накопления «ОстаткиМатериалов», состав данных

5. 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**.

6. Откроем окно редактирования документа **ПоступлениеМатериалов**, перейдем на вкладку **Движения** (рисунок 12.3.) и нажмем на кнопку **Конструктор движений.**

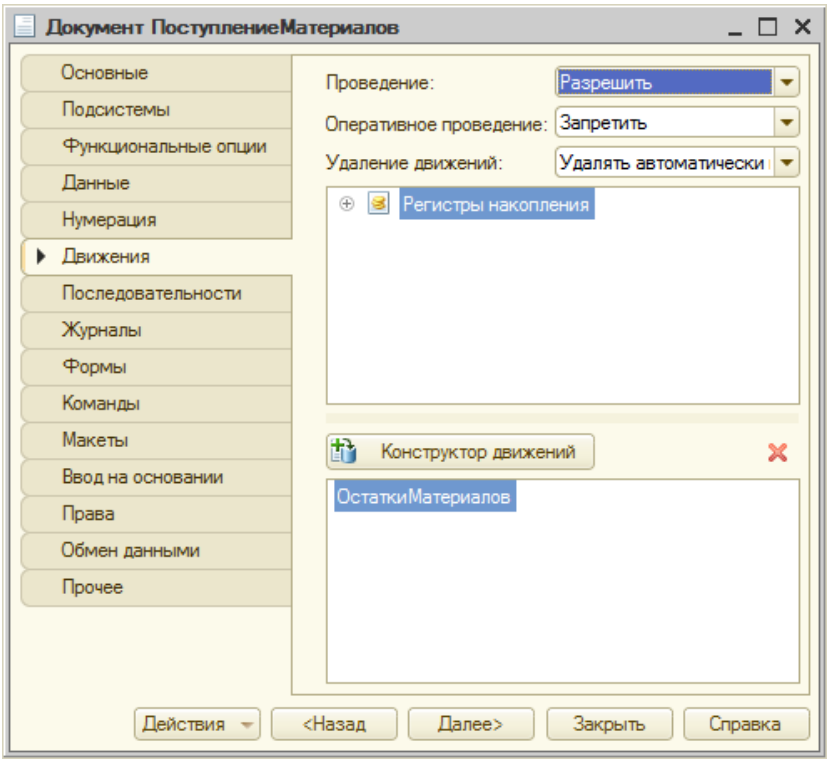


Рисунок 12.3 - Документ «ПоступлениеМатериалов», вкладка «Движения»

7. 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 12.4.

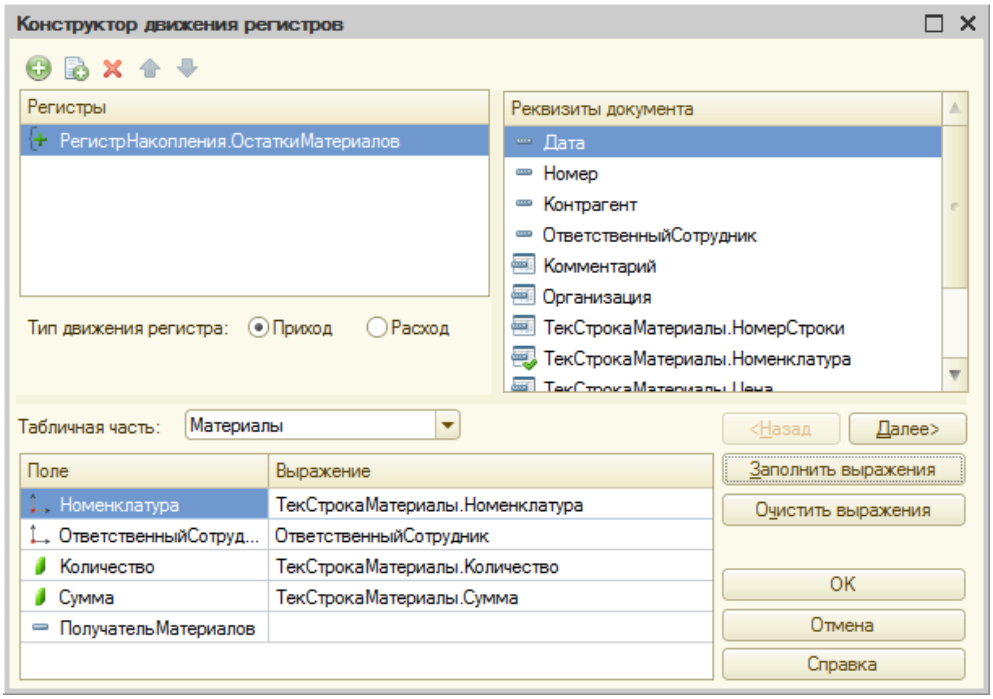
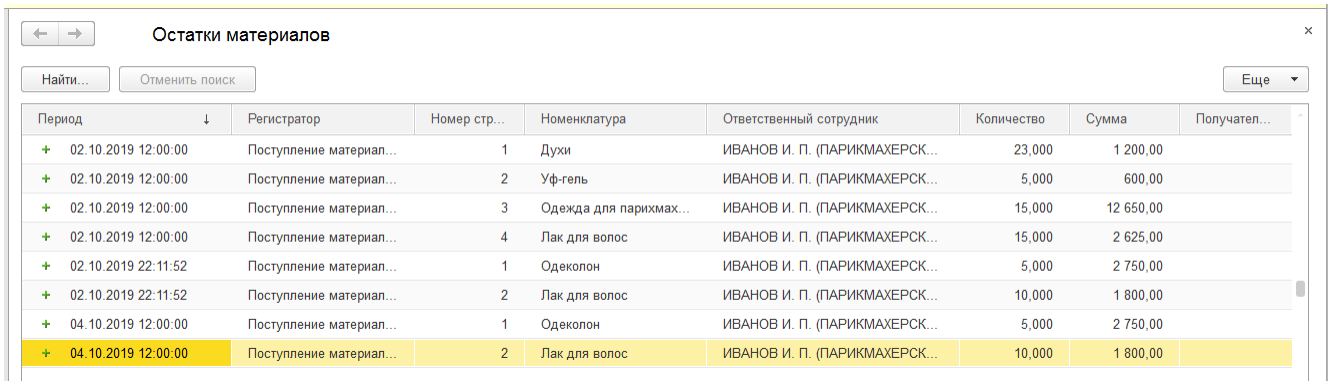


Рисунок 12.4- Конструктор движений

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
```

8. 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 12.5.



Период	Регистратор	Номер стр...	Номенклатура	Ответственный сотрудник	Количество	Сумма	Получател...
02.10.2019 12:00:00	Поступление материал...	1	Духи	ИВАНОВ И. П. (ПАРИКМАХЕРСК...	23,000	1 200,00	
02.10.2019 12:00:00	Поступление материал...	2	Уф-гель	ИВАНОВ И. П. (ПАРИКМАХЕРСК...	5,000	600,00	
02.10.2019 12:00:00	Поступление материал...	3	Одежда для парихмах...	ИВАНОВ И. П. (ПАРИКМАХЕРСК...	15,000	12 650,00	
02.10.2019 12:00:00	Поступление материал...	4	Лак для волос	ИВАНОВ И. П. (ПАРИКМАХЕРСК...	15,000	2 625,00	
02.10.2019 22:11:52	Поступление материал...	1	Одеколон	ИВАНОВ И. П. (ПАРИКМАХЕРСК...	5,000	2 750,00	
02.10.2019 22:11:52	Поступление материал...	2	Лак для волос	ИВАНОВ И. П. (ПАРИКМАХЕРСК...	10,000	1 800,00	
04.10.2019 12:00:00	Поступление материал...	1	Одеколон	ИВАНОВ И. П. (ПАРИКМАХЕРСК...	5,000	2 750,00	
04.10.2019 12:00:00	Поступление материал...	2	Лак для волос	ИВАНОВ И. П. (ПАРИКМАХЕРСК...	10,000	1 800,00	

Рисунок 12.5- 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.

Практическая работа №14 «Тестирование интеграции»

Цель работы: научиться выполнять тестирование интеграции

ХОД РАБОТЫ:

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

2. 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 13.1.

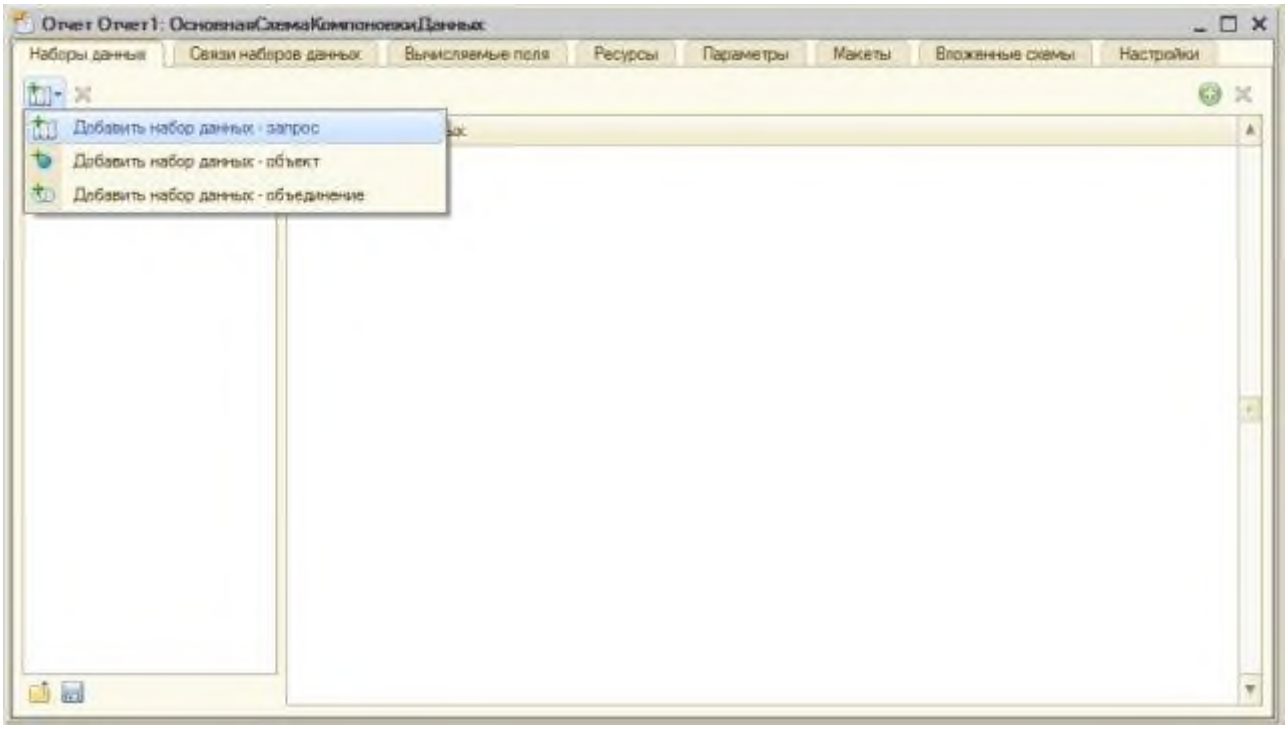


Рисунок 13.1- Добавление нового набора данных – запроса

3. 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.

4. Из виртуальной таблицы регистра накопления **ОстаткиМатериалов** выберем следующие поля, Рисунок 13.2.

- Номенклатура
- ОтвественныйСотрудник
- КоличествоОстаток

– СуммаОстаток

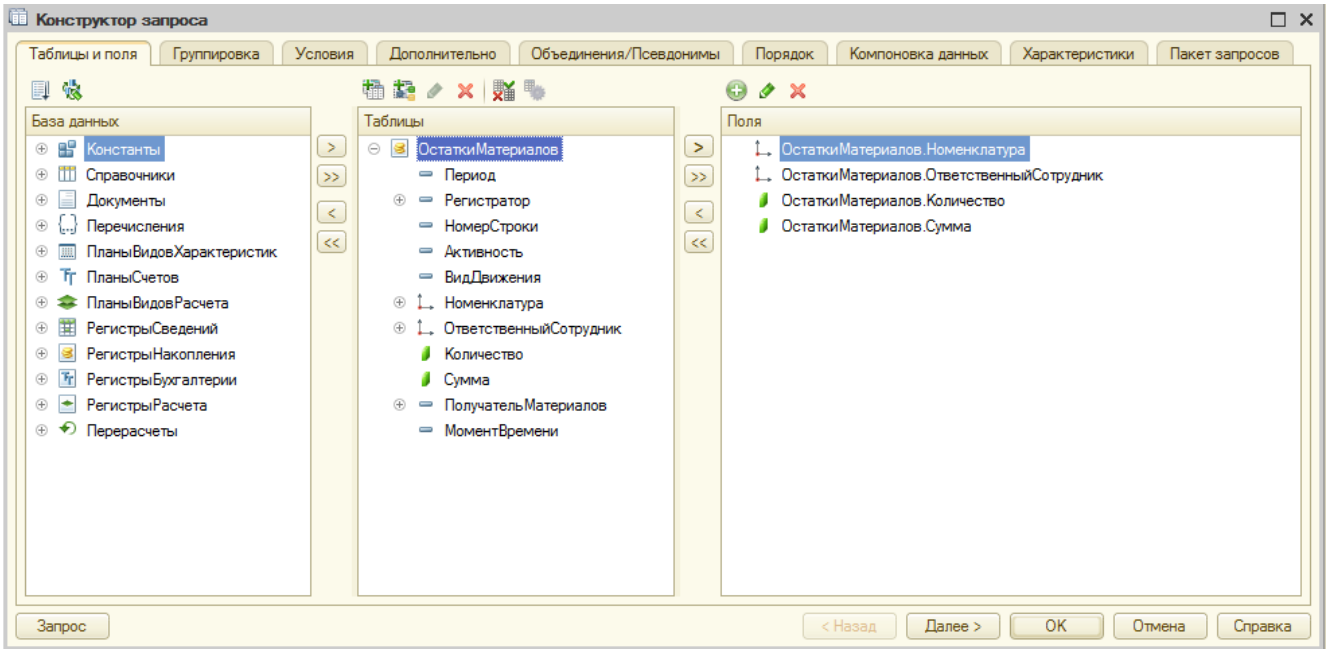


Рисунок 13.2- Создание запроса

5. 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 13.3. – с этими полями мы сможем работать при создании отчета.

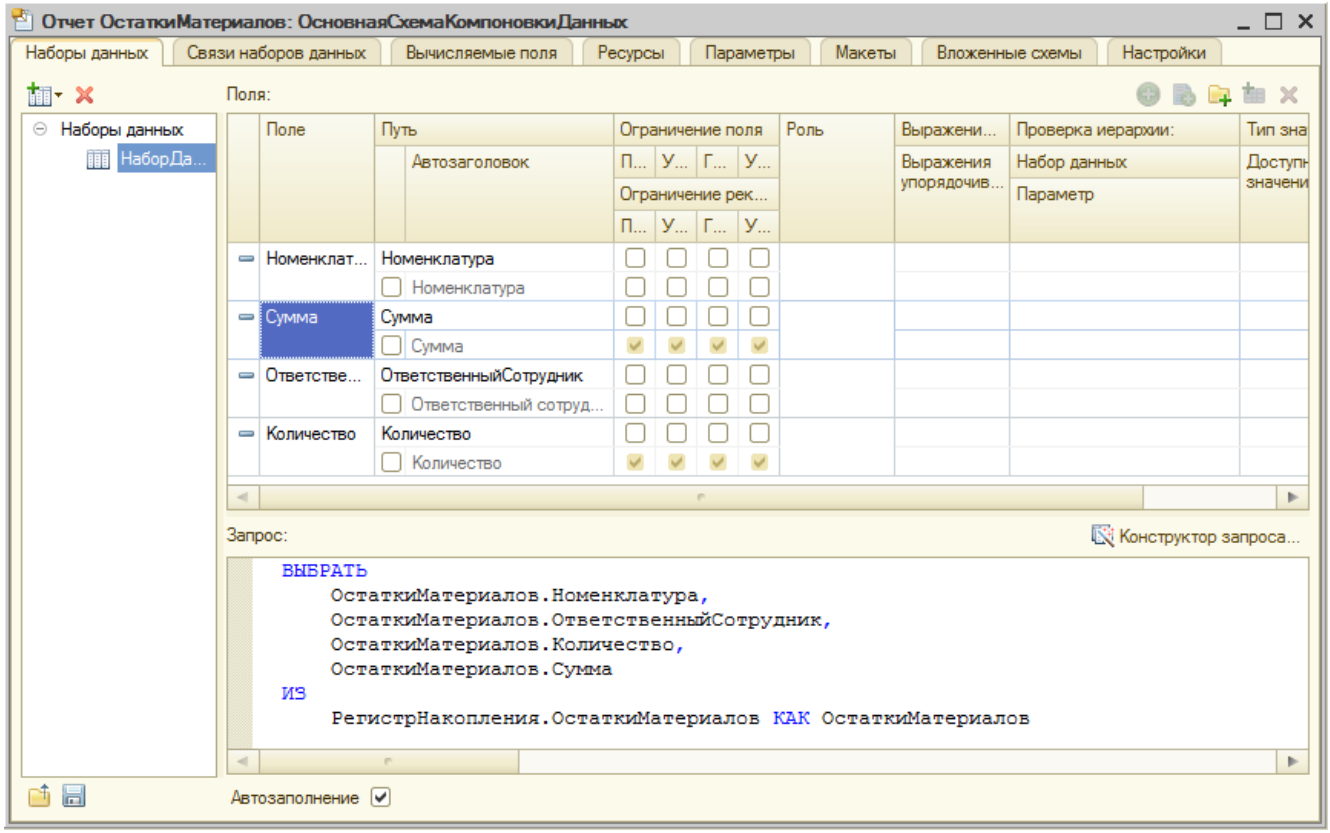


Рисунок 13.3- Автозаполнение списка полей на закладке Наборы данных

6. 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 13.4.

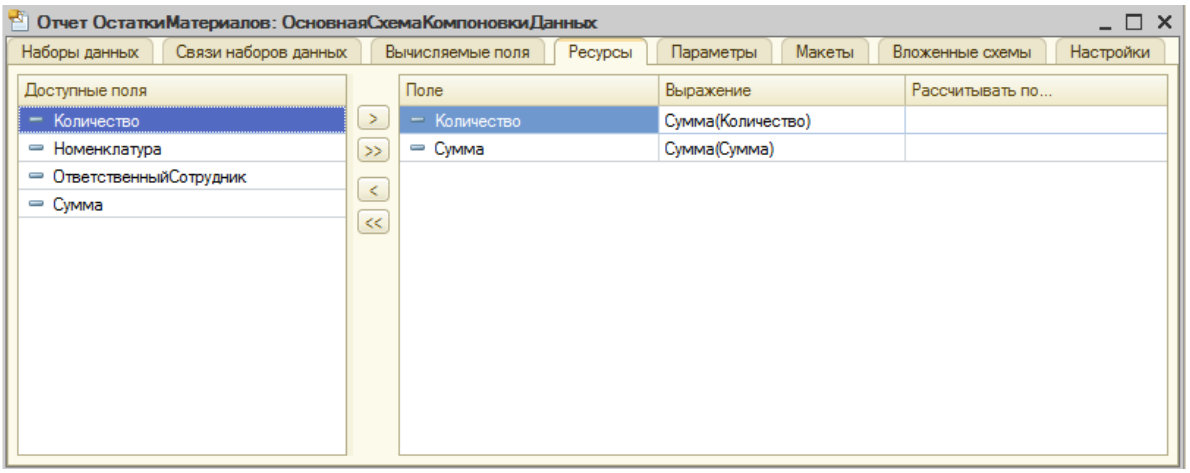


Рисунок 13.4 - Настройка состава ресурсов отчета

7. 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 13.5.):

- Номенклатура
- КоличествоОстаток
- СуммаОстаток.

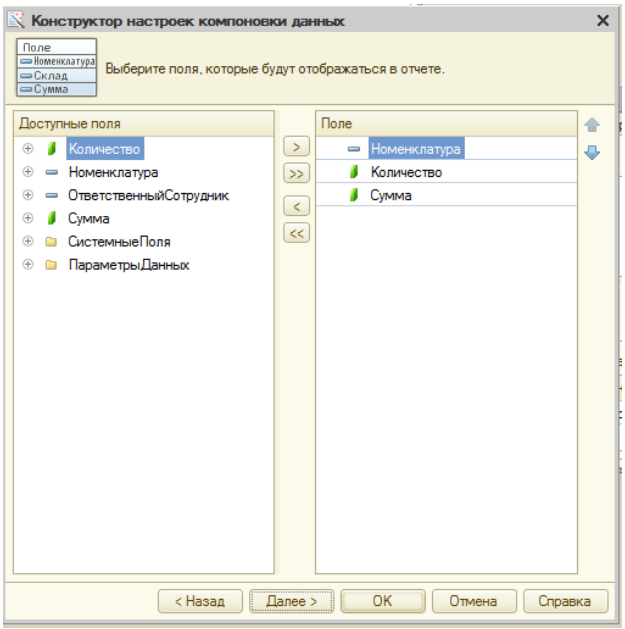


Рисунок 13.5- Выбор полей, которые будут отображаться в отчете

8. 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**.

9. В следующем окне конструктора, который позволяет задать упорядочение отчета, зададим упорядочивание по полю **Номенклатура**, по возрастанию, Рисунок 13.6.

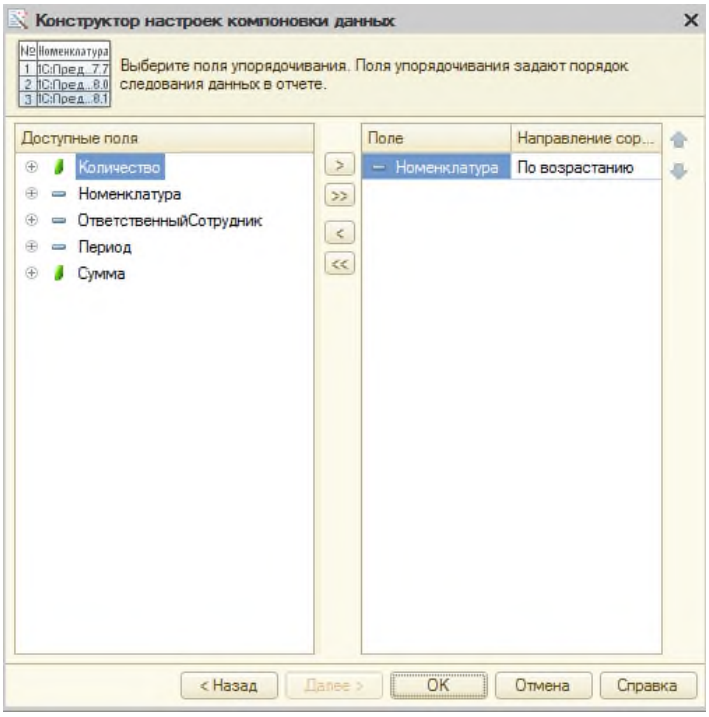


Рисунок 13.6- Настройка упорядочивания отчета

10. Нажмем **ОК**, в отчет будет добавлена новая таблица.
11. В верхней части формы конструктора СКД, на закладке **Параметры**, добавим параметр **Дата** и укажем тип Дата (рисунок 13.7).

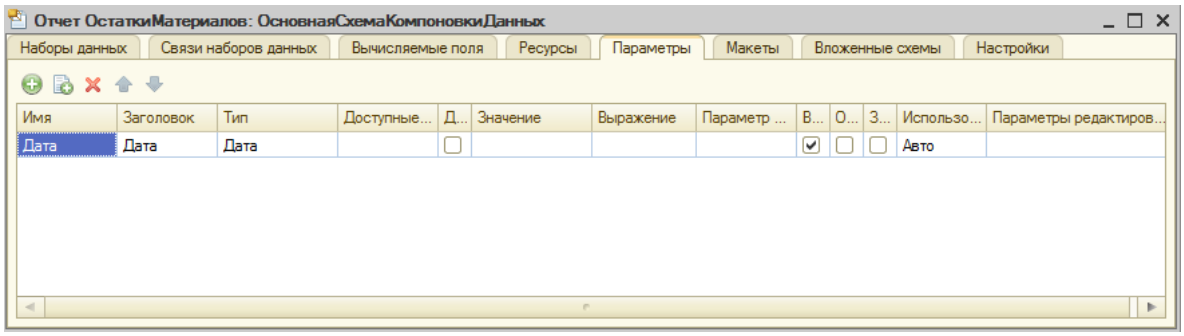


Рисунок 13.7- Добавление параметра отчета

12. Откроем вкладку **Настройки,** выберем внизу экрана вкладку **Параметры,** откроем настройку параметра **Дата** кнопкой **Свойства элемента пользовательских настроек** . В появившемся окне установим флаг **Включать в пользовательские настройки**, режим редактирования оставим в значении **Быстрый доступ**, Рисунок 13.8.

Это позволит нам вывести данный параметр в форму отчета, позволит пользователю выбирать нужную дату построения отчета по остаткам.

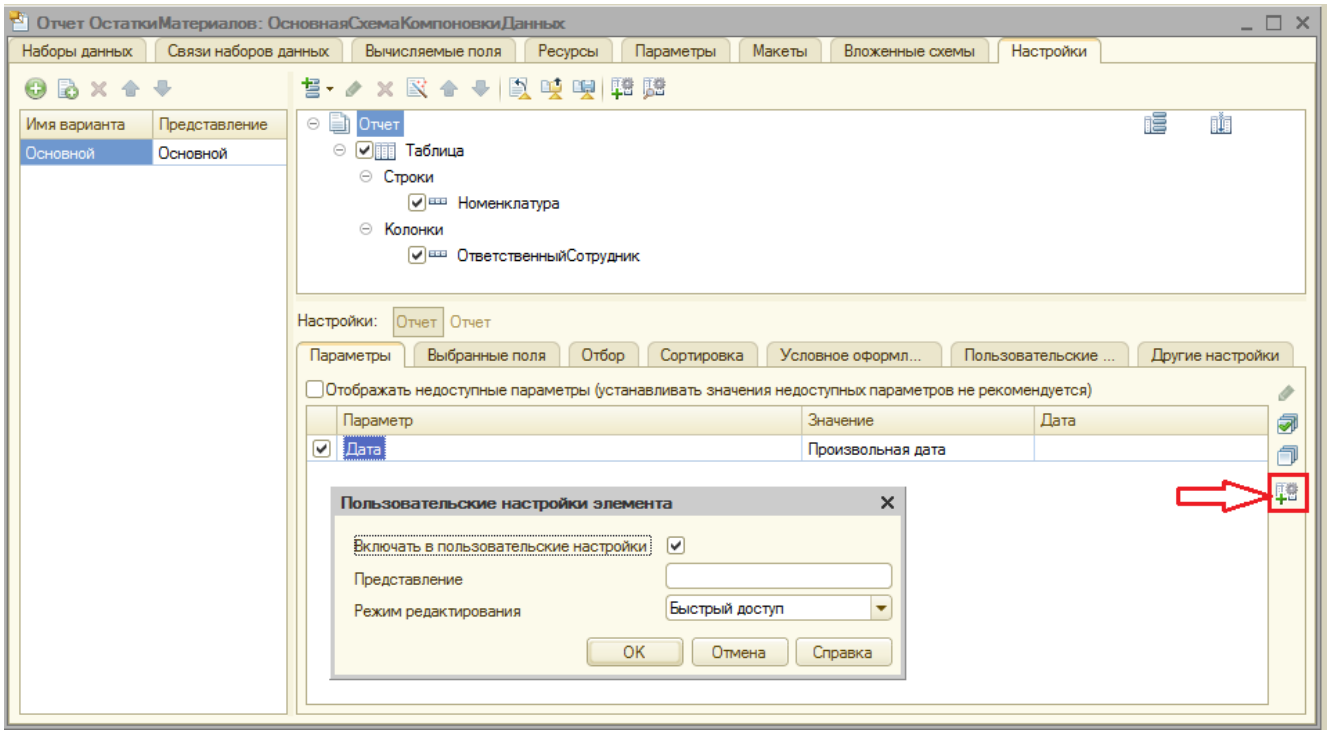
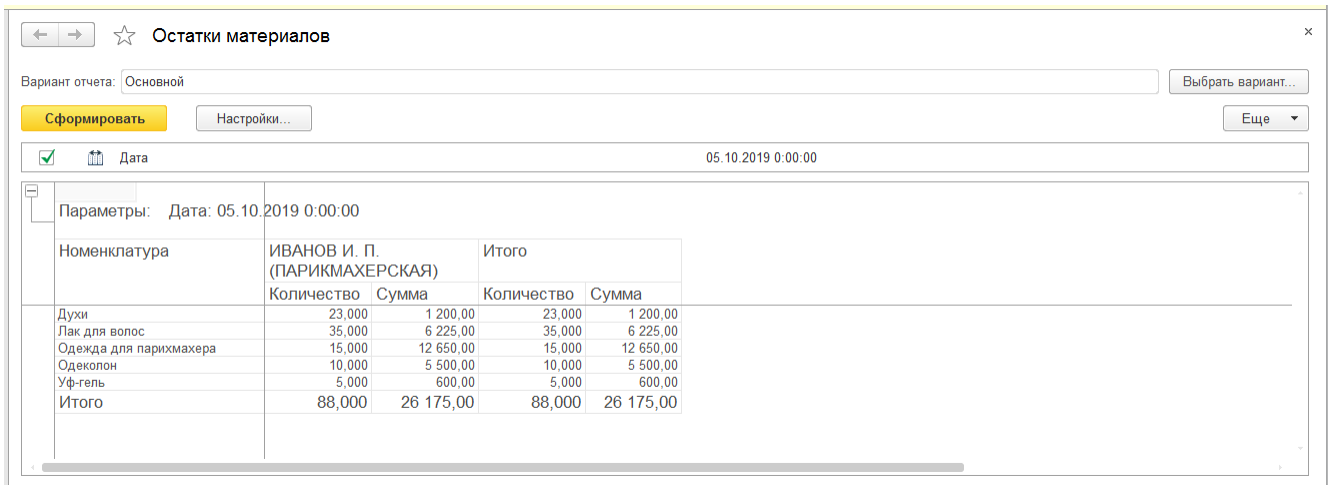


Рисунок 13.8- Настройка вывода параметра

11. Запустим систему в режиме 1С:Предприятие и построим отчет, Рисунок 13.9.



Номенклатура	ИВАНОВ И. П. (ПАРИКМАХЕРСКАЯ)		Итого	
	Количество	Сумма	Количество	Сумма
Духи	23,000	1 200,00	23,000	1 200,00
Лак для волос	35,000	6 225,00	35,000	6 225,00
Одежда для парихмахера	15,000	12 650,00	15,000	12 650,00
Одеколон	10,000	5 500,00	10,000	5 500,00
Уф-гель	5,000	600,00	5,000	600,00
Итого	88,000	26 175,00	88,000	26 175,00

Рисунок 13.9- Готовый отчет по количественным и суммовым остаткам материалов

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

Практическая работа №15 «Документирование результатов тестирования»

Цель работы: научиться выполнять документирование результатов тестирования.

ХОД РАБОТЫ:

1.Займемся проведением документа, отвечающего за списание материалов. Это – документ **ОтпускМатериаловМастеру**.

Наши две задачи:

Во-первых, мы хотим, чтобы система не позволяла списать больше материалов, чем числится за конкретным ответственным лицом. Это означает, что перед формированием движений мы должны сверить данные, введенные в табличную часть документа с данными по остаткам материалов, хранящимися в нашей базе, и, в том случае, если материалов нам не хватит – отказаться проводить документ и сообщить пользователю об ошибке.

Во-вторых, списывая материалы, мы должны придерживаться какой-либо политики оценки. Наиболее простая и широко используемая политика – это списание материалов по средней стоимости.

Определившись с нашими двумя основными задачами – реализации списания материалов по средней стоимости и контроля остатков, приступим к работе над процедурой для проведения нашего документа.

2. Перейдем в модуль объекта документа **ОтпускМатериаловМастеру**, с помощью панели инструментов **Модуль** создадим процедуру **ОбработкаПроведения**. Данные из табличной части мы будем получать с помощью запроса – в дальнейшем мы будем развивать этот запрос для получения необходимых сведений об остатках номенклатуры.

3. При создании запроса очень удобно пользоваться консолью запросов, которая позволяет в режиме 1С:Предприятие тут же проверять результаты, возвращаемые запросом. Подобные обработки можно найти на дисках ИТС, на различных Интернет-ресурсах.

4. Итак, обработку консоли запросов следует открыть командой **Главное меню > Файл > Открыть**. Начнем конструировать запрос, выбрав все поля из таблицы документа **ОтпускМатериаловМастеру**, рисунок 14.1.

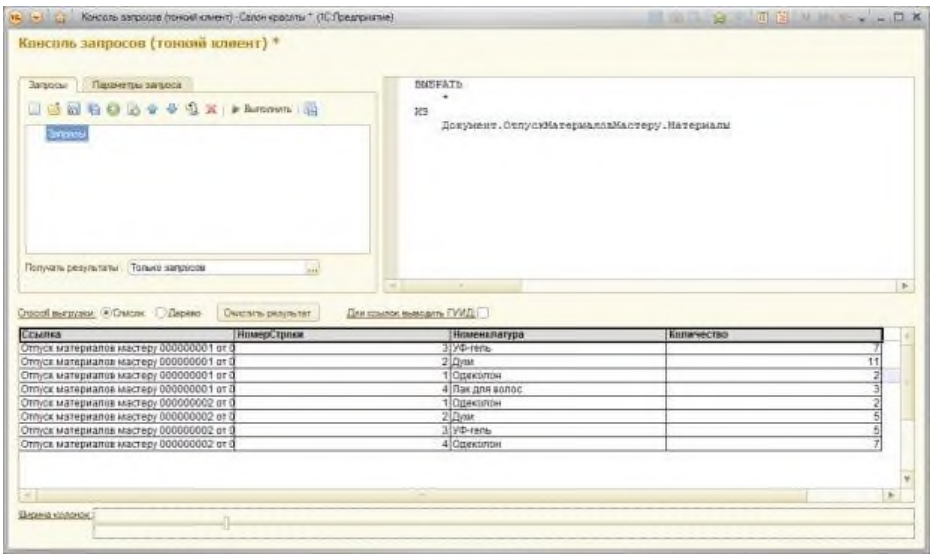


Рисунок 14.1- Конструирование запроса с помощью консоли запросов

5 В начале наш запрос имеет такой вид:

```
ВЫБРАТЬ
   *
ИЗ
   Документ.ОтпускМатериаловМастеру.Материалы
```

Мы выбрали все поля из таблицы, не вводя никаких ограничений. Мы собираемся получать данные из таблицы документа, проведением которого мы занимаемся. В запросе же мы получили данные по всем документам. Ограничим наш запрос по документу. Модифицируем запрос в консоли таким образом:

```
ВЫБРАТЬ
   *
ИЗ
   Документ.ОтпускМатериаловМастеру.Материалы
ГДЕ
   Ссылка=&Ссылка
```

6. Для того, чтобы задать параметр запроса в консоли запросов, перейдем на вкладку **Параметры запроса**, нажмем на кнопку **Заполнить**, после чего в поле **Значение параметра** выберем нужное его значение, в нашем случае – это будет один из документов **ОтпускМатериаловМастеру**, рисунок 14.2.

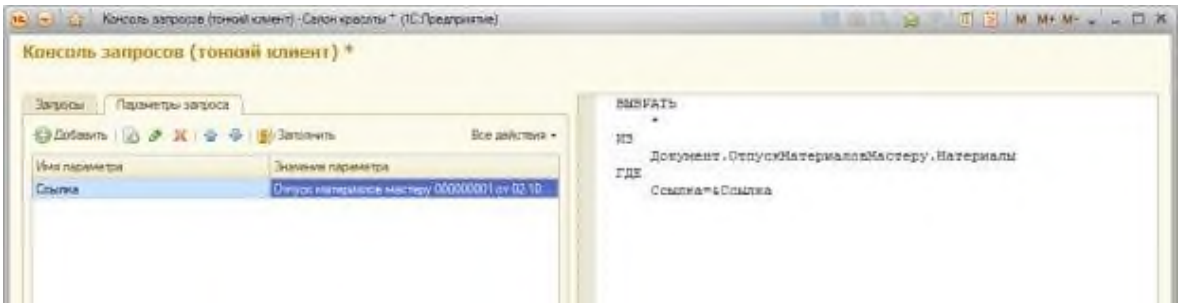


Рисунок 14.2 - Настройка параметра запроса

7. Выполнив этот запрос, получим таблицу **Материалы** из указанного документа. Теперь подумаем над тем, какие именно данные нас интересуют. Нам нужны, во-первых, сведения о номенклатуре (поле **Номенклатура**), во-вторых – о количестве номенклатуры, которую мы хотим списать (поле **Количество**). Модифицируем запрос следующим образом:

```
ВЫБРАТЬ
  ДокМ.Номенклатура,
  ДокМ.Количество
ИЗ
  Документ.ОтпускМатериаловМастеру.Материалы КАК ДокМ
ГДЕ
  Ссылка=&Ссылка
```

Этот запрос даст нам такой результат:

Номенклатура	Количество
Одеколон	2
Духи	11
УФ-гель	7
Лак для волос	3
Одеколон	3

8. В документе мы намеренно смоделировали ситуацию, в которой пользователь, заполняя его, два раза ввел одну и ту же номенклатурную позицию. Сгруппируем теперь результаты запроса по полю **Номенклатура** – придем к такому тексту запроса:

```
ВЫБРАТЬ
  ДокМ.Номенклатура,
  СУММА (ДокМ.Количество) КАК Количество
ИЗ
  Документ.ОтпускМатериаловМастеру.Материалы КАК ДокМ
ГДЕ
  Ссылка=&Ссылка
СГРУППИРОВАТЬ ПО ДокМ.Номенклатура
```

Здесь мы применили функцию **СУММА** к количественному показателю и с помощью выражения **СГРУППИРОВАТЬ ПО** сгруппировали результаты по полю **Номенклатура**. Это привело к такому результату:

Номенклатура	Количество
Духи	11
Одеколон	5
УФ-гель	7
Лак для волос	3

Теперь все данные, которые нужны нам для проведения документа, мы получили.

9. Следующим этапом работы над запросом будет добавление в него команд для выбора нужных данных из регистра накопления **ОстаткиМатериалов**. Мы приходим к такому запросу:

```
ВЫБРАТЬ
  ДокМ.Номенклатура,
  СУММА (ДокМ.Количество) КАК Количество,
  МАКСИМУМ (ОстМ.КоличествоОстаток) КАК КоличествоОстатков,
  МАКСИМУМ (ОстМ.СуммаОстаток) КАК СуммаОстатков
ИЗ
  Документ.ОтпускМатериаловМастеру.Материалы КАК ДокМ
  ЛЕВОЕ СОЕДИНЕНИЕ
  РегистрНакопления.ОстаткиМатериалов.Остатки(&МоментВремени)
Как ОстМ
  ПО
    ДокМ.Номенклатура = ОстМ.Номенклатура
ГДЕ
  Ссылка=&Ссылка
СГРУППИРОВАТЬ ПО ДокМ.Номенклатура
```

Здесь мы соединили таблицу регистра остатков с полученной таблицей документа по полю **Номенклатура**. К числовым полям, полученным из регистра, мы применили функцию **МАКСИМУМ** – иначе запрос будет выполняться неверно. В частности, если бы выше мы не выполнили группировку результатов запроса по полю **Номенклатура**, то в результатах запроса мы получили бы несколько полей с одной и той же номенклатурой, к каждому из которых было бы присоединено одно и то же поле из таблицы регистра.

Вышеописанный запрос привел к такому результату:

Номенклатура	Количество	КоличествоОстатков	СуммаОстатков
Духи	11	36	4 899
Одеколон	5	18	3 510
УФ-гель	7	11	3 350
Лак для волос	3	NULL	NULL

Здесь нас не устраивают два момента. Во-первых, в полях **КоличествоОстатков** и **СуммаОстатков** показаны данные по всем ответственным лицам – а нам нужно знать данные лишь по тому ответственному, с которого мы материалы списываем. Во-вторых, по номенклатурной позиции, по которой данных в регистре **ОстаткиМатериалов** не имеется, в полях находится значение **NULL**. Для того, чтобы попытка работать с этим значением не привела в будущем к возникновению ошибок, обработаем поля, полученные из регистра, функцией **ЕСТЬNULL**.

10. Модифицируем запрос в соответствии с последними соображениями.

```
ВЫБРАТЬ
  ДокМ.Номенклатура,
```

```
        СУММА (ДокМ.Количество) КАК Количество,
        МАКСИМУМ      (ЕСТЬNULL(ОстМ.КоличествоОстаток,      0))      КАК
КоличествоОстатков,
        МАКСИМУМ (ЕСТЬNULL(ОстМ.СуммаОстаток, 0)) КАК СуммаОстатков
      ИЗ
        Документ.ОтпускМатериаловМастеру.Материалы КАК ДокМ
        ЛЕВОЕ СОЕДИНЕНИЕ
        РегистрНакопления.ОстаткиМатериалов.Остатки(&МоментВремени,
ОтветственныйСотрудник = &ОтвСотр) Как ОстМ
        ПО
          ДокМ.Номенклатура = ОстМ.Номенклатура
      ГДЕ
        Ссылка=&Ссылка
      СГРУППИРОВАТЬ ПО ДокМ.Номенклатура
```

Здесь мы добавили отбор из регистра только записей, относящихся к заданному ответственному сотруднику (**ОтветственныйСотрудник = &ОтвСотр**) и применили к показателям количества и суммы, полученным из регистра, функцию **ЕСТЬNULL**. Если в поле находится **NULL**, мы заменяем это значение нулем.

Результат запроса теперь выглядит так:

Номенклатура	Количество	КоличествоОстатков	СуммаОстатков
Духи	11	15	1 860
Одеколон	5	2	400
УФ-гель	7	3	900
Лак для волос	3	0	0

11. Скопируем полученный текст запроса в буфер обмена и перейдем в Конфигуратор. В процедуре ОбработкаПроведения документаОтпускМатериаловМастеру. Процедура пока пуста, щелкнем в ней правой кнопкой мыши и вызовем из контекстного меню командуКонструктор запроса с обработкой результата. В ответ на вопрос конструктора о создании нового запроса, ответим утвердительно, после чего, в окне конструктора нажмем на кнопку Запрос и вставим в пустое поле для текста запроса полученный текст запроса (предварительно нажав на кнопку Редактировать запрос в окне Запрос), Рисунок 14.3.

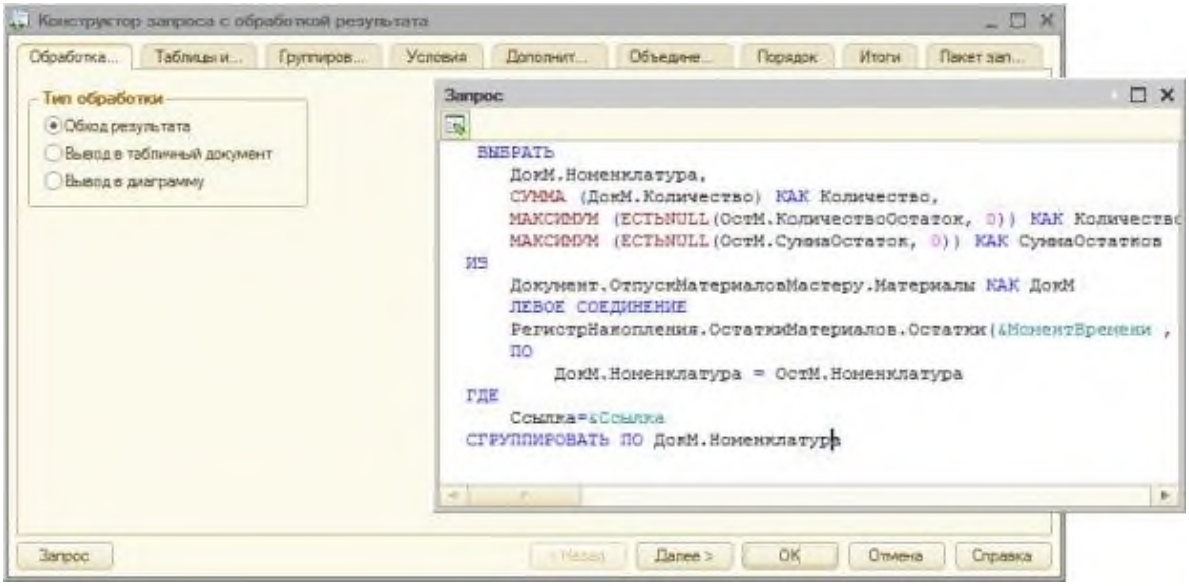


Рисунок 14.3 - Добавление сформированного текста запроса в конструктор

12. На закладке Обработка окна Конструктор запроса оставим переключатель Тип обработки в положении Обход результата. После закрытия окна Запрос конструктор автоматически разберет запрос, "разложит" по закладкам своего окна, при необходимости, его можно будет редактировать, пользуясь инструментами, расположенными на этих закладках. Нас запрос устраивает – поэтому мы можем нажимать в окне конструктора ОК и переходить к дальнейшему редактированию кода, рисунок 14.4.

Рисунок 14.4 - Добавление сформированного текста запроса в конструктор

13. Здесь нас, в первую очередь, не устраивает автоматическое заполнение параметров запроса Заменим код:

```
Запрос.УстановитьПараметр("МоментВремени", МоментВремени);
Запрос.УстановитьПараметр("ОтвСотр", ОтвСотр);
```

На код:

```
Запрос.УстановитьПараметр("МоментВремени", МоментВремени());
Запрос.УстановитьПараметр("ОтвСотр", ОтветственныйСотрудник);
```

Здесь мы, во-первых, вызвали метод МоментВремени(), возвращающий момент времени для нашего документа (то есть – для того, в модуле объекта которого мы сейчас работаем). Во-вторых, мы обратились к реквизиту документа ОтветственныйСотрудник для установки параметра ОтвСотр.

14. Проверим работу созданного механизма, запустив систему в режиме отладки и установив в коде модуля точку останова после получения выборки из результатов запроса.

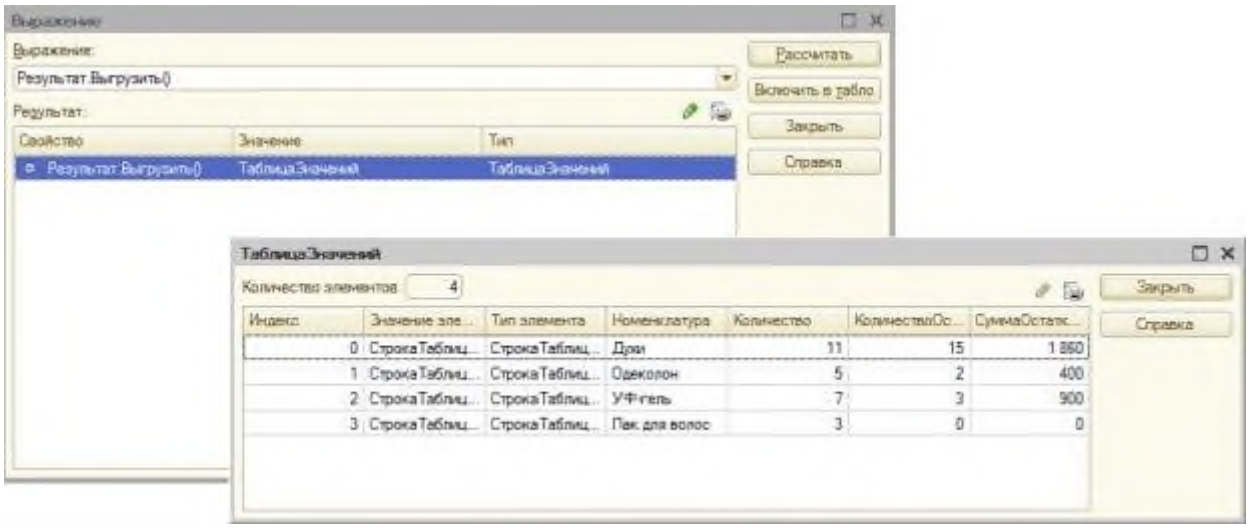


Рисунок 14.5 - Анализ результата выполнения запроса в коде процедуры проведения документа

15. Здесь мы выполнили метод Выгрузить() для результата выполнения запроса (переменная Результат), получили таблицу значений, которую можно проанализировать. Результат нас устраивает, поэтому мы принимаемся за дальнейшую работу над процедурой. В итоге у нас получился следующий код:

```
Процедура ОбработкаПроведения(Отказ, РежимПроведения)
  Запрос = Новый Запрос;
  Запрос.Текст =
    "ВЫБРАТЬ
    |  ДокМ.Номенклатура,
    |  СУММА(ДокМ.Количество) КАК Количество,
    |      МАКСИМУМ(ЕСТЬNULL(ОстМ.КоличествоОстаток,  0))  КАК
КоличествоОстатков,
    |  МАКСИМУМ(ЕСТЬNULL(ОстМ.СуммаОстаток, 0)) КАК СуммаОстатков
    |ИЗ
    |  Документ.ОтпускМатериаловМастеру.Материалы КАК ДокМ
    |    ЛЕВОЕ СОЕДИНЕНИЕ
РегистрНакопления.ОстаткиМатериалов.Остатки(&МоментВремени,
ОтветственныйСотрудник = &ОтвСотр) КАК ОстМ
    |    ПО ДокМ.Номенклатура = ОстМ.Номенклатура
    |ГДЕ
```

```
            |  ДокМ.Ссылка = &Ссылка
            |
            |СГРУППИРОВАТЬ ПО
            |  ДокМ.Номенклатура";

        Запрос.УстановитьПараметр("МоментВремени", МоментВремени());
        Запрос.УстановитьПараметр("ОтвСотр", ОтветственныйСотрудник);
        Запрос.УстановитьПараметр("Ссылка", Ссылка);

        Результат = Запрос.Выполнить();

        ВыборкаДЗ = Результат.Выбрать();

        Движения.ОстаткиМатериалов.Записывать=Истина;

        Пока ВыборкаДЗ.Следующий() Цикл
          Если ВыборкаДЗ.Количество>ВыборкаДЗ.КоличествоОстатков Тогда
            Сообщить("Недостаточное количество товара
"+ВыборкаДЗ.Номенклатура
            +", необходимо "+ВыборкаДЗ.Количество+", в наличии "
            +ВыборкаДЗ.КоличествоОстатков);
            Отказ=Истина;
            Движения.ОстаткиМатериалов.Записывать=Ложь;
          КонецЕсли;

          Если Отказ Тогда
            Продолжить;
          КонецЕсли;

          Движения.ОстаткиМатериалов.Записывать = Истина;
          Движение=Движения.ОстаткиМатериалов.Добавить();
          Движение.ВидДвижения=ВидДвиженияНакопления.Расход;
          Движение.Период=Дата;
          Движение.Номенклатура=ВыборкаДЗ.Номенклатура;
          Движение.Количество=ВыборкаДЗ.Количество;

Движение.Сумма=ВыборкаДЗ.Количество*ВыборкаДЗ.СуммаОстатков/
     ВыборкаДЗ.КоличествоОстатков;
          Движение.ОтветственныйСотрудник=ОтветственныйСотрудник;
          Движение.ПолучательМатериалов=ПолучательМатериалов;
        КонецЦикла;

     КонецПроцедуры
```

16. Проверим результаты работы нашего кода в режиме 1С:Предприятие. Если документ верно реагирует на попытку списания материалов, количество которых превышает имеющееся, и если анализ состава регистра накопления после проведения показывает, что списано нужное количество материалов и их стоимость определена верно – можно считать, что мы справились с поставленной задачей. Создадим два документа «Отпуск материалов мастеру» с учетом наличия остаток для расхода.

17. Мы собираемся построить отчет, который выводил бы сведения о начальном и конечном остатке материалов за определенный временной интервал, а так же – сведения о приходе и расходе материалов за этот период.

Создадим новый отчет, назовем его Материалы, включим в подсистему ОперативныйУчетМатериалов, добавим основную схему компоновки данных, создадим новый набор данных – Запрос. В конструкторе запроса выберем из виртуальной таблицы регистра накопления ОстаткиМатериалов следующие поля, Рисунок 14.6.:

- Номенклатура
- ОтветственныйСотрудник
- КоличествоНачальныйОстаток
- КоличестоПриход
- КоличествоРасход
- КоличествоКонечныйОстаток

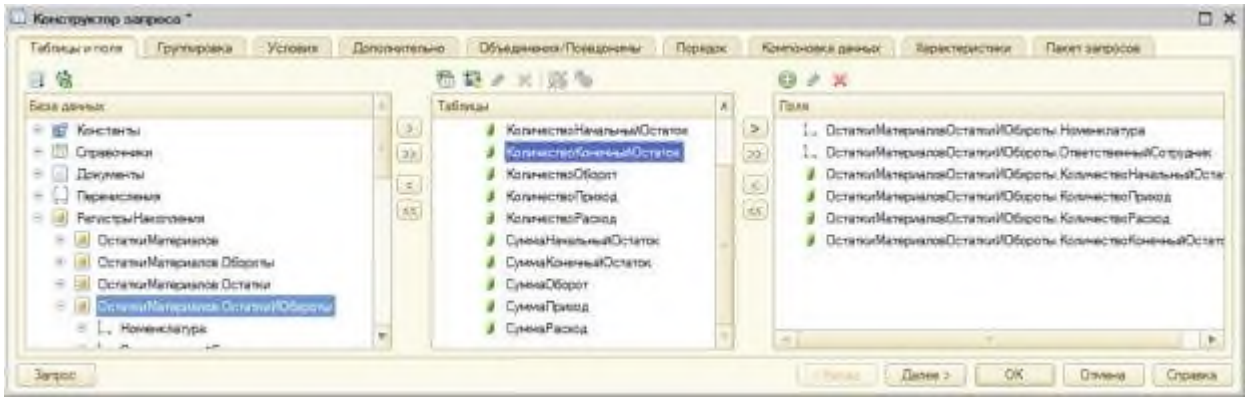


Рисунок 14.6 - Настройка запроса для отчета

18. Нажмем ОК в окне конструктора запроса, перейдем на закладку Ресурсы окна редактора СКД, добавим все количественные поля в состав ресурсов, рисунок 14.7.

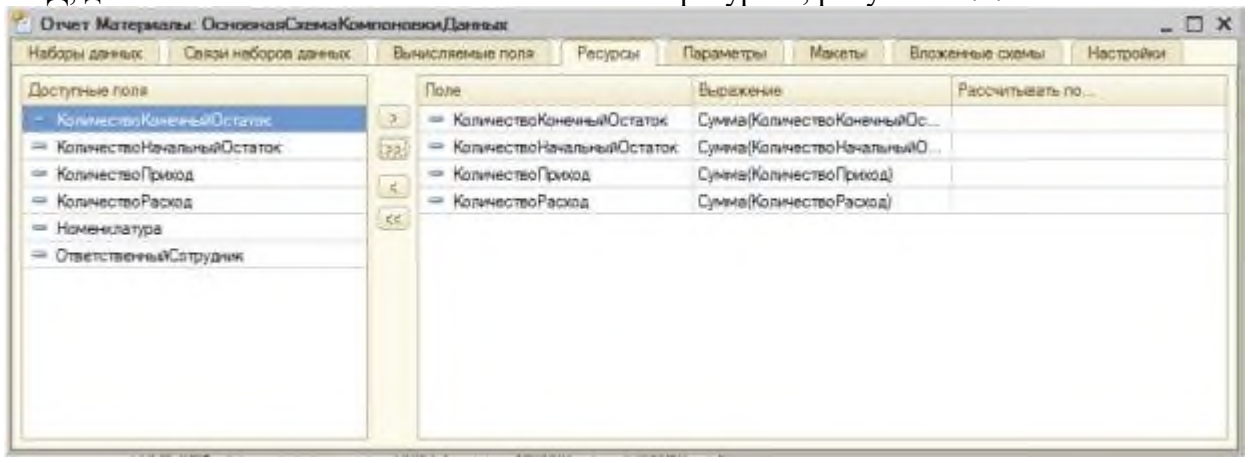


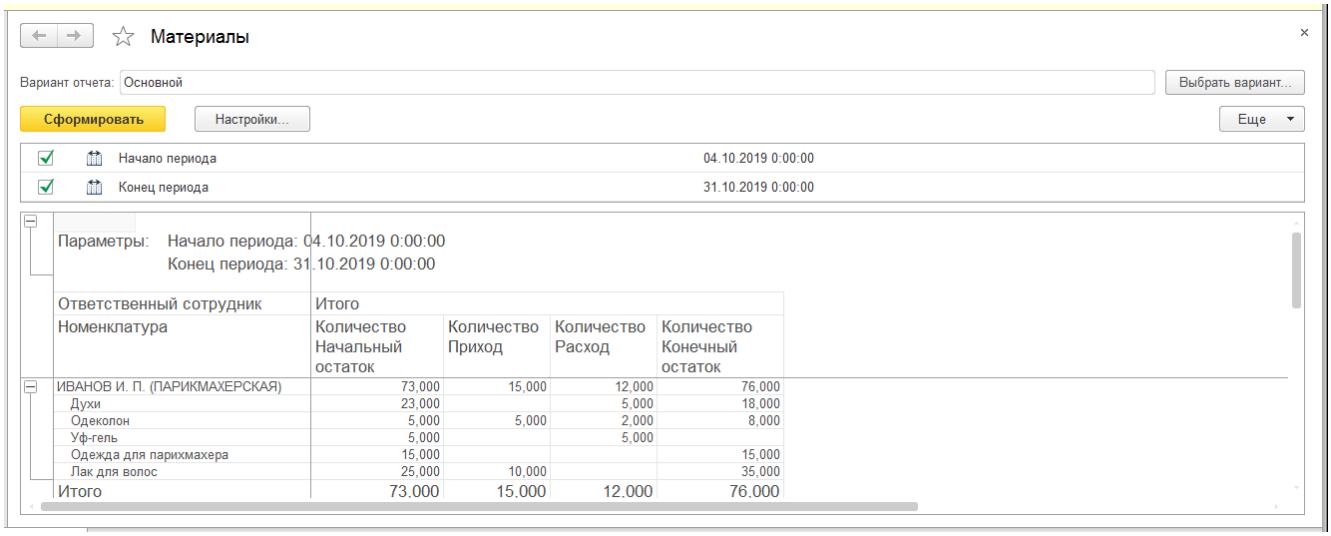
Рисунок 14.7 - Настройка состава ресурсов

19.На закладке Настройки воспользуемся конструктором настроек. Выберем табличный тип отчета, нажмем Далее, в окне настройки состава и порядка следования полей, которые будут отображаться в отчете, расположим поля следующим образом:

- Номенклатура
- ОтветственныйСотрудник

- КоличествоНачальныйОстаток
- КоличестоПриход
- КоличествоРасход
- КоличествоКонечныйОстаток

20. На следующем этапе укажем, что группировка строк будет осуществляться по полю Номенклатура, колонок – по полю **ОтветственныйСотрудник**.
21. На этапе настройки упорядочения укажем упорядочение по возрастанию по полю **Номенклатура**.
На этом настройка таблицы завершена.
22. На верхнем уровне настроек отчета укажем, что параметры Начало периода и Конец периода следует включать в пользовательские настройки.
23. Отчет готов, нам осталось лишь проверить его работу в режиме 1С:Предприятие, Рисунок 14.8.



Ответственный сотрудник / Номенклатура	Итого: Количество Начальный остаток	Количество Приход	Количество Расход	Количество Конечный остаток
ИВАНОВ И. П. (ПАРИКМАХЕРСКАЯ)	73,000	15,000	12,000	76,000
Духи	23,000		5,000	18,000
Одеколон	5,000	5,000	2,000	8,000
Уф-гель	5,000		5,000	
Одежда для парихмахера	15,000			15,000
Лак для волос	25,000	10,000		35,000
Итого	73,000	15,000	12,000	76,000

Рисунок 14.8 - Готовый отчет

24. В нашей конфигурации есть пара документов, относящихся к одной сфере деятельности – к учету материалов. Выше мы упоминали об объекте Журнал документов. Познакомимся с этим объектом поближе.
25. Добавим в конфигурацию новый журнал документов, назовем его ДокументыУчетаМатериалов. Включим журнал в подсистему ОперативныйУчетМатериалов.
На вкладке Данные добавим в состав документов, регистрируемых в журнале, документы ПоступлениеМатериалов и ОтпускМатериаловМастеру. Добавим в журнал графу с именем ОтветственныйСотрудник, заполним свойство Ссылки для этой графы, указав реквизиты ОтветственныйСотрудник из включенных в журнал документов, рисунок 14.9.

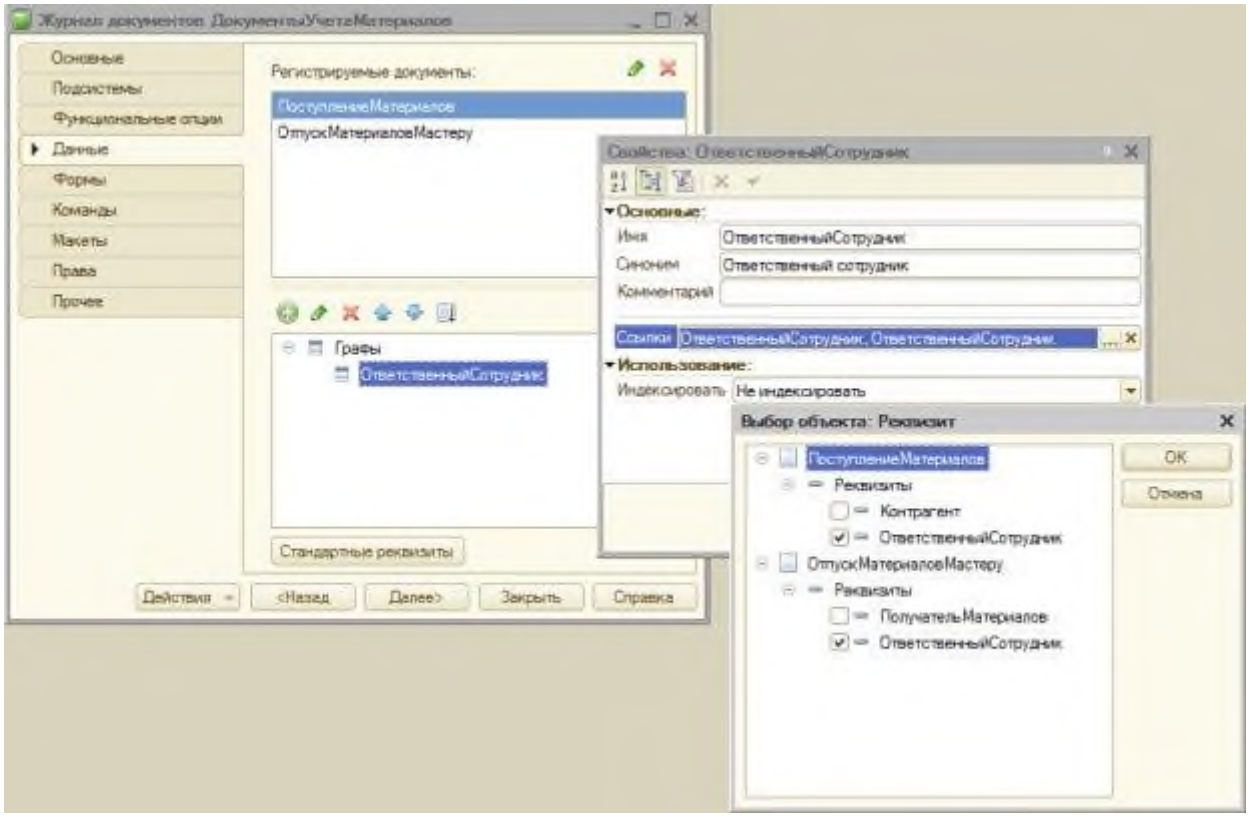


Рисунок 14.9 - Настройка журнала документов

26. В режиме 1С:Предприятие наш журнал позволит просматривать список документов разных типов, включенных в него, рисунок 14.10.

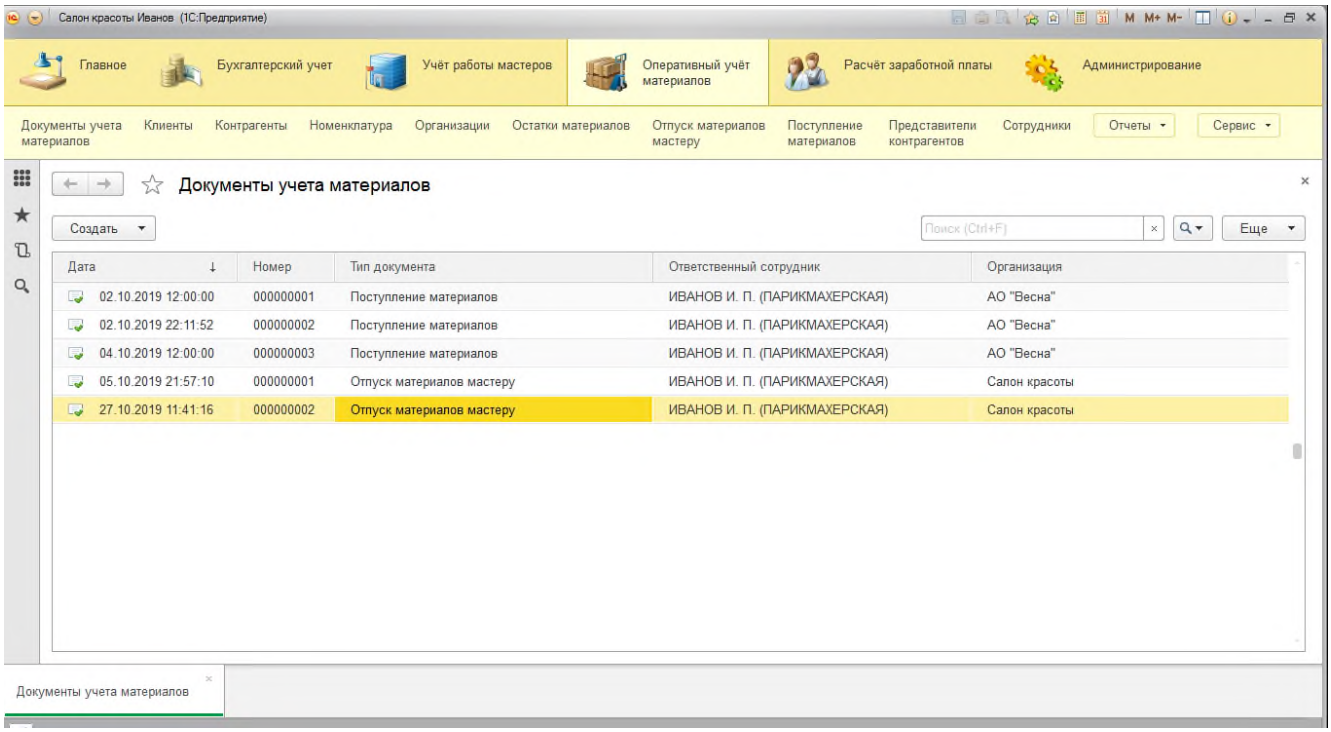


Рисунок 14.10 - Журнал документов в режиме 1С:Предприятие

27.Создадим новую обработку, назовем ее РаботаСДокументами. Включим в подсистему Администрирование.

28. Добавим в обработку команду с именем ВывестиСписокВидовДокументов, зададим обработчик для этой команды, выведем ее на форму обработки.
29. Сейчас мы воспользуемся свойством глобального контекста Метаданные для того, чтобы вывести пользователю список синонимов существующих в конфигурации документов. Для подобных действий нам понадобится серверная процедура, которую мы вызовем из клиентской процедуры обработчика ранее созданной команды. Выполнить запланированное можно с помощью следующего кода:

```
&НаКлиенте
Процедура ВывестиСписокВидовДокументов(Команда)
  ВывестиСинонимыДокументов();
КонецПроцедуры

Процедура ВывестиСинонимыДокументов()
  Для каждого Документ из Метаданные.Документы Цикл
    Сообщить(Документ.Синоним);
  КонецЦикла;
КонецПроцедуры
```

Результат выполнения показан на Рисунок 14.11.

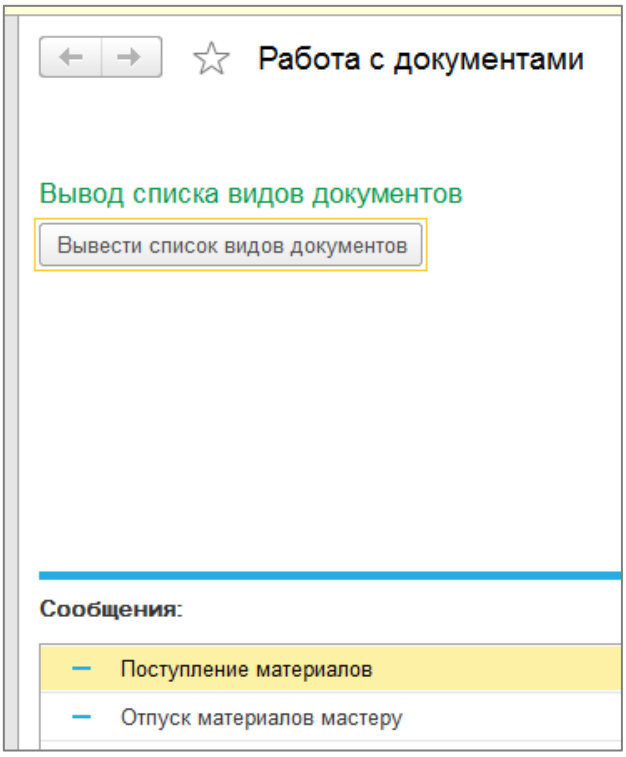


Рисунок 14.11 - Вывод списка синонимов документов

30. Добавим в нашу обработку новую команду – СоздатьДокументПоступлениеМатериалов. Так же добавим новый реквизит – ПроводитьДокумент, поместим его на форму, Рисунок 14.12.
Мы зададим все данные, в том числе – и тип документа для создания – в коде.

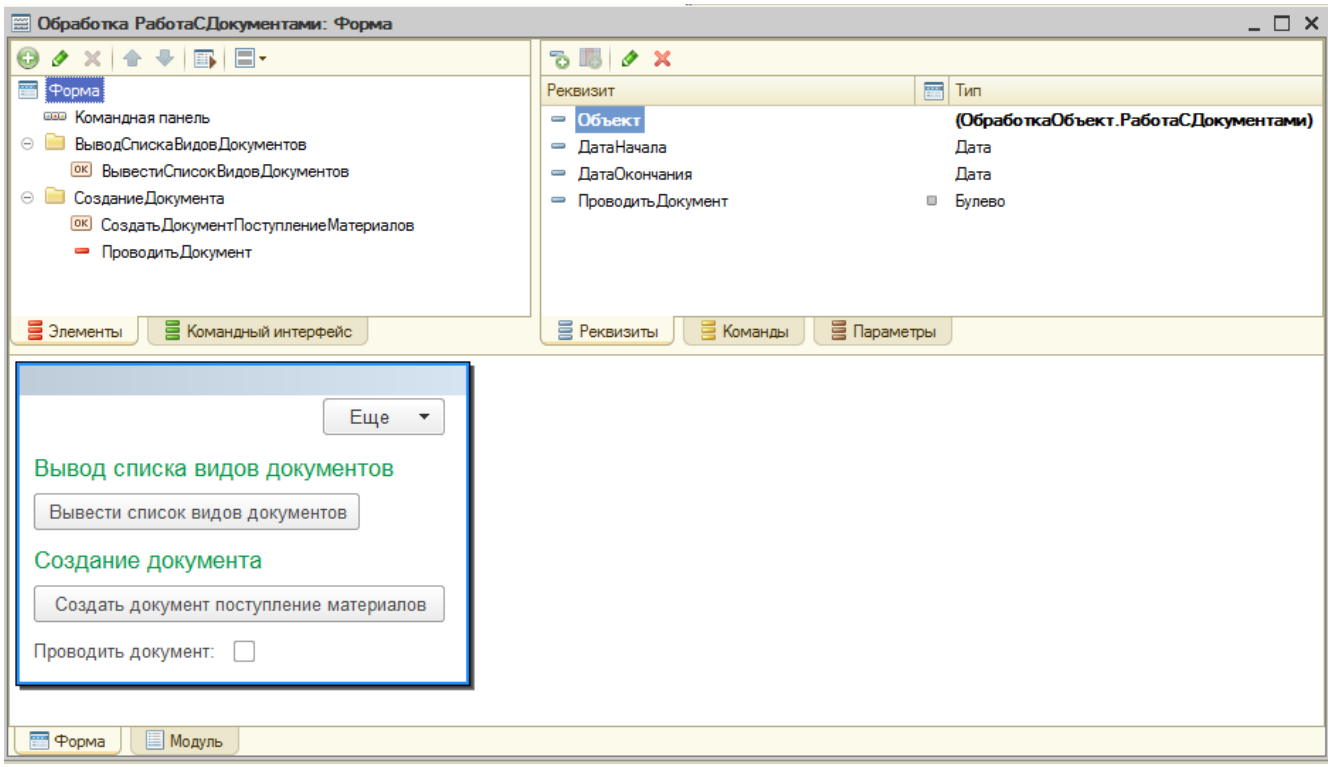


Рисунок 14.12 - Модификация формы обработки

Решить поставленную задачу можно с помощью следующего кода:

```
&НаКлиенте
Процедура СоздатьДокументПоступлениеМатериалов(Команда)
  //Настраиваем режим записи нового документа
  РежимЗаписи=РежимЗаписиДокумента.Запись;
  Если ПроводитьДокумент Тогда
    РежимЗаписи=РежимЗаписиДокумента.Проведение;
  КонецЕсли;
  //В функции будет создан новый документ
  //Она возвратит ссылку на него
  Документ=СоздатьДокумент(РежимЗаписи);
  //Открываем форму документа
  ОткрытьЗначение(Документ);
КонецПроцедуры

Функция СоздатьДокумент(РежимЗаписи)
  //Создаем новый документ
  Документ = Документы.ПоступлениеМатериалов.СоздатьДокумент();
  //Заполняем его реквизиты
  Документ.Дата=ТекущаяДата();

Документ.ОтветственныйСотрудник=Справочники.Сотрудники.НайтиПоКоду("
000000001");

Документ.Контрагент=Справочники.Контрагенты.НайтиПоРеквизиту("Контак
тныеСведения", "г Ростов н/Д ул Мира 1, телефон 253-34-34");
      Документ.Комментарий="Документ создан автоматически";
```

```
        //Заполняем строку табличной части
        НоваяСтрокаТЧ=Документ.Материалы.Добавить();

НоваяСтрокаТЧ.Номенклатура=Справочники.Номенклатура.НайтиПоНаименова
нию("Духи");
        НоваяСтрокаТЧ.Количество=10;
        НоваяСтрокаТЧ.Цена=200;
        НоваяСтрокаТЧ.Сумма=10*200;
        //Записываем документ
        Документ.Записать(РежимЗаписи);
        //Возвращаем ссылку на документ
        Возврат(Документ.Ссылка);
    КонецФункции
```

Перед созданием документа через обработку введите контрагенту ООО «Полет» контактные сведения: «г Ростов н/Д ул Мира 1, телефон 253-34-34».
Вот, как выглядит документ, созданный программно с помощью нашего кода, рисунок 14.13.

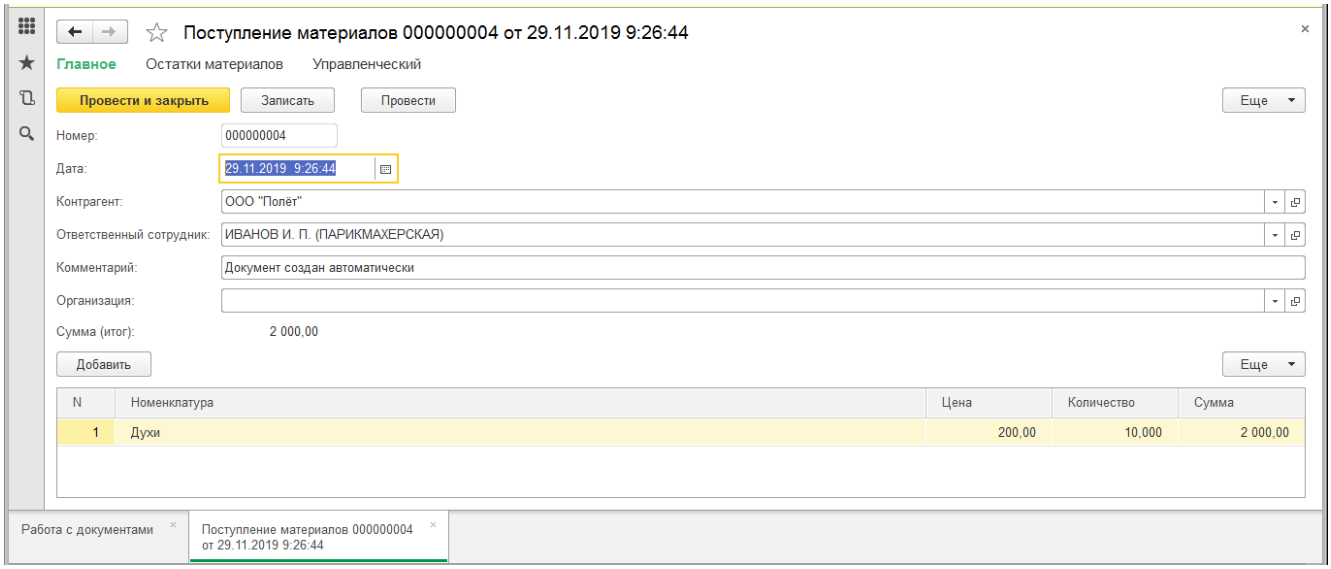


Рисунок 14.13 - Документ, созданный автоматически

31. Решим теперь следующую задачу. Нужно пометить на удаление все документы типа **Поступление Материалов**, которые созданы автоматически – их реквизит **Комментарий** содержит текст "Документ создан автоматически".
32. Добавим в форму обработки новую команду, назовем ее ПометитьНаУдаление. Поставленную задачу можно реализовать с помощью следующего кода:

```
    &НаКлиенте
    Процедура ПометитьНаУдаление(Команда)
        ПометитьДокументыНаУдаление();
        Предупреждение("Были помечены на удаление документы поступления
материалов");
    КонецПроцедуры
```

```
Процедура ПометитьДокументыНаУдаление()
  Запрос = Новый Запрос;
  Запрос.Текст =
    "ВЫБРАТЬ
    |  ПоступлениеМатериалов.Ссылка
    |ИЗ
    |  Документ.ПоступлениеМатериалов КАК ПоступлениеМатериалов
    |ГДЕ
    |  ПоступлениеМатериалов.Комментарий = &Комментарий";

  Запрос.УстановитьПараметр("Комментарий", "Документ создан автоматически");

  Результат = Запрос.Выполнить();

  ВыборкаДетальныеЗаписи = Результат.Выбрать();

  Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
    Документ=ВыборкаДетальныеЗаписи.Ссылка.ПолучитьОбъект();
    Документ.УстановитьПометкуУдаления(Истина);
  КонецЦикла;
КонецПроцедуры
```

Здесь мы, в серверной процедуре ПометитьДокументыНаУдаление(), получаем с помощью запроса список ссылок на документы, реквизит Комментарий которых равен нужному нам значению. После этого в цикле обхода выборки запроса переходим от ссылки на объект к объекту (тип ДокументОбъект) и устанавливаем у объектов пометки удаления.

При завершении серверной процедуры, мы, на клиенте, показываем пользователю окно сообщения, рисунок 14.14.

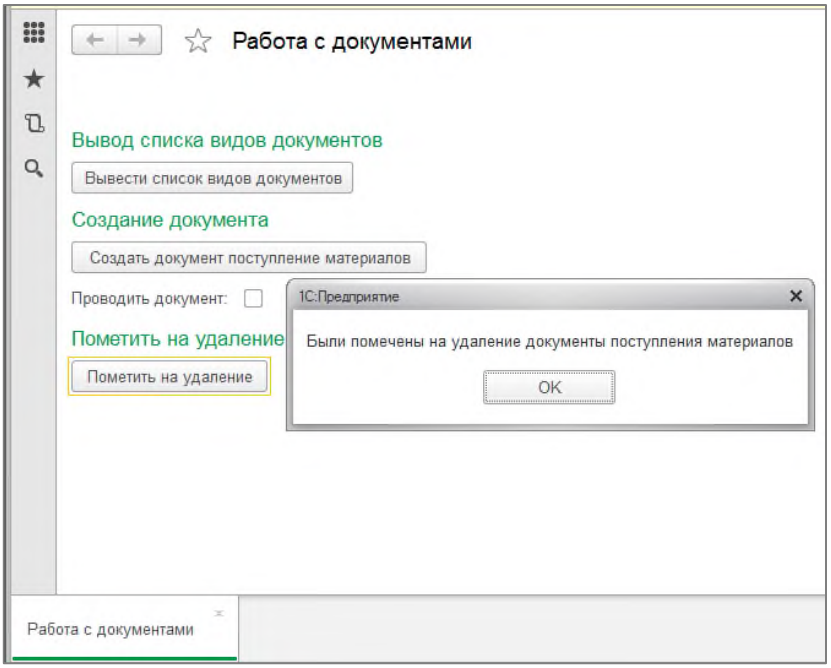


Рисунок 14.14 - Сообщение пользователю о пометке документов на удаление

33. Нашей следующей задачей будет вывод пользователю списка документов за заданный пользователем период. Добавим в форму обработки команду ВыводСпискаДокументовЗаПериод и два реквизита – ДатаНачала и ДатаОкончания – тип Дата, состав даты – Дата и время. Дата документа содержит сведения о дате и времени создания документа, поэтому для выбора периода, в который должны попасть искомые документы, нам понадобятся значения даты с датой и временем.

Решение задачи может выглядеть так:

```
&НаКлиенте
Процедура ВыводСпискаДокументовЗаПериод(Команда)
  Сообщить("Обнаружены следующие документы за период с "+ДатаНачала+" по "+ДатаОкончания);
  ВыводСписка();
КонецПроцедуры

Процедура ВыводСписка()
  Выборка=Документы.ПоступлениеМатериалов.Выбрать(ДатаНачала, ДатаОкончания);
  Пока Выборка.Следующий() Цикл
    Сообщить(Выборка.Ссылка);
  КонецЦикла
КонецПроцедуры
```

Здесь мы пользуемся методом Выбрать с параметрами, устанавливающими дату начала и дату окончания для выборки документов. Полученную выборку перебираем в цикле и сообщаем пользователю о найденных документах, рисунок 14.15.

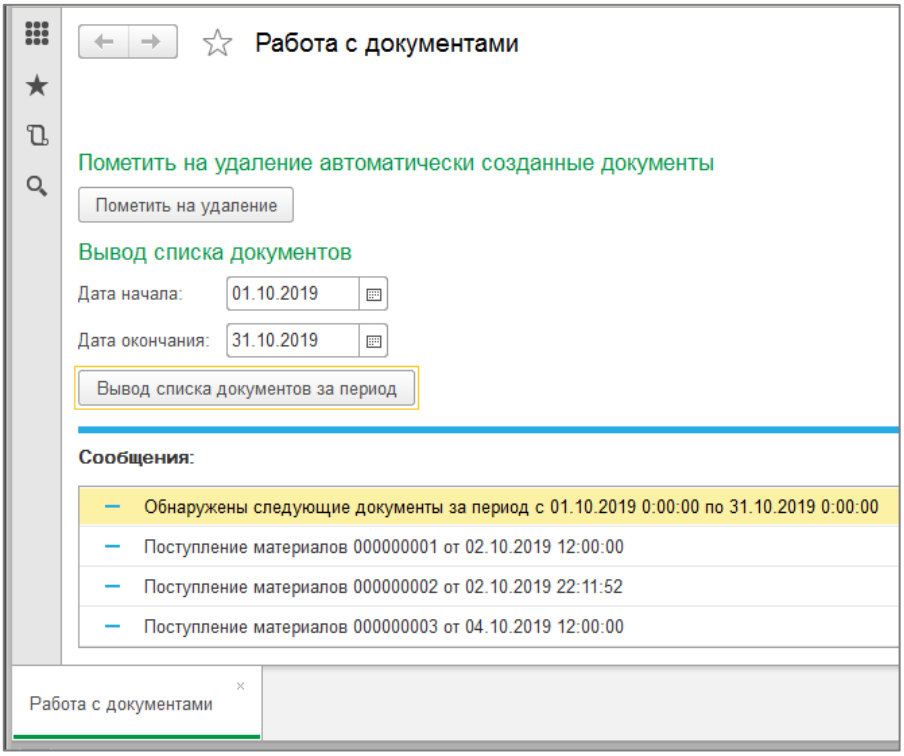


Рисунок 14.15 - Вывод списка документов, принадлежащих периоду, заданному пользователем

писок использованных источников

Интернет-ресурсы:

1. ЭБС «Университетская библиотека online».