Министерство образования Республики Беларусь

Учреждение образования «[Институт информационных технологий](https://www.google.by/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&sqi=2&ved=0CDcQFjAC&url=http%3A%2F%2Fwww.iit-bsuir.by%2F&ei=unOvUpDmGemB4gSZ3IGQDQ&usg=AFQjCNHEr4the3QhkSSjmxbzcNJBZi5-Tg&sig2=GxV3hq7_8t34Csduk0HElg&bvm=bv.57967247,d.bGE&cad=rja) Белорусского государственного университета информатики и радиоэлектроники»

Факультет компьютерных технологий

Кафедра информационных систем и технологий

Дисциплина: Современные платформы программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

«АРМ бухгалтера по ведению учёта основных средств»

БГУИР КП 1-40 01 01 010 ПЗ

Студент: гр. 681072 Иванов К.Е.

Руководитель: Бакунова О.М.

Минск 2019

**СОДЕРЖАНИЕ**

[Введение](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690542) 5

[1 Исследование](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690543) поставленной задачи 7

[1.1 Анализ](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690543) предметной области 7

[1.2 Обзор существующих аналогов](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690544) 7

[1.](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690544)3 Постановка задачи на курсовое проектирование 10

[1.4 Входные данные](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690543) 10

[1.5 Выходные данные](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690543) 10

[1.6 Разработка](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690543) алгоритмов программного средства общая 10

[1.7 Выбор](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690543) инструментов разработки 11

[2 Проектирование программного](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690543) средства 13

[2.1 Разаработка](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690543) диаграммы IDEF0 13

[2.2 Разграничение](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690543) прав доступа программного средства 15

[2.3 Справочная система](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690543) 16

[3 Разработка программного средства 1](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690552)8

[3.1 Разработка спецификаций требований](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690552) 18

[3.2 Разработка архитектуры программного средства](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690552) 18

[3.3 Обоснование выбора языка программирования 2](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690552)0

[3.4 Разработка алгоритмов программного средства укрупненная](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690552) 21

3.5 Построение диаграмм Use Case и Class 21

4 Тестирование программного средства 25

5 Руководство пользователя 32

Заключение 35

Список используемых источников 36

Приложение А Листинг программы 37

Приложение Б Схема алгоритма программы общая 43

**ВВЕДЕНИЕ**

Анализируя сущность автоматизированного рабочего места, специалисты определяют их чаще всего как профессионально-ориентированные малые вычислительные системы, расположенные непосредственно на рабочих местах специалистов и предназначенные для автоматизации их работ.

Автоматизация – это процесс, при котором функции управления и контроля, ранее выполнявшиеся человеком, передаются программному средству. Основная тенденция развития систем автоматизации идет в направлении создания автоматических систем, которые способны выполнять заданные функции или процедуры без участия человека. Роль человека заключается в подготовке исходных данных, выборе алгоритма (метода решения) и анализе полученных результатов.

Для каждого объекта управления нужно предусмотреть автоматизированные рабочие места, соответствующие их функциональному назначению. Однако принципы создания автоматизированного рабочего места должны быть общими: системность, гибкость, устойчивость, эффективность. Согласно принципу системности, автоматизированное рабочее место следует рассматривать как системы, структура которых определяется функциональным назначением. Принцип гибкости означает приспособляемость системы к возможным перестройкам благодаря модульности построения всех подсистем и стандартизации их элементов. Принцип устойчивости заключается в том, что система автоматизированного рабочего места должна выполнять основные функции независимо от воздействия на нее внутренних и внешних возможных факторов. Это значит, что неполадки в отдельных ее частях должны быть легко устранимы, а работоспособность системы – быстро восстановима. Эффективность автоматизированного рабочего места следует рассматривать как интегральный показатель уровня реализации приведенных выше принципов, отнесенного к затратам по созданию и эксплуатации системы. Функционирование автоматизированного рабочего места может дать численный эффект только при условии правильного распределения функций и нагрузки между человеком и машинными средствами обработки информации, ядром которых является электронно-вычислительная машина. Лишь тогда автоматизированное рабочего место станет средством повышения не только производительности труда и эффективности управления, но и социальной комфортности специалистов.

Основная цель курсового проекта максимально облегчить работу специалиста, высвобождая время и усилия, которые ранее расходовались на выполнение рутинных операций сбора данных и сложных расчетов, для творческой научно-обоснованной деятельности в решении профессиональных задач.

Пояснительная записка к курсовому включает в себя пять разделов, содержащих необходимую и достаточную информацию по использованию данного программного продукта, подробное описание программы и вычислительной системы.

Первый раздел «Исследование поставленной задачи» содержит анализ предметной области, обзор существующих аналогов. А также постановку задачи с входными и выходными данными, разработку алгоритмов и выбор инструментов разработки.

Второй раздел «Проектирование программного средства» имеет разработку диаграммы IDEF0, описание разграничения прав доступа программного средства и проектирование справочной системы.

Третий раздел «Разработка программного средства включает в себя разработку спецификации требований, описание разработки архитектуры программного средства, обоснование выбора языка программирования, а также описание построения диаграмм Use Case и Class.

В четвертом разделе «Тестирование программного средства» рассматриваются требования к техническим средствам для проведения тестирования, требования к характеристикам программы применительно к условиям эксплуатации, требования к информационной и программной совместимости. Также описывается порядок проведения функционального тестирования.

В заключении подведены итоги проведенной работы по разработке программного продукта.

**1 ИССЛЕДОВАНИЕ ПОСТАВЛЕННОЙ ЗАДАЧИ**

**1.1 Анализ предметной области**

Занять лидирующее положение на рынке, повысить эффективность работы персонала, создать оптимальную структуру управления - вот первоочередные задачи руководителя предприятия. В бухгалтерской деятельности это особенно важно, что обуславливает широкое применение бухгалтерских пакетов и программ, благодаря внедрению которых повышается оперативность обработки данных и достоверность деловой информации, принимаются более объективные финансовые и управленческие решения.

Бухгалтер выполняет такие действия как работа с отчетностью, ведение административного документооборота по учету основных средств, ведение архивов без ограничения сроков давности и т.д.

Описанные правила предметной области позволяют выявить следующие функции:

* поступлений основных средств;
* ликвидации основных средств;
* реализации основных средств;
* перемещений основных средств;
* анализ счета;
* ведение учёта основных средств.

А также следующие сущности:

* «Сотрудники» – сведения о сотрудниках;
* «Должности» – сведения о должностях;
* «Основные средства» – сведения об основных средствах;
* «Поставщики» – сведения о поставщиках;
* «Клиенты» – сведения о клиентах.

**1.2 Обзор существующих аналогов**

В качестве вариантов автоматизации существуют готовые типовые решения, также предлагается рассмотреть вариант индивидуальной разработки. Готовые решения предназначены для полной автоматизации деятельности кафе.

Широкое использование средств вычислительной техники привело к появлению новых приемов и способов организации учета. Разработаны машиноориентированные формы бухгалтерского учета. Отличительной их особенностью является возможность составления в автоматическом режиме машинограмм, обобщающих учетную информацию в разрезах, необходимых для управления, контроля, анализа, составления бухгалтерской отчетности и другой документации.

На текущий момент существует довольно большое количество универсальных комплексов ведения бухгалтерского учета, которые предназначены не только для автоматизации учета труда и ЗП на МП, но и для ведения всего бухгалтерского учета, как на малых, так и на крупных предприятиях.

Вот лишь несколько из них:

* Фолио; АО "Центр экономических компьютерных программ ФОЛИО".
* Инфин-бухгалтерия; Аудиторская компания "Инфин".
* AUBI; Фирма "О'стрим".
* 1С Бухгалтерия.

Подробно будут рассмотрены лишь два из них. Это такие системы, как AUBI и 1С (для Windows).

Программа 1С: Бухгалтерия является универсальной бухгалтерской программой и предназначена для ведения синтетического и аналитического бухгалтерского учета по различным разделам.

Аналитический учет ведется по объектам аналитического учета (субконто) в натуральном и стоимостном выражениях. Программа предоставляет возможность ручного и автоматического ввода проводок. Все проводки заносятся в журнал операций. При просмотре проводок в журнале операций их можно ограничить произвольным временным интервалом, группировать и искать по различным параметрам проводок.

Кроме журнала операций программа поддерживает несколько списков справочной информации (справочников):

* план счетов;
* список видов объектов аналитического учета;
* списки объектов аналитического учета (субконто);
* констант и т.д.

На основании введенных проводок может быть выполнен расчет итогов. Итоги могут выводиться за квартал, год, месяц и за любой период, ограниченный двумя датами. Расчет итогов может выполняться по запросу и одновременно с вводом проводок (в последнем случае не требуется пересчет).

После расчета итогов программа формирует различные ведомости:

* сводные проводки;
* оборотно-сальдовую ведомость;
* оборотно-сальдовую ведомость по объектам аналитического учета;
* карточка счета;
* карточка счета по одному объекту аналитического учета;
* анализ счета (аналог главной книги);
* анализ счета по датам;
* анализ счета по объектам аналитического учета;
* анализ объекта аналитического учета по всем счетам;
* карточка объекта аналитического учета по всем счетам;
* журнальный ордер.

Кроме того, программа имеет функции сохранения резервной копии информации и режим сохранения в архиве текстовых документов.

«АУБИ» – это зарегистрированное название интегрированной программной системы «Автоматизации Бухгалтерского Учета» малых, средних и больших предприятий. Аббревиатура названия комплекса “АУБИ” построена из ряда букв, входящих в приведенную выше фразу, заключенную в апострофы. Таким образом, сделана попытка отразить истинное предназначение программы.

“АУБИ” может быть с успехом использовано для автоматизации бухгалтерского учета предприятий различного рода деятельности. Программный комплекс представляет одинаковый интерес как для торговых (коммерческих) структур, так и для производственных предприятий.

Гибкая система программы позволяет настраивать “АУ-БИ” на нужды конкретного пользователя. При этом бухгалтер каждого предприятия, исходя из своих собственных потребностей, имеет возможность сформировать план счетов; информационные справочники, содержащие названия предприятий-партнеров и их банковские реквизиты; список материально ответственных лиц и т.д.

В зависимости от специфики деятельности предприятия “АУБИ” позволяет вести учет следующих элементов бухгалтерского производства:

* учет материалов (склад);
* учет малоценных и быстроизнашивающихся материалов (МБП) на складе и в эксплуатации;
* основные средства;
* учет кассовых операций - формирование приходных и расходных кассовых ордеров, ведение кассовой книги;
* учет банковских операций - платежных поручений, требований и реестров;
* учет счетов;
* ведение журнала хозяйственных операций;
* ведение главной бухгалтерской книги;
* формирование шахматной и оборотной ведомостей;
* формирование различных ведомостей аналитического учета и т. д.

Журнал хозяйственных операций является для “АУБИ” поистине основной информационной базой, используя которую, программа способна формировать множество отчетных документов по синтетическому и аналитическому учету.

Все сформированные “АУБИ” отчеты могут быть сохранены на жестком диске в виде ASKII файлов или выведены непосредственно на принтер. В случае сохранения выходных форм на жестком диске пользователь имеет возможность просматривать, корректировать и выводить на печать все документы, используя для этого имеющиеся у него стандартные программные средства (утилиты DOS, текстовые редакторы и т. д.).

# 1.3 Постановка задачи

Задачей курсового проекта является создание автоматизированного рабочего места для бухгалтера по ведению учёта основных средств в котором необходимо реализовать хранение информации о основных средствах, их движение, учёт и всю необходимую бумажную документацию.

Необходимо разработать программное средство для автоматизации работы бухгалтера, в котором необходимо реализовать:

* создание документов по приходу и реализации основных средств;
* создание справочников для хранения информации о основных средствах;
* хранение данных в регистрах;
* формирование отчетов по реализации основных средств;
* контроль и просмотр основных средств.

[**1.4 Входные данные**](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690543)

Входные данные – это величины, которые задаются до начала работы алгоритма. Входными данными для работы программы являются:

* пароль администратора при авторизации в программу;
* информация о расходных накладных;
* информация о приходных накладных;
* информация о поставщиках;
* информация о основных средствах;
* информация о сотрудниках;
* информация о компании;
* параметры поиска.

[**1.5 Выходные данные**](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690543)

Выходными данными для разрабатываемого программного средства являются:

* обновленная информация, хранимая в справочниках и документах;
* результаты операций добавления и изменения информации в таблицах;
* результаты поиска по заданным параметрам;
* сформированные отчеты.

[**1.6 Разработка**](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690543) **алгоритмов программного средства общая**

Алгоритм работы программы начинается с запуска приложения.

Следующим шагом начинается подключение к базе данных. В случае возникновения ошибки – например, если база данных не была найдена, то выводится сообщение об ошибке. Если база данных была найдена и успешно подключена, то переходим к следующему шагу.

Появляется окно с выбором пользователя, так как вход в программу можно осуществить с помощью таких ролей как: администратора и бухгалтера.

При первом или новом входе потребуется авторизация с вводом имя и пароля. Если авторизоваться не удалось, то выводится сообщение об ошибке и предлагается ввести пароль повторно.

После успешной авторизации пользователь может взаимодействовать с программой, получая различные данные из операций по работе с базой данных на свои запросы.

Схема алгоритма работы программы представлена в приложении Б.

[**1.7 Выбор**](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690543) **инструментов разработки**

В настоящее время на рынке программных продуктов предлагаются различные информационные системы, предназначенные для решения задач бухгалтерского, налогового, складского учета и торговой деятельности предприятия. Среди них выделяется «1С:Предприятие 8» – комплексная система автоматизации управления предприятиям, в которой реализованы в полном объеме все необходимые функции, так же данная система отличается высокой производительностью и мобильностью. Она постоянно дорабатывается и модернизируется, приобретая всё больше дополнительного функционала и обладает всеми возможностями, необходимыми для ведения учета на любом предприятии.

При разработке платформы «1С: Предприятие 8» ее создатели сразу ориентировались на получение целостной базовой платформы, которая будет использоваться для построения разнообразных прикладных решений. При этом данные прикладные решения могут создаваться не только специалистами фирмы «1С», но и множеством других разработчиков, которые являются экспертами в тех или иных отраслях производственной деятельности. Поэтому платформа изначально проектировалась как тиражируемый продукт.

В принципах построения платформы «1С:Предприятие 8» изначально заложена строгая ориентация на разработку прикладного решения на основе определенной модели.

Модель – это вся идеология построения прикладного решения. Здесь и способы построения структур данных, и типы связей между данными, и принципы управления данными, и форматы описания бизнес-логики, и способы связи данных с объектами интерфейса, и разделение функциональности по уровням системы, а также многое другое.

Важно, что все приложения должны следовать принятой модели. Этим обеспечивается единообразие и предсказуемость их поведения. Таким образом, разработчик, который желает отразить в прикладном бизнес-решении специфику той или иной предметной области, имеет вполне определенный набор способов реализации задачи средствами, заложенными в платформу. Безусловно, такой подход в чем-то ограничивает свободу разработчика, но в то же время он защищает от множества ошибок и позволяет довольно быстро получить работоспособное решение, которое в дальнейшем сможет развиваться и поддерживаться не только им, но и другими специалистами. Кроме того, наличие единой модели принципиальным образом сказывается и на простоте освоения системы.

Платформа обеспечивает работу конфигурации и позволяет вносить в нее изменения или создавать собственную конфигурацию. Существует одна платформа – «1С:Предприятие 8», и множество конфигураций. Для функционирования какого-либо прикладного решения всегда необходима платформа и какая-либо (одна) конфигурация. Сама по себе платформа не может выполнять никаких задач автоматизации, так как она создана для обеспечения работы какой-либо конфигурации. То же самое с конфигурацией: чтобы выполнить те задачи, для которых она создана, необходимо наличие платформы, которая и управляет ее работой.

Тут и выходит на передний план конфигурируемость системы, поскольку платформа, помимо управления работы конфигурацией, содержит средства, позволяющие вносить изменения в используемую конфигурацию. Более того, платформа позволяет создавать свою собственную конфигурацию «с нуля», если по каким-либо причинам использование типовой конфигурации представляется нецелесообразным.

[**2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО**](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690543) **СРЕДСТВА**

[**2.1 Разработка**](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690543) **диаграммы IDEF0**

Известная сегодня уже не только в узких кругах аббревиатура IDEF0 является первой методологией, стандартизирующей работу над бизнес-процессами. Она была разработана в середине прошлого века в рамках аэрокосмического проекта в США и, показав свою эффективность, стала федеральным стандартом. В нашей стране в 2000 году подготовлен документ «Методология функционального моделирования IDEF0. Руководящий документ», но как стандарт он так и не был утвержден. Хотя это не помешало данной методологии стать в нашей стране одним из наиболее популярных инструментов графического моделирования бизнес-процессов.

Графический стандарт IDEF0 является частью методологии SADT (Structured Analysis and Design Technique – метод структурного анализа и проектирования). IDEF – это сокращение от ICAM Definition, а ICAM образовано от Integrated Computer Aided Manufacturing, что переводится как интегрированная компьютеризация производства. Методология SADT – это целое семейство из 15 разных моделей, которые в комплексе должны были позволить исследовать структуру, параметры и характеристики производственно-технических и организационно-экономических систем.

Центральным элементом модели IDEF0 является функция, которая на схеме отображается в виде функционального блока – прямоугольника, внутри которого указано действие в форме отглагольного существительного. Действие может быть очень разным по масштабу – от деятельности компании вообще и до конкретной манипуляции в частности.

Независимо от масштаба действий все функции отображаются единообразно и обязательно содержат 4 ключевых потока, которые жестко закреплены за сторонами функционального блока:

* слева: входы или используемые ресурсы для выполнения функции;
* справа: выходы или результаты выполнения функции;
* сверху: управляющие воздействия, которые определяют, как и сколько нужно произвести результатов;
* снизу: механизмы, которые отражают, кто и с помощью чего должен выполнить эту работу.

Такой подход позволяет немного сэкономить на пояснениях в схемах и добиться однозначности в отображении потоков, что придает стройности всей модели.

Для построения функциональной модели методология IDEF0 требует соблюдать следующие правила.

* Входы – это ресурсы, которые переносят свою стоимость в выходы полностью, то есть расходуются на создание результата полностью, а механизмы – это ресурсы, которые переносят свою стоимость только частично (оборудование – через амортизацию, а люди – через заработную плату).
* Управление – это необходимый элемент модели, так как он привязывает все действия к системе регламентов компании, четко обозначая, какие правила и требования должны быть соблюдены в процессе выполнения функции. Часто к этому потоку относятся формально, но при этом схема теряет строгость, а иногда даже смысл.
* У каждого функционального блока должна быть как минимум одна стрелка с каждой стороны (так как не может быть работы без ресурсов или результатов, а также неполной будет инструкция без исполнителя или инструкции).

Функциональная модель IDEF0 программного средства, которая представлена на рисунках 2.1 и 2.2 включает в себя следующие этапы.

Имеются основные средства, которые поступают бухгалтеру. Бухгалтер обрабатывает заявку поступления основных средств, оформляет приходные документы. Всю информацию обрабатывает с последующим сохранением в таблицы базы данных.

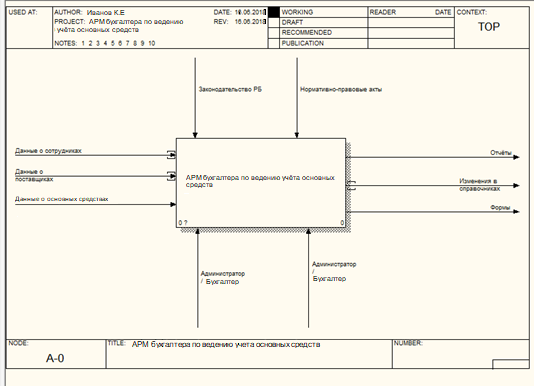
****

Рисунок 2.1 – Диаграмма А-0

Входными данными для базы данных являются – данные о основных средствах, данные о сотрудниках и поставщиках.

Выходными данными являются изменения в справочниках, а также обновленные отчеты и формы.

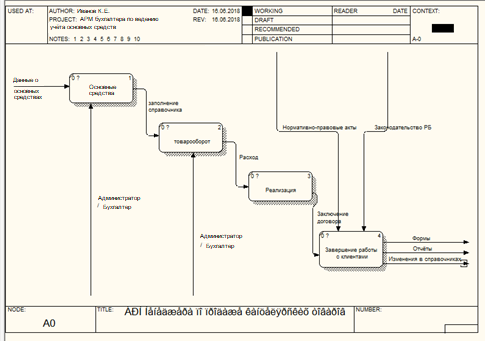


Рисунок 2.2 – Диаграмма А0

[**2.2 Разграничение**](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690543) **прав доступа программного средства**

Предусмотреть в программе разграничение прав доступа по ролям: роль администратора и роль бухгалтера.

Роль пользователя определяет и устанавливает администратор. Администратором по умолчанию является владелец главный бухгалтер компании.

Роль администратора не имеет ограничений, роль бухгалтера имеет следующие ограничения:

* запрещено удаление любой информации из базы;
* ограничен доступ к некоторым документам базы.

На рисунке 2.3 представлены все роли и список пользователей, а также на рисунке 2.4 показан перечень прав доступа, которые есть возможность корректировать для определенных пользователей

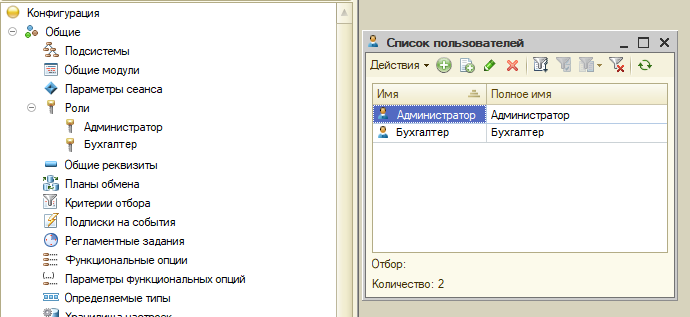


Рисунок 2.3 – Роли и список пользователей

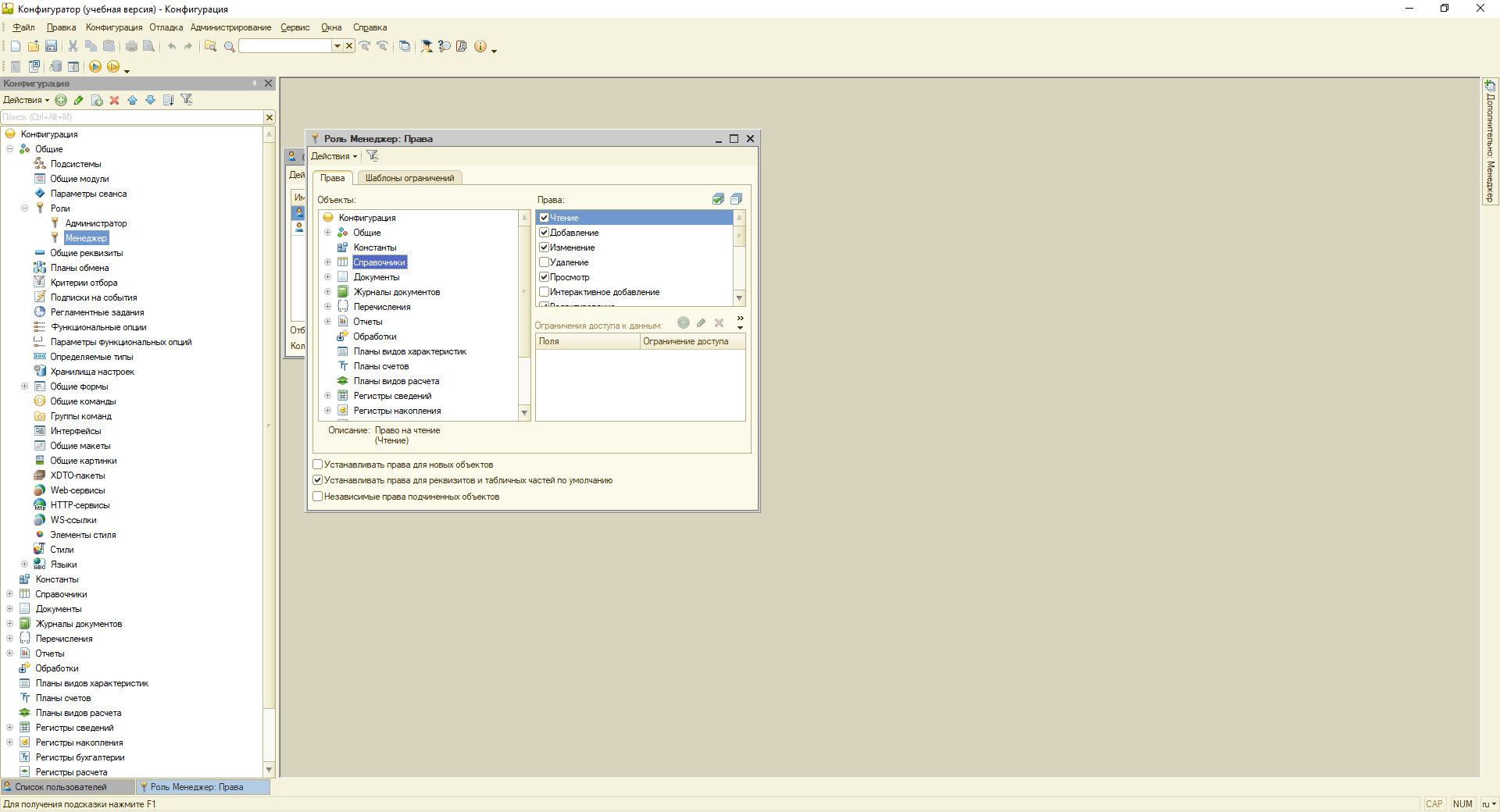


Рисунок 2.4 – Роли и список пользователей

Доступ к програмнному средству «АРМ бухгалтера по ведению учёта основных средств» имеет только бухгалтерский отдел и администратор.

**2.3 Справочная система**

В системе 1С: Предприятие существует механизм, позволяющий пользователю во время работы с программой получить справочную информацию по применению различных функций и режимов программы.

В главном меню 1С: Предприятие для вызова помощи (справки) существует подменю «Справка».

Встроенная система получения справочной информации.

Какую-либо информацию по работе с системой 1С: Предприятие и по конкретным объектам и режимам можно получить, используя систему справочной информации программы.

Данная система вызывается в любой момент вызовом пункта меню «Справка» или нажатием клавиши F1. При выполнении данной процедуры вызывается раздел справочной информации, соответствующий тому режиму, в котором сейчас работает пользователь.

«Справка: Содержание» или клавиша Shift+F1 – позволяют обратиться к общему содержанию справочной информации, из которого можно получить информацию по различным функциям программы.

В справке даны ответы на типичные вопросы, возникающие при работе с приложением, что, несомненно, должно помочь при освоении программного продукта.

Копия экрана справочной системы представлена на рисунке 2.5.

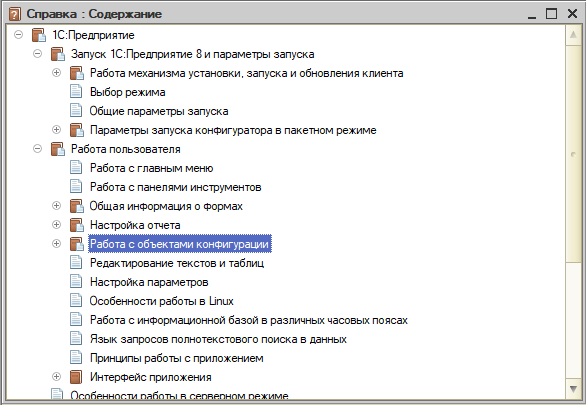


Рисунок 2.5 – Стандартная справочная система

Система справочной информации представляет собой всплывающие подсказки, которые возникают при наведении на элемент управления или поле ввода.

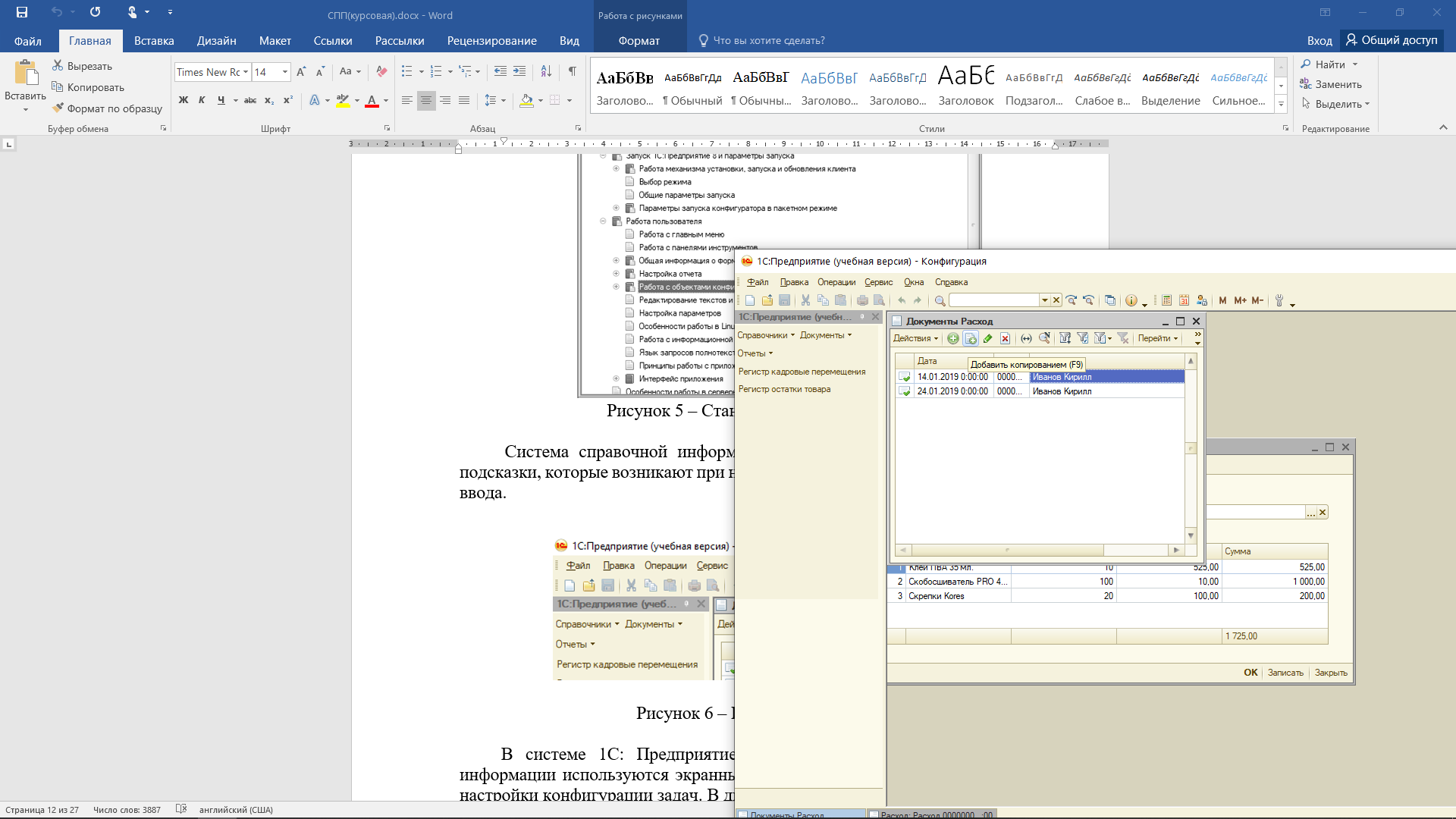


Рисунок 2.6 – Всплывающие подсказки

В системе 1С: Предприятие для ввода или просмотра различной информации используются экранные формы, которые создаются в процессе настройки конфигурации задач. В диалогах ведется работа:

* по вводу и редактированию документов;
* в диалогах могут редактироваться элементы справочников;
* перед формированием отчета выдается диалог для установки параметров его формирования и др.

Внешний вид подобных диалогов полностью задается в конфигурации. Однако система 1С: Предприятие содержит средства создания пользовательской подсказки, в которых при настройке конфигурации задачи описан смысл его реквизитов (F1).

Чтобы получить подсказку по конкретным реквизитам диалога, следует поместить указатель мыши над нужным реквизитом и немного подождать.

**3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**

**3.1 Разработка спецификаций требований**

Спецификация требований состоит из требований по созданию справочников, документов и регистров.

Справочники – это прикладные объекты конфигурации. Они позволяют хранить в информационной базе данные, имеющие одинаковую структуру и списочный характер. Это может быть, например, список сотрудников, перечень основных средств, список поставщиков или покупателей.

В данном курсовом проекте будут реализованы следующие справочники:

* Сотрудники – справочник, для хранения информации о всех сотрудниках работающих в организации;
* Должности – справочник, в котором хранится информация о всех должностях;
* Вид основных средств – справочник, в котором хранится информация о всех основных средствах;
* Основные средства – справочник, который содержит более подробную информацию о основных средствах;
* Поставщики – справочник, отображающий список поставщиков и их адреса;
* Клиенты – справочник, для хранения информации о всех клиентах;

**3.2 Разработка архитектуры программного средства**

Архитектура программного обеспечения – совокупность важнейших решений об организации программной системы. Архитектура включает:

* выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
* соединение выбранных элементов структуры и поведения во всё более крупные системы;
* архитектурный стиль, который направляет всю организацию – все элементы, их интерфейсы, их сотрудничество и их соединение.

Документирование архитектуры программного обеспечения упрощает процесс коммуникации между разработчиками, позволяет зафиксировать принятые проектные решения и предоставить информацию о них эксплуатационному персоналу системы, повторно использовать компоненты и шаблоны проекта в других.

Общепринятого определения «архитектуры программного обеспечения» не существует. Так, сайт Software Engineering Institute приводит более 150 определений этого понятия.

Архитектура ПО обычно содержит несколько видов, которые аналогичны различным типам чертежей в строительстве зданий. В онтологии, установленной ANSI / IEEE 1471-2000, виды являются экземплярами точки зрения, где точка зрения существует для описания архитектуры с точки зрения заданного множества заинтересованных лиц.

Архитектурный вид состоит из 2 компонентов:

* элементы;
* отношения между элементами.

Архитектурные виды можно поделить на 3 основных типа:

* модульные виды – показывают систему как структуру из различных программных блоков;
* компоненты-и-коннекторы – показывают систему как структуру из параллельно запущенных элементов (компонентов) и способов их взаимодействия (коннекторов);
* размещение – показывает размещение элементов системы во внешних средах;
* Примеры модульных видов:
* декомпозиция – состоит из модулей в контексте отношения «является подмодулем»;
* использование – состоит из модулей в контексте отношения «использует» (т.е. один модуль использует сервисы другого модуля);
* вид уровней – показывает структуру, в которой связанные по функциональности модули объединены в группы (уровни);
* вид классов/обобщений – состоит из классов, связанные через отношения «наследуется от» и «является экземпляром».

Примеры видов компонентов-и-коннекторов:

* процессный вид – состоит из процессов, соединённых операциями коммуникации, синхронизации и/или исключения;
* параллельный вид – состоит из компонентов и коннекторов, где коннекторы представляют собой «логические потоки»;
* вид обмена данными – состоит из компонентов и коннекторов, которые создают, сохраняют и получают постоянные данные;
* вид клиент-сервер – состоит из взаимодействующих клиентов и серверов, а также коннекторов между ними (например, протоколов и общих сообщений).

Примеры видов размещения:

* развертывание – состоит из программных элементов, их размещения на физических носителях и коммуникационных элементов;
* внедрение – состоит из программных элементов и их соответствия файловым структурам в различных средах (разработческой, интеграционной и т.д.);
* распределение работы – состоит из модулей и описания того, кто ответственен за внедрение каждого из них.

Хотя было разработано несколько языков для описания архитектуры программного обеспечения, в настоящий момент нет согласия по поводу того, какой набор видов должен быть принят в качестве эталона. В качестве стандарта «для моделирования программных систем (и не только)» был создан язык UML.

**3.3 Обоснование выбора языка программирования**

Встроенный язык программирования 1С:Предприятие – язык программирования, который используется в семействе программ «1С:Предприятие». Данный язык является интерпретируемым языком высокого уровня. Интерпретация текста программного модуля в байт-код выполняется в момент обращения к этому модулю в процессе работы, таким образом обычно интерпретируется только часть текстов программных модулей (в версиях 7.7 и старше). Начиная с версии 8.2 модули компилируются.

Средой исполнения языка является программная платформа «1С:Предприятие». Визуальная среда разработки («Конфигуратор») является неотъемлемой частью пакета программ «1С:Предприятие».

Диалекты языка для платформ 1С 7 версий (7.0, 7.5, 7.7) совместимы «снизу вверх» с незначительными исключениями. Языки для платформ 1С:7х и 1С:8х совместимы по основным операторам, но значительно отличаются в работе с прикладными объектами, вследствие чего перенос кода из 1С:7х в 1С:8х не имеет смысла.

Встроенный язык 1С:8 наиболее подобен по своему синтаксису языку Visual Basic.

Платформой предоставляется фиксированный набор базовых классов, ориентированных на решение типовых задач прикладной области:

* константа,
* справочник,
* документ,
* журнал документов,
* перечисление,
* отчёт,
* обработка
* план счетов и др.

На основании базовых классов средствами визуального конфигурирования можно создавать любое количество порождённых классов (возможность определить новый класс программно – отсутствует). Допускается только одна явная ступень наследования классов. Как правило, объекты порождённых классов представляют собой записи (или некоторые наборы записей) в базе данных. Такие классы образуют «Дерево метаданных». В терминах встроенного языка программирования 1С такие классы называются объектами метаданных.

Основными видами объектов метаданных являются: Справочники, Документы, Отчёты, Обработки, Планы видов характеристик, Планы счетов, Планы видов расчёта, Регистры сведений, Регистры накопления, Регистры расчёта, Бизнес-процессы, Задачи.

Поддерживаются русский и английский синтаксис команд.

Проекты на встроенном языке 1С:Предприятия называются конфигурациями. Распространение (продажа) и внедрение таких конфигураций – это основная коммерческая деятельность фирм-партнёров 1С.

Рабочее название языка – «1Сик» («одинэсик») – очень быстро исчезло из официальных источников. Сейчас при упоминании этого языка в письменных документах нужно писать 1С Язык программирования. Впрочем, часто этот язык называют «встроенный язык», в контексте обсуждения 1С:Предприятия.

[**3.4 Разработка алгоритмов программного средства укрупненная**](file:///C:\Users\Алексей\Desktop\Курсовой%20проект\Мое\2.%20Содержание.doc#_Toc263690552)

При моделировании поведения системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой операции в предыдущем состоянии. Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия, а дугами - переходы от одного состояния действия к другому. На диаграмме деятельности отображается логика или последовательность перехода от одной деятельности к другой, при этом внимание фиксируется на результате деятельности.

Компонентами диаграммы деятельности являются:

* состояния действия;
* переходы;
* дорожки;
* объекты.

**3.5 Построение диаграмм Use Case и Class**

Диаграмма вариантов использования (представлена на рисунке 3.1)

Визуальное моделирование в UML можно представить, как процесс поуровневого спуска от наиболее обшей и абстрактной концептуальной модели исходной системы к логической, а затем и к физической модели соответствующей программной системы. Для достижения этих целей вначале строится модель в форме так называемой диаграммы вариантов использования (use case diagram), которая описывает функциональное назначение системы. Диаграмма вариантов использования является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

Диаграммы вариантов использования определяют общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы; формулируют общие требования к функциональному поведению проектируемой системы.

Основными компонентами диаграммы вариантов использования являются:

* варианты использования;
* актеры;
* интерфейсы;
* примечания;
* отношения.

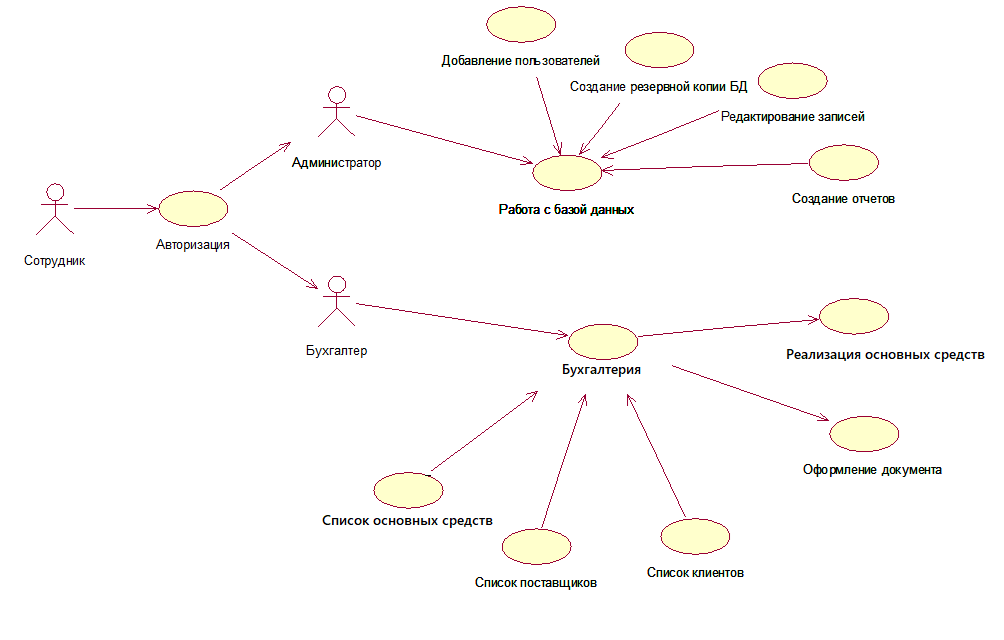
****

Рисунок 3.1 – Use Сase диаграмма

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером (actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик. В свою очередь, вариант использования (use case) служит для описания сервисов, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие актеров с системой. Отдельные компоненты диаграммы могут быть заключены в прямоугольник, который обозначает проектируемую систему в целом.

Диаграмма классов (представлена на рисунке 3.2)

Диаграммы классов используются при моделировании программных средств наиболее часто. Они являются одной из форм статического описания системы с точки зрения ее проектирования, показывая ее структуру. Диаграмма классов не отображает динамическое поведение объектов изображенных на ней классов. На диаграммах классов показываются классы, интерфейсы и отношения между ними.

Класс – это основной строительный блок программного средства. Это понятие присутствует и в объектно-ориентированных языках программирования, то есть между классами UML и программными классами есть соответствие, являющееся основой для автоматической генерации программных кодов или для выполнения реинжиниринга. Каждый класс имеет название, атрибуты и операции. Класс на диаграмме показывается в виде прямоугольника, разделенного на 3 области. В верхней содержится название класса, в средней – описание атрибутов (свойств), в нижней – названия операций – услуг, предоставляемых объектами этого класса.

Атрибуты класса определяют состав и структуру данных, хранимых в объектах этого класса. Каждый атрибут имеет имя и тип, определяющий, какие данные он представляет. При реализации объекта в программном коде для атрибутов будет выделена память, необходимая для хранения всех атрибутов, и каждый атрибут будет иметь конкретное значение в любой момент времени работы программы. Объектов одного класса в программе может быть сколь угодно много, все они имеют одинаковый набор атрибутов, описанный в классе, но значения атрибутов у каждого объекта свои и могут изменяться в ходе выполнения программы.

Для каждого атрибута класса можно задать видимость (visibility). Эта характеристика показывает, доступен ли атрибут для других классов. В UML определены следующие уровни видимости атрибутов:

* открытый (public) – атрибут виден для любого другого класса (объекта);
* защищенный (protected) – атрибут виден для потомков данного класса;
* закрытый (private) – атрибут не виден внешними классами (объектами) и может использоваться только объектом, его содержащим.

Последнее значение позволяет реализовать свойство инкапсуляции данных. Например, объявив все атрибуты класса закрытыми, можно полностью скрыть от внешнего мира его данные, гарантируя отсутствие несанкционированного доступа к ним. Это позволяет сократить число ошибок в программе. При этом любые изменения в составе атрибутов класса никак не скажутся на остальной части ПС.

Также на диаграмме указывается особый тип логических отношений между сущностями, показанных на диаграммах классов и объектов – их взаимосвязь. Обычно показываются ассоциации и обобщения, редко присутствует агрегация, композиция, реализация и зависимость.

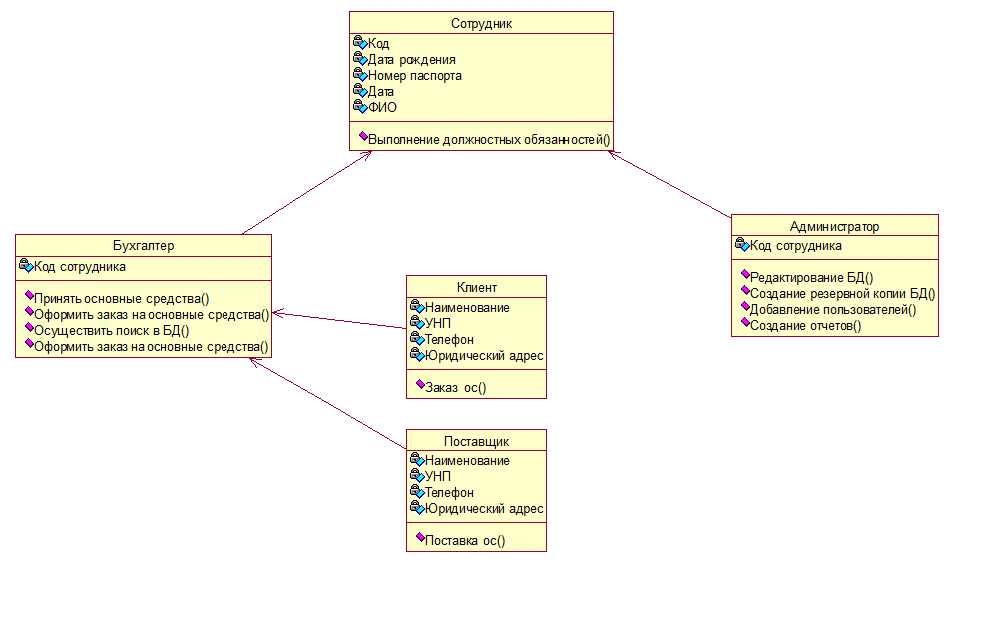


Рисунок 3.2 – Диаграмма классов

Все классы позволяют создавать, просматривать, редактировать и удалять информацию, хранящуюся в них.

Целью создания диаграммы классов является графическое представление статической структуры декларативных элементов системы (классов, типов и т. п.) Она содержит в себе также некоторые элементы поведения (например — операции), однако их динамика должна быть отражена на диаграммах других видов (диаграммах коммуникации, диаграммах состояний). Для удобства восприятия диаграмму классов можно также дополнить представлением пакетов, включая вложенные.

**4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определенным образом (ISO/IEC TR 19759:2005).

В разное время и в различных источниках тестированию давались различные определения, в том числе:

* процесс выполнения программы с целью нахождения ошибок;
* интеллектуальная дисциплина, имеющая целью получение надежного программного обеспечения без излишних усилий на его проверку;
* техническое исследование программы для получения информации о её качестве с точки зрения определенного круга заинтересованных лиц;
* проверка соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выполненных определенным образом;
* процесс наблюдения за выполнением программы в специальных условиях и вынесения на этой основе оценки каких-либо аспектов её работы;
* процесс, имеющий целью выявление ситуаций, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации;
* процесс, содержащий в себе все активности жизненного цикла, как динамические, так и статические, касающиеся планирования, подготовки и оценки программного продукта и связанных с этими результатами работ с целью определить, что они соответствуют описанным требованиям, показать, что они подходят для заявленных целей и для определения дефектов.

Первые программные системы разрабатывались в рамках программ научных исследований или программ для нужд министерств обороны. Тестирование таких продуктов проводилось строго формализованно с записью всех тестовых процедур, тестовых данных, полученных результатов. Тестирование выделялось в отдельный процесс, который начинался после завершения кодирования, но при этом, как правило, выполнялось тем же персоналом.

В 1960-х много внимания уделялось «исчерпывающему» тестированию, которое должно проводиться с использованием всех путей в коде или всех возможных входных данных. Было отмечено, что в этих условиях полное тестирование программного обеспечения невозможно, потому что, во-первых, количество возможных входных данных очень велико, во-вторых, существует множество путей, в-третьих, сложно найти проблемы в архитектуре и спецификациях. По этим причинам «исчерпывающее» тестирование было отклонено и признано теоретически невозможным.

В начале 1970-х годов тестирование программного обеспечения обозначалось как «процесс, направленный на демонстрацию корректности продукта» или как «деятельность по подтверждению правильности работы программного обеспечения».

В зарождавшейся программной инженерии верификация ПО значилась как «доказательство правильности». Хотя концепция была теоретически перспективной, на практике она требовала много времени и была недостаточно всеобъемлющей. Было решено, что доказательство правильности – неэффективный метод тестирования программного обеспечения. Однако, в некоторых случаях демонстрация правильной работы используется и в наши дни, например, приёмо-сдаточные испытания. Во второй половине 1970-х тестирование представлялось как выполнение программы с намерением найти ошибки, а не доказать, что она работает.

Успешный тест – это тест, который обнаруживает ранее неизвестные проблемы. Данный подход прямо противоположен предыдущему. Указанные два определения представляют собой «парадокс тестирования», в основе которого лежат два противоположных утверждения: с одной стороны, тестирование позволяет убедиться, что продукт работает хорошо, а с другой – выявляет ошибки в программах, показывая, что продукт не работает. Вторая цель тестирования является более продуктивной с точки зрения улучшения качества, так как не позволяет игнорировать недостатки программного обеспечения.

В 1980-е годы тестирование расширилось таким понятием, как предупреждение дефектов. Проектирование тестов – наиболее эффективный из известных методов предупреждения ошибок. В это же время стали высказываться мысли, что необходима методология тестирования, в частности, что тестирование должно включать проверки на всем протяжении цикла разработки, и это должен быть управляемый процесс.

В ходе тестирования надо проверить не только собранную программу, но и требования, код, архитектуру, сами тесты. «Традиционное» тестирование, существовавшее до начала 1980-х, относилось только к скомпилированной, готовой системе (сейчас это обычно называется системное тестирование), но в дальнейшем тестировщики стали вовлекаться во все аспекты жизненного цикла разработки. Это позволяло раньше находить проблемы в требованиях и архитектуре и тем самым сокращать сроки и бюджет разработки. В середине 1980-х появились первые инструменты для автоматизированного тестирования. Предполагалось, что компьютер сможет выполнить больше тестов, чем человек, и сделает это более надёжно. Поначалу эти инструменты были крайне простыми и не имели возможности написания сценариев на скриптовых языках.

В начале 1990-х годов в понятие «тестирование» стали включать планирование, проектирование, создание, поддержку и выполнение тестов и тестовых окружений, и это означало переход от тестирования к обеспечению качества, охватывающего весь цикл разработки программного обеспечения. В это время начинают появляться различные программные инструменты для поддержки процесса тестирования: более продвинутые среды для автоматизации с возможностью создания скриптов и генерации отчетов, системы управления тестами, ПО для проведения нагрузочного тестирования. В середине 1990-х годов с развитием Интернета и разработкой большого количества веб-приложений особую популярность стало получать «гибкое тестирование» (по аналогии с гибкими методологиями программирования).

В 2000-х появилось ещё более широкое определение тестирования, когда в него было добавлено понятие «оптимизация бизнес-технологий» Основной подход заключается в оценке и максимизации значимости всех этапов жизненного цикла разработки программного обеспечения для достижения необходимого уровня качества, производительности, доступности.

Существует несколько признаков, по которым принято производить классификацию видов тестирования. Обычно выделяют следующие:

* по объекту тестирования:
  + функциональное тестирование;
  + тестирование производительности:
    - нагрузочное тестирование;
    - стресс-тестирование;
    - тестирование стабильности;
  + конфигурационное тестирование;
  + юзабилити-тестирование;
  + тестирование интерфейса пользователя;
  + тестирование безопасности;
  + тестирование локализации;
  + тестирование совместимости;
* по знанию внутреннего строения системы:
  + тестирование чёрного ящика;
  + тестирование белого ящика;
  + тестирование серого ящика;
* по степени автоматизации:
  + ручное тестирование;
  + автоматизированное тестирование;
  + полуавтоматизированное тестирование;
* по степени изолированности:
  + тестирование компонентов;
  + интеграционное тестирование;
  + системное тестирование;
* по времени проведения тестирования:
  + альфа-тестирование:
    - дымовое тестирование;
    - тестирование новой функции;
    - подтверждающее тестирование;
    - регрессионное тестирование;
    - приёмочное тестирование;
  + бета-тестирование;
* по признаку позитивности сценариев:
  + позитивное тестирование;
  + негативное тестирование;
* по степени подготовленности к тестированию:
  + тестирование по документации (формальное тестирование);
  + интуитивное тестирование.

Также существуют различные уровни тестирования:

* Тестирование компонентов. Тестируется минимально возможный для тестирования компонент, например, отдельный класс или функция. Часто тестирование компонентов осуществляется разработчиками программного обеспечения.
* Интеграционное тестирование. Тестируются интерфейсы между компонентами, подсистемами или системами. При наличии резерва времени на данной стадии тестирование ведётся итерационно, с постепенным подключением последующих подсистем.
* Системное тестирование. Тестируется интегрированная система на её соответствие требованиям.
* Альфа-тестирование. Имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальными пользователями/заказчиком. Чаще всего альфа-тестирование проводится на ранней стадии разработки продукта, но в некоторых случаях может применяться для законченного продукта в качестве внутреннего приёмочного тестирования. Иногда альфа-тестирование выполняется под отладчиком или с использованием окружения, которое помогает быстро выявлять найденные ошибки. Обнаруженные ошибки могут быть переданы тестировщикам для дополнительного исследования в окружении, подобном тому, в котором будет использоваться программа.
* Бета-тестирование. В некоторых случаях выполняется распространение предварительной версии (в случае проприетарного программного обеспечения иногда с ограничениями по функциональности или времени работы) для некоторой большей группы лиц с тем, чтобы убедиться, что продукт содержит достаточно мало ошибок. Иногда бета-тестирование выполняется для того, чтобы получить обратную связь о продукте от его будущих пользователей.

Часто для свободного и открытого программного обеспечения стадия альфа-тестирования характеризует функциональное наполнение кода, а бета-тестирования — стадию исправления ошибок. При этом как правило на каждом этапе разработки промежуточные результаты работы доступны конечным пользователям.

В зависимости от доступа разработчика тестов к исходному коду тестируемой программы различают «тестирование (по стратегии) белого ящика» и «тестирование (по стратегии) чёрного ящика».

При тестировании белого ящика (также говорят – прозрачного ящика), разработчик теста имеет доступ к исходному коду программ и может писать код, который связан с библиотеками тестируемого программного обеспечения. Это типично для компонентного тестирования, при котором тестируются только отдельные части системы. Оно обеспечивает то, что компоненты конструкции работоспособны и устойчивы, до определённой степени. При тестировании белого ящика используются метрики покрытия кода или мутационное тестирование.

При тестировании чёрного ящика тестировщик имеет доступ к программе только через те же интерфейсы, что и заказчик или пользователь, либо через внешние интерфейсы, позволяющие другому компьютеру либо другому процессу подключиться к системе для тестирования. Например, тестирующий компонент может виртуально нажимать клавиши или кнопки мыши в тестируемой программе с помощью механизма взаимодействия процессов, с уверенностью в том, все ли идёт правильно, что эти события вызывают тот же отклик, что и реальные нажатия клавиш и кнопок мыши. Как правило, тестирование чёрного ящика ведётся с использованием спецификаций или иных документов, описывающих требования к системе. Обычно в данном виде тестирования критерий покрытия складывается из покрытия структуры входных данных, покрытия требований и покрытия модели (в тестировании на основе моделей).

При тестировании серого ящика разработчик теста имеет доступ к исходному коду, но при непосредственном выполнении тестов доступ к коду, как правило, не требуется.

Если «альфа-» и «бета-тестирование» относятся к стадиям до выпуска продукта (а также, неявно, к объёму тестирующего сообщества и ограничениям на методы тестирования), тестирование «белого ящика» и «чёрного ящика» имеет отношение к способам, которыми тестировщик достигает цели.

Бета-тестирование в целом ограничено техникой чёрного ящика (хотя постоянная часть тестировщиков обычно продолжает тестирование белого ящика параллельно бета-тестированию). Таким образом, термин «бета-тестирование» может указывать на состояние программы (ближе к выпуску, чем «альфа»), или может указывать на некоторую группу тестировщиков и процесс, выполняемый этой группой. То есть, тестировщик может продолжать работу по тестированию белого ящика, хотя программа уже «бета-стадии», но в этом случае он не является частью «бета-тестирования».

Существующие на сегодняметоды тестирования программного обеспечения не позволяют однозначно и полностью выявить все дефекты и установить корректность функционирования анализируемой программы, поэтому все существующие методы тестирования действуют в рамках формального процесса проверки исследуемого или разрабатываемого программного обеспечения. Такой процесс формальной проверки, или верификации, может доказать, что дефекты отсутствуют с точки зрения используемого метода (то есть нет никакой возможности точно установить или гарантировать отсутствие дефектов в программном продукте с учётом человеческого фактора, присутствующего на всех этапах жизненного цикла программного обеспечения).

Существует множество подходов к решению задачи тестирования и верификации программного обеспечения, но эффективное тестирование сложных программных продуктов – это процесс в высшей степени творческий, не сводящийся к следованию строгим и чётким процедурам или созданию таковых.

Качество программного обеспечения можно определить как совокупную характеристику исследуемого программного обеспечения с учётом следующих составляющих:

* надёжность;
* сопровождаемость;
* практичность;
* эффективность;
* мобильность;
* функциональность.

Для разработанного программного средства было проведено тестирование с использованием тестовых сценариев (см. таблицу 4.1 – Добавление документа, 4.2 – Проверка корректности работы отчётов).

Таблица 4.1 – Добавление документа.

|  |  |  |
| --- | --- | --- |
| Действие | Ожидаемый результат | Результат теста |
| Запустить приложение | Отображение актуальных данных сохраненных в БД | OK |
| Авторизоваться в системе |
| Дождаться загрузки данных | Данные в БД изменены |
| Выбрать документы «Приход» в меню «Документы» и выбрать «Ввести на основании расход» | Документ создан |
| Заполнение информации | Информация заполнена |
| Нажать на кнопку «Провести документ» | Документ сохранен и проведён |

Таблица 4.1 – Проверка корректности работы отчётов.

|  |  |  |
| --- | --- | --- |
| Действие | Ожидаемый результат | Результат теста |
| Запустить приложение | Отображение актуальных данных сохраненных в БД | OK |
| Авторизоваться в системе |
| Дождаться загрузки данных | Данные в БД изменены |
| Выбрать в меню раздел отчёты | Открытие страницы с отчётами |
| Выбрать интересующий отчёт и ввести диапазон времени и нажать кнопку «Сформировать» | Открылся отчёт |

Профессиональные тестировщики используют тестовые сценарии на разных уровнях: как в модульном, так и в интеграционном и системном тестировании. Тестовые сценарии, как правило, пишутся для проверки компонентов, в которых наиболее высока вероятность появления отказов или вовремя не найденная ошибка может быть дорогостоящей.

В ходе проведения тестирования приложения ошибок выявлено не было. Тестирование показало, что данное приложение работает корректно.

**5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Запустить приложение и авторизоваться в системе. Форма авторизации представлена на рисунке 5.1.

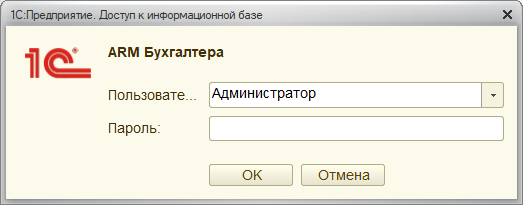


Рисунок 5.1 – Форма авторизации

После успешной авторизации откроется главная форма приложения (представлена на рисунке 5.2).

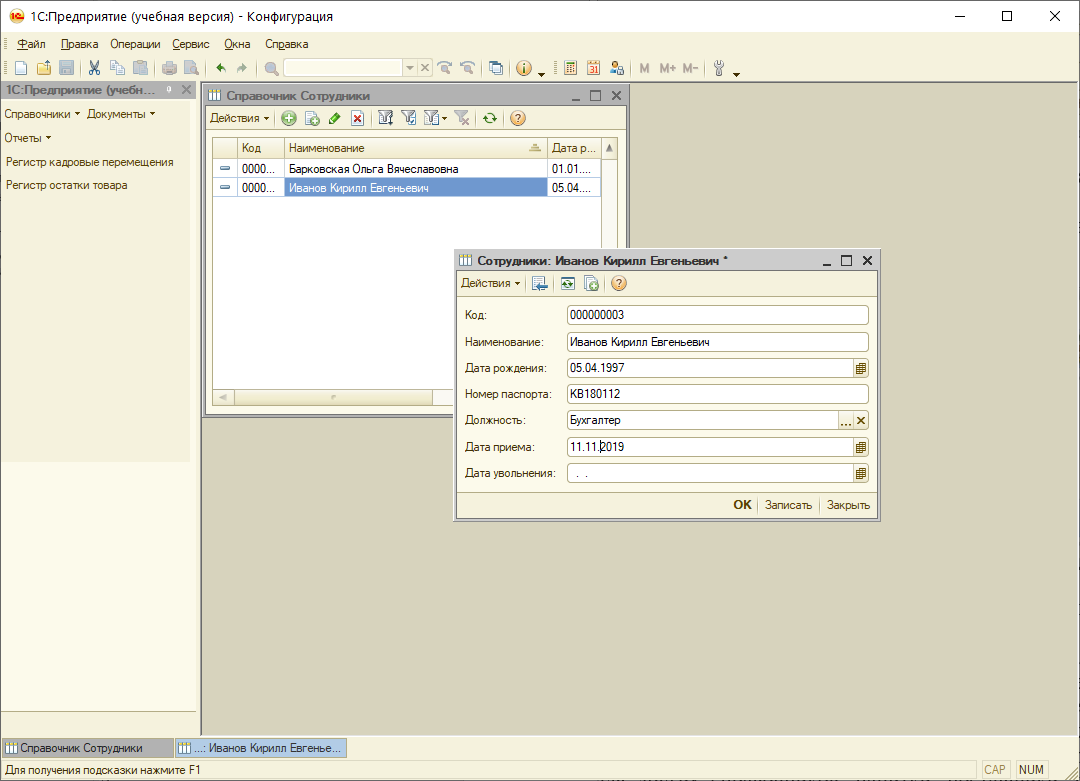


Рисунок 5.2 – Главная форма приложения

Для редактирования информации необходимо нажать на соответствующую иконку в главном меню. После чего пользователю будет доступно создание, редактирование и удалении.

Возможности по созданию, редактированию и удалению представлены на рисунках 5.3.

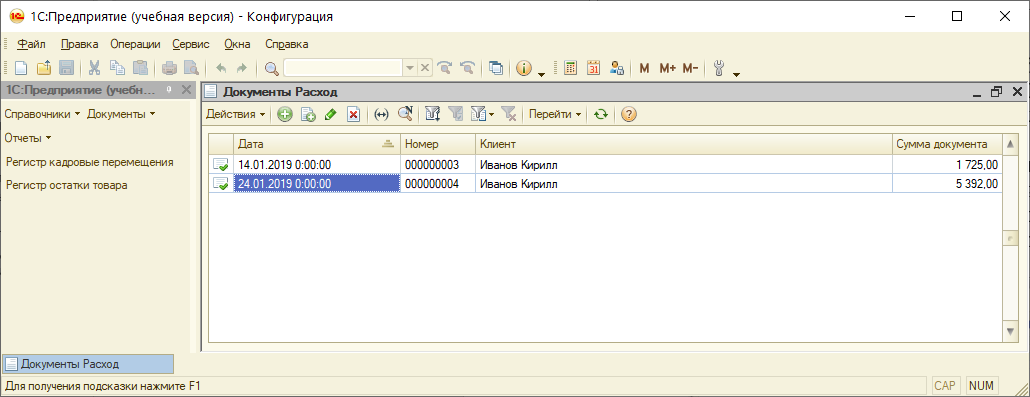


Рисунок 5.3 – Справочник «Основные средства»

Чтобы перейти в необходимый справочник/документ необходимо нажать соответствующую кнопку в меню (см. рисунок 5.4).

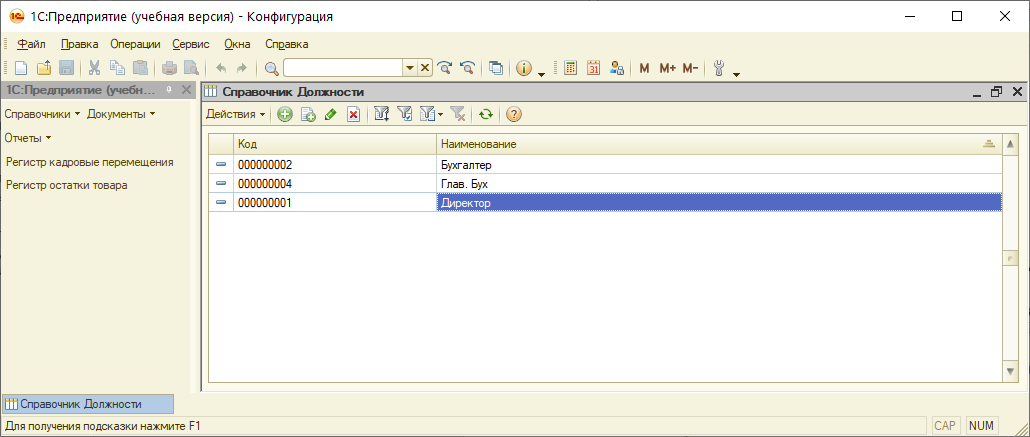


Рисунок 5.4 – Справочник «Сотрудники»

Чтобы сформировать отчет для получения информации о движении основных средств необходимо нажать по соответствующей иконке в главном меню. После чего потребуется указать диапазон дат дня сформировать отчет.

Отчет сформируется автоматически по указанным датам (см. рисунок 5.5).

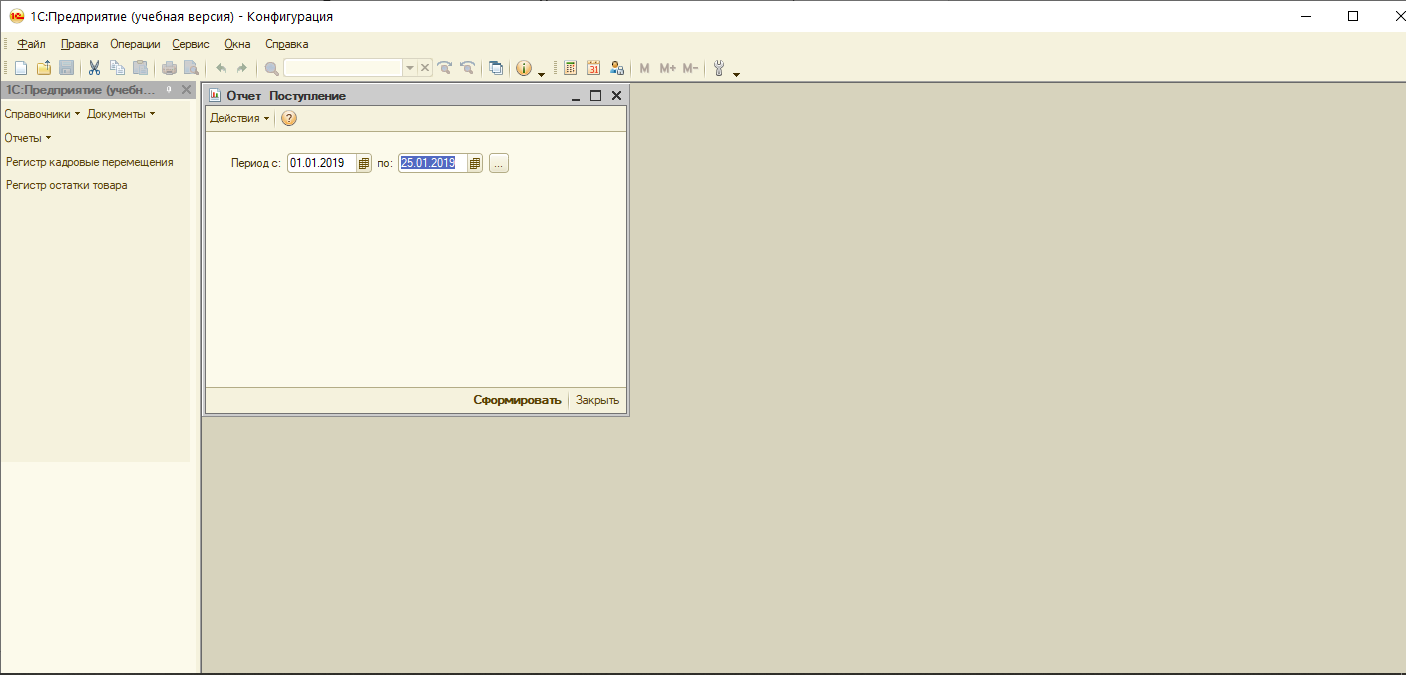


Рисунок 5.5 – Формирование отчётов

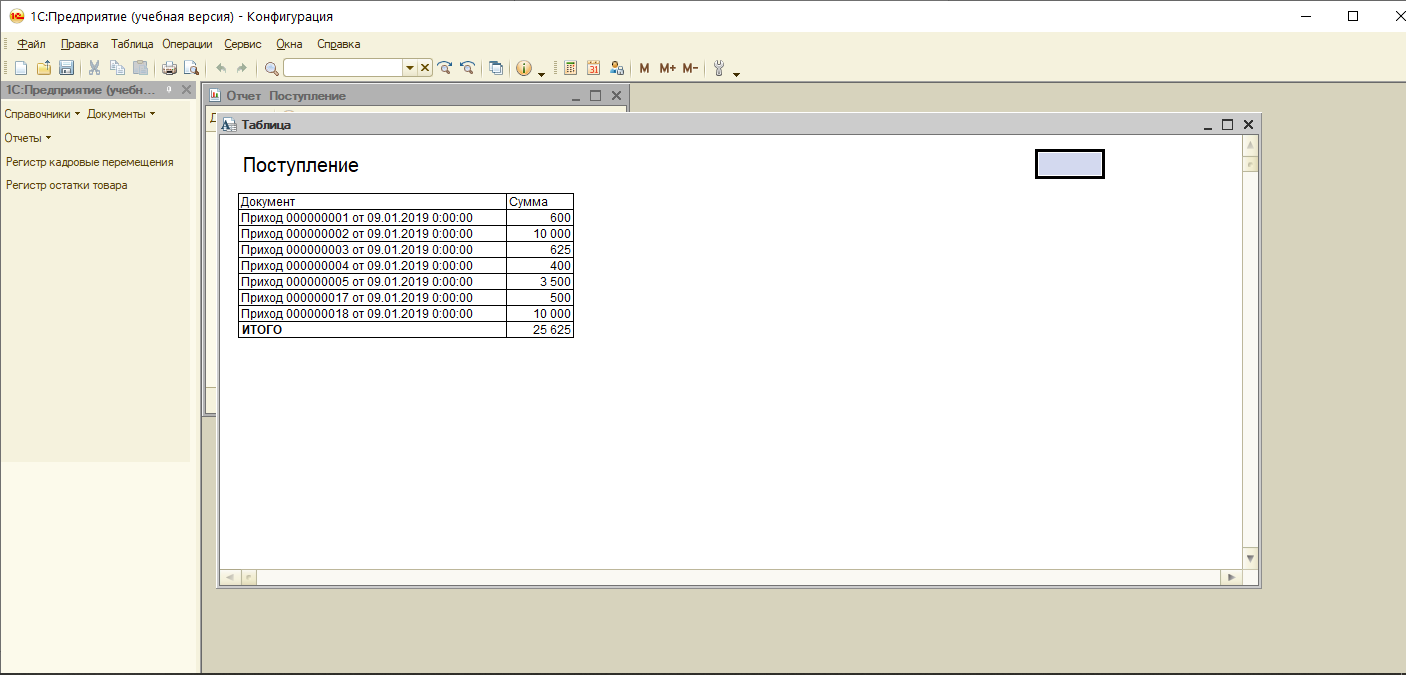


Рисунок 5.6 – Сформированный отчёт

Сформированный отчёт представлен на рисунке 5.6.

**ЗАКЛЮЧЕНИЕ**

Для разработки курсового проекта была выбрана платформа «1С:Предприятие». Данная платформа обладает возможностями, которых хватает для решения поставленной задачи по созданию автоматизированного рабочего места бухгалтера по ведению учета основных средств.

Разработанное в ходе выполнения курсового проекта программное средство на тему: «Автоматизированное рабочее место бухгалтера по ведению учёта основных средств» отвечает всем поставленным требованиям и имеет следующие функции: создание документов на приход основных средств, создание документов для реализации основных средств, хранение сведение о поставщиках, клиентах, пользователях, формирование отчетов за определенный период времени с возможностью просмотра каждого документа через отчет. Так же имеются регистры накоплений, которые отображают всю историю движения основных средств. Имеется регистр для кадровых перемещений в котором отображается должность сотрудника, оклад, дата приема.

Разработанное программное средство прошло тестирование на конкретно поставленных задачах, которые в свою очередь подтвердили корректную работу данного программного средства.

Программное средство может быть доработано по желанию заказчика.

Достигнута основная цель курсового проекта, максимально облегчить работу специалиста, высвобождая время и усилия, которые ранее расходовались на выполнение рутинных операций сбора данных и сложных расчетов в решении профессиональных задач.

Поставленные задачи достигнуты в полном объеме, а именно:

* создание документов по приходу и реализации основных средств;
* создание справочников для хранения информации о основных средствах;
* хранение данных в регистрах;
* формирование отчетов по реализации основных средств;
* контроль и просмотр остатков основных средств.

Пояснительная записка к курсовому включает в себя пять разделов, содержащих необходимую и достаточную информацию по использованию данного программного продукта, подробное описание программы и вычислительной системы.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] 1C:Бухгалтерия предприятия 8.1. Практическое пособие; КноРус - Москва, 2012. - 368 c.

[2] Баймакова И., Новиков А., Рогачев А., Хыдыров А. Обеспечение защиты персональных данных (+ CD-ROM); 1С-Паблишинг - Москва, 2011. - 272 c.

[3] Богаченко В. М., Кириллова Н. А., Сухарева Е. М. Практический консультант бухгалтера; Феникс - Москва, 2011. - 416 c.

[4] Бойко Э. В. 1С: Предприятие 8.0. Универсальный самоучитель; Омега-Л - Москва, 2011. - 232 c.

[5] Габец А. П., Козырев Д. В., Кухлевский Д. С., Хрусталева Е. Ю. Реализация прикладных задач в системе "1С:Предприятие 8.2" (+ CD-ROM); 1С-Паблишинг - Москва, 2010. - 720 c.

[6] Гладкий А. А. 1С:Управление торговлей 8.2 с нуля. 100 уроков для начинающих; БХВ-Петербург - Москва, 2011. - 448 c.

[7] Гладкий А. А. Ведение первичной документации в 1С 8.2; Феникс - Москва, 2011. - 192 c.

[8] Гладкий А. А. Учет складских операций в 1С 8.2; Феникс - Москва, 2011. - 160 c.

[9] Жадаев Александр Наглядный самоучитель 1C:Бухгалтерия 8.1 (+ DVD-ROM); БХВ-Петербург - Москва, 2010. - 268 c.

[10] Каргина Е. Н. Учет бизнес-процессов в системе "1С:Бухгалтерия 8.1"; Феникс - Москва, 2010. - 192 c.

[11] Князев А. М. 1С:Предприниматель 7.7. Учет и отчетность индивидуального предпринимателя; Триумф - Москва, 2013. - 224 c.

[12] Колесников С. Н. Планирование деятельности производственного предприятия. От промфинтехпланирования к МRP II и дальше; 1С-Паблишинг - Москва, 2013. - 384 c.

[13] Крюков Андрей 1С: Бухгалтерия 8: 250 вопросов и ответов; Эксмо - Москва, 2011. - 320 c.

[14] Кузнецов Владимир , Засорин Сергей Новейший самоучитель по 1С:Бухгалтерии 7.7; БХВ-Петербург - Москва, 2010. - 496 c.

[15] Кузнецов Владимир , Засорин Сергей Самоучитель 1С:Упрощенка 8; БХВ-Петербург - Москва, 2010. - 336 c.

[16] Несвижский Всеволод 1С:Предприятие 8.0. Приемы программирования (+ CD-ROM); БХВ-Петербург - Москва, 2012. - 512 c.

[17] Ощенко Игорь 1C:Предприятие. Торговля и склад для начинающих; БХВ-Петербург - Москва, 2013. - 256 c.

[18] Радченко М. Г. 1С:Предприятие 8.2. Коротко о главном. Новые возможности версии 8.2 (+ CD-ROM); 1С-Паблишинг - Москва, 2010. - 416 c.

**ПРИЛОЖЕНИЕ А**

Листинг программы

Процедура КнопкаСформироватьНажатие(Кнопка)

Макет = ПолучитьМакет("Макет");

Область = Макет.ПолучитьОбласть("Шапка");

ТабДок = Новый ТабличныйДокумент;

ТабДок.Вывести(Область);

Сумма = 0;

Выборка = Документы.Расход.Выбрать(НачПериода, КонПериода);

Пока Выборка.Следующий() Цикл

Если Выборка.Проведен Тогда

Область = Макет.ПолучитьОбласть("Строка");

Область.Параметры.Документ = Выборка.Ссылка;

Область.Параметры.Сумма = Выборка.СуммаДокумента;

ТабДок.Вывести(Область);

Сумма = Сумма + Выборка.СуммаДокумента;

КонецЕсли;

КонецЦикла;

Область = Макет.ПолучитьОбласть("Итого");

Область.Параметры.Сумма = Сумма;

ТабДок.Вывести(Область);

ТабДок.ОтображатьЗаголовки = Ложь;

ТабДок.ОтображатьСетку = Ложь;

ТабДок.ТолькоПросмотр = Истина;

ТабДок.Показать();

КонецПроцедуры

Процедура ВыбПериодНажатие(Элемент)

НастройкаПериода = Новый НастройкаПериода;

НастройкаПериода.РедактироватьКакИнтервал = Истина;

НастройкаПериода.РедактироватьКакПериод = Истина;

НастройкаПериода.ВариантНастройки = ВариантНастройкиПериода.Период;

НастройкаПериода.УстановитьПериод(НачПериода, ?(КонПериода='0001-01-01', КонПериода, КонецДня(КонПериода)));

Если НастройкаПериода.Редактировать() Тогда

НачПериода = НастройкаПериода.ПолучитьДатуНачала();

КонПериода = НастройкаПериода.ПолучитьДатуОкончания();

КонецЕсли;

КонецПроцедуры

Процедура КнопкаСформироватьНажатие(Кнопка)

Макет = ПолучитьМакет("Макет");

Область = Макет.ПолучитьОбласть("Шапка");

ТабДок = Новый ТабличныйДокумент;

ТабДок.Вывести(Область);

Сумма = 0;

Выборка = Документы.Приход.Выбрать(НачПериода, КонПериода);

Пока Выборка.Следующий() Цикл

Если Выборка.Проведен Тогда

Область = Макет.ПолучитьОбласть("Строка");

Область.Параметры.Документ = Выборка.Ссылка;

Область.Параметры.Сумма = Выборка.СуммаДокумента;

ТабДок.Вывести(Область);

Сумма = Сумма + Выборка.СуммаДокумента;

КонецЕсли;

КонецЦикла;

Область = Макет.ПолучитьОбласть("Итого");

Область.Параметры.Сумма = Сумма;

ТабДок.Вывести(Область);

ТабДок.ОтображатьЗаголовки = Ложь;

ТабДок.ОтображатьСетку = Ложь;

ТабДок.ТолькоПросмотр = Истина;

ТабДок.Показать();

КонецПроцедуры

Процедура ВыбПериодНажатие(Элемент)

НастройкаПериода = Новый НастройкаПериода;

НастройкаПериода.РедактироватьКакИнтервал = Истина;

НастройкаПериода.РедактироватьКакПериод = Истина;

НастройкаПериода.ВариантНастройки = ВариантНастройкиПериода.Период;

НастройкаПериода.УстановитьПериод(НачПериода, ?(КонПериода='0001-01-01', КонПериода, КонецДня(КонПериода)));

Если НастройкаПериода.Редактировать() Тогда

НачПериода = НастройкаПериода.ПолучитьДатуНачала();

КонПериода = НастройкаПериода.ПолучитьДатуОкончания();

КонецЕсли;

КонецПроцедуры

Процедура КнопкаСформироватьНажатие(Кнопка)

Остатки = РегистрыНакопления.ОстаткиТоваров.Остатки(Дата);

Макет = ПолучитьМакет("Макет");

ТабДок = Новый ТабличныйДокумент;

Область = Макет.ПолучитьОбласть("Шапка");

ТабДок.Вывести(Область);

Для Каждого Строка Из Остатки Цикл

Область = Макет.ПолучитьОбласть("Строка");

Область.Параметры.Номенклатура = Строка.Товар;

Область.Параметры.Количество = Строка.Количество;

ТабДок.Вывести(Область);

КонецЦикла;

Область = Макет.ПолучитьОбласть("Итого");

Область.Параметры.Количество = Остатки.Итог("Количество");

ТабДок.Вывести(Область);

ТабДок.Показать();

ТабДок.ОтображатьЗаголовки = Ложь;

ТабДок.ОтображатьСетку = Ложь;

ТабДок.ТолькоПросмотр = Истина;

КонецПроцедуры

Процедура ТоварыСуммаНачалоВыбора(Элемент, СтандартнаяОбработка)

// Вставить содержимое обработчика.

КонецПроцедуры

Процедура ТоварыЦенаПриИзменении(Элемент)

стр=ЭлементыФормы.Товары.ТекущаяСтрока;

стр.Сумма=стр.Количество\*стр.Цена;

КонецПроцедуры

Процедура ТоварыКоличествоПриИзменении(Элемент)

стр=ЭлементыФормы.Товары.ТекущаяСтрока;

стр.Сумма=стр.Количество\*стр.Цена;

КонецПроцедуры

Процедура СоздатьНажатие(Элемент)

Объект = Документы.Приход.СоздатьДокумент();

Объект.Дата = ТекущаяДата();

Объект.Записать();

КонецПроцедуры

Процедура УдалитьНажатие(Элемент)

Ссылка = ЭлементыФормы.ДокументСписок.ТекущиеДанные.Ссылка;

Объект = Ссылка.ПолучитьОбъект();

Объект.Удалить();

КонецПроцедуры

Процедура ТоварыСуммаНачалоВыбора(Элемент, СтандартнаяОбработка)

// Вставить содержимое обработчика.

КонецПроцедуры

Процедура ТоварыЦенаПриИзменении(Элемент)

КонецПроцедуры

Процедура ТоварыКоличествоПриИзменении(Элемент)

стр=ЭлементыФормы.Товары.ТекущаяСтрока;

стр.Сумма=стр.Количество\*стр.Цена;

КонецПроцедуры

**ПРИЛОЖЕНИЕ Б**

Схема алгоритма программы общая

