МИНИСТЕРСТВО ОБРАЗОВАНИЯ КИРОВСКОЙ ОБЛАСТИ

Кировское областное государственное профессиональное образовательное бюджетное учреждение

"Слободской колледж педагогики и социальных отношений"

ДИПЛОМНЫЙ ПРОЕКТ

Разработка программного обеспечения для комплексной автоматизации библиотеки

Збруев Антон Владимирович

Специальность 09.02.07

Информационные системы и программирование

Группа 21П-1

Форма обучения: очная

Руководитель:

Пентин Николай Сергеевич

Дипломный проект защищен

"__" _____2025 г.

Оценка _____

Секретарь ГЭК_____

Слободской

2025

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В условиях современного информационного общества библиотеки играют важную роль, предоставляя доступ к знаниям и культурному наследию. Однако эффективное выполнение этой роли требует постоянной модернизации и оптимизации библиотечных процессов. Традиционные методы учета книжного фонда, выдачи и возврата литературы, регистрации читателей и других операций становятся все менее эффективными в условиях растущих информационных потоков и изменяющихся потребностей пользователей. Внедрение современных информационных систем позволяет библиотекам решать эти задачи, обеспечивая более высокий уровень обслуживания, сокращение трудозатрат и эффективное управление ресурсами.

Актуальность разработки информационных систем для библиотек подтверждается стремлением превратить их в интерактивные информационно-культурные центры. Автоматизация позволяет библиотекам расширить спектр предоставляемых услуг, упростить доступ к информации, интегрироваться с электронными ресурсами и повысить общую эффективность своей деятельности.

Платформа «1С:Предприятие 8.5» является одним из наиболее подходящих инструментов для разработки библиотечных ИС. Ее гибкость, масштабируемость, широкие интеграционные возможности и наличие развитой инфраструктуры поддержки позволяют создавать системы, максимально удовлетворяющие специфическим требованиям библиотек различного профиля и масштаба. Доступность квалифицированных специалистов и консультационных услуг в области «1С:Предприятие» минимизирует риски и затраты на внедрение и сопровождение систем.

Цель данной дипломной работы – разработка информационной системы для автоматизации работы библиотеки на платформе «1С:Предприятие 8.5». Для достижения поставленной цели необходимо решить следующие задачи: провести анализ предметной области, разработать структуру базы данных и пользовательский интерфейс, реализовать функционал учета книжного фонда и

работы с читателями, провести тестирование и отладку системы, подготовить пользовательскую документацию.

Объектом исследования являются библиотечные процессы, а предметом — информационная система для их автоматизации. Методы исследования включают анализ предметной области, моделирование бизнес-процессов, объектно-ориентированное программирование и тестирование.

Практическая значимость дипломного проекта заключается в автоматизации ключевых библиотечных процессов, что позволит повысить эффективность работы библиотеки, улучшить качество обслуживания читателей и оптимизировать использование ресурсов.

Методологический аппарат исследования базируется на современных методологиях разработки программного обеспечения на платформе «1С:Предприятие», а также на методах анализа и проектирования информационных систем.

Дипломная работа состоит из введения, трех глав, заключения, списка литературы и приложений. Первая глава посвящена анализу предметной области и формулированию требований к системе. Во второй главе описана разработка информационной системы, ее структура и функциональные возможности. Третья глава содержит результаты тестирования и руководство пользователя.

В данном введении обоснована актуальность разработки ИС для библиотеки, определены цель, задачи, объект и предмет исследования, методы и методологический аппарат, а также отмечена практическая значимость проекта. Описание структуры работы завершает введение.

ГЛАВА 1. АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1 Анализ предметной области

Современная библиотека – это сложный организм, эффективное функционирование которого зависит от множества взаимосвязанных процессов. Для успешной работы библиотеки необходимо обеспечить точный учет книжного фонда, оперативный контроль за выдачей и возвратом литературы, а также вести базу данных читателей. Кроме того, важными аспектами деятельности являются закупка новых изданий, списание устаревших или поврежденных книг и прием пожертвований.

В данном разделе будет проведен детальный анализ предметной области, охватывающий ключевые библиотечные процессы и участников этих процессов. Это позволит выявить требования к разрабатываемой информационной системе и обеспечить ее максимальное соответствие потребностям библиотеки.

Основой любой библиотеки является ее книжный фонд. Эффективное управление этим фондом, как ключевым ресурсом библиотеки, невозможно без точного и подробного учета каждого экземпляра [12]. Разрабатываемая информационная система должна обеспечивать хранение и обработку всей необходимой информации о книгах, включая такие данные, как название, автор , жанр, издательство, год издания, ISBN, количество страниц и краткое описание содержания. Кроме того, система должна учитывать возрастной рейтинг книг, чтобы контролировать доступ к литературе для разных возрастных групп читателей. Информация о месте хранения каждой книги (зал, стеллаж, полка) позволит быстро находить нужные экземпляры. Важным аспектом учета является также статус книги, который может принимать значения "Есть на складе", "Отсутствует на складе", отражающие текущее состояние книги. Для удобства поиска и идентификации книг система должна предоставлять возможность добавления изображения обложки.

ISBN (International Standard Book Number) - это международный стандартный книжный номер, уникальный идентификатор для каждой книги. Он используется для упрощения учета и поиска книг в библиотеках, книжных магазинах и других организациях. Наличие ISBN в системе позволит быстро и точно идентифицировать книги, а также интегрировать систему с внешними базами данных и каталогами. Принципы использования стандартных идентификаторов, таких как ISBN, являются важной частью проектирования современных информационных систем [13].

Работа с читателями: для эффективного обслуживания читателей, система должна обеспечивать быструю и удобную регистрацию новых пользователей. При регистрации в системе фиксируются стандартные данные: фамилия, имя, отчество, дата рождения, полный почтовый адрес и контактная информация (телефон, электронная почта). Хранение этой информации позволяет библиотекарям легко идентифицировать читателей, связываться с ними по возникшим вопросам, а также вести статистику пользователей библиотеки. Кроме того, система должна предоставлять возможность просмотра истории взаимодействия с каждым читателем, включая информацию о выданных и возвращенных книгах, а также наличие задолженностей по штрафам. Разработка требований к хранению и обработке персональных данных пользователей является важным этапом создания любой информационной системы [14].

Выдача и возврат книг: центральным процессом в работе библиотеки является выдача и возврат книг. Система должна автоматизировать и упростить эти операции. При выдаче книги библиотекарь регистрирует в системе читателя, выбранную книгу, количество выдаваемых экземпляров, дату выдачи и расчетную дату возврата. При возврате книги фиксируется фактическая дата возврата и оценивается состояние возвращенной книги (на предмет повреждений). Автоматизация этого процесса позволяет контролировать соблюдение сроков возврата, вести точный учет книг, находящихся в обращении, и своевременно выявлять просроченную задолженность. При реализации таких операций в системах на платформе 1С:Предприятие

используются механизмы проведения документов и движения по регистрам накопления [5].

Продление книг: система должна предусматривать возможность продления срока выдачи книг по запросу читателя. При продлении в системе регистрируется новая дата планируемого возврата, что позволяет избежать начисления необоснованных штрафов.

Учет читательских билетов: читательские билеты являются необходимым условием для пользования библиотекой. Система должна обеспечивать регистрацию и учет читательских билетов, храня информацию о читателе, дате выдачи билета, выбранном абонементе, сроке действия абонемента, а также возрастном рейтинге, установленном для данного билета. Система должна поддерживать различные типы абонементов, отличающиеся стоимостью и сроком действия, что позволяет библиотеке предоставлять гибкие условия обслуживания для разных категорий читателей. Продление читательских билетов также должно отражаться в системе с фиксацией новой даты окончания срока действия.

Учет штрафов: система должна автоматически рассчитывать штрафы за просроченные книги на основании стоимости штрафа за день просрочки и количества просроченных дней. Должна быть предусмотрена возможность регистрации оплаты штрафов.

Закупка книг: процесс закупки новых книг должен быть автоматизирован. Система должна позволять создавать заказы на закупку, указывая поставщика (или издательство), список заказываемых книг и их количество. Необходимо отслеживать статус заказа (сформирован, оплачен, в пути, получен).

Списание книг: система должна позволять списывать книги с учетом причины списания (износ, повреждение, утрата).

Учет пожертвований: библиотеки часто получают книги в качестве пожертвований. Система должна предусматривать функционал для регистрации пожертвованных книг с указанием жертвователя и списка книг.

Отчетность: для анализа работы библиотеки и принятия управленческих решений система должна формировать различные отчеты. Например, отчеты по остаткам книг, популярности книг среди читателей, читателям с просроченными книгами, закупкам и списаниям.

Сотрудники, отделы и должности: для обеспечения бесперебойной работы библиотеки и корректного функционирования информационной системы, в библиотеке выделены следующие отделы и должности:

- Отдел обслуживания читателей: библиотекари этого отдела взаимодействуют с читателями, оформляют выдачу и возврат книг, проводят регистрацию новых читателей, оформляют и продлевают читательские билеты, а также взимают штрафы (Рисунок 1). В системе они работают с документами "ВыдачаКниг", "ВозвратКниг", "ЧитательскийБилет", "ПродлениеЧитательскогоБилета", "ОплатаШтрафов".

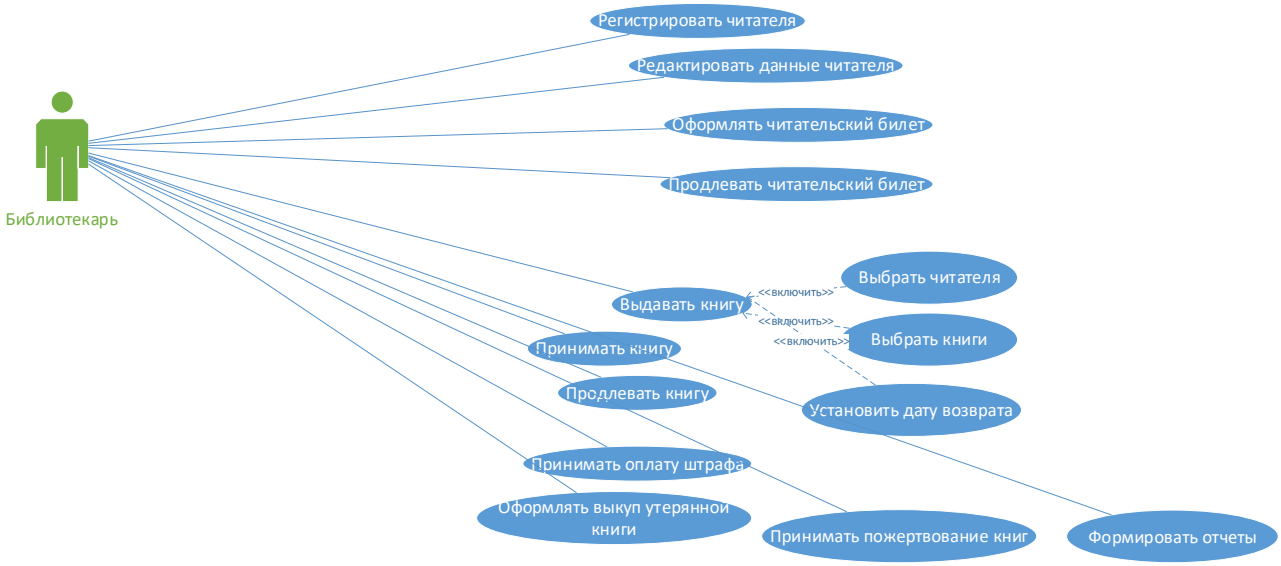


Рисунок 1 - Сценарии работы роли "библиотекарь" в ИС

- Отдел комплектования и обработки литературы: сотрудники этого отдела (библиографы, заведующий отделом) отвечают за формирование и обновление книжного фонда (Рисунок 2). Они проводят закупку новых книг, регистрируют их поступление в библиотеку, списывают устаревшую или поврежденную литературу, а также оформляют пожертвования. В системе они

используют документы "ЗакупкаКниг", "ПриходКниг", "СписаниеКниг", "Пожертвование".

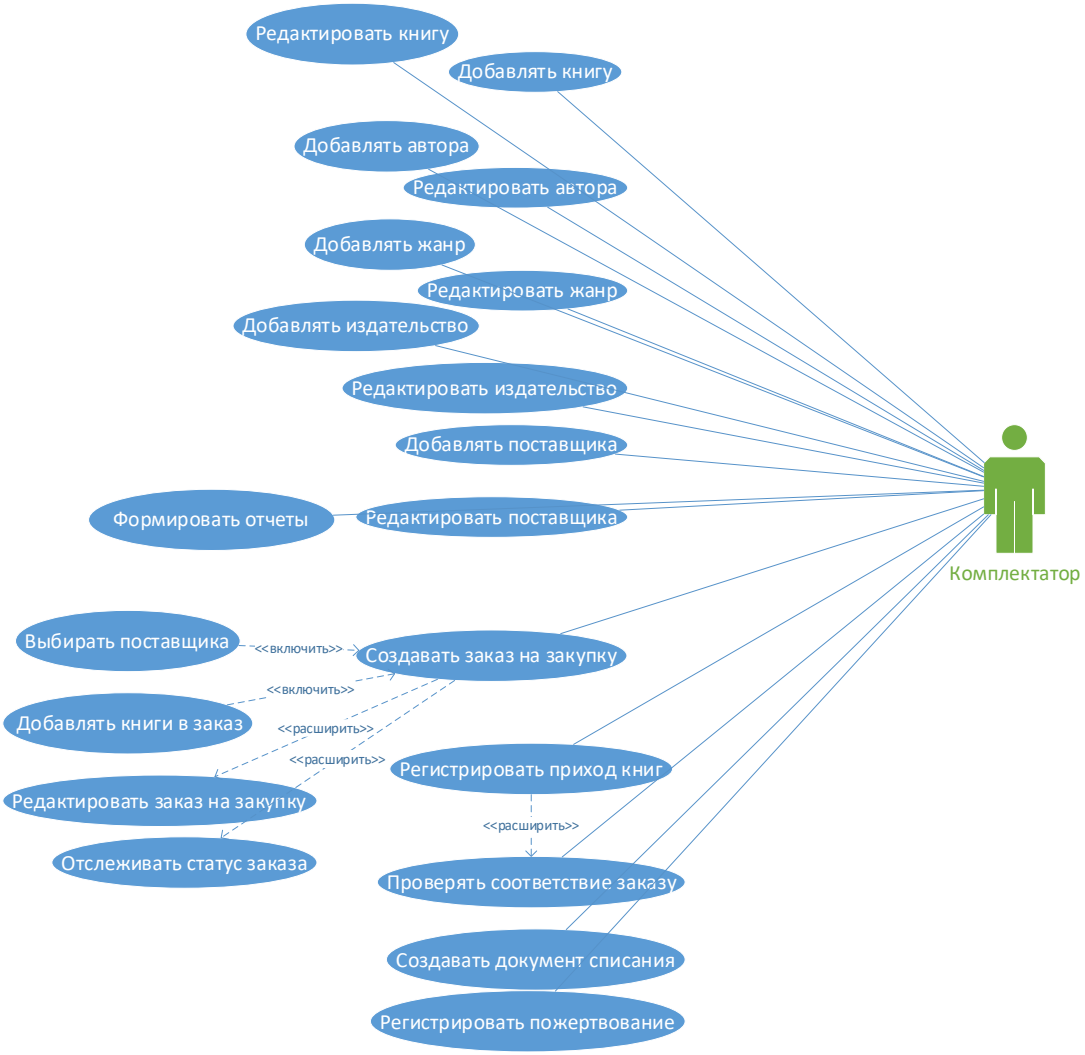


Рисунок 2 - Сценарии работы роли "комплектатор" в ИС

- Административно-управленческий отдел: руководители (директор, заместитель директора) и администраторы системы имеют полный доступ к функционалу ИС (Рисунок 3). Они отвечают за общее управление библиотекой, настройку системы, управление пользователями и их правами доступа, а также формирование отчетности. Руководители, в отличие от администраторов, могут быть ограничены в правах на удаление информации. Реализация ролевой модели и настройка прав доступа являются стандартными задачами при разработке конфигураций на платформе 1С:Предприятие [6].

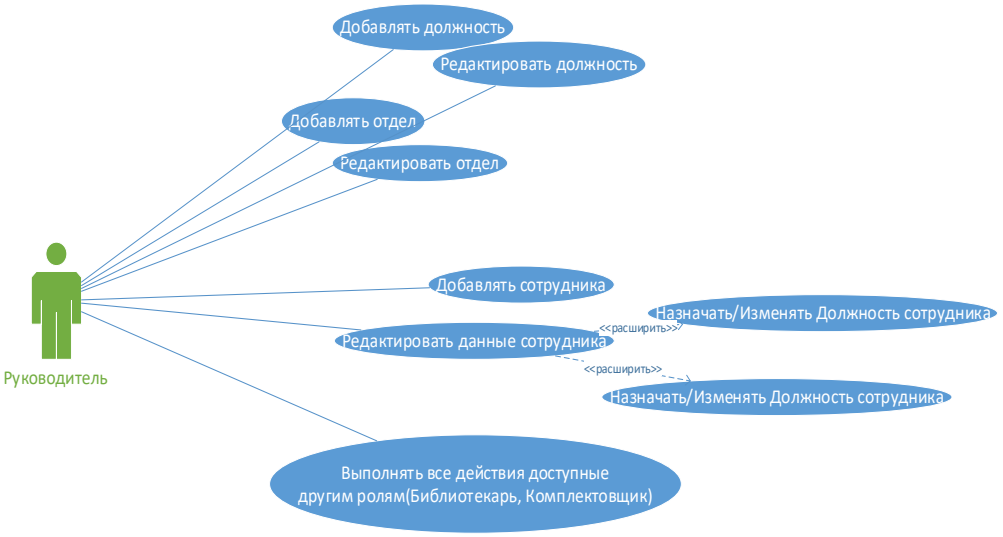


Рисунок 3 - Сценарии работы роли "Руководитель" в ИС

Программы-аналоги: перед разработкой собственной информационной системы важно проанализировать существующие на рынке решения, чтобы учесть их достоинства и недостатки. Анализ аналогов позволяет уточнить требования к собственной разработке и избежать повторения известных проблем [14]. Рассмотрим две программы-аналога от разных компаний:

1. АИБС "МАРК-SQL" (разработчик – компания "РУСЛАН"):

- **Описание:** Комплексная автоматизированная информационно-библиотечная система, предназначенная для автоматизации всех основных библиотечных процессов: комплектование, каталогизация, читательское обслуживание, учет фондов и т.д.
- **Плюсы:**
  - Широкий функционал, покрывающий большинство библиотечных процессов.
  - Возможность работы с большими объемами данных.
  - Интеграция с различными внешними системами.
- **Минусы:**
  - Высокая стоимость лицензий.
  - Сложность внедрения и настройки.
  - Требует высокой квалификации персонала.

2. АИБС "Ирбис64" (разработчик – ГПНТБ России):

- **Описание:** Автоматизированная информационно-библиотечная система, предоставляющая инструменты для автоматизации библиотечных процессов.
- **Плюсы:**
  - Бесплатное распространение.
  - Открытый исходный код.
  - Большое сообщество пользователей.
- **Минусы:**
  - Ограниченный функционал по сравнению с коммерческими АИБС.
  - Может требовать дополнительной доработки под специфические потребности библиотеки.
  - Менее удобный интерфейс.

Вывод по анализу программ-аналогов: анализ программ-аналогов показал, что существующие решения либо обладают широким функционалом, но имеют высокую стоимость и сложность внедрения ("МАРК-SQL"), либо бесплатны и доступны, но имеют ограниченный функционал и требуют доработки ("Ирбис64"). Разрабатываемая в рамках данного дипломного проекта система на платформе "1С:Предприятие 8.5" направлена на то, чтобы сочетать в себе достоинства обоих подходов: предоставить необходимый функционал для автоматизации библиотечных процессов, обладая при этом гибкостью, масштабируемостью и относительно невысокой стоимостью внедрения и сопровождения благодаря использованию популярной платформы.

Общий вывод по анализу предметной области: проведенный анализ предметной области, включающий рассмотрение основных библиотечных процессов, ролей пользователей и доступных программных аналогов, позволил сформулировать четкие требования к разрабатываемой информационной системе. Система должна обеспечить автоматизацию учета книжного фонда, работы с читателями, закупок, списаний и других важных аспектов библиотечной деятельности. Полученные результаты будут использованы для проектирования базы данных, разработки пользовательского интерфейса и

реализации функциональных возможностей системы. Таким образом, этап анализа предметной области закладывает фундамент для последующих стадий разработки программного обеспечения в соответствии с процессами жизненного цикла [4]. Выбранная платформа "1С:Предприятие 8.5" позволит создать гибкое и масштабируемое решение, адаптированное к специфическим потребностям библиотеки, а так же имеет современный стиль, который является более удобным решением в сравнений с более старыми версиями.

1.2. Техническое задание

Наименование программы: "Автоматизированная библиотечная информационная система" (АБИС).

Заказчик: МастерСофт-Сервис.

Назначение: программа предназначена для автоматизации работы библиотеки, включая учет библиотечного фонда, обслуживание читателей, закупку и списание книг, учет пожертвований и формирование отчетов.

Область применения: программа может эксплуатироваться на ПК на любом предприятии, связанном с библиотечной деятельностью.

Функциональные требования.

1. Учет библиотечного фонда.

- Добавление, редактирование и удаление записей о книгах, журналах, газетах и других изданиях с указанием названия, автора, жанра, издательства, года издания, ISBN, количества страниц, описания, возрастного рейтинга, статуса, места хранения и изображения обложки.
- Каталогизация и классификация изданий.
- Учет поступлений (закупка, пожертвование) и выбытия (списание, утеря).
- Инвентаризация фонда.
- Поиск изданий по различным критериям.

2. Обслуживание читателей.

- Регистрация читателей с возможностью хранения полной информации о каждом читателе (ФИО, дата рождения, адрес, контакты, выданные книги и т.д.).
- Выдача и возврат книг с автоматическим учетом сроков и начислением штрафов за просрочку.
- Продление сроков пользования.
- Резервирование книг.
- Формирование читательских билетов с различными типами абонементов и возрастными рейтингами.
- Учет и оплата штрафов.

3. Закупка книг.

- Формирование заказов на закупку книг у поставщиков и издательств.
- Учет статусов заказов (сформирован, оплачен, в пути, получен).
- Регистрация поступления книг по заказам.

4. Списание книг.

- Регистрация списания книг с указанием причины.

5. Учет пожертвований.

- Регистрация пожертвованных книг с указанием жертвователя.

6. Формирование отчетов.

- Отчеты по библиотечному фонду (остатки, движение, популярность, анализ операций с книгами).
- Отчеты по читателям (задолженности).
- Отчеты по закупкам и списаниям.

7. Система разграничения прав доступа.

- Различные роли пользователей (администратор, комплектатор фондов, библиотекарь) с разными уровнями доступа к функциям системы.

Требования к пользовательскому интерфейсу.

- Интуитивно понятный и удобный интерфейс.
- Наличие системы подсказок и помощи.
- Соответствие принципам эргономики.

Требования к техническому обеспечению.

- Операционная система: Windows 10/11.
- Платформа разработки: 1С:Предприятие 8.5.
- СУБД: Использование внутренней СУБД у 1С:Предприятие 8.5.

Требования к надежности и безопасности.

- Устойчивость к ошибкам пользователя.
- Резервное копирование данных.
- Авторизация и разграничение прав доступа.
- Защита от несанкционированного доступа.
- Журналирование действий пользователей.

Стадии и этапы разработки.

Разработка АБИС будет осуществляться в несколько стадий, каждая из которых включает определенные этапы:

1. Анализ и проектирование.

- **Анализ требований:** сбор и анализ требований к системе, детальное изучение бизнес-процессов библиотеки, определение функциональных возможностей будущей системы. Результатом этого этапа является техническое задание.
- **Проектирование базы данных:** разработка структуры базы данных, определение сущностей, атрибутов и связей между ними. Создание логической и физической моделей базы данных.
- **Проектирование пользовательского интерфейса:** разработка макетов форм и отчетов, определение навигации и взаимодействия пользователя с системой.

2. Разработка:

- **Разработка модулей системы:** написание программного кода для реализации функциональных модулей системы (учет книг, работа с читателями, закупки, списания и т.д.).
- **Создание форм и отчетов:** реализация пользовательского интерфейса на основе разработанных макетов.

- **Настройка прав доступа:** настройка ролей пользователей и их прав доступа к различным функциям системы.

3. Тестирование:

- **Модульное тестирование:** проверка корректности работы отдельных модулей системы.
- **Интеграционное тестирование:** тестирование взаимодействия между модулями системы.
- **Системное тестирование:** проверка работы системы в целом на соответствие требованиям технического задания.
- **Нагрузочное тестирование:** оценка производительности системы при большой нагрузке.

4. Внедрение:

- **Установка и настройка системы:** установка системы на сервере и рабочих станциях пользователей, настройка параметров подключения к базе данных.
- **Обучение пользователей:** проведение обучения пользователей работе с системой.
- **Миграция данных:** перенос данных из существующих систем в новую АБИС (при необходимости).

5. Сопровождение:

- **Исправление ошибок:** оперативное исправление выявленных ошибок и недочетов.
- **Доработка функционала:** добавление нового функционала по запросу заказчика.
- **Консультационная поддержка:** предоставление консультаций пользователям по работе с системой.

Документация:

- Техническое задание.
- Руководство пользователя.

ВЫВОД ПО ГЛАВЕ 1

В первой главе дипломного проекта проведен анализ предметной области, определяющей контекст разработки, автоматизированной библиотечной информационной системы (АБИС). Детально рассмотрены основные бизнес-процессы библиотеки, включая учет книжного фонда, обслуживание читателей, процессы закупки и списания литературы, а также учет пожертвований. Особое внимание уделено ролям пользователей системы – библиотекарей, комплектаторов и администраторов, – и их взаимодействию с АБИС в рамках выполняемых ими функций.

Анализ существующих программных аналогов, таких как АИБС "МАРК-SQL" и "Ирбис64", позволил выявить их сильные и слабые стороны, а также обосновать выбор платформы "1С:Предприятие 8.5" для разработки АБИС. Преимущества данной платформы, такие как гибкость, масштабируемость и наличие широкой сети специалистов, делают ее оптимальным решением для создания системы, удовлетворяющей специфическим потребностям библиотеки.

На основе проведенного анализа сформулированы функциональные, технические и пользовательские требования к разрабатываемой АБИС, которые зафиксированы в техническом задании. Это задание служит основой для дальнейших стадий проектирования и разработки системы, обеспечивая ее соответствие потребностям библиотеки и эффективность ее использования. Результаты анализа предметной области будут использованы на последующих этапах разработки для создания оптимальной архитектуры системы, проектирования базы данных и разработки удобного и функционального пользовательского интерфейса.

ГЛАВА 2. КОНСТРУКТОРСКАЯ ЧАСТЬ

2.1. Архитектура программы

Разработанная автоматизированная библиотечная информационная система (АБИС) представляет собой прикладное решение, созданное на технологической платформе "1С:Предприятие 8.5". В состав программного обеспечения входит основной файл конфигурации Библиотека.cf, который содержит все объекты метаданных, определяющие структуру данных и логику работы системы.

Платформа и среда разработки:

Выбор платформы "1С:Предприятие 8.5" обусловлен ее широкими возможностями для быстрой разработки учетных систем, гибкостью настройки, масштабируемостью и наличием развитых механизмов интеграции. Платформа обеспечивает высокую производительность, надежность хранения данных и современные средства безопасности. Весь процесс разработки, включая создание объектов метаданных, написание программного кода и проектирование пользовательского интерфейса, осуществлялся в интегрированной среде разработки – Конфигураторе 1С.

Архитектура:

АБИС реализована с использованием **клиент-серверной архитектуры**, которая является рекомендуемой для многопользовательских систем и обеспечивает оптимальное распределение нагрузки.

- **Серверная часть:** отвечает за основную бизнес-логику, обработку данных и взаимодействие с системой управления базами данных (СУБД). Сервер "1С:Предприятие" управляет подключениями клиентских приложений, выполняет серверные процедуры и функции, обеспечивает целостность данных при одновременной работе нескольких пользователей.
- **Клиентская часть:** представляет собой пользовательский интерфейс, с которым взаимодействуют сотрудники библиотеки. В данной

разработке используется **тонкий клиент** и **веб-клиент**, что позволяет пользователям работать с системой как через установленное приложение, так и через веб-браузер, обеспечивая гибкость доступа. Взаимодействие между клиентом и сервером происходит по протоколу TCP/IP.

- **База данных:** для хранения всей информации системы (данные справочников, документы, записи регистров) используется **встроенная файловая СУБД платформы "1С:Предприятие"**. Это упрощает развертывание и администрирование системы для небольших и средних библиотек.

- **Основные компоненты системы (объекты метаданных):**

структура данных и функциональность АБИС определяются следующими основными объектами метаданных:

- **Справочники:** используются для хранения условно-постоянной, нормативно-справочной информации. Ключевыми справочниками являются: "Книги" (основной каталог фонда), "Читатели", "Авторы", "Жанры", "Издательства", "Поставщики", "Сотрудники" (для ведения списка персонала), "Должности", "Отделы", "Статусы" (для книг), "Руководители". Иерархическая структура справочника "Книги" позволяет группировать издания по типам (Книги, Журналы, Периодические издания).

- **Документы:** предназначены для регистрации хозяйственных операций и событий в хронологическом порядке. Основные документы системы: "ПриходКниг", "СписаниеКниг", "ЗакупкаКниг", "Пожертвование" (учет движения фонда); "ВыдачаКниг", "ВозвратКниг", "ПродлениеКниги" (обслуживание читателей); "ЧитательскийБилет", "ПродлениеЧитательскогоБилета" (учет билетов); "ВыкупУтерянной Книги", "ОплатаШтрафов" (финансовые операции с читателями).

- **Регистры сведений:** служат для хранения информации, характеризующей состояние объектов или взаимосвязи между ними на определенный момент времени или периодически. Используются регистры: "Цены" (возможно, для оценки книг), "ЦеныЧитательскихБилетов",

"ЦеныШтрафов", "СрокиДействияЧитательскихБилетов", "СрокВозвратаКниг", "ШтрафыЗаПросрочку".

- **Регистры накопления:** предназначены для хранения итоговой (агрегированной) информации о количественных или суммовых показателях в различных разрезах (измерениях). Ключевые регистры накопления: "ОстаткиКниг" (учет количества книг на складе), "ОборотЗаказовКниг" (отслеживание выполнения заказов), "ДоступныеКниги" (оперативный учет книг, доступных для выдачи), "ИсторияВыдачКниг" (отслеживание книг на руках у читателей), "ПопулярностьКниги" (сбор статистики для анализа спроса), "ОборотыПожертвований", "Взаиморасчеты" *(если используются)*. Регистр "РегистрБухУчет" был предусмотрен, но его использование в рамках данной работы ограничено.
- **Перечисления:** используются для хранения фиксированных наборов значений, определяющих различные классификационные признаки (например, ИсточникПолучения, СтатусЗакупки, СостояниеКниги, ПолМЖ и т.д.).
- **Обработки:** предназначены для выполнения сервисных функций и пакетной обработки данных. В системе реализованы обработки: "ЗагрузитьАвторов" (для импорта данных), "УстановкаЦенБилетов", "Виджеты" (для формирования данных начальной страницы), "ПервоначальноеЗаполнениеФонда".
- **Отчеты:** служат для анализа накопленной информации и представления ее в удобном для пользователя виде. Основные отчеты: ОтчетПоОстаткамКниг, СоотношениеКниг, ЗаказКниг, КнигиНаРуках, ДанныеОВозвратахКниг, ДоступныеКнигиДляВыдачи, ПопулярностьКнигСредиЧитателей, ОтчетПоВыкупуУтерянныхКниг, ПродажиДиаграмма.

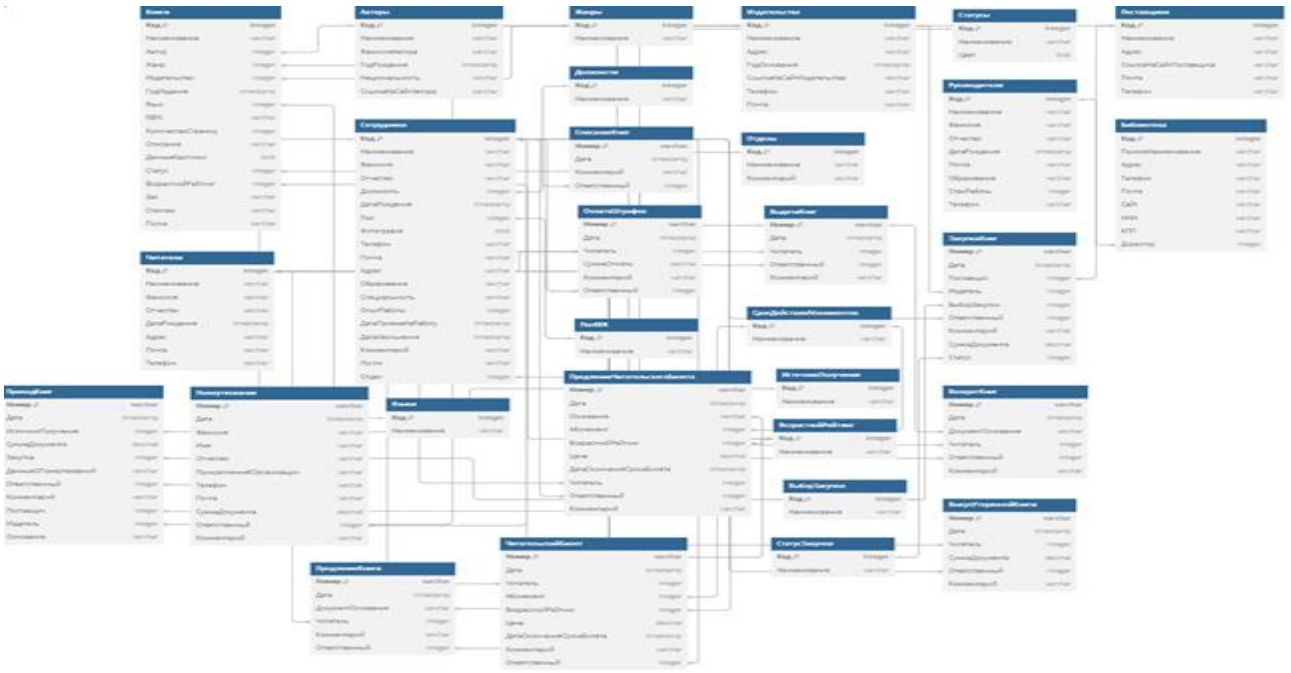


Рисунок 4 – Логическая структура базы данных (ER-диаграмма)

На рисунке 4 представлена ER-диаграмма, иллюстрирующая основные сущности базы данных и связи между ними, что дает наглядное представление о логической структуре хранения информации в АБИС.

Взаимодействие между компонентами системы осуществляется через вызовы процедур и функций общих модулей, а также через механизмы проведения документов, которые формируют записи в соответствующих регистрах. Пользовательский интерфейс построен на основе управляемых форм, обеспечивающих современный внешний вид и удобство работы.

Этот раздел описывает основные компоненты разработанной АБИС и используемые технологии. Более подробная информация о реализации отдельных объектов метаданных и алгоритмах их работы будет представлена в следующих разделах.

2.2. Описание алгоритмов и функционирования программы

- **Функционирование программы**

При запуске разработанной "Автоматизированной библиотечной информационной системы" (АБИС) происходит подключение к созданной базе

данных на платформе "1С:Предприятие 8.5". Затем система запрашивает у пользователя данные для авторизации. Механизмы подключения к базе данных и аутентификации пользователей являются стандартными возможностями платформы 1С:Предприятие [16].

Рассмотрим основные алгоритмы, используемые в программе. Ниже приведено их краткое описание.

- **Запуск программы**

При запуске АБИС происходит инициализация соединения с базой данных и отображение окна аутентификации.

- **Вход в программу**

Вход осуществляется после ввода логина и пароля. Если введенные данные соответствуют зарегистрированному пользователю (сотруднику или администратору), программа предоставляет доступ и открывает основной интерфейс (рабочий стол), соответствующий роли пользователя. Реализация разграничения доступа на основе ролей и управление пользовательскими интерфейсами выполняются с использованием встроенных средств конфигуратора [8]. Если логин или пароль неверны, доступ запрещается до ввода корректных данных.

- **Регистрация читателя и выдача билета**

Библиотекарь переходит в подсистему управления читателями, создает новую запись в справочнике "Читатели", заполняя необходимые данные. Затем создается документ "ЧитательскийБилет", где указывается читатель, выбирается абонемент и возрастной рейтинг. Система автоматически рассчитывает цену и дату окончания действия. Документ проводится, сохраняя информацию в регистре сведений. Для хранения атрибутов объектов, таких как данные читателей или параметры билетов, используются справочники и регистры сведений [5].

- **Выдача книги**

В подсистеме обслуживания читателей библиотекарь создает документ "ВыдачаКниг". Указывается читатель, в табличную часть добавляются книги,

устанавливается качество при выдаче и дата возврата. При проведении система проверяет наличие книги и действительность читательского билета. Данные о выдаче фиксируются в регистрах "ДоступныеКниги", "ИсторияВыдачКниг", "ПопулярностьКниги" и "СрокВозвратаКниг". Проведение документа "ВыдачаКниг" инициирует запись движений в соответствующие регистры накопления и сведений, отражая изменения в состоянии библиотечного фонда и истории взаимодействия с читателем [21].

- **Возврат книги**

Библиотекарь создает документ "ВозвратКниг", часто на основании выдачи. Указывается качество книги при возврате. Система сравнивает состояния и информирует о расхождениях. При проведении обновляются регистры "ИсторияВыдачКниг", "ДоступныеКниги", "СрокВозвратаКниг". Если возврат просрочен, рассчитывается и фиксируется штраф в регистре "ШтрафыЗаПросрочку". Алгоритмы проведения документа возврата включают логику контроля остатков по регистрам накопления и запись информации в периодические регистры сведений для отслеживания истории и начисления штрафов [22].

- **Закупка и приход книг**

Комплектатор создает документ "ЗакупкаКниг", указывая поставщика и заказываемые книги. Документ записывается, а при необходимости может формировать движения, например, для резервирования или планирования поступлений в регистре "ОборотЗаказовКниг" [9]. При фактическом поступлении книг создается документ "ПриходКниг" (возможно, на основании закупки). При проведении "ПриходКниг" книги поступают на учет в регистры "ОстаткиКниг" и "ДоступныеКниги", а также списывается информация из "ОборотЗаказовКниг".

- **Просмотр отчетов**

Пользователи с соответствующими правами могут перейти в раздел отчетов и сформировать необходимый отчет (например, "ОтчетПоОстаткамКниг", "ПопулярностьКнигСредиЧитателей",

"ДанныеОВозвратахКниг"). Отчеты формируются на основе актуальных данных из регистров системы. Для формирования сложных аналитических отчетов в системе используется механизм Системы Компоновки Данных (СКД) [7].

- **Работа с виджетами (на основной форме)**

При открытии основной формы автоматически загружаются данные для виджетов: информация о текущем пользователе, данные о библиотеке, список ожидаемых заказов и диаграмма популярности книг за выбранный период. Пользователь может изменять период для диаграммы популярности. Настройка начальной страницы и использование управляемых форм позволяют создавать удобные и информативные рабочие столы для пользователей [20]

- **Входные данные**

Входными данными программы является информация, вносимая пользователем с клавиатуры и путем выбора из справочников в поля документов и справочников. Эта информация хранится в базе данных в соответствующих таблицах (объектах метаданных). Структура хранения данных определяется объектами метаданных конфигурации, такими как справочники, документы и регистры [6].

- **Выходные данные**

Выходные данные выводятся в виде сформированных отчетов, информации в списках и формах документов и справочников, а также в виде печатных форм документов (например, акт пожертвования, регистрация книг, чеки об оплате). Формирование печатных форм осуществляется с использованием механизма макетов и языка запросов для выборки необходимых данных [24]. Отчеты и списки формируются на основе данных, хранящихся в регистрах и справочниках системы.

ВЫВОДЫ ПО ГЛАВЕ 2

Во второй главе дипломного проекта была рассмотрена конструкторская часть разработанной автоматизированной библиотечной информационной системы (АБИС).

Был проведен анализ и описание архитектуры программы. Система построена на платформе "1С:Предприятие 8.5" с использованием клиент-серверной архитектуры, что обеспечивает надежность, производительность и возможность масштабирования. Определен состав программного обеспечения, включая основную конфигурацию и потенциальные расширения. Перечислены ключевые компоненты системы: справочники, документы, регистры сведений и накопления, перечисления, обработки и отчеты, составляющие основу структуры данных и функциональности АБИС.

Далее было представлено описание основных алгоритмов и функционирования программы. Рассмотрены ключевые сценарии работы пользователей с системой, начиная от запуска и авторизации, до выполнения основных операций: регистрации читателей и выдачи билетов, выдачи и возврата книг, закупки и прихода литературы, просмотра отчетов и взаимодействия с информационными виджетами на основной форме. Описана логика проверок, выполняемых системой при проведении документов, и движений, формируемых в регистрах учета. Также определены основные входные и выходные данные программы.

Разработанная архитектура и реализованные алгоритмы обеспечивают выполнение всех функциональных требований, сформулированных в техническом задании, и создают основу для эффективной автоматизации библиотечных процессов. Структура системы является логичной и позволяет в дальнейшем развивать и модифицировать функционал АБИС.

ГЛАВА 3. ЭКСПЕРИМЕНТАЛЬНО-ПРИКЛАДНАЯ ЧАСТЬ

3.1. Тестирование и опытная эксплуатация программы

Тестирование программного обеспечения является неотъемлемым этапом разработки, позволяющим выявить и устранить ошибки, а также убедиться в соответствии системы заявленным требованиям. В рамках дипломного проекта было проведено функциональное тестирование основных модулей АБИС.

Подготовка к тестированию:

Для проведения тестирования была создана тестовая база данных, заполненная необходимыми начальными данными с помощью специально разработанных обработок. Были созданы записи в справочниках "Языки", "Жанры", "Авторы", "Издательства", "Читатели", "Сотрудники" и др. Также была имитирована начальная загрузка книжного фонда, выдача читательских билетов и выдача книг за предыдущий период.

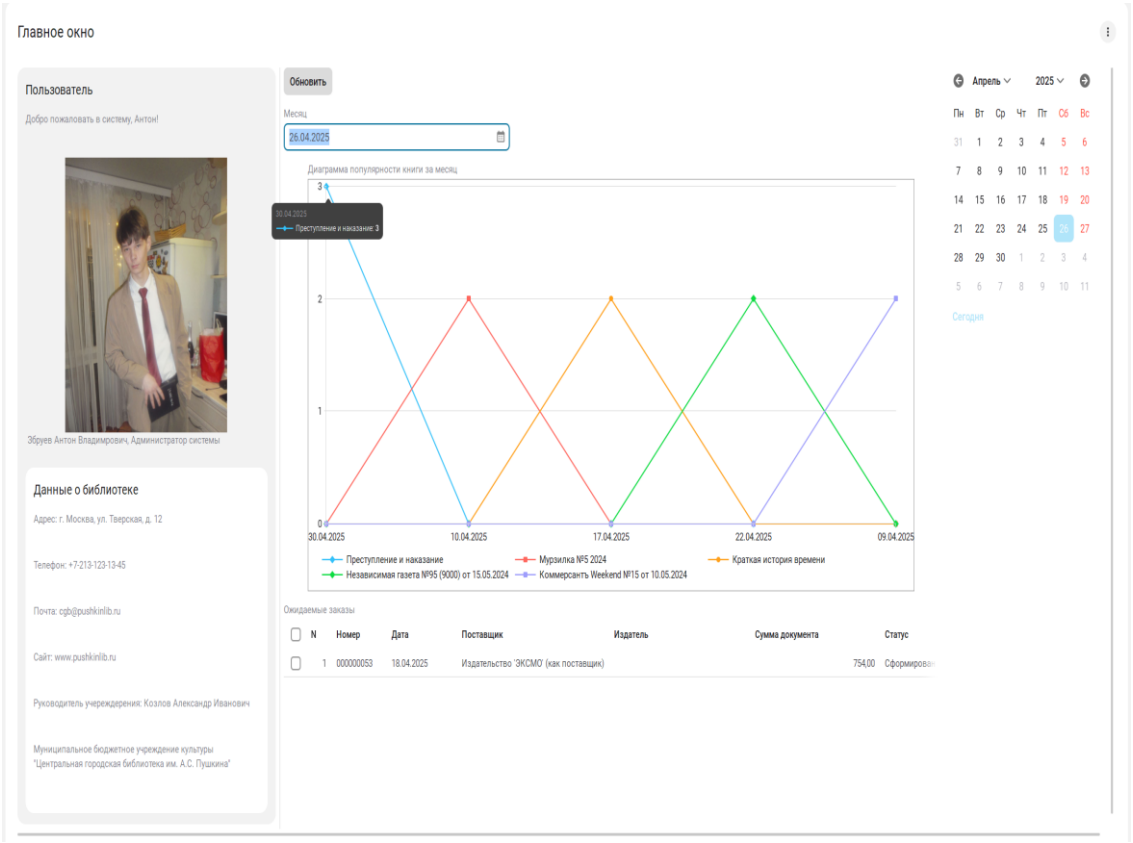


Рисунок 3.1 – Основной интерфейс АБИС

Тестирование основных сценариев:

Рассмотрим тестирование нескольких ключевых сценариев работы системы:

- **Сценарий 1: Регистрация читателя и выдача читательского билета.**
  - **Действия:** Библиотекарь входит в систему под своей учетной записью. Переходит в справочник "Читатели". Создает новую запись, заполняя ФИО, дату рождения и контактные данные.

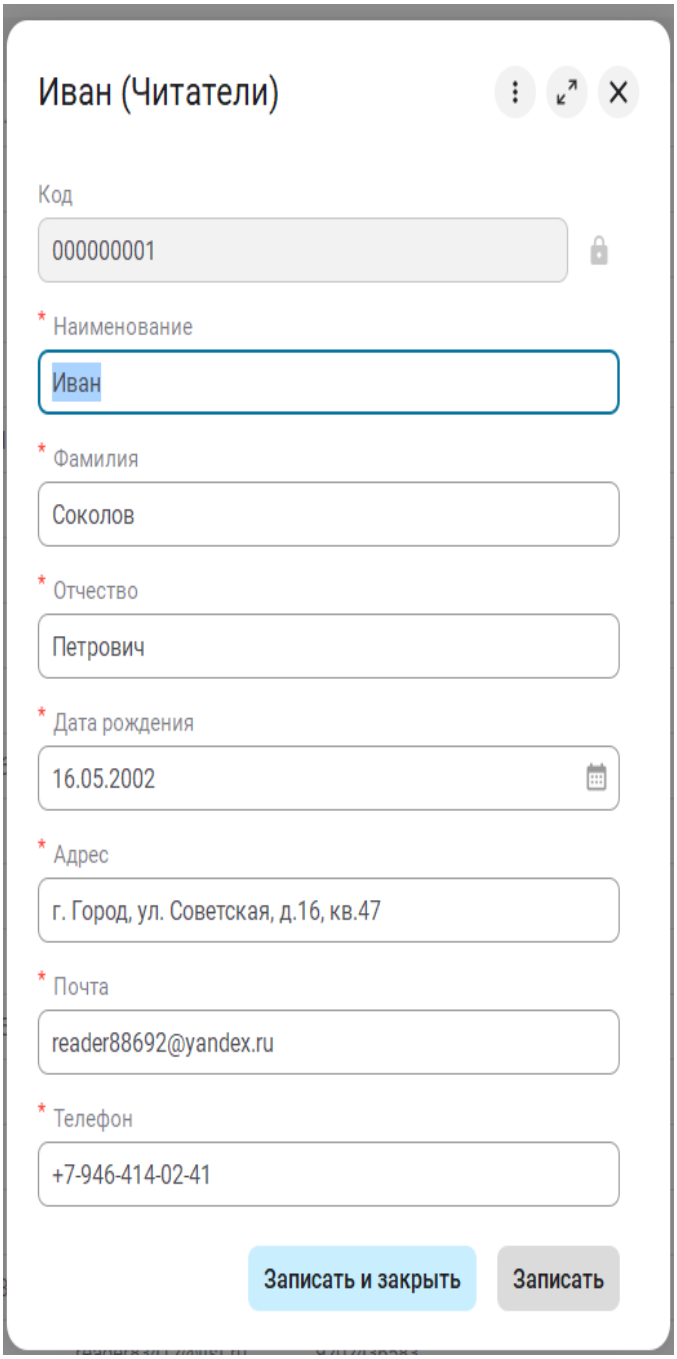
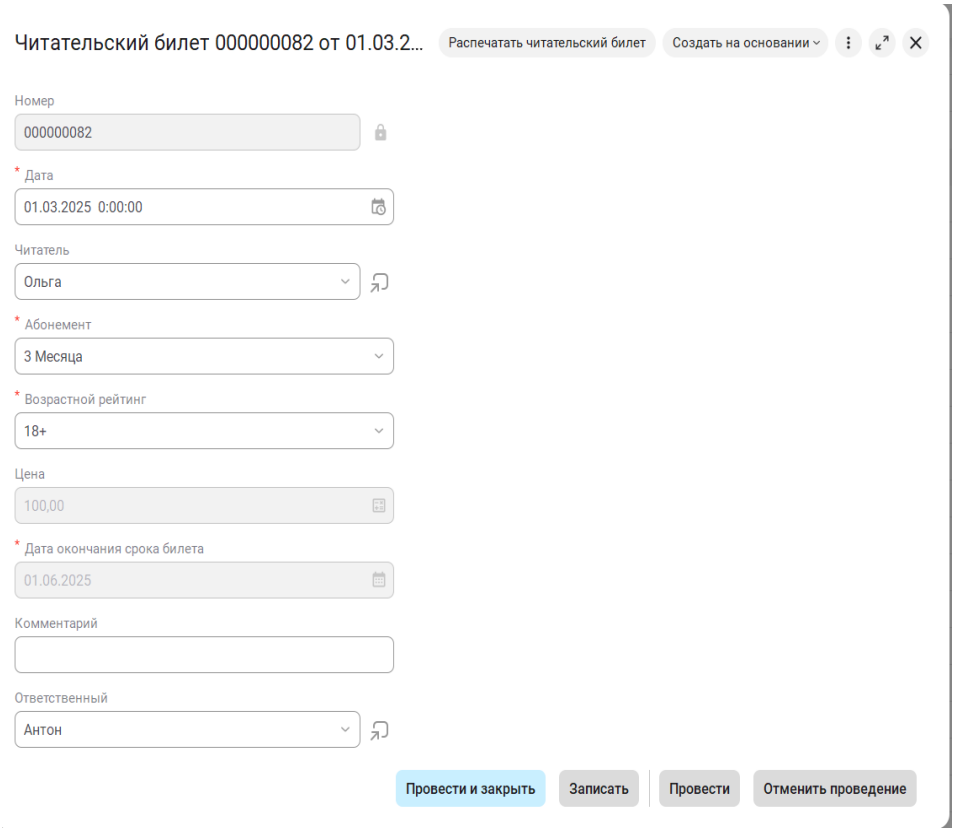


Рисунок 3.2 – Форма регистрации нового читателя

- Далее библиотекарь создает документ "ЧитательскийБилет". Выбирает созданного читателя, указывает тип абонемента (например, "Месяц_3") и возрастной рейтинг. Система автоматически рассчитывает цену и дату окончания срока действия билета.



3.3 – Оформление читательского билета

- Документ проводится.
- **Ожидаемый результат:** Запись о читателе успешно создана. Документ "ЧитательскийБилет" проведен, информация о сроке действия билета записана в регистр сведений "СрокиДействияЧитательскихБилетов".
- **Результат тестирования:** Сценарий выполнен успешно, ошибок не выявлено.

- **Сценарий 2: Выдача книги читателю.**
- **Действия:** Библиотекарь создает документ "ВыдачаКниг". Выбирает читателя (у которого есть действующий билет). В табличную часть добавляет одну или несколько книг из доступных. Устанавливает качество книги при выдаче и дату возврата (не более месяца).

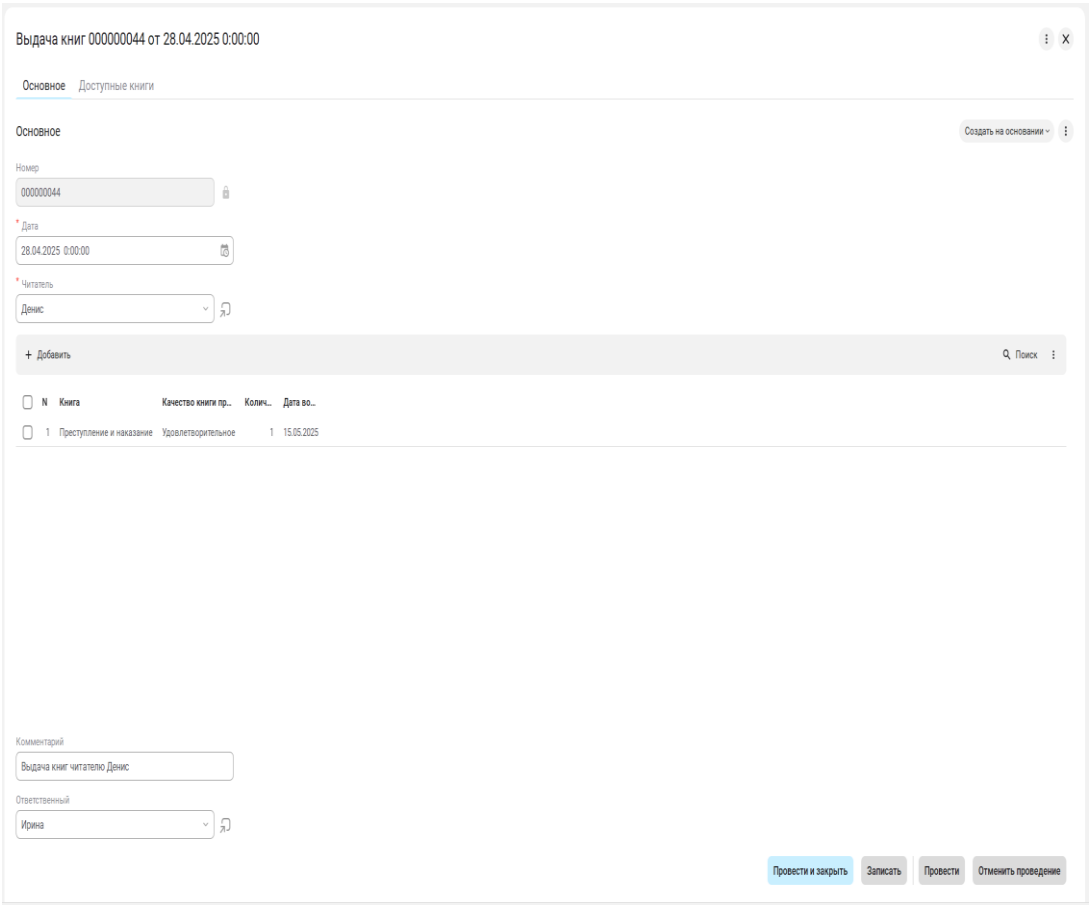


Рисунок 3.4 – Оформление выдачи книг

- Документ проводится.
- **Ожидаемый результат:** Система проверяет наличие книг и действительность билета. Документ успешно проводится. Формируются движения в регистрах "ДоступныеКниги" (расход), "ИсторияВыдачКниг" (приход), "ПопулярностьКниги" (приход), "СрокВозвратаКниг".
- **Результат тестирования:** Сценарий выполнен успешно. Проверка доступности книг и срока действия билета работает корректно. Движения по регистрам формируются правильно.

- **Сценарий 3: Возврат книги с просрочкой и начислением штрафа.**
  - **Действия:** Библиотекарь создает документ "ВозвратКниг" на основании ранее созданного документа "ВыдачаКниг". Дата документа устанавливается позже планируемой даты возврата. Указывается качество книги при возврате (отличное от качества при выдаче для проверки).

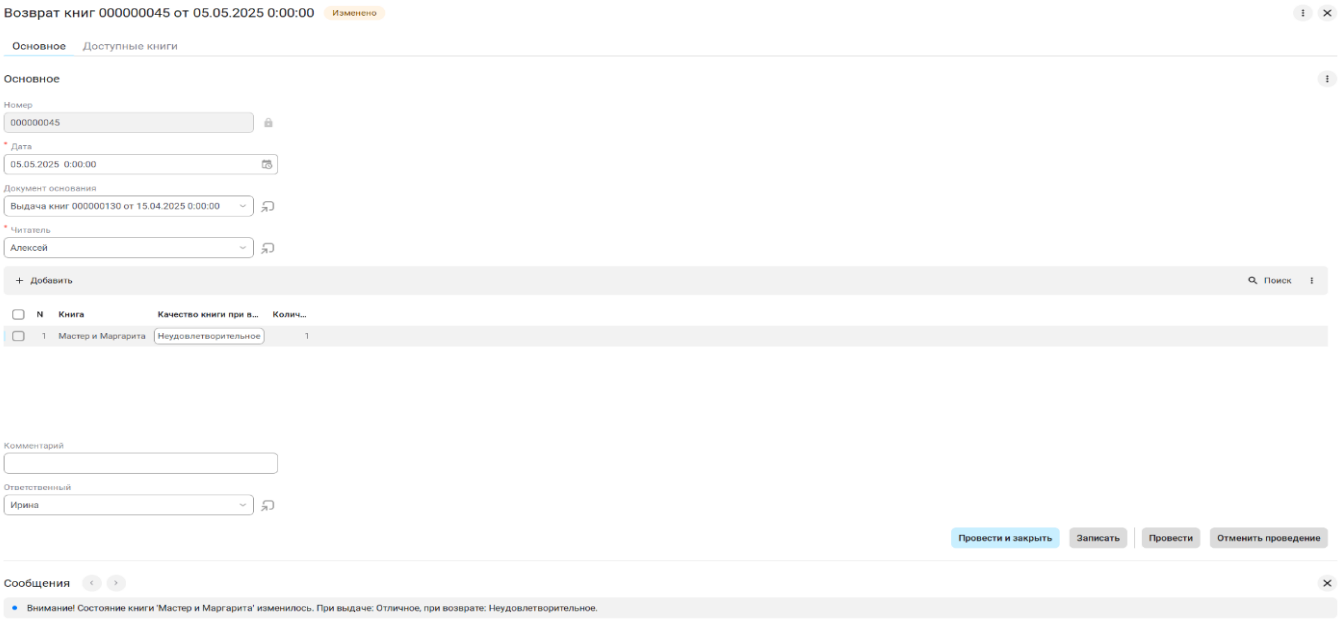


Рисунок 3.5 – Оформление возврата книг

- При изменении качества система выдает предупреждающее сообщение.
- Документ проводится.
- **Ожидаемый результат:** Документ проводится. Формируются движения в регистрах "ИсторияВыдачКниг" (расход), "ДоступныеКниги" (приход), "СрокВозвратаКниг" (запись фактической даты возврата). Автоматически рассчитывается и записывается штраф в регистр сведений "ШтрафыЗаПросрочку".
- **Результат тестирования:** Сценарий выполнен успешно. Сравнение качества книги при изменении в форме возврата работает. Штраф рассчитывается и регистрируется корректно при проведении.

- **Сценарий 4: Оплата штрафа.**
- **Действия:** Библиотекарь создает документ "ОплатаШтрафов". Выбирает читателя, имеющего неоплаченный штраф. Табличная часть автоматически заполняется неоплаченными штрафами этого читателя. Библиотекарь отмечает флажком штраф(ы) для оплаты. Сумма оплаты рассчитывается автоматически.

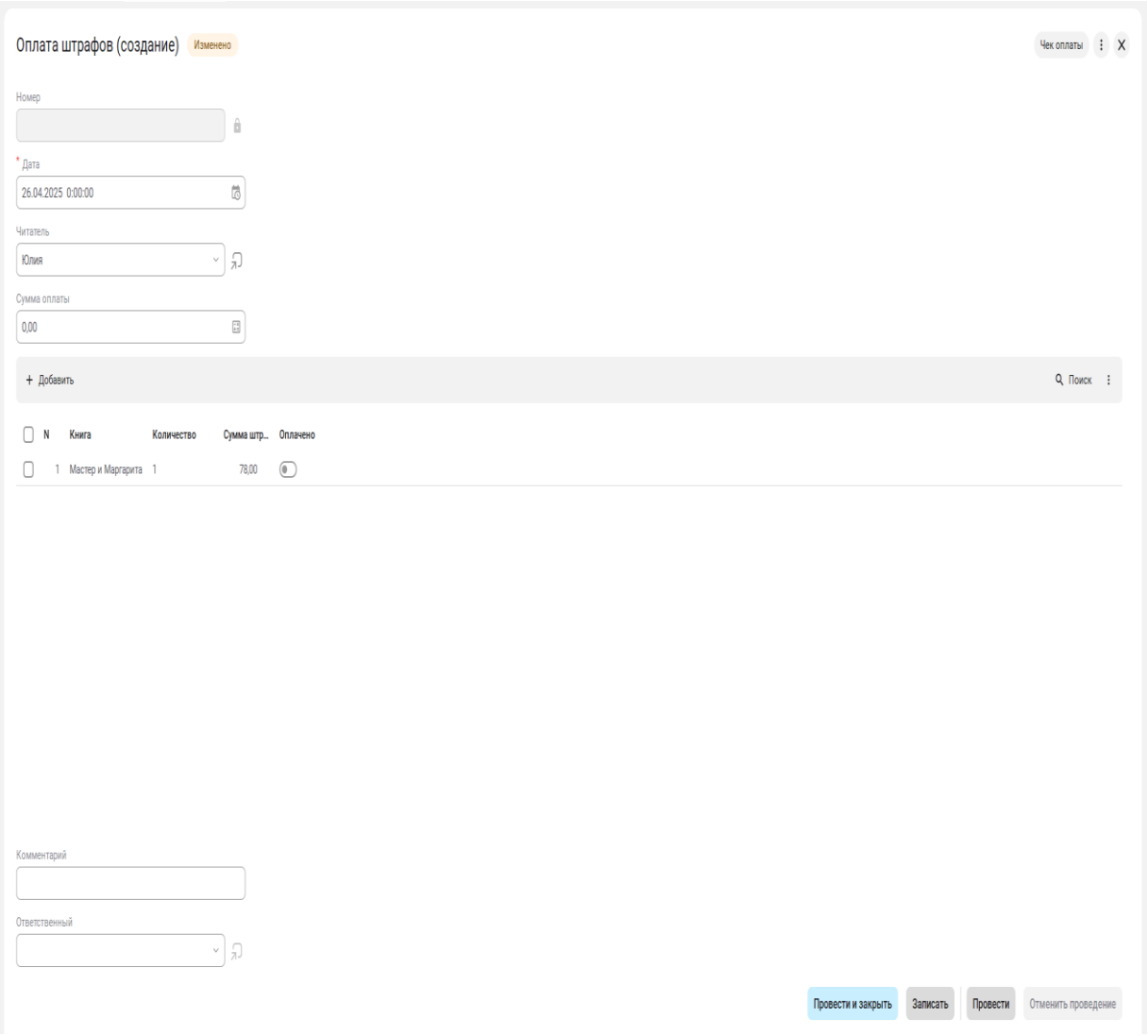


Рисунок 3.6 – Оформление оплаты штрафа

- Документ проводится.
- **Ожидаемый результат:** Документ проводится. В регистре сведений "ШтрафыЗаПросрочку" для оплаченного штрафа устанавливается флаг "Оплачено".
- **Результат тестирования:** Сценарий выполнен успешно. Статус штрафа обновляется корректно.

- **Сценарий 5: Закупка и приход книг.**
- **Действия:** Комплектатор создает документ "ЗакупкаКниг". Выбирает поставщика, заполняет табличную часть заказываемыми книгами, указывает количество и цену. Устанавливает статус "Согласовано". Документ записывается.

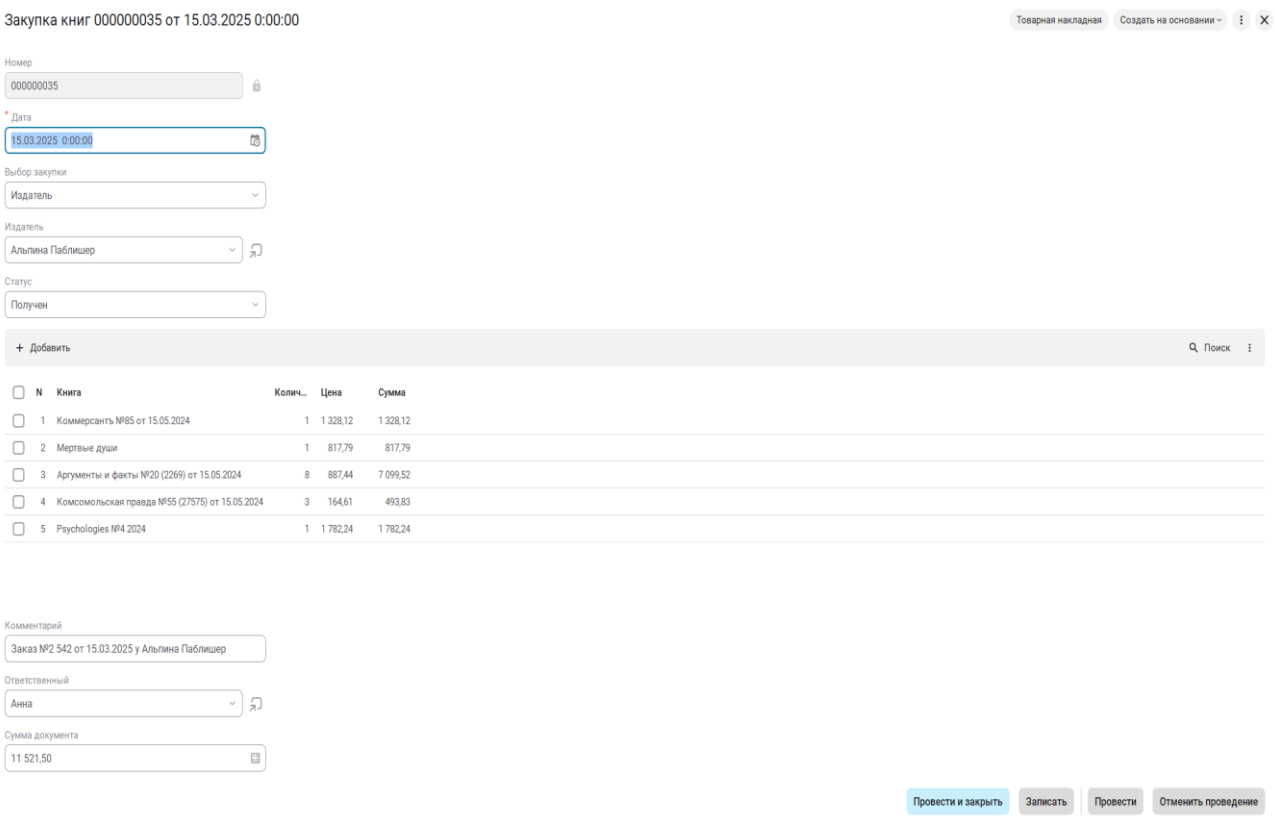


Рисунок 3.7 – Создание заказа на закупку

- После фактического получения книг комплектатор на основании документа "ЗакупкаКниг" создает документ "ПриходКниг". Данные автоматически переносятся. Пользователь проверяет и при необходимости корректирует количество.

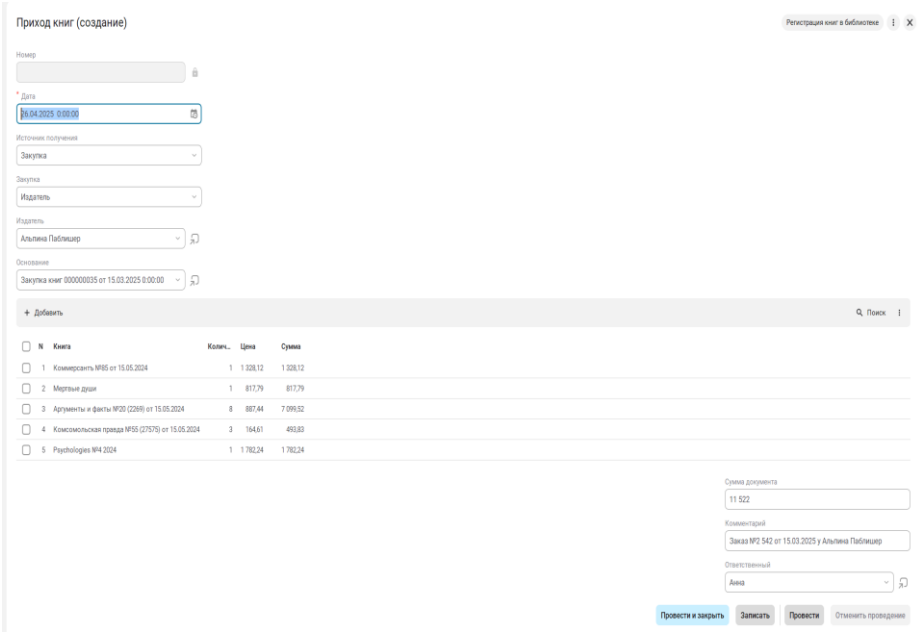


Рисунок 3.8 – Оформление прихода книг

- Документ "ПриходКниг" проводится.
- **Ожидаемый результат:** Документ "ЗакупкаКниг" записан. Документ "ПриходКниг" проведен. Книги оприходованы в регистрах "ОстаткиКниг" и "ДоступныеКниги". Информация о выполненном заказе отражена в регистре "ОборотЗаказовКниг" (расход по заказу).
- **Результат тестирования:** Сценарий выполнен успешно. Создание на основании работает, данные переносятся корректно. Движения по регистрам формируются правильно.

- **Сценарий 6: Списание книги.**

- **Действия:** Комплектатор создает документ "СписаниеКниг". В табличную часть добавляет книгу, подлежащую списанию, указывает количество и причину списания (например, "Износ").

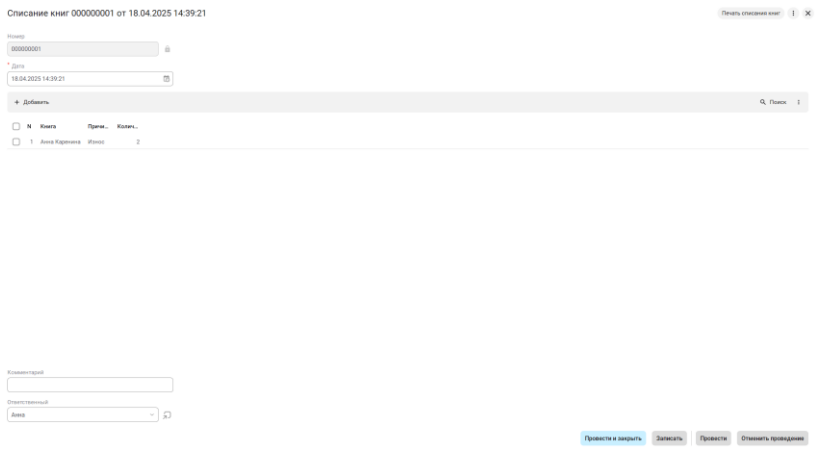


Рисунок 3.9 – Оформление списания книги

- Документ проводится.
- **Ожидаемый результат:** Документ проведен. Списанное количество отражено в регистрах "ОстаткиКниг" (расход) и "ДоступныеКниги" (расход).
- **Результат тестирования:** Сценарий выполнен успешно. Движения по регистрам формируются корректно.

- **Сценарий 7: Продление книги.**

- **Действия:** Библиотекарь создает документ "ПродлениеКниги" на основании документа "ВыдачаКниг". В табличной части для нужной книги указывает новую дату сдачи (не превышающую допустимый срок продления).

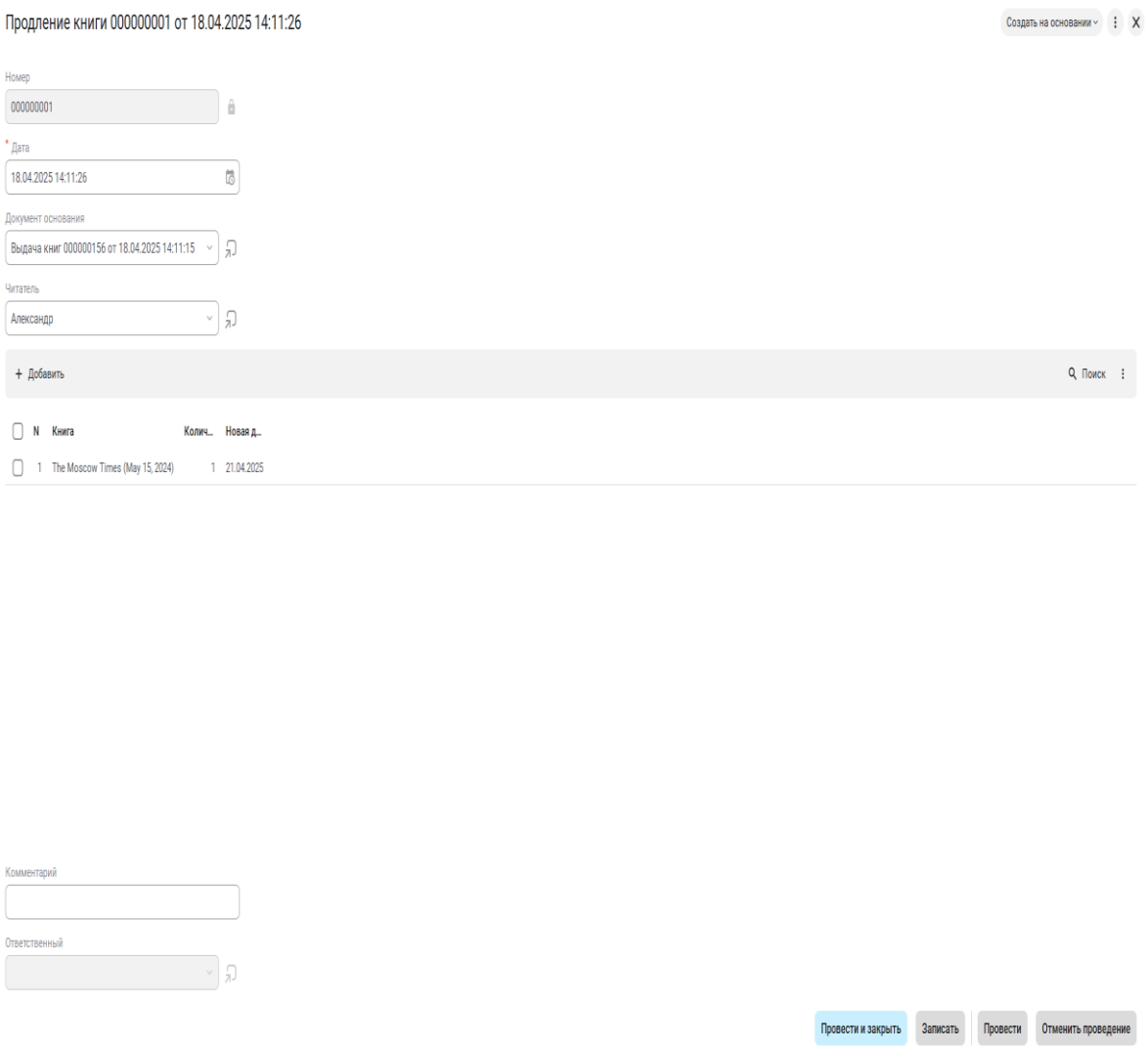


Рисунок 3.10 – Оформление продления книги

- Документ проводится.
- **Ожидаемый результат:** Документ проведен. В регистре сведений "СрокВозвратаКниг" создана новая запись с обновленной планируемой датой возврата.
- **Результат тестирования:** Сценарий выполнен успешно. Информация в регистре обновляется правильно.

- **Сценарий 8: Формирование отчета "Популярность книг".**

- **Действия:** Пользователь (например, руководитель или комплектатор) переходит в раздел отчетов. Выбирает отчет "ПопулярностьКнигСредиЧитателей". Устанавливает период (например, прошлый месяц). Нажимает "Сформировать".

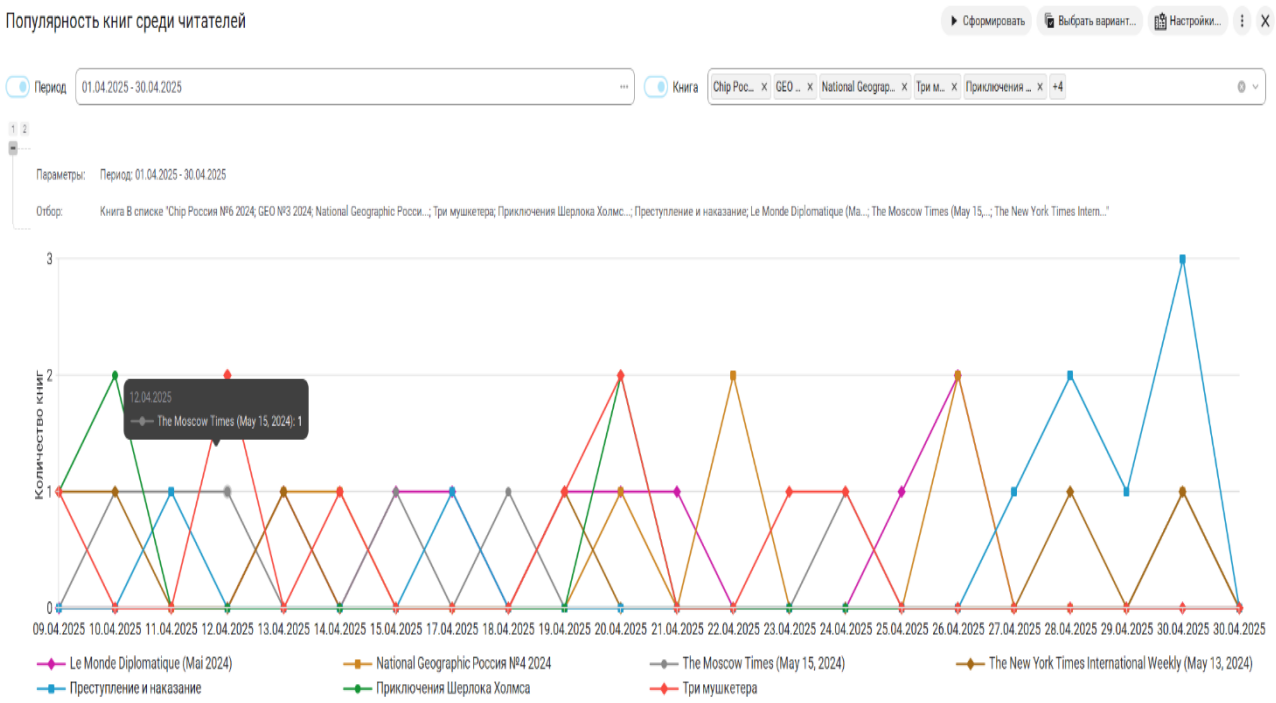


Рисунок 3.11 – Отчет по популярности книг

- **Ожидаемый результат:** Отчет сформирован, отображает список книг, отсортированный по количеству выдач за указанный период. Данные соответствуют движениям в регистре "ПопулярностьКниги".
- **Результат тестирования:** Отчет формируется корректно, данные соответствуют учетным данным.

Выводы по тестированию:

Проведенное функциональное тестирование показало, что основные модули и сценарии работы АБИС функционируют корректно в соответствии с требованиями технического задания. Выявленные в процессе тестирования незначительные ошибки были оперативно устранены. Система готова к опытной эксплуатации и дальнейшему развитию.

3.2. Руководство пользователя

Данное руководство предназначено для пользователей автоматизированной библиотечной информационной системы (АБИС) и описывает порядок выполнения основных операций.

1. Начало работы

- Запустите ярлык "1С:Предприятие" на рабочем столе или в меню "Пуск".
- В окне запуска выберите информационную базу вашей библиотеки.
- Введите ваш **Логин** и **Пароль**, предоставленные администратором системы.
- Нажмите кнопку "ОК".

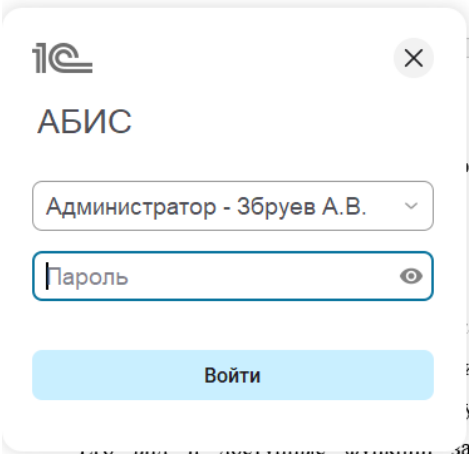


Рисунок 3.12 – Окно входа в систему

2. Основной интерфейс (Рабочий стол)

После успешного входа в систему откроется основной рабочий стол АБИС. Его вид и доступные функции зависят от вашей роли (Библиотекарь, Комплектатор фондов, Администратор).

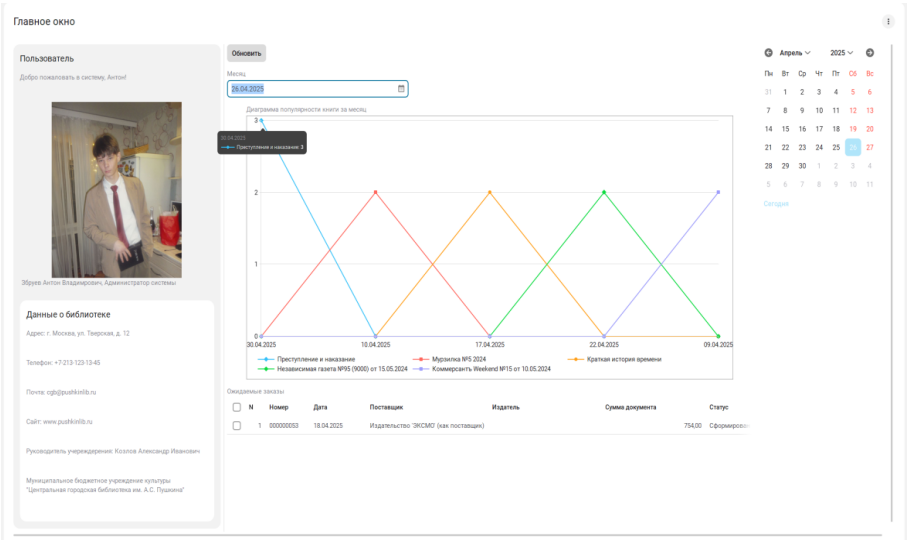


Рисунок 3.13 – Основной рабочий стол АБИС

Основные элементы интерфейса:

- **Панель разделов:** Обычно расположена слева или сверху и содержит основные функциональные блоки системы (например, "Главное", "Книжный фонд", "Читатели", "Обслуживание", "Закупки", "Отчеты", "Администрирование").
- **Панель навигации:** При выборе раздела в этой панели отображаются команды для перехода к справочникам, документам и отчетам, относящимся к данному разделу.
- **Рабочая область:** Основное пространство окна, где открываются списки, формы объектов (справочников, документов), отчеты и обработки.
- **Начальная страница (виджеты):** На начальной странице могут быть размещены информационные блоки (виджеты) для быстрого доступа к важной информации (например, данные о пользователе, ожидаемые заказы, диаграмма популярности книг).

3. Общие принципы работы

- **Списки:** Для просмотра и выбора данных используются формы списков (справочников, документов). В списках доступны функции сортировки (щелчком по заголовку колонки), поиска (в строке поиска над списком), отбора (через кнопку "Еще" -> "Настроить список").
- **Создание объектов:** Для создания нового элемента справочника или документа используется кнопка "Создать" в форме списка или соответствующая команда в панели навигации.
- **Редактирование объектов:** Для изменения существующего объекта выберите его в списке и нажмите кнопку "Изменить" (или иконку карандаша), либо дважды щелкните по строке.
- **Запись и проведение:** После внесения данных в форму документа необходимо его сохранить. Используйте кнопки:
  - "Записать": Сохраняет документ без формирования движений в регистрах.

- "Провести": Сохраняет документ и формирует движения в учетных регистрах.
- "Провести и закрыть": Проводит документ и закрывает его форму.

- **Печать:** Большинство документов и отчетов имеют кнопку "Печать" для вывода соответствующих печатных форм.

4. Основные операции (для роли Библиотекарь)

- **Регистрация нового читателя:**

1. Перейдите в раздел "Читатели".
2. В панели навигации выберите "Читатели".
3. Нажмите "Создать".
4. Заполните поля: ФИО, Дата рождения, Контакты, Адрес. (Рисунок 3.2)
5. Нажмите "Записать и закрыть".

- **Оформление читательского билета:**

1. Перейдите в раздел "Обслуживание" (или "Читатели").
2. Выберите "Читательские билеты".
3. Нажмите "Создать".
4. Выберите читателя.
5. Выберите тип "Абонемента" и "Возрастной рейтинг". Цена и дата окончания рассчитаются автоматически. (Рисунок 3.3)
6. Нажмите "Провести и закрыть".

- **Выдача книги:**

1. Перейдите в раздел "Обслуживание".
2. Выберите "Выдача книг".
3. Нажмите "Создать".
4. Укажите "Дату" выдачи.
5. Выберите "Читателя". Система проверит наличие действующего билета.
6. В табличной части "Книги" нажмите "Добавить".

7. Выберите "Книгу" из справочника (можно использовать поиск). Система проверит доступность книги.

8. Укажите "Качество книги при выдаче".

9. Установите планируемую "Дату возврата книги" (обычно не более месяца).

10. Повторите шаги 6-9 для других книг, если необходимо. (Рисунок 3.4)

11. Нажмите "Провести и закрыть".

- **Возврат книги:**

1. Перейдите в раздел "Обслуживание".

2. Выберите "Возврат книг".

3. Нажмите "Создать".

4. *Рекомендуемый способ:* Нажмите "Создать на основании" и выберите документ "Выдача книг", по которому возвращаются книги. Форма заполнится автоматически.

5. *Альтернативный способ:* Заполните "Дату", выберите "Читателя" и "Документ основания" (выдачи) вручную. Нажмите "Заполнить" -> "Заполнить по основанию".

6. В табличной части "Книги" для каждой возвращаемой книги укажите "Качество книги при возврате". Если оно отличается от качества при выдаче, система выдаст сообщение. (Рисунок 3.5)

7. Нажмите "Провести и закрыть". Система автоматически рассчитает и зарегистрирует штраф, если возврат просрочен.

- **Оплата штрафа:**

1. Перейдите в раздел "Обслуживание".

2. Выберите "Оплата штрафов".

3. Нажмите "Создать".

4. Выберите "Читателя". Табличная часть "Штрафы" заполнится его неоплаченными штрафами.

5. Установите флажок "Оплачено" для тех штрафов, которые оплачиваются. Сумма к оплате рассчитается автоматически. (Рисунок 3.6)

6. Нажмите "Провести и закрыть".

5. Основные операции (для роли Комплектатор фондов)

- **Создание заказа на закупку:**

1. Перейдите в раздел "Закупки".
2. Выберите "Заказы поставщикам" (или "Закупки книг").
3. Нажмите "Создать".
4. Выберите "Поставщика" или "Издательство".
5. Заполните табличную часть "Книги", указав книги, количество и цену.
6. Установите статус заказа.
7. Нажмите "Записать" или "Провести". (Рисунок 3.7)

- **Оформление прихода книг:**

1. Перейдите в раздел "Закупки" или "Книжный фонд".
2. Выберите "Приход книг".
3. Нажмите "Создать".
4. *Рекомендуемый способ:* Нажмите "Создать на основании" и выберите документ "Закупка книг".
5. *Альтернативный способ:* Заполните шапку вручную (источник, поставщик/издатель), затем табличную часть.
6. Проверьте/скорректируйте количество и цены. (Рисунок 3.8)
7. Нажмите "Провести и закрыть".

- **Списание книг:**

1. Перейдите в раздел "Книжный фонд".
2. Выберите "Списание книг".
3. Нажмите "Создать".
4. В табличной части "Книги" нажмите "Добавить".
5. Выберите "Книгу" для списания, укажите "Количество" и "Причину списания". (Рисунок 3.9)
6. Нажмите "Провести и закрыть".

6. Формирование отчетов

- Перейдите в раздел "Отчеты".
- Выберите необходимый отчет из списка (например, "Остатки книг", "Задолженности читателей").
- В форме отчета установите параметры (период, отборы по читателю, книге и т.п.).
- Нажмите кнопку "Сформировать". (Рисунок 3.11)

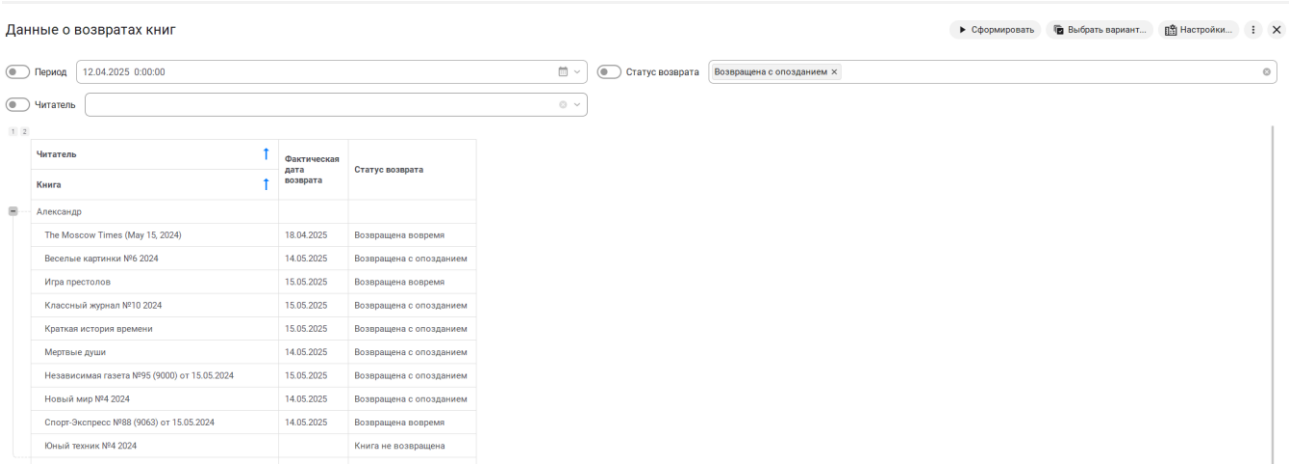


Рисунок 3.14 – Пример отчета "Задолженности читателей"

- Сформированный отчет можно распечатать или сохранить в файл (через меню "Файл" или кнопку "Еще").

7. Завершение работы

- Для выхода из программы закройте главное окно приложения или выберите пункт меню "Файл" -> "Выход".

ВЫВОДЫ ПО ГЛАВЕ 3

В третьей главе дипломного проекта была представлена экспериментально-прикладная часть работы, посвященная проверке работоспособности и описанию использования разработанной автоматизированной библиотечной информационной системы (АБИС).

В рамках функционального тестирования были проверены ключевые сценарии использования системы, охватывающие основные бизнес-процессы библиотеки. Тестирование включало проверку регистрации читателей и выдачи читательских билетов, операций выдачи и возврата книг (в том числе с просрочкой и начислением штрафов), оплаты штрафов, процессов закупки и прихода книг, списания литературы, продления книг и формирования основных отчетов. По результатам тестирования можно сделать вывод о корректной работе реализованного функционала и соответствии системы требованиям, изложенным в техническом задании. Обнаруженные незначительные недочеты были устранены.

Для пользователей системы было разработано подробное руководство, описывающее порядок выполнения основных операций: от входа в систему и навигации по интерфейсу до работы со справочниками, документами и отчетами. Руководство пользователя содержит пошаговые инструкции и иллюстрировано скриншотами интерфейса программы, что облегчает освоение системы сотрудниками библиотеки с разным уровнем компьютерной грамотности.

Таким образом, экспериментально-прикладная часть работы подтвердила работоспособность разработанной АБИС и ее готовность к практическому применению. Тестирование показало стабильность системы и корректность выполнения основных функций, а подготовленное руководство пользователя обеспечивает необходимую поддержку для эффективного использования системы сотрудниками библиотеки.

ЗАКЛЮЧЕНИЕ

В рамках данного дипломного проекта была успешно решена задача разработки автоматизированной библиотечной информационной системы (АБИС) на платформе "1С:Предприятие 8.5. Актуальность данной разработки обусловлена необходимостью модернизации библиотечных процессов, повышения эффективности управления фондами и улучшения качества обслуживания читателей в современных условиях.

В ходе работы над проектом были выполнены все поставленные задачи. Проведен детальный анализ предметной области, позволивший выявить ключевые бизнес-процессы библиотеки и сформулировать четкие требования к функциональности информационной системы. На основе этого анализа было разработано техническое задание, которое легло в основу проектирования и разработки АБИС.

Была спроектирована архитектура системы, основанная на клиент-серверной технологии платформы "1С:Предприятие", обеспечивающая надежность, производительность и масштабируемость решения. Разработана структура базы данных, включающая необходимые справочники, документы, регистры сведений и накопления для хранения и обработки всей информации, связанной с деятельностью библиотеки.

Реализованы основные функциональные модули системы, автоматизирующие процессы учета книжного фонда (поступление, списание, инвентаризация), обслуживания читателей (регистрация, выдача и возврат книг, продление, учет билетов и штрафов), закупки литературы и учета пожертвований. Разработан интуитивно понятный пользовательский интерфейс и настроена ролевая модель доступа, соответствующая структуре персонала библиотеки. Реализован набор отчетов, предоставляющих необходимую аналитическую информацию для принятия управленческих решений.

Проведено функциональное тестирование разработанной системы, подтвердившее ее работоспособность и соответствие заявленным требованиям.

Выявленные в ходе тестирования недочеты были устранены. Подготовлено руководство пользователя, облегчающее освоение системы сотрудниками библиотеки.

Таким образом, цель дипломного проекта – создание программного обеспечения для комплексной автоматизации библиотеки – была достигнута. Разработанная АБИС представляет собой готовое к внедрению решение, способное значительно повысить эффективность работы библиотеки, оптимизировать использование ресурсов и улучшить качество предоставляемых услуг. Система обладает потенциалом для дальнейшего развития и модификации, например, путем интеграции с веб-сайтом библиотеки или добавления новых функциональных модулей.

СПИСОК ЛИТЕРАТУРЫ

1. ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов. – Введ. 1980-01-01. – М.: Издательство стандартов, 1978. – 9 с.

2. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения. – Введ. 1992-01-01. – М.: Издательство стандартов, 1991. – 27 с.

3. ГОСТ Р 7.0.5-2008 СИБИД. Библиографическая ссылка. Общие требования и правила составления. – Введ. 2009-01-01. – М.: Стандартинформ, 2008. – 23 с.

4. ГОСТ Р ИСО/МЭК 12207-2010 Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств. – Введ. 2012-01-01. – М.: Стандартинформ, 2011. – 147 с.

5. Гартсуева, Е. А. 1С:Предприятие 8.3. Практическое пособие разработчика. Примеры и типовые приемы / Е.А. Гартсуева. – М.: ООО "1С-Паблишинг", 2020. – 977 с.

6. Радченко, М. Г. 1С:Программирование для начинающих. Разработка в системе 1С:Предприятие 8.3 / М. Г. Радченко. – М.: ООО "1С-Паблишинг", 2022. – 494 с.

7. Хрусталева, Е. Ю. Разработка сложных отчетов в "1С:Предприятии 8". Система компоновки данных / Е.Ю. Хрусталева. – М.: ООО "1С-Паблишинг", 2019. – 458 с.

8. Ажеронок, В.А. Разработка управляемого интерфейса / В.А. Ажеронок, А.В. Островерх. – М.: ООО «1С-Паблишинг», 2018. – 374 с.

9. Насибов, И. И. Секреты профессиональной работы с «1С:Предприятие 8.3». Анализ и разработка конфигураций / И.И. Насибов. – СПб.: БХВ-Петербург, 2021. – 688 с.

10. Чистов, Д.В. Хозяйственные операции в "1С:Бухгалтерии 8". Задачи, решения, результаты / Д.В. Чистов, С.А. Харитонов. - М.: ООО "1С-Паблишинг", 2021. - 477 с.

11. Островерх А.В. Практика применения 1С:Библиотеки стандартных подсистем / А.В. Островерх. – М.: ООО «1С-Паблишинг», 2020. - 654 с.

12. Бойко, В.В. Проектирование баз данных информационных систем / В.В. Бойко, В.М. Савинков. – М.: Финансы и статистика, 2002. – 352 с.

13. Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг. – 3-е изд. – М.: Вильямс, 2016. – 1436 с.

14. Вигерс, К. Разработка требований к программному обеспечению / К. Вигерс, Дж. Битти. – 3-е изд. – М.: Русская Редакция, 2014. – 736 с.

15. Макконнелл, С. Совершенный код. Мастер-класс / С. Макконнелл. – М.: Русская Редакция, 2017. – 896 с.

16. 1С:Предприятие 8. Руководство разработчика [Электронный ресурс]. – Режим доступа: https://its.1c.ru/db/v8devdoc

17. Встроенный язык. Синтакс-помощник [Электронный ресурс] // 1С:Предприятие 8. Информационно-технологическое сопровождение. – Режим доступа: https://its.1c.ru/db/v83doc/bookmark/dev/SyntaxHelper.

18. Система стандартов и методик разработки конфигураций для платформы 1С:Предприятие 8 [Электронный ресурс] // Фирма "1С". – Режим доступа: https://its.1c.ru/db/v8std.

19. Основы работы с запросами в 1С [Электронный ресурс] // Инфостарт. – Режим доступа: https://infostart.ru/public/15708/

20. Работа с формами в 1С: Управляемое приложение [Электронный ресурс] // Хабр. – Режим доступа: https://habr.com/ru/articles/186150/.

21. Проведение документов в 1С [Электронный ресурс] // Программист1С.РФ. – Режим доступа: https://программист1с.рф/stati/provedenie-dokumentov-1s.html

22. Использование регистров накопления в 1С [Электронный ресурс] // 1С: Лекторий. – Режим доступа: https://its.1c.ru/lector/.

23. Работа с хранилищем значения в 1С [Электронный ресурс] // Клерк.ру. – Режим доступа: https://www.klerk.ru/1c/articles/512367/.

24. Разработка печатных форм в 1С: Управляемое приложение [Электронный ресурс] // Инфостарт. – Режим доступа: https://infostart.ru/public/115953/.

25. Моделирование бизнес-процессов. Нотация UML. Use Case Diagram [Электронный ресурс] // intuit.ru. – Режим доступа: https://intuit.ru/studies/courses/6/6/lecture/174.

ПРИЛОЖЕНИЯ

Приложение 1

Компакт-диск с материалами проекта

На диске располагается:

- Проект программы.
- Файл дипломного проекта в формате MS Word

Приложение 2

КОД ПРОГРАММЫ

Код общего модуля «ОбщийМодульФункции»

```
Функция ЦеныИзРегистраСведений
(Дата,КнигиИзРегистра) Экспорт
	Отбор = Новый
Структура("Книги",КнигиИзРегистра);
	ЗначениеЦены =
РегистрыСведений.Цены.ПолучитьПоследнее(Дата,
Отбор);
	Возврат ЗначениеЦены.Цена;
КонецФункции


Функция ПолучитьЦенуБилета(Абонемент,
ВозрастнойРейтинг) Экспорт

	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ ПЕРВЫЕ 1
	|	ЦеныЧитательскихБилетов.Цена
	|ИЗ
	|
РегистрСведений.ЦеныЧитательскихБилетов КАК
ЦеныЧитательскихБилетов
	|ГДЕ
	|	ЦеныЧитательскихБилетов.Абонемент =
&Абонемент
	|	И
ЦеныЧитательскихБилетов.ВозрастнойРейтинг =
&ВозрастнойРейтинг
	|УПОРЯДОЧИТЬ ПО
	|	ЦеныЧитательскихБилетов.Период
УБЫВ";

	Запрос.УстановитьПараметр("Абонемент",
Абонемент);
	Запрос.УстановитьПараметр("ВозрастнойР
ейтинг", ВозрастнойРейтинг);
	РезультатЗапроса = Запрос.Выполнить();
	Выборка = РезультатЗапроса.Выбрать();

	Если Выборка.Следующий() Тогда
		Возврат Выборка.Цена;
	Иначе
		Возврат 0; // Или другое значение
по умолчанию, или выбросить исключение
	КонецЕсли;

КонецФункции

Функция ПолучитьЦенуШтрафа(Штраф) Экспорт

   Запрос = Новый Запрос;
   Запрос.Текст =
   "ВЫБРАТЬ ПЕРВЫЕ 1
   |	ЦеныШтрафов.Цена
   |ИЗ
   |	РегистрСведений.ЦеныШтрафов КАК
ЦеныШтрафов
   |ГДЕ
   |	ЦеныШтрафов.Штраф = &Штраф
   |УПОРЯДОЧИТЬ ПО
   |	ЦеныШтрафов.Период УБЫВ";

   Запрос.УстановитьПараметр("Штраф", Штраф);
// Передаем параметр в запрос

   Результат = Запрос.Выполнить();
   Выборка = Результат.Выбрать();

   Если Выборка.Следующий() Тогда
      Возврат Выборка.Цена;
   Иначе
      Возврат 0; // Или другое значение по
умолчанию, или выбросить исключение
   КонецЕсли;
```

```
КонецФункции
```

Код общего модуля «ВалидацияПолей»

```
&НаСервере
Процедура ПроверитьEmail(Почта, ЭлементПочты = Неопределено) Экспорт

	Шаблон = ".+@.+\..+"; // Регулярное выражение для проверки email

	СтрокаАдреса = Почта;

	RegExp = Новый COMОбъект("VBScript.RegExp");
	RegExp.MultiLine = Ложь;
	RegExp.Global = Истина;
	RegExp.IgnoreCase = Истина;
	RegExp.Pattern = Шаблон;

	Если НЕ RegExp.Test(Почта) Тогда
		ВызватьИсключение "Некорректный email адрес. Пожалуйста, введите адрес в формате ""name@domain.com"".";
	КонецЕсли;

КонецПроцедуры


Функция КорректноВведенНомерТелефона(НомерТелефона) Экспорт

	СимволыНомера = "+1234567890";
	СимволыРазделители = "()- ";

	ФорматированныйНомер = "";
	Для Позиция = 1 По СтрДлина(НомерТелефона) Цикл
		Символ = Сред(НомерТелефона, Позиция, 1);
		Если Символ = "+" И Не ПустаяСтрока(ФорматированныйНомер) Тогда
			Возврат Ложь;
		КонецЕсли;
		Если СтрНайти(СимволыНомера, Символ) > 0 Тогда
			ФорматированныйНомер = ФорматированныйНомер + Символ;
		ИначеЕсли СтрНайти(СимволыРазделители, Символ) = 0 Тогда
			Возврат Ложь;
		КонецЕсли;
	КонецЦикла;

	Если ПустаяСтрока(ФорматированныйНомер) Тогда
		Возврат Ложь;
	КонецЕсли;

	Если СтрДлина(ФорматированныйНомер) < 3 Тогда
		Возврат Ложь;
	КонецЕсли;

	Возврат Истина;

КонецФункции
```

Код модуля объекта справочника «Книги»

```
&НаСервере
Процедура ПередУдалением(Отказ)
   // Получаем остаток книги в регистре "ОстаткиКниг"
   Остаток = ПолучитьОстатокКниги(ЭтотОбъект.Ссылка);

   // Если остаток больше 0, запрещаем удаление
   Если Остаток < 0 Тогда
      Сообщить("Нельзя удалить книгу, так как она есть в наличии в библиотеке. Остаток: " + Остаток);
      Отказ = Истина;
   КонецЕсли;
КонецПроцедуры
```

```
&НаСервере
Функция ПолучитьОстатокКниги(Книга)
   Запрос = Новый Запрос;
   Запрос.Текст =
      "ВЫБРАТЬ
      |   ОстаткиКниг.Количество
      |ИЗ
      |   РегистрНакопления.ОстаткиКниг КАК
ОстаткиКниг
      |ГДЕ
      |   ОстаткиКниг.Книги = &Книга";

   Запрос.УстановитьПараметр("Книга", Книга);

   Результат = Запрос.Выполнить().Выбрать();

   Если Результат.Следующий() Тогда
      Возврат Результат.Количество;
   Иначе
      Возврат 0;
   КонецЕсли;
КонецФункции
```

Код формы элемента справочника «Книги»

```
&НаКлиенте
Процедура ВставитьКартинку(Команда)

        Оповещение = Новый
ОписаниеОповещения("ОбработатьВыборФайла",Э
тотОбъект);
        НачатьПомещениеФайла(Оповещение,,,Ис
тина, УникальныйИдентификатор);

КонецПроцедуры

&НаКлиенте
Процедура ОбработатьВыборФайла(Результат,
Адрес,
ВыбранноеИмяФайла,ДополнительныйПараметр)
Экспорт

        Если НЕ Результат Тогда
            Возврат;
        КонецЕсли;
        ССылкаНаКартинку = Адрес;

КонецПроцедуры

&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ,
ТекущийОбъект, ПараметрыЗаписи)

        Если
ЭтоАдресВременногоХранилища(ССылкаНаКартин
ку) Тогда
            ТекущийОбъект.ДанныеКартинки
= Новый

        ХранилищеЗначения(ПолучитьИзВременн
огоХранилища(ССылкаНаКартинку));
        КонецЕсли;

КонецПроцедуры

&НаСервере
Процедура ПриСозданииНаСервере(Отказ,
СтандартнаяОбработка)

        ССылкаНаКартинку =
ПолучитьНавигационнуюССылку(Объект.Ссылка,
"ДанныеКартинки");

КонецПроцедуры

&НаСервере
Функция ПолучитьОстаток(Книга)

        Результат = 0;
        Отбор = Новый Структура;
        Отбор.Вставить("Книги", Книга);
        НайденныеЗначения =
РегистрыНакопления.ОстаткиКниг.Остатки(Текущ
аяДата(), Отбор);
```

```
        Если НайденныеЗначения.Количество() > 0
Тогда
                Результат =
НайденныеЗначения[0].Количество;
        КонецЕсли;
        Возврат Результат;

КонецФункции
```

Код формы списка справочника «Книги»

```
Процедура ПриСозданииНаСервере(Отказ,
СтандартнаяОбработка)

        ОбновитьОстаткиКниг();
        УстановитьУсловноеОформление();


КонецПроцедуры

Процедура УстановитьУсловноеОформление()

        УсловноеОформлениеСписка=Список.Ком
поновщикНастроек.Настройки.УсловноеОформлен
ие;

        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       Статус.Ссылка КАК Ссылка,
        |       Статус.Цвет КАК Цвет
        |ИЗ
        |       Справочник.Статус КАК Статус";

        РезультатЗапроса = Запрос.Выполнить();

        ВыборкаДетальныеЗаписи =
РезультатЗапроса.Выбрать();

        Пока
ВыборкаДетальныеЗаписи.Следующий() Цикл

                ЭлементОформления =
УсловноеОформлениеСписка.Элементы.Добавить()
;


        ЭлементОформления.Оформление.Установ
итьЗначениеПараметра("ЦветФона",
ВыборкаДетальныеЗаписи.Цвет.Получить());

                ЭлементОтбора =
ЭлементОформления.Отбор.Элементы.Добавить(Т
ип("ЭлементОтбораКомпоновкиДанных"));

        ЭлементОтбора.ЛевоеЗначение=Новый
ПолеКомпоновкиДанных("Статус");
                ЭлементОтбора.ВидСравнения =
ВидСравненияКомпоновкиДанных.Равно;
                ЭлементОтбора.ПравоеЗначение =
ВыборкаДетальныеЗаписи.Ссылка;
        КонецЦикла;


КонецПроцедуры

&НаСервере
Процедура ОбновитьОстаткиКниг()

        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |   Книги.Ссылка
        |ИЗ
        |   Справочник.Книги КАК Книги"; //
Выбираем все книги

        Результат = Запрос.Выполнить();
        Выборка = Результат.Выбрать();

        Пока Выборка.Следующий() Цикл

                Книга =
Выборка.Ссылка.ПолучитьОбъект();
```

```
                Книга.КоличествоНаСкладе =
ПолучитьОстаток(Выборка.Ссылка);

                // Проверка и установка статуса
                Если Книга.КоличествоНаСкладе >
0 Тогда
                        Книга.Статус =
Справочники.Статус.НайтиПоНаименованию("Есть
на складе");
                Иначе
                        Книга.Статус =
Справочники.Статус.НайтиПоНаименованию("Отс
утствует на складе");
                КонецЕсли;

                Попытка
                        Книга.Записать();
                Исключение
                        Сообщить("Ошибка при
записи книги " + Книга + ": " +
ОписаниеОшибки());
                КонецПопытки;
        КонецЦикла;

        Элементы.Список.Обновить();

КонецПроцедуры

&НаСервере
Функция ПолучитьОстаток(Книга)

        Результат = 0;
        Отбор = Новый Структура;
        Отбор.Вставить("Книги", Книга);
        НайденныеЗначения =
РегистрыНакопления.ОстаткиКниг.Остатки(Текущ
аяДата(), Отбор);
        Если НайденныеЗначения.Количество() > 0
Тогда
                Результат =
НайденныеЗначения[0].Количество;
        КонецЕсли;
        Возврат Результат;

КонецФункции
```

Код формы элемента справочника «Статус»

```
Процедура ПередЗаписьюНаСервере(Отказ,
ТекущийОбъект, ПараметрыЗаписи)

        ЗначениеЦвета = Цвет;
        Если ЗначениеЦвета = Новый Цвет(0,0,0)
Тогда
                ЗначениеЦвета = Неопределено;
        КонецЕсли;
        ТекущийОбъект.Цвет=Новый
ХранилищеЗначения(ЗначениеЦвета);

КонецПроцедуры

Процедура ПриЧтенииНаСервере(ТекущийОбъект)

        Цвет = ТекущийОбъект.Цвет.Получить();

КонецПроцедуры
```

Код формы элемента справочника «Поставщики»

```
&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ,
ТекущийОбъект, ПараметрыЗаписи)

        Попытка

        ВалидацияПолей.ПроверитьEmail(Текущий
Объект.Почта);
        Исключение
                Сообщить(ОписаниеОшибки());
                Отказ = Истина;
                Возврат;
        КонецПопытки;
```

```
        Если НЕ
ВалидацияПолей.КорректноВведенНомерТелефона
(ТекущийОбъект.Телефон) Тогда
                ВызватьИсключение
"Некорректный номер телефона. Пожалуйста,
введите номер в международном формате.";
        КонецЕсли;

КонецПроцедуры
```

Код формы элемента справочника «Читатели»

```
&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ,
ТекущийОбъект, ПараметрыЗаписи)

        Попытка

        ВалидацияПолей.ПроверитьEmail(Текущий
Объект.Почта);
        Исключение
                Сообщить(ОписаниеОшибки());
                Отказ = Истина;
                Возврат;
        КонецПопытки;


        Если НЕ
ВалидацияПолей.КорректноВведенНомерТелефона
(ТекущийОбъект.Телефон) Тогда
                ВызватьИсключение
"Некорректный номер телефона. Пожалуйста,
введите номер в международном формате.";
        КонецЕсли;

КонецПроцедуры
```

Код формы элемента справочника «Сотрудники»

```
&НаКлиенте
Процедура ВставитьФото(Команда)
        Оповещение = Новый
ОписаниеОповещения("ОбработатьВыборФайла",Э
тотОбъект);
        НачатьПомещениеФайла(Оповещение,,,Ис
тина, УникальныйИдентификатор);

КонецПроцедуры

&НаКлиенте
Процедура ОбработатьВыборФайла(Результат,
Адрес,
        ВыбранноеИмяФайла,ДополнительныйПар
аметр) Экспорт

        Если НЕ Результат Тогда
                Возврат;
        КонецЕсли;
        ССылкаНаКартинку = Адрес;

КонецПроцедуры

&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ,
ТекущийОбъект, ПараметрыЗаписи)

        Если
ЭтоАдресВременногоХранилища(ССылкаНаКартин
ку) Тогда
                ТекущийОбъект.Фотография =
Новый

        ХранилищеЗначения(ПолучитьИзВременн
огоХранилища(ССылкаНаКартинку));
        КонецЕсли;

        Попытка

        ВалидацияПолей.ПроверитьEmail(Текущий
Объект.Почта);
        Исключение
                Сообщить(ОписаниеОшибки());
                Отказ = Истина;
```

```
            Возврат;
        КонецПопытки;

        Если НЕ
ВалидацияПолей.КорректноВведенНомерТелефона
(ТекущийОбъект.Телефон) Тогда
            ВызватьИсключение
"Некорректный номер телефона. Пожалуйста,
введите номер в международном формате.";
        КонецЕсли;



КонецПроцедуры

&НаСервере
Процедура ПриСозданииНаСервере(Отказ,
СтандартнаяОбработка)

        СсылкаНаКартинку =
ПолучитьНавигационнуюССылку(Объект.Ссылка,
        "Фотография");

КонецПроцедуры
```

Код формы элемента справочника «Руководители»

```
&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ,
ТекущийОбъект, ПараметрыЗаписи)

        Попытка

        ВалидацияПолей.ПроверитьEmail(Текущий
Объект.Почта);
        Исключение
            Сообщить(ОписаниеОшибки());
            Отказ = Истина;
            Возврат;
        КонецПопытки;

        Если НЕ
ВалидацияПолей.КорректноВведенНомерТелефона
(ТекущийОбъект.Телефон) Тогда
            ВызватьИсключение
"Некорректный номер телефона. Пожалуйста,
введите номер в международном формате.";
        КонецЕсли;

КонецПроцедуры
```

Код формы элемента справочника «Библиотека»

```
&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ,
ТекущийОбъект, ПараметрыЗаписи)

        Попытка

        ВалидацияПолей.ПроверитьEmail(Текущий
Объект.Почта);
        Исключение
            Сообщить(ОписаниеОшибки());
            Отказ = Истина;
            Возврат;
        КонецПопытки;

        Если НЕ
ВалидацияПолей.КорректноВведенНомерТелефона
(ТекущийОбъект.Телефон) Тогда
            ВызватьИсключение
"Некорректный номер телефона. Пожалуйста,
введите номер в международном формате.";
        КонецЕсли;

        Если
ЭтоАдресВременногоХранилища(ССылкаНаКартин
ку) Тогда
            ТекущийОбъект.ДанныеКартинки
= Новый
```

```
        ХранилищеЗначения(ПолучитьИзВременн
огоХранилища(ССылкаНаКартинку));
        КонецЕсли;



КонецПроцедуры

&НаКлиенте
Процедура ВставитьКартинку(Команда)

        Оповещение = Новый
ОписаниеОповещения("ОбработатьВыборФайла",Э
тотОбъект);
        НачатьПомещениеФайла(Оповещение,,,Ис
тина, УникальныйИдентификатор);

КонецПроцедуры

&НаКлиенте
Процедура ОбработатьВыборФайла(Результат,
Адрес,
ВыбранноеИмяФайла,ДополнительныйПараметр)
Экспорт

        Если НЕ Результат Тогда
                Возврат;
        КонецЕсли;
        ССылкаНаКартинку = Адрес;

КонецПроцедуры



&НаСервере
Процедура ПриСозданииНаСервере(Отказ,
СтандартнаяОбработка)

        ССылкаНаКартинку =
ПолучитьНавигационнуюССылку(Объект.Ссылка,
"ДанныеКартинки");
КонецПроцедуры
```

Код модуля объекта документа «ПриходКниг»

```
Процедура ОбработкаПроведения(Отказ, Режим)

        // регистр ОстаткиКниг Приход
        Движения.ОстаткиКниг.Записывать =
Истина;
        Для Каждого ТекСтрокаКниги Из Книги
Цикл
                Движение =
Движения.ОстаткиКниг.Добавить();
                Движение.ВидДвижения =
ВидДвиженияНакопления.Приход;
                Движение.Период = Дата;
                Движение.Книги =
ТекСтрокаКниги.Книга;
                Движение.Количество =
ТекСтрокаКниги.Количество;
        КонецЦикла;



        // регистр ОборотЗаказовКниг Расход
        Движения.ОборотЗаказовКниг.Записывать
= Истина;

        // Проверяем источник получения
        Если ИсточникПолучения <>
Перечисления.ИсточникПолучения.Пожертвование
Тогда

                Для Каждого ТекСтрокаКниги Из
Книги Цикл
                        // Получаем остаток по
книге, поставщику и издателю
                        ОстатокКниг =
ПолучитьОстаток(ТекСтрокаКниги.Книга,
Поставщик, Издатель);

                        Если ОстатокКниг <
ТекСтрокаКниги.Количество Тогда
```

```
                    Сообщить("Не
достаточно книг " + ТекСтрокаКниги.Книга + " у
поставщика " + Поставщик + " и издателя " +
Издатель + ". Доступный остаток: " +
ОстатокКниг);
                    Отказ = Истина;
                    Возврат; //
Прерываем проведение, если не хватает книг
                КонецЕсли;

                Движение =
Движения.ОборотЗаказовКниг.Добавить();
                Движение.ВидДвижения =
ВидДвиженияНакопления.Расход;
                Движение.Период = Дата;
                Движение.Книга =
ТекСтрокаКниги.Книга;
                Движение.ВыборЗакупки
= Закупка;
                Движение.Поставщик =
Поставщик;
                Движение.Издатель =
Издатель;
                Движение.Количество =
ТекСтрокаКниги.Количество;
            КонецЦикла;

        КонецЕсли;

        Если ЭтотОбъект.ИсточникПолучения =
Перечисления.ИсточникПолучения.Пожертвование
Тогда
            Если Не
ПожертвованиеДоступно(ЭтотОбъект.ДанныеОПо
жертвований) Тогда
                Сообщить("Документ
пожертвования уже использован!");
                Отказ = Истина;
                Возврат; // Прерываем
проведение
            КонецЕсли;

        Движения.ОборотыПожертвований.Записы
вать = Истина;
            Движение =
Движения.ОборотыПожертвований.Добавить();
            Движение.ВидДвижения =
ВидДвиженияНакопления.Расход;
            Движение.Период = Дата;

        Движение.ДокументПожертвования =
ДанныеОПожертвований;
            Движение.Количество = 1;
        КонецЕсли;

        // регистр ДоступныеКниги Приход
        Движения.ДоступныеКниги.Записывать =
Истина;
        Для Каждого ТекСтрокаКниги Из Книги
Цикл
            Движение =
Движения.ДоступныеКниги.Добавить();
            Движение.ВидДвижения =
ВидДвиженияНакопления.Приход;
            Движение.Период = Дата;
            Движение.Книги =
ТекСтрокаКниги.Книга;
            Движение.Количество =
ТекСтрокаКниги.Количество;
        КонецЦикла;



КонецПроцедуры

Функция ПолучитьОстаток(Книга, Поставщик =
Неопределено, Издатель = Неопределено)

        Результат = 0;

        // Создаем структуру отборов
```

```
	Отбор = Новый Структура("Книга", Книга);

	// Добавляем отбор по поставщику, если он передан
	Если ЗначениеЗаполнено(Поставщик) Тогда
		Отбор.Вставить("Поставщик", Поставщик);
	КонецЕсли;

	// Добавляем отбор по издателю, если он передан
	Если ЗначениеЗаполнено(Издатель) Тогда
		Отбор.Вставить("Издатель", Издатель);
	КонецЕсли;

	// Получаем остатки с учетом отбора
	НайденныеЗначения = РегистрыНакопления.ОборотЗаказовКниг.Остатки(Дата, Отбор);

	Если НайденныеЗначения.Количество() > 0 Тогда
		Результат = НайденныеЗначения[0].Количество;
	КонецЕсли;

	Возврат Результат;

КонецФункции

Процедура ПередЗаписью(Отказ, РежимЗаписи, РежимПроведения)

	СуммаДокумента =0;
	Для каждого ТабличнаяЧасть из Книги Цикл
		СуммаДокумента = СуммаДокумента +ТабличнаяЧасть.Сумма;
	КонецЦикла;

	Если Не ЗначениеЗаполнено(ЭтотОбъект.Ответственный) Тогда
		ЭтотОбъект.Ответственный = ПользователиИнформационнойБазы.ТекущийПользователь();
	КонецЕсли;

	Если ЭтотОбъект.ИсточникПолучения = Перечисления.ИсточникПолучения.Пожертвование Тогда
		Если Не ПожертвованиеДоступно(ЭтотОбъект.ДанныеОПожертвований) Тогда
			Сообщить("Документ пожертвования уже использован!");
			Отказ = Истина;
		КонецЕсли;
	КонецЕсли;

КонецПроцедуры

Процедура ПриЗаписи(Отказ)

	Если Не ЗначениеЗаполнено(ЭтотОбъект.Ответственный) Тогда
		ЭтотОбъект.Ответственный = ПользователиИнформационнойБазы.ТекущийПользователь();
	КонецЕсли;

КонецПроцедуры

Процедура ОбработкаЗаполнения(ДанныеЗаполнения, СтандартнаяОбработка)

	Если ТипЗнч(ДанныеЗаполнения) = Тип("ДокументСсылка.ЗакупкаКниг") Тогда
		// Проверяем статус закупки
```

```
            Если ДанныеЗаполнения.Статус =
Перечисления.СтатусЗакупки.Получен Тогда
                // Заполнение шапки
                Закупка =
ДанныеЗаполнения.ВыборЗакупки;
                Издатель =
ДанныеЗаполнения.Издатель;
                Комментарий =
ДанныеЗаполнения.Комментарий;
                Ответственный =
ДанныеЗаполнения.Ответственный;
                Поставщик =
ДанныеЗаполнения.Поставщик;
                СуммаДокумента =
ДанныеЗаполнения.СуммаДокумента;
                ИсточникПолучения =
Перечисления.ИсточникПолучения.Закупка;
                Основание =
ДанныеЗаполнения.Ссылка;
                Для Каждого
ТекСтрокаКниги Из ДанныеЗаполнения.Книги
Цикл
                    НоваяСтрока =
Книги.Добавить();

    НоваяСтрока.Книга =
ТекСтрокаКниги.Книга;

    НоваяСтрока.Количество =
ТекСтрокаКниги.Количество;

    НоваяСтрока.Сумма =
ТекСтрокаКниги.Сумма;
                    НоваяСтрока.Цена
= ТекСтрокаКниги.Цена;
                КонецЦикла;
            Иначе
                // Выводим сообщение об
ошибке
                Сообщить("Ввод на
основании доступен только для закупок со
статусом 'Получен'!");
                // Прерываем заполнение
                СтандартнаяОбработка =
Ложь;
                Возврат;
            КонецЕсли;
        ИначеЕсли ТипЗнч(ДанныеЗаполнения) =
Тип("ДокументСсылка.Пожертвование") Тогда
            // Заполнение шапки
            Комментарий =
ДанныеЗаполнения.Комментарий;
            Ответственный =
ДанныеЗаполнения.Ответственный;
            ДанныеОПожертвований =
ДанныеЗаполнения.Ссылка;
            СуммаДокумента =
ДанныеЗаполнения.СуммаДокумента;
            ИсточникПолучения =
Перечисления.ИсточникПолучения.Пожертвование
;
            Для Каждого ТекСтрокаКниги Из
ДанныеЗаполнения.Книги Цикл
                НоваяСтрока =
Книги.Добавить();
                НоваяСтрока.Книга =
ТекСтрокаКниги.Книга;
                НоваяСтрока.Количество
= ТекСтрокаКниги.Количество;
                НоваяСтрока.Сумма =
ТекСтрокаКниги.Сумма;
                НоваяСтрока.Цена =
ТекСтрокаКниги.Цена;
            КонецЦикла;
        КонецЕсли;

КонецПроцедуры

&НаСервере
Функция
ПожертвованиеДоступно(ДокументПожертвования
)
```

```
        Отбор = Новый
Структура("ДокументПожертвования",
ДокументПожертвования);
        Остатки =
РегистрыНакопления.ОборотыПожертвований.Ост
атки(Дата, Отбор);

        // Проверяем, что есть остатки и
количество равно 1
        Если Остатки.Количество() > 0 Тогда
                Возврат Остатки[0].Количество =
1;
        КонецЕсли;

        Возврат Ложь;

КонецФункции

&НаСервере
Процедура ПередУдалением(Отказ)

        Для Каждого СтрокаТЧ Из Книги Цикл
                Отбор = Новый
Структура("Книги", СтрокаТЧ.Книга);
                Остатки =
РегистрыНакопления.ОстаткиКниг.Остатки(Дата,
Отбор);

                ТекущийОстаток = 0;
                Если Остатки.Количество() > 0
Тогда
                        ТекущийОстаток =
Остатки[0].Количество;
                КонецЕсли;

                // Вычисляем остаток после
удаления документа (вычитаем количество, так как
удаление прихода уменьшает остаток)
                ОстатокПослеУдаления =
ТекущийОстаток;
                Если ОстатокПослеУдаления < 0
Тогда
                        Сообщить("Нельзя
удалить документ! Недостаточно книг на складе
для книги """ + СтрокаТЧ.Книга + """. Остаток
после удаления: " + ОстатокПослеУдаления);
                        Отказ = Истина;
                        Возврат; // Прерываем
цикл и отменяем удаление
                КонецЕсли;
        КонецЦикла;

КонецПроцедуры
```

Код формы документа «Приход книг»

```
&НаКлиенте
Перем Пожертвование;
Перем Закупка;

&НаКлиенте
Процедура
КнигиЦенаЗаШтПриИзменении(Элемент)

        СтрокаТабличнойЧасти =
Элементы.Книги.ТекущиеДанные;
        СтрокаТабличнойЧасти.Сумма =
СтрокаТабличнойЧасти.Количество*СтрокаТаблич
нойЧасти.Цена;

КонецПроцедуры

&НаКлиенте
Процедура
КнигиСуммаЦенаПриИзменении(Элемент)

        СтрокаТабличнойЧасти =
Элементы.Книги.ТекущиеДанные;
        Если СтрокаТабличнойЧасти.Количество =
0 тогда
                СтрокаТабличнойЧасти.ЦенаЗаШ
= 0;
        Иначе
```

```
            СтрокаТабличнойЧасти.Цена  =
СтрокаТабличнойЧасти.Сумма /
СтрокаТабличнойЧасти.Количество;
        КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура
КнигиКоличествоПриИзменении(Элемент)

        СтрокаТабличнойЧасти =
Элементы.Книги.ТекущиеДанные;
        СтрокаТабличнойЧасти.Сумма =
СтрокаТабличнойЧасти.Количество*СтрокаТаблич
нойЧасти.Цена;

КонецПроцедуры

&НаСервере
Процедура
ИсточникПолученияПриИзмененииНаСервере()

        // Скрываем все поля выбора сначала
        Элементы.ДанныеОпожертвований.Видим
ость = Ложь;
        Элементы.Закупка.Видимость = Ложь;
        Элементы.Поставщик.Видимость = Ложь;
        Элементы.Издатель.Видимость = Ложь;

        Если Объект.ИсточникПолучения =
Перечисления.ИсточникПолучения.Пожертвование
Тогда

        Элементы.ДанныеОпожертвований.Видим
ость = Истина;
            // Очищаем поля, связанные с
закупкой, только если меняем источник получения
            Если Закупка <> 0 Тогда // Или
Если ЗначениеЗаполнено(Закупка) Тогда
                Объект.Закупка = 0;
                Объект.Поставщик = 0;
                Объект.Издатель = 0;
            КонецЕсли;
        ИначеЕсли Объект.ИсточникПолучения =
Перечисления.ИсточникПолучения.Закупка Тогда
            Элементы.Закупка.Видимость =
Истина;
            // Очищаем поле пожертвования,
только если меняем источник получения
            Если
Объект.ДанныеОПожертвований <> 0 Тогда // Или
Если ЗначениеЗаполнено(ДанныеОПожертвований)
Тогда

        Объект.ДанныеОПожертвований = 0;
            КонецЕсли;
            // Устанавливаем видимость полей
выбора в зависимости от значения "Закупка"

        ЗакупкаПриИзмененииНаСервере(); //
Вызываем обработчик изменения закупки
        КонецЕсли;

        // Управляем видимостью "Основание"
        Элементы.Основание.Видимость =
(Объект.ИсточникПолучения =
Перечисления.ИсточникПолучения.Закупка);

        // Очищаем "Основание", если источник
получения не "Закупка"
        Если Объект.ИсточникПолучения <>
Перечисления.ИсточникПолучения.Закупка И
ЗначениеЗаполнено(Объект.Основание) Тогда
            Объект.Основание =
Неопределено;
        КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура
ИсточникПолученияПриИзменении(Элемент)
```

```
        ИсточникПолученияПриИзмененииНаСерв
ере();

КонецПроцедуры

&НаСервере
Процедура ПриОткрытииНаСервере()

        ИсточникПолученияПриИзмененииНаСерв
ере();

КонецПроцедуры

&НаКлиенте
Процедура ПриОткрытии(Отказ)

        ПриОткрытииНаСервере();

КонецПроцедуры

&НаСервере
Процедура ЗакупкаПриИзмененииНаСервере()

        // Скрываем оба поля сначала
        Элементы.Поставщик.Видимость = Ложь;
        Элементы.Издатель.Видимость = Ложь;

        Если Объект.Закупка =
Перечисления.ВыборЗакупки.Издатель Тогда
                Элементы.Издатель.Видимость =
Истина;
                // Очищаем поле поставщика,
только если меняем выбор закупки
                Если Объект.Поставщик <> 0
Тогда // Или Если ЗначениеЗаполнено(Поставщик)
Тогда
                        Объект.Поставщик = 0;
                КонецЕсли;
        ИначеЕсли Объект.Закупка =
Перечисления.ВыборЗакупки.Поставщик Тогда
                Элементы.Поставщик.Видимость =
Истина;
                // Очищаем поле издателя, только
если меняем выбор закупки
                Если Объект.Издатель <> 0 Тогда
// Или Если ЗначениеЗаполнено(Издатель) Тогда
                        Объект.Издатель = 0;
                КонецЕсли;
        КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура ЗакупкаПриИзменении(Элемент)

        ЗакупкаПриИзмененииНаСервере();

КонецПроцедуры

&НаКлиенте
Процедура КнигиКнигаПриИзменении(Элемент)

        СтрокаТЧ =
Элементы.Книги.ТекущиеДанные;
        СтрокаТЧ.Цена=ОбщийМодульФункции.Ц
еныИзРегистраСведений(Объект.Дата,СтрокаТЧ.К
нига);

КонецПроцедуры

&НаСервере
Процедура
ДанныеОпожертвованийПриИзмененииНаСервере(
)

        // Проверяем, выбран ли документ
пожертвования
        Если
ЗначениеЗаполнено(Объект.ДанныеОПожертвован
ий) Тогда
                // Получаем документ
пожертвования
```

```
                Пожертвование =
Объект.ДанныеОПожертвований.ПолучитьОбъект(
);

                // Очищаем табличную часть
"Книги" в документе "Приход книг"
                Объект.Книги.Очистить();

                // Копируем данные из табличной
части "Книги" документа "Пожертвование"
                Для Каждого
СтрокаПожертвования Из Пожертвование.Книги
Цикл
                    НоваяСтрока =
Объект.Книги.Добавить();
                    НоваяСтрока.Книга =
СтрокаПожертвования.Книга;
                    НоваяСтрока.Количество
= СтрокаПожертвования.Количество;
                    НоваяСтрока.Цена =
СтрокаПожертвования.Цена;
                    НоваяСтрока.Сумма =
СтрокаПожертвования.Сумма;
                КонецЦикла;
            КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура
ДанныеОпожертвованийПриИзменении(Элемент)

            ДанныеОпожертвованийПриИзмененииНа
Сервере();

КонецПроцедуры

&НаСервере
Процедура ОснованиеПриИзмененииНаСервере()
    Если Объект.Основание = Неопределено Тогда
        Возврат; // Если основание не выбрано, ничего
не проверяем
    КонецЕсли;

    ЗакупкаДокумент =
Объект.Основание.ПолучитьОбъект();
    ВыборЗакупкиЗначение =
ЗакупкаДокумент.ВыборЗакупки; // Используйте
отдельную переменную

    // Заполнение шапки
            Объект.Закупка = ВыборЗакупкиЗначение;
// Устанавливаем перечисление в документ
"ПриходКниг"
    Объект.Издатель = ЗакупкаДокумент.Издатель;
    Объект.Комментарий =
ЗакупкаДокумент.Комментарий;
    Объект.Ответственный =
ЗакупкаДокумент.Ответственный;
    Объект.Поставщик =
ЗакупкаДокумент.Поставщик;
    Объект.СуммаДокумента =
ЗакупкаДокумент.СуммаДокумента;
    Объект.ИсточникПолучения =
Перечисления.ИсточникПолучения.Закупка;
    Объект.Основание = ЗакупкаДокумент.Ссылка;

    Объект.Книги.Очистить(); // Очищаем
табличную часть перед заполнением
    Для Каждого ТекСтрокаКниги Из
ЗакупкаДокумент.Книги Цикл
        НоваяСтрока = Объект.Книги.Добавить();
        НоваяСтрока.Книга = ТекСтрокаКниги.Книга;
        НоваяСтрока.Количество =
ТекСтрокаКниги.Количество;
        НоваяСтрока.Сумма =
ТекСтрокаКниги.Сумма;
        НоваяСтрока.Цена = ТекСтрокаКниги.Цена;
    КонецЦикла;

    // Проверка соответствия данных *после*
заполнения
```

```
    Если Не
СоответствуютДанные(ЗакупкаДокумент, Объект)
Тогда // Передаем документ
        Сообщить("Данные прихода не соответствуют
данным закупки!");
        Объект.Основание = Неопределено;
    КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура ОснованиеПриИзменении(Элемент)

        ОснованиеПриИзмененииНаСервере();

КонецПроцедуры


&НаСервере
Функция СоответствуютДанные(Закупка, Приход)

        // Проверка соответствия
поставщика/издателя
        Если Закупка.ВыборЗакупки =
Перечисления.ВыборЗакупки.Поставщик Тогда
                Если Закупка.Поставщик <>
Приход.Поставщик Тогда
                        Возврат Ложь;
                КонецЕсли;
        ИначеЕсли Закупка.ВыборЗакупки =
Перечисления.ВыборЗакупки.Издатель Тогда
                Если Закупка.Издатель <>
Приход.Издатель Тогда
                        Возврат Ложь;
                КонецЕсли;
        КонецЕсли;



        // Проверяем, что все книги из прихода
есть в закупке с учетом количества
        Для Каждого СтрокаПрихода Из
Приход.Книги Цикл
                КоличествоВЗакупке = 0;
                Для Каждого СтрокаЗакупки Из
Закупка.Книги Цикл
                        Если
СтрокаПрихода.Книга = СтрокаЗакупки.Книга
Тогда

                КоличествоВЗакупке =
СтрокаЗакупки.Количество;
                        Прервать; // Книга
найдена, выходим из внутреннего цикла
                КонецЕсли;
                КонецЦикла;

                Если КоличествоВЗакупке = 0
Тогда
                Возврат Ложь; // Книга из
прихода не найдена в закупке
                ИначеЕсли
СтрокаПрихода.Количество > КоличествоВЗакупке
Тогда
                Возврат Ложь; //
Количество в приходе превышает количество в
закупке
                КонецЕсли;
        КонецЦикла;



        Возврат Истина; // Данные соответствуют

КонецФункции

&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ,
ТекущийОбъект, ПараметрыЗаписи)

        ТекущийПользователь =
ПользователиИнформационнойБазы.ТекущийПоль
зователь();

        // Поиск сотрудника по логину только если
поле "Ответственный" не заполнено
```

```
        Если Не
ЗначениеЗаполнено(ТекущийОбъект.Ответственны
й) Тогда
                НайденныйСотрудник =
Справочники.Сотрудники.НайтиПоРеквизиту("Лог
ин", ТекущийПользователь.Имя);

                Если НайденныйСотрудник <>
Неопределено Тогда

        ТекущийОбъект.Ответственный =
НайденныйСотрудник;
                Иначе
                        Сообщить("Сотрудник не
найден по логину текущего пользователя. Укажите
ответственного вручную.");
                КонецЕсли;
        КонецЕсли;

КонецПроцедуры
```

Код печати «РегистрацияКнигВБиблиотеке»

```
 &НаКлиенте
 Процедура
ОбработкаКоманды(ПараметрКоманды,
ПараметрыВыполненияКоманды)

        ТабДок = Новый ТабличныйДокумент;
        Печать (ТабДок, ПараметрКоманды);
        ТабДок.Показать("Печать Регистрации
книг");

 КонецПроцедуры

 &НаСервере
 Процедура Печать(ТабДок, ССылкаНаДокумент)

        Макет =
Документы.ПриходКниг.ПолучитьМакет("Регистра
цияКнигВБиблиотеке");
        ОбластьЗаголовок =
Макет.ПолучитьОбласть("Заголовок");
        ОбластьТекст =
Макет.ПолучитьОбласть("Текст");
        ОбластьШапкаТаблици =
Макет.ПолучитьОбласть("ШапкаТаблици");
        ОбластьТаблица =
Макет.ПолучитьОбласть("Таблица");
        ОбластьПодвалТаблици =
Макет.ПолучитьОбласть("ПодвалТаблици");
        ОбластьПодвал =
Макет.ПолучитьОбласть("Подвал");

        ОбластьЗаголовок.Параметры.Номер
=ССылкаНаДокумент.Номер;
        ОбластьЗаголовок.Параметры.Дата =
Формат(ССылкаНаДокумент.Дата,"ДЛФ=DD");
        ТабДок.Вывести(ОбластьЗаголовок);

        Запрос = Новый Запрос;
   Запрос.Текст =
      "ВЫБРАТЬ ПЕРВЫЕ 1
      |   Библиотека.ПолноеНаименование
      |ИЗ
      |   Справочник.Библиотека КАК Библиотека
      |АВТОУПОРЯДОЧИВАНИЕ";

   Результат = Запрос.Выполнить();
   Выборка = Результат.Выбрать();

   Если Выборка.Следующий() Тогда
      НазваниеБиблиотеки =
Выборка.ПолноеНаименование;
   Иначе
      НазваниеБиблиотеки = "";
   КонецЕсли;

   ОбластьТекст.Параметры.НазваниеБибилиотеки
= НазваниеБиблиотеки;
```

```
	ТекущийПользователь = ПользователиИнформационнойБазы.ТекущийПользователь();
	Сотрудник = Справочники.Сотрудники.НайтиПоРеквизиту("Логин", ТекущийПользователь.Имя);

	Если Сотрудник <> Неопределено И ЗначениеЗаполнено(Сотрудник.Отдел) Тогда
		НазваниеОтдела = Сотрудник.Отдел.Наименование;
	Иначе
		НазваниеОтдела = "";
	КонецЕсли;


ОбластьТекст.Параметры.СтруктурноеПодразделение = НазваниеОтдела;



	ДокументПриходКниг = ССылкаНаДокумент.ПолучитьОбъект();

	// Определяем наименование
	НаименованиеОрганизации = "";
	ФИО = "";

	Если ДокументПриходКниг.ИсточникПолучения = Перечисления.ИсточникПолучения.Пожертвование Тогда
		// Получаем данные о пожертвовании
		Пожертвование = ДокументПриходКниг.ДанныеОПожертвований.ПолучитьОбъект();
		ФИО = Пожертвование.Фамилия + " " + Пожертвование.Имя + " " + Пожертвование.Отчество;
		НаименованиеОрганизации = "Пожертвование №" + Пожертвование.Номер + " от " + ФИО;
	ИначеЕсли ДокументПриходКниг.Закупка = Перечисления.ВыборЗакупки.Поставщик Тогда
		НаименованиеОрганизации = ДокументПриходКниг.Поставщик.Наименование;
	ИначеЕсли ДокументПриходКниг.Закупка = Перечисления.ВыборЗакупки.Издатель Тогда
		НаименованиеОрганизации = ДокументПриходКниг.Издатель.Наименование;
	КонецЕсли;

	ОбластьТекст.Параметры.Источник = НаименованиеОрганизации;


	Если ДокументПриходКниг.Основание <> Неопределено Тогда
		ДокументОснование = ДокументПриходКниг.Основание.ПолучитьОбъект();

ОбластьТекст.Параметры.Основание = "Документ №" + ДокументОснование.Номер + " от " + Формат(ДокументОснование.Дата, "ДЛФ=DD");
	Иначе
	КонецЕсли;

	ТабДок.Вывести(ОбластьТекст);



	ТабДок.Вывести(ОбластьШапкаТаблицы);

	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|	ПриходКнигКниги.НомерСтроки КАК НомерСтроки,
	|	ПриходКнигКниги.Книга КАК Книга,
	|	ПриходКнигКниги.Количество КАК Количество,
```

```
        |        ПриходКнигКниги.Цена КАК
Цена,
        |        ПриходКнигКниги.Сумма КАК
Сумма
        |ИЗ
        |        Документ.ПриходКниг.Книги КАК
ПриходКнигКниги
        |ГДЕ
        |        ПриходКнигКниги.Ссылка =
&ССылкаНаДокумент";


Запрос.УстановитьПараметр("ССылкаНаДокумент",
ССылкаНаДокумент);

        РезультатЗапроса = Запрос.Выполнить();

        ВыборкаДетальныеЗаписи =
РезультатЗапроса.Выбрать();
        СуммаКоличство = 0;
        СуммаВсего = 0;
        Пока
ВыборкаДетальныеЗаписи.Следующий() Цикл

ОбластьТаблица.Параметры.НомерСтроки =
ВыборкаДетальныеЗаписи.НомерСтроки;

ОбластьТаблица.Параметры.Книга =
ВыборкаДетальныеЗаписи.Книга;

ОбластьТаблица.Параметры.Количество =
ВыборкаДетальныеЗаписи.Количество;
                ОбластьТаблица.Параметры.Цена
= ВыборкаДетальныеЗаписи.Цена;

ОбластьТаблица.Параметры.Сумма =
ВыборкаДетальныеЗаписи.Сумма;
                СуммаКоличство =
СуммаКоличство+
ВыборкаДетальныеЗаписи.Количество;
                СуммаВсего = СуммаВсего +
ВыборкаДетальныеЗаписи.Сумма;

ТабДок.Вывести(ОбластьТаблица);
        КонецЦикла;


ОбластьПодвалТаблици.Параметры.ВсегоКоличест
во = СуммаКоличство;

ОбластьПодвалТаблици.Параметры.ВсегоСумма =
СуммаВсего;

ТабДок.Вывести(ОбластьПодвалТаблици);

        ТабДок.Вывести(ОбластьПодвал);

КонецПроцедуры
```

Код модуля объекта документа «СписаниеКниг»

```
Процедура ОбработкаПроведения(Отказ, Режим)

        // регистр ОстаткиКниг Расход
        Движения.ОстаткиКниг.Записывать =
Истина;
        Для Каждого ТекСтрокаКниги Из Книги
Цикл
                ОстатокКниг =
ПолучитьОстаток(ТекСтрокаКниги.Книга);
                Если
ОстатокКниг<ТекСтрокаКниги.Количество Тогда
                        Сообщить(" Не достаточно
книг "+ТекСтрокаКниги.Книга+" Доступный
остаток: "+ОстатокКниг);
                        Отказ = Истина;
                КонецЕсли;
                Движение =
Движения.ОстаткиКниг.Добавить();
                Движение.ВидДвижения =
ВидДвиженияНакопления.Расход;
                Движение.Период = Дата;
                Движение.Книги =
ТекСтрокаКниги.Книга;
```

```
            Движение.Количество =
ТекСтрокаКниги.Количество;
        КонецЦикла;

        // регистр ДоступныеКниги Расход
        Движения.ДоступныеКниги.Записывать =
Истина;
        Для Каждого ТекСтрокаКниги Из Книги
Цикл
            Движение =
Движения.ДоступныеКниги.Добавить();
            Движение.ВидДвижения =
ВидДвиженияНакопления.Расход;
            Движение.Период = Дата;
            Движение.Книги =
ТекСтрокаКниги.Книга;
            Движение.Количество =
ТекСтрокаКниги.Количество;
        КонецЦикла;



КонецПроцедуры

Функция ПолучитьОстаток(Книга)

        Результат =0;
        Отбор = Новый Структура;
        Отбор.Вставить("Книги", Книга);
        НайденныеЗначения =
РегистрыНакопления.ОстаткиКниг.Остатки(Дата,О
тбор);
        Если НайденныеЗначения.Количество()>0
Тогда
            Результат =
НайденныеЗначения[0].Количество;
        КонецЕсли;
        Возврат Результат;

КонецФункции

Процедура ПриЗаписи(Отказ)

        Если Не
ЗначениеЗаполнено(ЭтотОбъект.Ответственный)
Тогда
            ЭтотОбъект.Ответственный =
ПользователиИнформационнойБазы.ТекущийПоль
зователь();

        КонецЕсли;
КонецПроцедуры

Процедура
ОбработкаЗаполнения(ДанныеЗаполнения,
СтандартнаяОбработка)

        Если ТипЗнч(ДанныеЗаполнения) =
Тип("ДокументСсылка.ВыкупУтерянныйКниги")
Тогда
            // Заполнение шапки
            Для Каждого ТекСтрокаКниги Из
ДанныеЗаполнения.Книги Цикл
                НоваяСтрока =
Книги.Добавить();
                НоваяСтрока.Книга =
ТекСтрокаКниги.Книга;
                НоваяСтрока.Количество
= ТекСтрокаКниги.Количество;

        НоваяСтрока.ПричинаСписания =
Перечисления.ПричинаСписания.Утрата;
            КонецЦикла;
        КонецЕсли;

КонецПроцедуры
```

Код формы документа «СписаниеКниг»

```
&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ,
ТекущийОбъект, ПараметрыЗаписи)
```

```
	ТекущийПользователь = ПользователиИнформационнойБазы.ТекущийПользователь();

	// Поиск сотрудника по логину только если поле "Ответственный" не заполнено
	Если Не ЗначениеЗаполнено(ТекущийОбъект.Ответственный) Тогда
		НайденныйСотрудник = Справочники.Сотрудники.НайтиПоРеквизиту("Логин", ТекущийПользователь.Имя);

		Если НайденныйСотрудник <> Неопределено Тогда

	ТекущийОбъект.Ответственный = НайденныйСотрудник;
		Иначе
			Сообщить("Сотрудник не найден по логину текущего пользователя. Укажите ответственного вручную.");
		КонецЕсли;
	КонецЕсли;

КонецПроцедуры
```

Код печати «ПечатьСписанияКниг»

```
&НаКлиенте
Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполненияКоманды)

	ТабДок = Новый ТабличныйДокумент;
	Печать (ТабДок, ПараметрКоманды);
	ТабДок.Показать("Печать Акта Пожертвования");

КонецПроцедуры

&НаСервере
Процедура Печать(ТабДок, ССылкаНаДокумент)
	Макет = Документы.СписаниеКниг.ПолучитьМакет("СписаниеКниг");
	ОбластьЧердак = Макет.ПолучитьОбласть("Чердак");
	ОбластьЗаголовок = Макет.ПолучитьОбласть("Заголовок");
	ОбластьТекст = Макет.ПолучитьОбласть("Текст");
	ОбластьШапкаТаблици = Макет.ПолучитьОбласть("ШапкаТаблици");
	ОбластьТаблица = Макет.ПолучитьОбласть("Таблица");
	ОбластьПодвалТаблици = Макет.ПолучитьОбласть("ПодвалТаблици");
	ОбластьПодвал = Макет.ПолучитьОбласть("Подвал");

	ОбластьЧердак.Параметры.Дата = Формат(ССылкаНаДокумент.Дата,"ДЛФ=DD");
	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ ПЕРВЫЕ 1
	| Библиотека.Директор.Фамилия КАК ДиректорФамилия,
	| Библиотека.Директор.Наименование КАК ДиректорИмя,
	| Библиотека.Директор.Отчество КАК ДиректорОтчество
	|ИЗ
	| Справочник.Библиотека КАК Библиотека
	|АВТОУПОРЯДОЧИВАНИЕ";

	Результат = Запрос.Выполнить();
	Выборка = Результат.Выбрать();

	Если Выборка.Следующий() Тогда
		ФИОРуководителя = Выборка.ДиректорФамилия + " " +
```

```
Выборка.ДиректорИмя + " " +
Выборка.ДиректорОтчество;

        ОбластьЧердак.Параметры.РуководительУ
чреждения = ФИОРуководителя;
        Иначе

        ОбластьЧердак.Параметры.РуководительУ
чреждения = "";
        КонецЕсли;

        ТабДок.Вывести(ОбластьЧердак);

        ОбластьЗаголовок.Параметры.Номер =
ССылкаНаДокумент.Номер;
        ОбластьЗаголовок.Параметры.Дата =
Формат(ССылкаНаДокумент.Дата,"ДЛФ=DD");
        ТабДок.Вывести(ОбластьЗаголовок);

        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       СписаниеКнигКниги.НомерСтроки
КАК НомерСтроки,
        |       СписаниеКнигКниги.Количество
КАК Количество
        |ИЗ
        |       Документ.СписаниеКниг.Книги
КАК СписаниеКнигКниги
        |ГДЕ
        |       СписаниеКнигКниги.Ссылка =
&ССылкаНаДокумент";

        Запрос.УстановитьПараметр("ССылкаНаДо
кумент", ССылкаНаДокумент);

        РезультатЗапроса = Запрос.Выполнить();

        ВыборкаДетальныеЗаписи =
РезультатЗапроса.Выбрать();
        СуммаКоличство = 0;
        Пока
ВыборкаДетальныеЗаписи.Следующий() Цикл

        ОбластьТаблица.Параметры.Количество =
ВыборкаДетальныеЗаписи.Количество;
                СуммаКоличство =
СуммаКоличство+
ВыборкаДетальныеЗаписи.Количество;
        КонецЦикла;


        // Получаем данные библиотеки
        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ ПЕРВЫЕ 1
        |   Библиотека.ПолноеНаименование
        |ИЗ
        |   Справочник.Библиотека КАК
Библиотека
        |АВТОУПОРЯДОЧИВАНИЕ";

        Результат = Запрос.Выполнить();
        Выборка = Результат.Выбрать();

        Если Выборка.Следующий() Тогда
                НазваниеБиблиотеки =
Выборка.ПолноеНаименование;
        Иначе
                НазваниеБиблиотеки = ""; // Или
другое значение по умолчанию
        КонецЕсли;

        ОбластьТекст.Параметры.НазваниеБибили
отеки = НазваниеБиблиотеки;

        ТекущийПользователь =
ПользователиИнформационнойБазы.ТекущийПольз
ователь();
        Сотрудник =
Справочники.Сотрудники.НайтиПоРеквизиту("Лог
ин", ТекущийПользователь.Имя);
```

```
	Если Сотрудник <> Неопределено И ЗначениеЗаполнено(Сотрудник.Отдел) Тогда
		НазваниеОтдела = Сотрудник.Отдел.Наименование;
	Иначе
		НазваниеОтдела = "";
	КонецЕсли;

	ОбластьТекст.Параметры.СтруктурноеПодразделение = НазваниеОтдела;



	ОбластьТекст.Параметры.Дата = Формат(ССылкаНаДокумент.Дата,"ДЛФ=D");
	ОбластьТекст.Параметры.Номер = ССылкаНаДокумент.Номер;
	ОбластьТекст.Параметры.Количество = СуммаКоличство;



	ТабДок.Вывести(ОбластьТекст);

	ТабДок.Вывести(ОбластьШапкаТаблици);

	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|	СписаниеКнигКниги.НомерСтроки КАК НомерСтроки,
	|	СписаниеКнигКниги.Книга КАК Книга,
	|	СписаниеКнигКниги.ПричинаСписания КАК ПричинаСписания,
	|	СписаниеКнигКниги.Количество КАК Количество
	|ИЗ
	|	Документ.СписаниеКниг.Книги КАК СписаниеКнигКниги
	|ГДЕ
	|	СписаниеКнигКниги.Ссылка = &ССылкаНаДокумент";

	Запрос.УстановитьПараметр("ССылкаНаДокумент", ССылкаНаДокумент);

	РезультатЗапроса = Запрос.Выполнить();

	ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
	СуммаКоличство = 0;
	СуммаВсего = 0;
	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

	ОбластьТаблица.Параметры.НомерСтроки = ВыборкаДетальныеЗаписи.НомерСтроки;
		ОбластьТаблица.Параметры.Книга = ВыборкаДетальныеЗаписи.Книга;

	ОбластьТаблица.Параметры.Количество = ВыборкаДетальныеЗаписи.Количество;
		СуммаКоличство = СуммаКоличство+ ВыборкаДетальныеЗаписи.Количество;

	ОбластьТаблица.Параметры.ПричинаСписания = ВыборкаДетальныеЗаписи.ПричинаСписания;
		ТабДок.Вывести(ОбластьТаблица);
	КонецЦикла;

	ОбластьПодвалТаблици.Параметры.ВсегоКоличество = СуммаКоличство;

	ТабДок.Вывести(ОбластьПодвалТаблици);

	ТабДок.Вывести(ОбластьПодвал);

КонецПроцедуры
```

Код модуля объекта «ЗакупкаКниг»

```
Процедура ПриЗаписи(Отказ)
```

```
        Если Не
ЗначениеЗаполнено(ЭтотОбъект.Ответственный)
Тогда
                ЭтотОбъект.Ответственный =
ПользователиИнформационнойБазы.ТекущийПоль
зователь();
        КонецЕсли;

КонецПроцедуры

Процедура ПередЗаписью(Отказ, РежимЗаписи,
РежимПроведения)

        СуммаДокумента =0;
        Для каждого ТабличнаяЧасть из Книги
Цикл
                СуммаДокумента =
СуммаДокумента +ТабличнаяЧасть.Сумма;
        КонецЦикла;
        Если Не
ЗначениеЗаполнено(ЭтотОбъект.Ответственный)
Тогда
                ЭтотОбъект.Ответственный =
ПользователиИнформационнойБазы.ТекущийПоль
зователь();
        КонецЕсли;

КонецПроцедуры

Процедура ОбработкаПроведения(Отказ, Режим)

        // регистр ОборотЗаказовКниг Приход
        Движения.ОборотЗаказовКниг.Записывать
= Истина;
        Для Каждого ТекСтрокаКниги Из Книги
Цикл
                Движение =
Движения.ОборотЗаказовКниг.Добавить();
                Движение.ВидДвижения =
ВидДвиженияНакопления.Приход;
                Движение.Период = Дата;
                Движение.Книга =
ТекСтрокаКниги.Книга;
                Движение.ВыборЗакупки =
ВыборЗакупки;
                Движение.Поставщик =
Поставщик;
                Движение.Издатель = Издатель;
                Движение.Количество =
ТекСтрокаКниги.Количество;
        КонецЦикла;


        // регистр СредняяСобестоимостьТоваров
Приход
        Движения.СредняяСобестоимостьТоваров.
Записывать = Истина;
        Для Каждого ТекСтрокаКниги Из Книги
Цикл
                Движение =
Движения.СредняяСобестоимостьТоваров.Добавит
ь();
                Движение.ВидДвижения =
ВидДвиженияНакопления.Приход;
                Движение.Период = Дата;
                Движение.Книга =
ТекСтрокаКниги.Книга;
                Движение.Сумма =
ТекСтрокаКниги.Сумма;
                Движение.Количество =
ТекСтрокаКниги.Количество;
        КонецЦикла;

        // регистр Взаиморасчеты
        Движения.Взаиморасчеты.Записывать=Ист
ина;
        Движение =
Движения.Взаиморасчеты.Добавить();
        Движение.ВидДвижения =
ВидДвиженияНакопления.Приход;
        Движение.Период=Дата;
        Движение.Поставщик = Поставщик;
        Движение.Сумма = СуммаДокумента;
```

```
КонецПроцедуры
```

Код формы документа «ЗакупкаКниг»

```
&НаСервере
Процедура
ВыборЗакупкиПриИзмененииНаСервере()

    // Очищаем оба поля перед изменением
видимости
    Объект.Поставщик = 0;
    Объект.Издатель = 0;

    Если Объект.ВыборЗакупки =
Перечисления.ВыборЗакупки.Поставщик Тогда
        Элементы.Поставщик.Видимость =
Истина;
        Элементы.Издатель.Видимость =
Ложь;
    ИначеЕсли Объект.ВыборЗакупки =
Перечисления.ВыборЗакупки.Издатель Тогда
        Элементы.Поставщик.Видимость =
Ложь;
        Элементы.Издатель.Видимость =
Истина;
    КонецЕсли;

КонецПроцедуры
&НаКлиенте
Процедура ВыборЗакупкиПриИзменении(Элемент)

    ВыборЗакупкиПриИзмененииНаСервере();

КонецПроцедуры

&НаСервере
Процедура ПриОткрытииНаСервере()

    Если Объект.ВыборЗакупки =
Перечисления.ВыборЗакупки.Поставщик тогда
        Элементы.Поставщик.Видимость =
Истина;
        Элементы.Издатель.Видимость =
Ложь;
    ИначеЕсли Объект.ВыборЗакупки =
Перечисления.ВыборЗакупки.Издатель тогда
        Элементы.Поставщик.Видимость =
Ложь;
        Элементы.Издатель.Видимость =
Истина;
    КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура ПриОткрытии(Отказ)

    ПриОткрытииНаСервере();

КонецПроцедуры

&НаКлиенте
Процедура КнигиЦенаПриИзменении(Элемент)

    СтрокаТабличнойЧасти =
Элементы.Книги.ТекущиеДанные;
    СтрокаТабличнойЧасти.Сумма =
СтрокаТабличнойЧасти.Количество*СтрокаТаблич
нойЧасти.Цена;

КонецПроцедуры

&НаКлиенте
Процедура КнигиСуммаПриИзменении(Элемент)

    СтрокаТабличнойЧасти =
Элементы.Книги.ТекущиеДанные;
    Если СтрокаТабличнойЧасти.Количество =
0 тогда
        СтрокаТабличнойЧасти.ЦенаЗаШ
= 0;
    Иначе
```

```
            СтрокаТабличнойЧасти.Цена  =
СтрокаТабличнойЧасти.Сумма /
СтрокаТабличнойЧасти.Количество;
        КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура
КнигиКоличествоПриИзменении(Элемент)

        СтрокаТабличнойЧасти =
Элементы.Книги.ТекущиеДанные;
        СтрокаТабличнойЧасти.Сумма =
СтрокаТабличнойЧасти.Количество*СтрокаТаблич
нойЧасти.Цена;

КонецПроцедуры

&НаКлиенте
Процедура КнигиКнигаПриИзменении(Элемент)

        СтрокаТЧ =
Элементы.Книги.ТекущиеДанные;
        СтрокаТЧ.Цена=ОбщийМодульФункции.Ц
еныИзРегистраСведений(Объект.Дата,СтрокаТЧ.К
нига);

КонецПроцедуры

&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ,
ТекущийОбъект, ПараметрыЗаписи)

        // Проверяем, заполнено ли поле "Статус"
        Если Не
ЗначениеЗаполнено(ТекущийОбъект.Статус) Тогда
            // Устанавливаем статус
"Сформирован"
            ТекущийОбъект.Статус =
Перечисления.СтатусЗакупки.Сформирован;
        КонецЕсли;

        ТекущийПользователь =
ПользователиИнформационнойБазы.ТекущийПольз
ователь();

        // Поиск сотрудника по логину только если
поле "Ответственный" не заполнено
        Если Не
ЗначениеЗаполнено(ТекущийОбъект.Ответственны
й) Тогда
            НайденныйСотрудник =
Справочники.Сотрудники.НайтиПоРеквизиту("Лог
ин", ТекущийПользователь.Имя);

            Если НайденныйСотрудник <>
Неопределено Тогда

        ТекущийОбъект.Ответственный =
НайденныйСотрудник;
            Иначе
                Сообщить("Сотрудник не
найден по логину текущего пользователя. Укажите
ответственного вручную.");
            КонецЕсли;
        КонецЕсли;
КонецПроцедуры
```

Код печати «ТоварнаяНакладная»

```
&НаСервере
Процедура
ВыборЗакупкиПриИзмененииНаСервере()

        // Очищаем оба поля перед изменением
видимости
        Объект.Поставщик = 0;
        Объект.Издатель = 0;

        Если Объект.ВыборЗакупки =
Перечисления.ВыборЗакупки.Поставщик Тогда
```

```
                Элементы.Поставщик.Видимость =
Истина;
                Элементы.Издатель.Видимость =
Ложь;
        ИначеЕсли Объект.ВыборЗакупки =
Перечисления.ВыборЗакупки.Издатель Тогда
                Элементы.Поставщик.Видимость =
Ложь;
                Элементы.Издатель.Видимость =
Истина;
        КонецЕсли;

КонецПроцедуры
&НаКлиенте
Процедура ВыборЗакупкиПриИзменении(Элемент)

        ВыборЗакупкиПриИзмененииНаСервере();

КонецПроцедуры

&НаСервере
Процедура ПриОткрытииНаСервере()

        Если Объект.ВыборЗакупки =
Перечисления.ВыборЗакупки.Поставщик тогда
                Элементы.Поставщик.Видимость =
Истина;
                Элементы.Издатель.Видимость =
Ложь;
        ИначеЕсли Объект.ВыборЗакупки =
Перечисления.ВыборЗакупки.Издатель тогда
                Элементы.Поставщик.Видимость =
Ложь;
                Элементы.Издатель.Видимость =
Истина;
        КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура ПриОткрытии(Отказ)

        ПриОткрытииНаСервере();

КонецПроцедуры

&НаКлиенте
Процедура КнигиЦенаПриИзменении(Элемент)

        СтрокаТабличнойЧасти =
Элементы.Книги.ТекущиеДанные;
        СтрокаТабличнойЧасти.Сумма =
СтрокаТабличнойЧасти.Количество*СтрокаТаблич
нойЧасти.Цена;

КонецПроцедуры

&НаКлиенте
Процедура КнигиСуммаПриИзменении(Элемент)

        СтрокаТабличнойЧасти =
Элементы.Книги.ТекущиеДанные;
        Если СтрокаТабличнойЧасти.Количество =
0 тогда
                СтрокаТабличнойЧасти.ЦенаЗаШ
= 0;
        Иначе
                СтрокаТабличнойЧасти.Цена  =
СтрокаТабличнойЧасти.Сумма /
СтрокаТабличнойЧасти.Количество;
        КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура
КнигиКоличествоПриИзменении(Элемент)

        СтрокаТабличнойЧасти =
Элементы.Книги.ТекущиеДанные;
        СтрокаТабличнойЧасти.Сумма =
СтрокаТабличнойЧасти.Количество*СтрокаТаблич
нойЧасти.Цена;
```

```
КонецПроцедуры

&НаКлиенте
Процедура КнигиКнигаПриИзменении(Элемент)

	СтрокаТЧ = Элементы.Книги.ТекущиеДанные;
	СтрокаТЧ.Цена=ОбщийМодульФункции.ЦеныИзРегистраСведений(Объект.Дата,СтрокаТЧ.Книга);

КонецПроцедуры

&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи)

	// Проверяем, заполнено ли поле "Статус"
	Если Не ЗначениеЗаполнено(ТекущийОбъект.Статус) Тогда
		// Устанавливаем статус "Сформирован"
		ТекущийОбъект.Статус = Перечисления.СтатусЗакупки.Сформирован;
	КонецЕсли;



	ТекущийПользователь = ПользователиИнформационнойБазы.ТекущийПользователь();

	// Поиск сотрудника по логину только если поле "Ответственный" не заполнено
	Если Не ЗначениеЗаполнено(ТекущийОбъект.Ответственный) Тогда
		НайденныйСотрудник = Справочники.Сотрудники.НайтиПоРеквизиту("Логин", ТекущийПользователь.Имя);

		Если НайденныйСотрудник <> Неопределено Тогда

	ТекущийОбъект.Ответственный = НайденныйСотрудник;
		Иначе
			Сообщить("Сотрудник не найден по логину текущего пользователя. Укажите ответственного вручную.");
		КонецЕсли;
	КонецЕсли;
КонецПроцедуры
```

Код модуля документа «Пожертвования»

```
Процедура ПриЗаписи(Отказ)

	Если Не ЗначениеЗаполнено(ЭтотОбъект.Ответственный) Тогда
		ЭтотОбъект.Ответственный = ПользователиИнформационнойБазы.ТекущийПользователь();
	КонецЕсли;

КонецПроцедуры

Процедура ПередЗаписью(Отказ, РежимЗаписи, РежимПроведения)
	СуммаДокумента =0;
	Для каждого ТабличнаяЧасть из Книги Цикл
		СуммаДокумента = СуммаДокумента +ТабличнаяЧасть.Сумма;
	КонецЦикла;
	Если Не ЗначениеЗаполнено(ЭтотОбъект.Ответственный) Тогда

	ЭтотОбъект.Ответственный = ПользователиИнформационнойБазы.ТекущийПользователь();
	КонецЕсли;
```

```
КонецПроцедуры

Процедура ОбработкаПроведения(Отказ, Режим)
	Движения.ОборотыПожертвований.Записывать = Истина;
	Движение = Движения.ОборотыПожертвований.Добавить();
   Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
   Движение.Период = Дата;
   Движение.ДокументПожертвования = ЭтотОбъект.Ссылка;
   Движение.Количество = 1;

КонецПроцедуры
```

Код формы документа «Пожертвования»

```
&НаСервере
Процедура ВыборЗакупкиПриИзмененииНаСервере()

	Если Объект.ВыборЗакупки = Перечисления.ВыборЗакупки.Поставщик тогда
		Элементы.Поставщик.Видимость = Истина;
		Элементы.Издатель.Видимость = Ложь;
	ИначеЕсли Объект.ВыборЗакупки = Перечисления.ВыборЗакупки.Издатель тогда
		Элементы.Поставщик.Видимость = Ложь;
		Элементы.Издатель.Видимость = Истина;
	КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура ВыборЗакупкиПриИзменении(Элемент)

	ВыборЗакупкиПриИзмененииНаСервере();

КонецПроцедуры

&НаСервере
Процедура ПриОткрытииНаСервере()

	Если Объект.ВыборЗакупки = Перечисления.ВыборЗакупки.Поставщик тогда
		Элементы.Поставщик.Видимость = Истина;
		Элементы.Издатель.Видимость = Ложь;
	ИначеЕсли Объект.ВыборЗакупки = Перечисления.ВыборЗакупки.Издатель тогда
		Элементы.Поставщик.Видимость = Ложь;
		Элементы.Издатель.Видимость = Истина;
	КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура ПриОткрытии(Отказ)

	ПриОткрытииНаСервере();

КонецПроцедуры

&НаКлиенте
Процедура КнигиЦенаПриИзменении(Элемент)

	СтрокаТабличнойЧасти = Элементы.Книги.ТекущиеДанные;
	СтрокаТабличнойЧасти.Сумма = СтрокаТабличнойЧасти.Количество*СтрокаТабличнойЧасти.Цена;

КонецПроцедуры

&НаКлиенте
Процедура КнигиСуммаПриИзменении(Элемент)
```

```
	СтрокаТабличнойЧасти = Элементы.Книги.ТекущиеДанные;
	Если СтрокаТабличнойЧасти.Количество = 0 тогда
		СтрокаТабличнойЧасти.ЦенаЗаШ = 0;
	Иначе
		СтрокаТабличнойЧасти.Цена  = СтрокаТабличнойЧасти.Сумма / СтрокаТабличнойЧасти.Количество;
	КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура КнигиКоличествоПриИзменении(Элемент)

	СтрокаТабличнойЧасти = Элементы.Книги.ТекущиеДанные;
	СтрокаТабличнойЧасти.Сумма = СтрокаТабличнойЧасти.Количество*СтрокаТабличнойЧасти.Цена;

КонецПроцедуры

&НаКлиенте
Процедура КнигиКнигаПриИзменении(Элемент)

	СтрокаТЧ = Элементы.Книги.ТекущиеДанные;
	СтрокаТЧ.Цена=ОбщийМодульФункции.ЦеныИзРегистраСведений(Объект.Дата,СтрокаТЧ.Книга);

КонецПроцедуры

&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи)

	ТекущийПользователь = ПользователиИнформационнойБазы.ТекущийПользователь();

	// Поиск сотрудника по логину только если поле "Ответственный" не заполнено
	Если Не ЗначениеЗаполнено(ТекущийОбъект.Ответственный) Тогда
		НайденныйСотрудник = Справочники.Сотрудники.НайтиПоРеквизиту("Логин", ТекущийПользователь.Имя);

		Если НайденныйСотрудник <> Неопределено Тогда

	ТекущийОбъект.Ответственный = НайденныйСотрудник;
		Иначе
			Сообщить("Сотрудник не найден по логину текущего пользователя. Укажите ответственного вручную.");
		КонецЕсли;
	КонецЕсли;

	Попытка

	ВалидацияПолей.ПроверитьEmail(ТекущийОбъект.Почта);
	Исключение
		Сообщить(ОписаниеОшибки());
		Отказ = Истина;
		Возврат;
	КонецПопытки;

	Если НЕ ВалидацияПолей.КорректноВведенНомерТелефона(ТекущийОбъект.Телефон) Тогда
		ВызватьИсключение "Некорректный номер телефона. Пожалуйста, введите номер в международном формате.";
	КонецЕсли;
```

```
КонецПроцедуры
```

Код печати «ПечатьАктаПожертвования»

```
&НаКлиенте
Процедура ОбработкаКоманды(ПараметрКоманды,
ПараметрыВыполненияКоманды)

        ТабДок = Новый ТабличныйДокумент;
        Печать (ТабДок, ПараметрКоманды);
        ТабДок.Показать("Печать Акта
Пожертвования");

КонецПроцедуры

&НаСервере
Процедура Печать(ТабДок, ССылкаНаДокумент)

        Макет =
Документы.Пожертвование.ПолучитьМакет("АктП
ожертвования");
        ОбластьЧердак =
Макет.ПолучитьОбласть("Чердак");
        ОбластьЗаголовок =
Макет.ПолучитьОбласть("Заголовок");
        ОбластьТекст =
Макет.ПолучитьОбласть("Текст");
        ОбластьШапкаТаблици =
Макет.ПолучитьОбласть("ШапкаТаблици");
        ОбластьТаблица =
Макет.ПолучитьОбласть("Таблица");
        ОбластьПодвалТаблици =
Макет.ПолучитьОбласть("ПодвалТаблици");
        ОбластьПодвал =
Макет.ПолучитьОбласть("Подвал");

        ОбластьЧердак.Параметры.Дата =
Формат(ССылкаНаДокумент.Дата,"ДЛФ=DD");
         Запрос = Новый Запрос;
   Запрос.Текст =
      "ВЫБРАТЬ ПЕРВЫЕ 1
      |   Библиотека.Директор.Фамилия КАК
ДиректорФамилия,
      |   Библиотека.Директор.Наименование КАК
ДиректорИмя,
      |   Библиотека.Директор.Отчество КАК
ДиректорОтчество
      |ИЗ
      |   Справочник.Библиотека КАК Библиотека
      |АВТОУПОРЯДОЧИВАНИЕ";

   Результат = Запрос.Выполнить();
   Выборка = Результат.Выбрать();

   Если Выборка.Следующий() Тогда
      ФИОРуководителя =
Выборка.ДиректорФамилия + " " +
Выборка.ДиректорИмя + " " +
Выборка.ДиректорОтчество;

ОбластьЧердак.Параметры.РуководительУчрежден
ия = ФИОРуководителя;
   Иначе

ОбластьЧердак.Параметры.РуководительУчрежден
ия = "";
        КонецЕсли;

        ТабДок.Вывести(ОбластьЧердак);

        ОбластьЗаголовок.Параметры.Номер =
ССылкаНаДокумент.Номер;
        ОбластьЗаголовок.Параметры.Дата =
Формат(ССылкаНаДокумент.Дата,"ДЛФ=DD");
        ТабДок.Вывести(ОбластьЗаголовок);

        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |        ПожертвованиеКниги.Количество
КАК Количество,
        |        ПожертвованиеКниги.Цена КАК
Цена,
        |        ПожертвованиеКниги.Сумма КАК
Сумма
```

```
        |ИЗ
        |        Документ.Пожертвование.Книги
КАК ПожертвованиеКниги
        |ГДЕ
        |        ПожертвованиеКниги.Ссылка =
&ССылкаНаДокумент";

        Запрос.УстановитьПараметр("ССылкаНаДо
кумент", ССылкаНаДокумент);

        РезультатЗапроса = Запрос.Выполнить();

        ВыборкаДетальныеЗаписи =
РезультатЗапроса.Выбрать();
        СуммаКоличство = 0;
        СуммаВсего = 0;
        Пока
ВыборкаДетальныеЗаписи.Следующий() Цикл

        ОбластьТаблица.Параметры.Количество =
ВыборкаДетальныеЗаписи.Количество;
                ОбластьТаблица.Параметры.Цена
= ВыборкаДетальныеЗаписи.Цена;

        ОбластьТаблица.Параметры.Сумма =
ВыборкаДетальныеЗаписи.Сумма;
                СуммаКоличство =
СуммаКоличство+
ВыборкаДетальныеЗаписи.Количество;
                СуммаВсего = СуммаВсего +
ВыборкаДетальныеЗаписи.Сумма;
        КонецЦикла;
```

```
        |АВТОУПОРЯДОЧИВАНИЕ";

    Результат = Запрос.Выполнить();
    Выборка = Результат.Выбрать();

    Если Выборка.Следующий() Тогда
        НазваниеБиблиотеки =
Выборка.ПолноеНаименование;
    Иначе
        НазваниеБиблиотеки = ""; // Или другое
значение по умолчанию
    КонецЕсли;

    ОбластьТекст.Параметры.НазваниеБибилиотеки
= НазваниеБиблиотеки;


            ТекущийПользователь =
ПользователиИнформационнойБазы.ТекущийПоль
зователь();
    Сотрудник =
Справочники.Сотрудники.НайтиПоРеквизиту("Лог
ин", ТекущийПользователь.Имя);

    Если Сотрудник <> Неопределено И
ЗначениеЗаполнено(Сотрудник.Отдел) Тогда
        НазваниеОтдела =
Сотрудник.Отдел.Наименование;
    Иначе
        НазваниеОтдела = ""; // Или другое значение
по умолчанию, например, "Общий отдел"
    КонецЕсли;


ОбластьТекст.Параметры.СтруктурноеПодразделен
ие = НазваниеОтдела;


            ОбластьТекст.Параметры.ФИО =
ССылкаНаДокумент.Фамилия + " " +
```

```
        // Получаем данные библиотеки
    Запрос = Новый Запрос;
    Запрос.Текст =
        "ВЫБРАТЬ ПЕРВЫЕ 1
        |    Библиотека.ПолноеНаименование
        |ИЗ
        |    Справочник.Библиотека КАК Библиотека
```

```
ССылкаНаДокумент.Имя + " " +
ССылкаНаДокумент.Отчество;
	ОбластьТекст.Параметры.Дата =
Формат(ССылкаНаДокумент.Дата,"ДЛФ=D");
	ОбластьТекст.Параметры.Номер =
ССылкаНаДокумент.Номер;
	ОбластьТекст.Параметры.Количество =
СуммаКоличство;
	ОбластьТекст.Параметры.Сумма
=СуммаВсего;
	ОбластьТекст.Параметры.СуммаСловами =
ЧислоПрописью(СуммаВсего,
"Л=ru_RU;ДП=Истина",
"рубль,рубля,рублей,м,копейка,копейки,копеек,ж,2
");

	ТабДок.Вывести(ОбластьТекст);

	ТабДок.Вывести(ОбластьШапкаТаблицы);

	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|
	ПожертвованиеКниги.НомерСтроки КАК
НомерСтроки,
	|	ПожертвованиеКниги.Книга КАК
Книга,
	|	ПожертвованиеКниги.Количество
КАК Количество,
	|	ПожертвованиеКниги.Цена КАК
Цена,
	|	ПожертвованиеКниги.Сумма КАК
Сумма
	|ИЗ
	|	Документ.Пожертвование.Книги
КАК ПожертвованиеКниги
	|ГДЕ
	|	ПожертвованиеКниги.Ссылка =
&ССылкаНаДокумент";

	Запрос.УстановитьПараметр("ССылкаНаДо
кумент", ССылкаНаДокумент);

	РезультатЗапроса = Запрос.Выполнить();

	ВыборкаДетальныеЗаписи =
РезультатЗапроса.Выбрать();
	СуммаКоличство = 0;
	СуммаВсего = 0;
	Пока
ВыборкаДетальныеЗаписи.Следующий() Цикл

	ОбластьТаблица.Параметры.НомерСтроки
= ВыборкаДетальныеЗаписи.НомерСтроки;
		ОбластьТаблица.Параметры.Книга
= ВыборкаДетальныеЗаписи.Книга;

	ОбластьТаблица.Параметры.Количество =
ВыборкаДетальныеЗаписи.Количество;
		ОбластьТаблица.Параметры.Цена
= ВыборкаДетальныеЗаписи.Цена;

	ОбластьТаблица.Параметры.Сумма =
ВыборкаДетальныеЗаписи.Сумма;
		СуммаКоличство =
СуммаКоличство+
ВыборкаДетальныеЗаписи.Количество;
		СуммаВсего = СуммаВсего +
ВыборкаДетальныеЗаписи.Сумма;
		ТабДок.Вывести(ОбластьТаблица);
	КонецЦикла;

	ОбластьПодвалТаблицы.Параметры.ВсегоК
оличество = СуммаКоличство;
	ОбластьПодвалТаблицы.Параметры.ВсегоС
умма = СуммаВсего;
	ТабДок.Вывести(ОбластьПодвалТаблицы);

	ТабДок.Вывести(ОбластьПодвал);


КонецПроцедуры
```

Код модуля объекта документа «ВыдачаКниг»

```
Функция ПолучитьОстаток(Книга)

	Результат =0;
	Отбор = Новый Структура;
	Отбор.Вставить("Книги", Книга);
	НайденныеЗначения = РегистрыНакопления.ДоступныеКниги.Остатки(Дата,Отбор);
	Если НайденныеЗначения.Количество()>0 Тогда
		Результат = НайденныеЗначения[0].Количество;
	КонецЕсли;
	Возврат Результат;

КонецФункции

Процедура ОбработкаПроведения(Отказ, Режим)

	// регистр СрокВозвратаКниг
	Движения.СрокВозвратаКниг.Записывать = Истина;
	Для Каждого ТекСтрокаКниги Из Книги Цикл
		Движение = Движения.СрокВозвратаКниг.Добавить();
		Движение.Период = Дата; //МС()
		Движение.Книга = ТекСтрокаКниги.Книга;
		Движение.Читатель = Читатель;
		Движение.ВремяДействия = ЭтотОбъект.Дата;
		Движение.Количество = ТекСтрокаКниги.Количество;
		Движение.ДокументВыдачи = ЭтотОбъект.Ссылка;

	Движение.ПланируемаяДатаВозврата = ТекСтрокаКниги.ДатаВозвратаКниги;
	КонецЦикла;
	// регистр ДоступныеКниги Расход
	Движения.ДоступныеКниги.Записывать = Истина;
	Для Каждого ТекСтрокаКниги Из Книги Цикл
		ДоступныеКниги = ПолучитьОстаток(ТекСтрокаКниги.Книга);
		Если ДоступныеКниги<ТекСтрокаКниги.Количество Тогда
			Сообщить(" Не достаточно книг "+ТекСтрокаКниги.Книга+" Доступный остаток: "+ДоступныеКниги);
			Отказ = Истина;
		КонецЕсли;
		Движение = Движения.ДоступныеКниги.Добавить();
		Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
		Движение.Период = Дата;
		Движение.Книги = ТекСтрокаКниги.Книга;
		Движение.Количество = ТекСтрокаКниги.Количество;
	КонецЦикла;
	// регистр ИсторияВыдачКниг Приход
	Движения.ИсторияВыдачКниг.Записывать = Истина;
	Для Каждого ТекСтрокаКниги Из Книги Цикл
		Движение = Движения.ИсторияВыдачКниг.Добавить();
		Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
		Движение.Период = Дата;
		Движение.Книга = ТекСтрокаКниги.Книга;
		Движение.Читатель = Читатель;
		Движение.Количество = ТекСтрокаКниги.Количество;
	КонецЦикла;
```

```
        // регистр ПопулярностьКниги Приход
        Движения.ПопулярностьКниги.Записывать = Истина;
        Для Каждого ТекСтрокаКниги Из Книги Цикл
                Движение = Движения.ПопулярностьКниги.Добавить();
                Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
                Движение.Период = Дата;
                Движение.Книга = ТекСтрокаКниги.Книга;
                Движение.Количество = ТекСтрокаКниги.Количество;
        КонецЦикла;

        // Проверка читательского билета
        Если Не ПроверитьЧитательскийБилет(Читатель, Книги) Тогда
                Отказ = Истина;
                Возврат;
        КонецЕсли;

КонецПроцедуры

Процедура ПриЗаписи(Отказ)

        Если Не ЗначениеЗаполнено(ЭтотОбъект.Ответственный) Тогда
                ЭтотОбъект.Ответственный = ПользователиИнформационнойБазы.ТекущийПользователь();
        КонецЕсли;

КонецПроцедуры

Процедура ПередЗаписью(Отказ, РежимЗаписи, РежимПроведения)

        Если Не ЗначениеЗаполнено(ЭтотОбъект.Ответственный) Тогда
                ЭтотОбъект.Ответственный = ПользователиИнформационнойБазы.ТекущийПользователь();
        КонецЕсли;

КонецПроцедуры

&НаСервере
Функция ПроверитьЧитательскийБилет(Читатель, Книги)

        // Проверка срока действия билета
        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ ПЕРВЫЕ 1
        |
СрокиДействияЧитательскихБилетов.ДатаОкончания
        |ИЗ
        |
РегистрСведений.СрокиДействияЧитательскихБилетов КАК СрокиДействияЧитательскихБилетов
        |ГДЕ
        |
СрокиДействияЧитательскихБилетов.ЧитательскийБилет.Читатель = &Читатель
        |УПОРЯДОЧИТЬ ПО
        |
СрокиДействияЧитательскихБилетов.Период УБЫВ";

        Запрос.УстановитьПараметр("Читатель", Читатель);
        Результат = Запрос.Выполнить();
        Выборка = Результат.Выбрать();

        Если Выборка.Следующий() Тогда
                ДатаОкончанияБилета = Выборка.ДатаОкончания;
```

```
            Если ДатаОкончанияБилета < Дата
Тогда
                Сообщить("Срок действия
читательского билета истек!");
                Возврат Ложь;
            КонецЕсли;
        Иначе
            Сообщить("У читателя нет
читательского билета!");
            Возврат Ложь;
        КонецЕсли;
        // Получение возрастного рейтинга
читательского билета
        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ ПЕРВЫЕ 1
        |
        ЧитательскийБилет.ВозрастнойРейтинг
КАК ВозрастнойРейтинг
        |ИЗ
        |    Документ.ЧитательскийБилет КАК
ЧитательскийБилет
        |ГДЕ
        |    ЧитательскийБилет.Читатель =
&Читатель
        |УПОРЯДОЧИТЬ ПО
        |    ЧитательскийБилет.Дата УБЫВ";

        Запрос.УстановитьПараметр("Читатель",
Читатель);
        Результат = Запрос.Выполнить();
        Выборка = Результат.Выбрать();

        Если Выборка.Следующий() Тогда
            ВозрастнойРейтингБилета =
Выборка.ВозрастнойРейтинг;
        Иначе
            Сообщить("У читателя нет
читательского билета или у него не указан
возрастной рейтинг!"); // Более информативное
сообщение
            Возврат Ложь;
        КонецЕсли;

        // Проверка возрастного рейтинга книг
        Для Каждого СтрокаКниги Из Книги Цикл
            Если Не
ПроверитьВозрастнойРейтинг(Читатель,
СтрокаКниги.Книга, ВозрастнойРейтингБилета)
Тогда
                Сообщить("Возрастной
рейтинг книги """ + СтрокаКниги.Книга + """ не
соответствует возрастному рейтингу читательского
билета.");
                Возврат Ложь;
            КонецЕсли;
        КонецЦикла;

        Возврат Истина;
КонецФункции

&НаСервере
Функция ПроверитьВозрастнойРейтинг(Читатель,
Книга, ВозрастнойРейтингБилета)

        ВозрастЧитателя =
СколькоЛетТебе(ТекущаяДата(),
Читатель.ДатаРождения);
        РейтингКниги = Книга.ВозрастнойРейтинг;

        Если ВозрастнойРейтингБилета =
Перечисления.ВозрастнойРейтинг.БезОграничений
Тогда
            Возврат Истина;
        ИначеЕсли ВозрастнойРейтингБилета =
Перечисления.ВозрастнойРейтинг.ДляДетейОт6лет
            И (РейтингКниги =
Перечисления.ВозрастнойРейтинг.ДляДетейОт6лет
            ИЛИ РейтингКниги =
Перечисления.ВозрастнойРейтинг.БезОграничений
) Тогда
            Возврат Истина;
```

```
        ИначеЕсли ВозрастнойРейтингБилета =
Перечисления.ВозрастнойРейтинг.ДляДетейОт12Л
ет
                И (РейтингКниги =
Перечисления.ВозрастнойРейтинг.ДляДетейОт6лет
                ИЛИ РейтингКниги =
Перечисления.ВозрастнойРейтинг.ДляДетейОт12Л
ет
                ИЛИ РейтингКниги =
Перечисления.ВозрастнойРейтинг.БезОграничений
) Тогда
                Возврат Истина;
        ИначеЕсли ВозрастнойРейтингБилета =
Перечисления.ВозрастнойРейтинг.ДляПодростков
От16Лет
                И (РейтингКниги =
Перечисления.ВозрастнойРейтинг.ДляДетейОт6лет
                ИЛИ РейтингКниги =
Перечисления.ВозрастнойРейтинг.ДляДетейОт12Л
ет
                ИЛИ РейтингКниги =
Перечисления.ВозрастнойРейтинг.ДляПодростков
От16Лет
                ИЛИ РейтингКниги =
Перечисления.ВозрастнойРейтинг.БезОграничений
) Тогда
                Возврат Истина;
        ИначеЕсли ВозрастнойРейтингБилета =
Перечисления.ВозрастнойРейтинг.ДляВзрослых
Тогда
                Возврат Истина; // Взрослым
можно выдавать любые книги
        Иначе
                Возврат Ложь;
        КонецЕсли;

КонецФункции

&НаСервере
Функция СколькоЛетТебе(ТекДата, ДатаРождения)

        Если ДатаРождения = Неопределено Тогда
                Возврат 0;
        КонецЕсли;

        Возраст = Год(ТекДата) -
Год(ДатаРождения);
        Если Месяц(ТекДата) <
Месяц(ДатаРождения) Или (Месяц(ТекДата) =
Месяц(ДатаРождения) И День(ТекДата) <
День(ДатаРождения)) Тогда
                Возраст = Возраст - 1;
        КонецЕсли;

        Возврат Возраст;

КонецФункции
```

Код формы документа «ВыдачаКниг»

```
&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ,
ТекущийОбъект, ПараметрыЗаписи)

        Для Каждого СтрокаТЧ Из
ТекущийОбъект.Книги Цикл
                Если
СтрокаТЧ.ДатаВозвратаКниги >
ДобавитьМесяц(ТекущийОбъект.Дата, 1) Тогда
                        Сообщить("Срок выдачи
книги не может превышать один месяц!");
                        Отказ = Истина;
                        Возврат; // Прерываем
выполнение, если ошибка найдена
                КонецЕсли;
        КонецЦикла;

        Если Не
ЗначениеЗаполнено(Объект.Ответственный) Тогда
                НайденныйСотрудник =
Справочники.Сотрудники.НайтиПоНаименованию(
ПользователиИнформационнойБазы.ТекущийПоль
зователь().Имя);
                Если НайденныйСотрудник <>
Неопределено Тогда
```

```
                Объект.Ответственный =
НайденныйСотрудник;
            КонецЕсли;
        КонецЕсли;



        ТекущийПользователь =
ПользователиИнформационнойБазы.ТекущийПоль
зователь();

        // Поиск сотрудника по логину только если
поле "Ответственный" не заполнено
        Если Не
ЗначениеЗаполнено(ТекущийОбъект.Ответственны
й) Тогда
            НайденныйСотрудник =
Справочники.Сотрудники.НайтиПоРеквизиту("Лог
ин", ТекущийПользователь.Имя);

            Если НайденныйСотрудник <>
Неопределено Тогда

        ТекущийОбъект.Ответственный =
НайденныйСотрудник;
            Иначе
                Сообщить("Сотрудник не
найден по логину текущего пользователя. Укажите
ответственного вручную.");
            КонецЕсли;
        КонецЕсли;

КонецПроцедуры
```

Код модуля объекта документа «ВозвратКниг»

```
&НаСервере
Процедура ОбработкаПроведения(Отказ, Режим)

        Движения.ИсторияВыдачКниг.Записывать
= Истина;
        Для Каждого ТекСтрокаКниги Из Книги
Цикл
            // Поиск записи о выдаче данной
книги этому читателю
            ЗаписьВыдачи =
НайтиЗаписьВыдачи(ТекСтрокаКниги.Книга,
Читатель);

            Если ЗаписьВыдачи =
Неопределено Тогда
                Сообщить("Не найдена
запись о выдаче книги " + ТекСтрокаКниги.Книга +
" читателю " + Читатель);
                Отказ = Истина;
                Продолжить; // Переходим
к следующей книге
            КонецЕсли;
            // Проверка количества
            Если ЗаписьВыдачи.Количество <
ТекСтрокаКниги.Количество Тогда
                Сообщить("Попытка
вернуть больше книг, чем было выдано. Доступно
для возврата: " + ЗаписьВыдачи.Количество);
                Отказ = Истина;
                Продолжить;
            КонецЕсли;
            // Создание движения по регистру
            Движение =
Движения.ИсторияВыдачКниг.Добавить();
            Движение.ВидДвижения =
ВидДвиженияНакопления.Расход;
            Движение.Период = Дата;
            Движение.Книга =
ТекСтрокаКниги.Книга;
            Движение.Читатель = Читатель;
            Движение.Количество =
ТекСтрокаКниги.Количество;
        КонецЦикла;
        // регистр ДоступныеКниги Приход
        Движения.ДоступныеКниги.Записывать =
Истина;
        Для Каждого ТекСтрокаКниги Из Книги
Цикл
```

```
                Движение =
Движения.ДоступныеКниги.Добавить();
                Движение.ВидДвижения =
ВидДвиженияНакопления.Приход;
                Движение.Период = Дата;
                Движение.Книги =
ТекСтрокаКниги.Книга;
                Движение.Количество =
ТекСтрокаКниги.Количество;
        КонецЦикла;
        // регистр СрокВозвратаКниг
        Движения.СрокВозвратаКниг.Записывать
= Истина;
        Для Каждого ТекСтрокаКниги Из Книги
Цикл
                Движение =
Движения.СрокВозвратаКниг.Добавить();
                Движение.Период = Дата; //МС()
                Движение.Книга =
ТекСтрокаКниги.Книга;
                Движение.Читатель = Читатель;
                Движение.ВремяДействия =
ЭтотОбъект.Дата;
                Движение.Количество =
ТекСтрокаКниги.Количество;

        Движение.ФактическаяДатаВозврата =
Дата;
                Движение.ДокументВыдачи =
ДокументОснования;

                ЗаписьВыдачи =
НайтиДатуВыдачи(ТекСтрокаКниги.Книга);

                Если Не ЗаписьВыдачи =
Неопределено Тогда

        Движение.ПланируемаяДатаВозврата =
ЗаписьВыдачи.ПланируемаяДатаВозврата;
                КонецЕсли;

        КонецЦикла;

        // регистр ШтрафыЗаПросрочку
        ЦенаЗаДеньПросрочки =
ОбщийМодульФункции.ПолучитьЦенуШтрафа(Пе
речисления.НазваниеШтрафа.Просрочка);
        Движения.ШтрафыЗаПросрочку.Записыват
ь = Истина;
        Для Каждого ТекСтрокаКниги Из Книги
Цикл
                ЗаписьВыдачи =
НайтиДатуВыдачи(ТекСтрокаКниги.Книга);

                Если Не ЗаписьВыдачи =
Неопределено Тогда
                        ПланируемаяДатаВозврата
= ЗаписьВыдачи.ПланируемаяДатаВозврата;

                        // Расчет просроченных
дней
                        ПросроченныеДни =
?(Дата > ПланируемаяДатаВозврата, Цел((Дата -
ПланируемаяДатаВозврата) / 86400), 0);
                        Если ПросроченныеДни >
0 Тогда
                                Движение =
Движения.ШтрафыЗаПросрочку.Добавить();
                                Движение.Период
= Дата;

        Движение.Основание =
ЭтотОбъект.Ссылка; // Заполнение основания
ссылкой на текущий документ
                                Движение.Книга =
ТекСтрокаКниги.Книга;

        Движение.Читатель = Читатель;

        Движение.Количество =
ТекСтрокаКниги.Количество;

        Движение.СуммаШтрафа =
```

```
ПросроченныеДни * ЦенаЗаДеньПросрочки *
ТекСтрокаКниги.Количество;

        Движение.Оплачено = Ложь;
                КонецЕсли;
            КонецЕсли;
        КонецЦикла;

КонецПроцедуры

// Функция для поиска записи о выдаче
Функция НайтиЗаписьВыдачи(Книга, Читатель)

        Отбор = Новый Структура("Книга,
Читатель", Книга, Читатель);
        // Ищем запись с нужной книгой,
читателем
        // и ПУСТЫМ ДокументВыдачи - то есть,
не возвращенную ранее
        НайденныеЗначения =
РегистрыНакопления.ИсторияВыдачКниг.Остатки(
Дата, Отбор);

        Если НайденныеЗначения.Количество() > 0
Тогда
            Возврат НайденныеЗначения[0];
        Иначе
            Возврат Неопределено;
        КонецЕсли;

КонецФункции

//Функция для поиска даты выдачи
Функция НайтиДатуВыдачи(Книга)

        Отбор = Новый Структура("Книга,
Читатель, ДокументВыдачи", Книга, Читатель,
ДокументОснования);
        НайденныеЗначения =
РегистрыСведений.СрокВозвратаКниг.ПолучитьПо
следнее(Дата, Отбор);

        Возврат НайденныеЗначения;

КонецФункции

Процедура
ОбработкаЗаполнения(ДанныеЗаполнения,
СтандартнаяОбработка)

        Если ТипЗнч(ДанныеЗаполнения) =
Тип("ДокументСсылка.ВыдачаКниг") Тогда
            // Заполнение шапки
            ДокументОснования =
ДанныеЗаполнения.Ссылка;
            Читатель =
ДанныеЗаполнения.Читатель;
            Для Каждого ТекСтрокаКниги Из
ДанныеЗаполнения.Книги Цикл
                НоваяСтрока =
Книги.Добавить();
                НоваяСтрока.Книга =
ТекСтрокаКниги.Книга;
                НоваяСтрока.Количество
= ТекСтрокаКниги.Количество;

            НоваяСтрока.КачествоКнигиПриВозврате
=ТекСтрокаКниги.КачествоКнигиПриВыдаче;
            КонецЦикла;
        ИначеЕсли ТипЗнч(ДанныеЗаполнения) =
Тип("ДокументСсылка.ПродлениеКниги") Тогда
            // Заполнение шапки
            ДокументОснования =
ДанныеЗаполнения.ДокументОснования;
            Читатель =
ДанныеЗаполнения.Читатель;
            Для Каждого ТекСтрокаКниги Из
ДанныеЗаполнения.Книги Цикл
                НоваяСтрока =
Книги.Добавить();
                НоваяСтрока.Книга =
ТекСтрокаКниги.Книга;
                НоваяСтрока.Количество
= ТекСтрокаКниги.Количество;
```

```
            КонецЦикла;
        КонецЕсли;

КонецПроцедуры
```

Код модуля объекта документа «ВозвратКниг»

```
&НаСервере
Процедура ОбработкаПроведения(Отказ, Режим)

        Движения.ИсторияВыдачКниг.Записывать = Истина;
        Для Каждого ТекСтрокаКниги Из Книги Цикл
            // Поиск записи о выдаче данной книги этому читателю
            ЗаписьВыдачи = НайтиЗаписьВыдачи(ТекСтрокаКниги.Книга, Читатель);

            Если ЗаписьВыдачи = Неопределено Тогда
                Сообщить("Не найдена запись о выдаче книги " + ТекСтрокаКниги.Книга + " читателю " + Читатель);
                Отказ = Истина;
                Продолжить; // Переходим к следующей книге
            КонецЕсли;
            // Проверка количества
            Если ЗаписьВыдачи.Количество < ТекСтрокаКниги.Количество Тогда
                Сообщить("Попытка вернуть больше книг, чем было выдано. Доступно для возврата: " + ЗаписьВыдачи.Количество);
                Отказ = Истина;
                Продолжить;
            КонецЕсли;
            // Создание движения по регистру
            Движение = Движения.ИсторияВыдачКниг.Добавить();
            Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
            Движение.Период = Дата;
            Движение.Книга = ТекСтрокаКниги.Книга;
            Движение.Читатель = Читатель;
            Движение.Количество = ТекСтрокаКниги.Количество;
        КонецЦикла;
        // регистр ДоступныеКниги Приход
        Движения.ДоступныеКниги.Записывать = Истина;
        Для Каждого ТекСтрокаКниги Из Книги Цикл
            Движение = Движения.ДоступныеКниги.Добавить();
            Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
            Движение.Период = Дата;
            Движение.Книги = ТекСтрокаКниги.Книга;
            Движение.Количество = ТекСтрокаКниги.Количество;
        КонецЦикла;
        // регистр СрокВозвратаКниг
        Движения.СрокВозвратаКниг.Записывать = Истина;
        Для Каждого ТекСтрокаКниги Из Книги Цикл
            Движение = Движения.СрокВозвратаКниг.Добавить();
            Движение.Период = Дата; //МС()
            Движение.Книга = ТекСтрокаКниги.Книга;
            Движение.Читатель = Читатель;
            Движение.ВремяДействия = ЭтотОбъект.Дата;
            Движение.Количество = ТекСтрокаКниги.Количество;

        Движение.ФактическаяДатаВозврата = Дата;
            Движение.ДокументВыдачи = ДокументОснования;
```

```
            ЗаписьВыдачи =
НайтиДатуВыдачи(ТекСтрокаКниги.Книга);

            Если Не ЗаписьВыдачи =
Неопределено Тогда

        Движение.ПланируемаяДатаВозврата =
ЗаписьВыдачи.ПланируемаяДатаВозврата;
            КонецЕсли;

        КонецЦикла;

        // регистр ШтрафыЗаПросрочку
        ЦенаЗаДеньПросрочки =
ОбщийМодульФункции.ПолучитьЦенуШтрафа(Пе
речисления.НазваниеШтрафа.Просрочка);
        Движения.ШтрафыЗаПросрочку.Записыват
ь = Истина;
        Для Каждого ТекСтрокаКниги Из Книги
Цикл
            ЗаписьВыдачи =
НайтиДатуВыдачи(ТекСтрокаКниги.Книга);

            Если Не ЗаписьВыдачи =
Неопределено Тогда
                ПланируемаяДатаВозврата
= ЗаписьВыдачи.ПланируемаяДатаВозврата;

                // Расчет просроченных
дней
                ПросроченныеДни =
?(Дата > ПланируемаяДатаВозврата, Цел((Дата -
ПланируемаяДатаВозврата) / 86400), 0);
                Если ПросроченныеДни >
0 Тогда
                    Движение =
Движения.ШтрафыЗаПросрочку.Добавить();
                    Движение.Период
= Дата;

        Движение.Основание =
ЭтотОбъект.Ссылка; // Заполнение основания
ссылкой на текущий документ
                    Движение.Книга =
ТекСтрокаКниги.Книга;

        Движение.Читатель = Читатель;

        Движение.Количество =
ТекСтрокаКниги.Количество;

        Движение.СуммаШтрафа =
ПросроченныеДни * ЦенаЗаДеньПросрочки *
ТекСтрокаКниги.Количество;

        Движение.Оплачено = Ложь;
                КонецЕсли;
            КонецЕсли;
        КонецЦикла;

КонецПроцедуры

// Функция для поиска записи о выдаче
Функция НайтиЗаписьВыдачи(Книга, Читатель)

        Отбор = Новый Структура("Книга,
Читатель", Книга, Читатель);
        // Ищем запись с нужной книгой,
читателем
        // и ПУСТЫМ ДокументВыдачи - то есть,
не возвращенную ранее
        НайденныеЗначения =
РегистрыНакопления.ИсторияВыдачКниг.Остатки(
Дата, Отбор);

        Если НайденныеЗначения.Количество() > 0
Тогда
            Возврат НайденныеЗначения[0];
        Иначе
            Возврат Неопределено;
        КонецЕсли;

КонецФункции
```

```
//Функция для поиска даты выдачи
Функция НайтиДатуВыдачи(Книга)

	Отбор = Новый Структура("Книга, Читатель, ДокументВыдачи", Книга, Читатель, ДокументОснования);
	НайденныеЗначения = РегистрыСведений.СрокВозвратаКниг.ПолучитьПоследнее(Дата, Отбор);

	Возврат НайденныеЗначения;

КонецФункции

Процедура ОбработкаЗаполнения(ДанныеЗаполнения, СтандартнаяОбработка)

	Если ТипЗнч(ДанныеЗаполнения) = Тип("ДокументСсылка.ВыдачаКниг") Тогда
		// Заполнение шапки
		ДокументОснования = ДанныеЗаполнения.Ссылка;
		Читатель = ДанныеЗаполнения.Читатель;
		Для Каждого ТекСтрокаКниги Из ДанныеЗаполнения.Книги Цикл
			НоваяСтрока = Книги.Добавить();
			НоваяСтрока.Книга = ТекСтрокаКниги.Книга;
			НоваяСтрока.Количество = ТекСтрокаКниги.Количество;

		НоваяСтрока.КачествоКнигиПриВозврате =ТекСтрокаКниги.КачествоКнигиПриВыдаче;
		КонецЦикла;
	ИначеЕсли ТипЗнч(ДанныеЗаполнения) = Тип("ДокументСсылка.ПродлениеКниги") Тогда
		// Заполнение шапки
		ДокументОснования = ДанныеЗаполнения.ДокументОснования;
		Читатель = ДанныеЗаполнения.Читатель;
		Для Каждого ТекСтрокаКниги Из ДанныеЗаполнения.Книги Цикл
			НоваяСтрока = Книги.Добавить();
			НоваяСтрока.Книга = ТекСтрокаКниги.Книга;
			НоваяСтрока.Количество = ТекСтрокаКниги.Количество;
		КонецЦикла;
	КонецЕсли;

КонецПроцедуры
```

Код формы документа «ВозвратКниг»

```
&НаСервере
Процедура ДокументОснованияПриИзмененииНаСервере()

	Если Объект.ДокументОснования = Неопределено Тогда
		Объект.Книги.Очистить(); // Очищаем табличную часть, если основание не выбрано
		Возврат;
	КонецЕсли;

	Выдача = Объект.ДокументОснования.ПолучитьОбъект();

	Объект.Читатель = Выдача.Читатель;
	Объект.Книги.Очистить();

	Для Каждого СтрокаВыдачи Из Выдача.Книги Цикл
		НоваяСтрока = Объект.Книги.Добавить();
```

```
            НоваяСтрока.Книга =
СтрокаВыдачи.Книга;
            НоваяСтрока.Количество =
СтрокаВыдачи.Количество;
        КонецЦикла;

КонецПроцедуры


&НаКлиенте
Процедура
ДокументОснованияПриИзменении(Элемент)

        ДокументОснованияПриИзмененииНаСерв
ере();

КонецПроцедуры


&НаСервере
Процедура ЧитательПриИзмененииНаСервере()

        Если Объект.ДокументОснования =
Неопределено Тогда
            Возврат; // Если основание не
выбрано, ничего не проверяем
        КонецЕсли;

        ЧитательИзОснования =
Объект.ДокументОснования.ПолучитьОбъект().Чит
атель;

        Если Объект.Читатель <>
ЧитательИзОснования Тогда
            Сообщить("Читатель не
соответствует выбранному основанию! Читатель
должен быть: " + ЧитательИзОснования);
            Объект.Читатель =
ЧитательИзОснования; // Восстанавливаем
читателя из основания
        КонецЕсли;
КонецПроцедуры


&НаКлиенте
Процедура ЧитательПриИзменении(Элемент)

        ЧитательПриИзмененииНаСервере();

КонецПроцедуры


&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ,
ТекущийОбъект, ПараметрыЗаписи)

        ТекущийПользователь =
ПользователиИнформационнойБазы.ТекущийПольз
ователь();

        // Поиск сотрудника по логину только если
поле "Ответственный" не заполнено
        Если Не
ЗначениеЗаполнено(ТекущийОбъект.Ответственны
й) Тогда
            НайденныйСотрудник =
Справочники.Сотрудники.НайтиПоРеквизиту("Лог
ин", ТекущийПользователь.Имя);

            Если НайденныйСотрудник <>
Неопределено Тогда

        ТекущийОбъект.Ответственный =
НайденныйСотрудник;
            Иначе
                Сообщить("Сотрудник не
найден по логину текущего пользователя. Укажите
ответственного вручную.");
            КонецЕсли;
        КонецЕсли;

КонецПроцедуры
```

```
&НаСервере
Процедура СравнитьСостояниеКнигиНаСервере(ССылкаНаКнигу, НовоеСостояние)

    ДокументОснованияССылка = Объект.ДокументОснования;

    Если НЕ ЗначениеЗаполнено(ДокументОснованияССылка) Тогда
        Возврат;
    КонецЕсли;

    ДокументВыдачиОбъект = ДокументОснованияССылка.ПолучитьОбъект();
    СтрокаВыдачи = ДокументВыдачиОбъект.Книги.Найти(ССылкаНаКнигу, "Книга");

    Если СтрокаВыдачи = Неопределено Тогда
        Сообщить("В документе выдачи не найдена информация о книге: " + ССылкаНаКнигу);
        Возврат;
    КонецЕсли;

    СостояниеПриВыдаче = СтрокаВыдачи.КачествоКнигиПриВыдаче;

    Если СостояниеПриВыдаче <> НовоеСостояние Тогда
        ТекстСообщения = "Внимание! Состояние книги '" + ССылкаНаКнигу +
                         "' изменилось. При выдаче: " + СостояниеПриВыдаче +
                         ", при возврате: " + НовоеСостояние + ".";
        Сообщить(ТекстСообщения);
    КонецЕсли;

КонецПроцедуры
&НаКлиенте
Процедура КнигиКачествоКнигиПриВозвратеПриИзменении(Элемент)
    // Получаем текущую строку табличной части
    СтрокаТЧ = Элементы.Книги.ТекущиеДанные;

    // Проверяем, что выбрана книга и указан документ основания
    Если СтрокаТЧ = Неопределено ИЛИ НЕ ЗначениеЗаполнено(СтрокаТЧ.Книга) ИЛИ НЕ ЗначениеЗаполнено(Объект.ДокументОснования) Тогда
        Возврат; // Нечего проверять
    КонецЕсли;

    // Получаем новое состояние из текущей строки
    НовоеСостояние = СтрокаТЧ.КачествоКнигиПриВозврате;

    // Вызываем серверную функцию для получения состояния при выдаче и сравнения
    СравнитьСостояниеКнигиНаСервере(СтрокаТЧ.Книга, НовоеСостояние);

КонецПроцедуры
```

Код модуля объекта «ПродлениеКниги»

```
Процедура ОбработкаЗаполнения(ДанныеЗаполнения, СтандартнаяОбработка)

    Если ТипЗнч(ДанныеЗаполнения) = Тип("ДокументСсылка.ВыдачаКниг") Тогда
        // Заполнение шапки
        ДокументОснования = ДанныеЗаполнения.Ссылка;
        Читатель = ДанныеЗаполнения.Читатель;
```

```
            Для Каждого ТекСтрокаКниги Из
ДанныеЗаполнения.Книги Цикл
                НоваяСтрока =
Книги.Добавить();
                НоваяСтрока.Книга =
ТекСтрокаКниги.Книга;
                НоваяСтрока.Количество
= ТекСтрокаКниги.Количество;
            КонецЦикла;
        ИначеЕсли ТипЗнч(ДанныеЗаполнения) =
Тип("ДокументСсылка.ПродлениеКниги") Тогда
            // Заполнение шапки
            ДокументОснования =
ДанныеЗаполнения.ДокументОснования;
            Комментарий =
ДанныеЗаполнения.Комментарий;
            Ответственный =
ДанныеЗаполнения.Ответственный;
            Читатель =
ДанныеЗаполнения.Читатель;
            Для Каждого ТекСтрокаКниги Из
ДанныеЗаполнения.Книги Цикл
                НоваяСтрока =
Книги.Добавить();
                НоваяСтрока.Книга =
ТекСтрокаКниги.Книга;
                НоваяСтрока.Количество
= ТекСтрокаКниги.Количество;

        НоваяСтрока.НоваяДатаСдачи =
ТекСтрокаКниги.НоваяДатаСдачи;
            КонецЦикла;
        КонецЕсли;

КонецПроцедуры

Процедура ОбработкаПроведения(Отказ, Режим)

        // регистр СрокВозвратаКниг
        Движения.СрокВозвратаКниг.Записывать
= Истина;
        Для Каждого ТекСтрокаКниги Из Книги
Цикл
            Движение =
Движения.СрокВозвратаКниг.Добавить();
            Движение.Период = Дата;
            Движение.Книга =
ТекСтрокаКниги.Книга;
            Движение.Читатель = Читатель;
            Движение.ВремяДействия =
ЭтотОбъект.Дата;
            Движение.ДокументВыдачи =
ДокументОснования;
            Движение.Количество =
ТекСтрокаКниги.Количество;

        Движение.ПланируемаяДатаВозврата =
ТекСтрокаКниги.НоваяДатаСдачи;
        КонецЦикла;

КонецПроцедуры

Процедура ПриЗаписи(Отказ)

        Если Не
ЗначениеЗаполнено(ЭтотОбъект.Ответственный)
Тогда
            ЭтотОбъект.Ответственный =
ПользователиИнформационнойБазы.ТекущийПоль
зователь();
        КонецЕсли;

КонецПроцедуры
```

Код формы документа «ПродлениеКниги»

```
&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ,
ТекущийОбъект, ПараметрыЗаписи)

        Для Каждого СтрокаТЧ Из
ТекущийОбъект.Книги Цикл
```

```
                Если СтрокаТЧ.НоваяДатаСдачи > ДобавитьМесяц(ТекущийОбъект.Дата, 1) Тогда // Проверка новой даты сдачи
                    Сообщить("Срок продления книги не может превышать один месяц от текущей даты!");
                    Отказ = Истина;
                    Возврат; // Прерываем выполнение, если ошибка найдена
                КонецЕсли;
            КонецЦикла;

            Для Каждого СтрокаТЧ Из ТекущийОбъект.Книги Цикл

                // Получаем дату последнего продления или выдачи из регистра "СрокВозвратаКниг"
                Запрос = Новый Запрос;
                Запрос.Текст =
                "ВЫБРАТЬ ПЕРВЫЕ 1
                |
СрокВозвратаКниг.ПланируемаяДатаВозврата
                |ИЗ
                |
РегистрСведений.СрокВозвратаКниг КАК СрокВозвратаКниг
                |ГДЕ
                |    СрокВозвратаКниг.Книга = &Книга
                |    И СрокВозвратаКниг.Читатель = &Читатель
                |УПОРЯДОЧИТЬ ПО
                |    СрокВозвратаКниг.Период УБЫВ";

                Запрос.УстановитьПараметр("Книга", СтрокаТЧ.Книга);

                Запрос.УстановитьПараметр("Читатель", ТекущийОбъект.Читатель); // Читатель из объекта документа
                Результат = Запрос.Выполнить();

                Если Не Результат.Пустой() Тогда
                    Выборка = Результат.Выбрать();
                    Выборка.Следующий();
                    ТекущаяДатаВозврата = Выборка.ПланируемаяДатаВозврата;

                    // Проверяем, что новая дата возврата не меньше текущей
                    Если СтрокаТЧ.НоваяДатаСдачи < ТекущаяДатаВозврата Тогда
                        Сообщить("Новая дата возврата не может быть раньше текущей даты возврата!");
                        Отказ = Истина;
                        Возврат; // Прерываем процедуру, если ошибка найдена
                    КонецЕсли;

                Иначе
                    Сообщить("Не найдена запись о выдаче книги " + СтрокаТЧ.Книга);
                    Отказ = Истина;
                    Возврат;
                КонецЕсли;
            КонецЦикла;

КонецПроцедуры

&НаСервере
Процедура ДокументОснованияПриИзмененииНаСервере()

    Если Объект.ДокументОснования = Неопределено Тогда
```

```
            Объект.Книги.Очистить(); // Очищаем табличную часть, если основание не выбрано
            Возврат;
        КонецЕсли;

        Выдача = Объект.ДокументОснования.ПолучитьОбъект();

        Объект.Читатель = Выдача.Читатель;
        Объект.Книги.Очистить();

        Для Каждого СтрокаВыдачи Из Выдача.Книги Цикл
            НоваяСтрока = Объект.Книги.Добавить();
            НоваяСтрока.Книга = СтрокаВыдачи.Книга;
            НоваяСтрока.Количество = СтрокаВыдачи.Количество;
        КонецЦикла;

КонецПроцедуры

&НаКлиенте
Процедура ДокументОснованияПриИзменении(Элемент)

        ДокументОснованияПриИзмененииНаСервере();

КонецПроцедуры

&НаСервере
Процедура ЧитательПриИзмененииНаСервере()

        Если Объект.ДокументОснования = Неопределено Тогда
            Возврат; // Если основание не выбрано, ничего не проверяем
        КонецЕсли;

        ЧитательИзОснования = Объект.ДокументОснования.ПолучитьОбъект().Читатель;

        Если Объект.Читатель <> ЧитательИзОснования Тогда
            Сообщить("Читатель не соответствует выбранному основанию! Читатель должен быть: " + ЧитательИзОснования);
            Объект.Читатель = ЧитательИзОснования; // Восстанавливаем читателя из основания
        КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура ЧитательПриИзменении(Элемент)

        ЧитательПриИзмененииНаСервере();

КонецПроцедуры
```

Код модуля объекта «ОплатаШтрафов»

```
&НаСервере
Процедура ОбработкаПроведения(Отказ, РежимПроведения)

        Для Каждого СтрокаТЧ Из ЭтотОбъект.Штрафы Цикл

            Если СтрокаТЧ.Оплачено Тогда

                НаборЗаписей = РегистрыСведений.ШтрафыЗаПросрочку.СоздатьНаборЗаписей();

                // Отбор ТОЛЬКО по регистратору (Основание)

                НаборЗаписей.Отбор.Регистратор.Установить(СтрокаТЧ.Основание);
```

```
        НаборЗаписей.Прочитать();

                Для Каждого Запись Из НаборЗаписей Цикл
                    // Проверяем соответствие Книга и Читатель внутри цикла
                    Если Запись.Книга = СтрокаТЧ.Книга И Запись.Читатель = ЭтотОбъект.Читатель Тогда

        Запись.Оплачено = Истина;
                    КонецЕсли;
                КонецЦикла;

                НаборЗаписей.Записать();

            КонецЕсли;

        КонецЦикла;

        Движения.РегистрБухУчет.Записывать = Истина;
        Движение = Движения.РегистрБухУчет.Добавить();
        Движение.Период = Дата;
        Движение.СчетДт = ПланыСчетов.БухгалтерскийУчет.НайтиПоКоду("01.3"); // Расчетный счет (дебет)
        Движение.СчетКт = ПланыСчетов.БухгалтерскийУчет.НайтиПоКоду("02.2"); // Выручка (кредит)
        Движение.Сумма = СуммаОплаты; // Цена билета из реквизита документа

КонецПроцедуры
```

Код формы документа «ОплатаШтрафов»

```
&НаКлиенте
Процедура ЧитательПриИзменении(Элемент)
    ЗаполнитьТабличнуюЧастьШтрафов();

КонецПроцедуры

&НаСервере
Процедура ЗаполнитьТабличнуюЧастьШтрафов()

    // Очищаем табличную часть перед заполнением
    Объект.Штрафы.Очистить();

    // Выходим, если читатель не выбран
    Если Объект.Читатель = Неопределено Тогда
        Возврат;
    КонецЕсли;

    ТекущаяДата = Дата(Год(ТекущаяДата()), Месяц(ТекущаяДата()), День(ТекущаяДата())); // Только дата, без времени


    Запрос = Новый Запрос;
    Запрос.Текст =
    "ВЫБРАТЬ
    |    СрокВозвратаКниг.Книга КАК Книга,
    |    СрокВозвратаКниг.ПланируемаяДатаВозврата КАК ПланируемаяДатаВозврата,
    |    СрокВозвратаКниг.Количество КАК Количество,
    |    ВложенныйЗапрос.СуммаШтрафа КАК СуммаШтрафа,
    |    ВложенныйЗапрос.Оплачено КАК Оплачено,
    |    ВложенныйЗапрос.Основание КАК Основание
    |ИЗ
    |    РегистрСведений.СрокВозвратаКниг КАК СрокВозвратаКниг
```

```
        |                ЛЕВОЕ СОЕДИНЕНИЕ
(ВЫБРАТЬ
        |
        ШтрафыЗаПросрочку.СуммаШтрафа КАК
СуммаШтрафа,
        |
        ШтрафыЗаПросрочку.Оплачено КАК
Оплачено,
        |
        ШтрафыЗаПросрочку.Основание КАК
Основание,
        |
        ШтрафыЗаПросрочку.Книга КАК Книга,
        |
        ШтрафыЗаПросрочку.Читатель КАК
Читатель
        |                ИЗ
        |
        РегистрСведений.ШтрафыЗаПросрочку
КАК ШтрафыЗаПросрочку
        |                ГДЕ
        |
        ШтрафыЗаПросрочку.Оплачено = ЛОЖЬ)
КАК ВложенныйЗапрос
        |                ПО
СрокВозвратаКниг.Книга =
ВложенныйЗапрос.Книга
        |                        И
СрокВозвратаКниг.Читатель =
ВложенныйЗапрос.Читатель
        |ГДЕ
        |        СрокВозвратаКниг.Читатель =
&Читатель
        |        И
СрокВозвратаКниг.ПланируемаяДатаВозврата <
&ТекущаяДата
        |        И
СрокВозвратаКниг.ФактическаяДатаВозврата =
ДАТАВРЕМЯ(1, 1, 1)
        |        И ВложенныйЗапрос.Оплачено =
ЛОЖЬ";

        Запрос.УстановитьПараметр("Читатель",
Объект.Читатель);
        Запрос.УстановитьПараметр("ТекущаяДата
", ТекущаяДата);

        Результат = Запрос.Выполнить();
        Выборка = Результат.Выбрать();

        Пока Выборка.Следующий() Цикл
                НоваяСтрока =
Объект.Штрафы.Добавить();
                НоваяСтрока.Книга =
Выборка.Книга;
                НоваяСтрока.Количество =
Выборка.Количество;
                НоваяСтрока.СуммаШтрафа =
?(Выборка.СуммаШтрафа = NULL, 0,
Выборка.СуммаШтрафа);
                НоваяСтрока.Оплачено =
Выборка.Оплачено;
                // Заполняем Основание из запроса
                НоваяСтрока.Основание =
Выборка.Основание;
                // Расчет суммы штрафа за книгу
                ЦенаЗаДеньПросрочки =
ОбщийМодульФункции.ПолучитьЦенуШтрафа(Пе
речисления.НазваниеШтрафа.Просрочка);
                ПросроченныеДни =
?(ТекущаяДата() >
Выборка.ПланируемаяДатаВозврата,
Цел((ТекущаяДата() -
Выборка.ПланируемаяДатаВозврата) / 86400), 0);
                НоваяСтрока.СуммаШтрафа =
ПросроченныеДни * ЦенаЗаДеньПросрочки *
Выборка.Количество;
        КонецЦикла;

КонецПроцедуры

&НаКлиенте
Процедура ШтрафыПриИзменении(Элемент)
```

```
	ПересчитатьСуммуОплаты();

КонецПроцедуры

&НаКлиенте
Процедура ПересчитатьСуммуОплаты()

	СуммаОплаты = 0;

	Для Каждого СтрокаШтрафа Из Объект.Штрафы Цикл
		Если СтрокаШтрафа.Оплачено Тогда
			СуммаОплаты = СуммаОплаты + СтрокаШтрафа.СуммаШтрафа;
		КонецЕсли;
	КонецЦикла;

	Объект.СуммаОплаты = СуммаОплаты;

КонецПроцедуры

&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи)

	ТекущийПользователь = ПользователиИнформационнойБазы.ТекущийПользователь();

	// Поиск сотрудника по логину только если поле "Ответственный" не заполнено
	Если Не ЗначениеЗаполнено(ТекущийОбъект.Ответственный) Тогда
		НайденныйСотрудник = Справочники.Сотрудники.НайтиПоРеквизиту("Логин", ТекущийПользователь.Имя);

		Если НайденныйСотрудник <> Неопределено Тогда
			ТекущийОбъект.Ответственный = НайденныйСотрудник;
		Иначе
			Сообщить("Сотрудник не найден по логину текущего пользователя. Укажите ответственного вручную.");
		КонецЕсли;
	КонецЕсли;

КонецПроцедуры
```

Код печати «ЧекОплаты»

```
&НаКлиенте
Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполненияКоманды)

	ТабДок = Новый ТабличныйДокумент;
	Печать (ТабДок, ПараметрКоманды);
	ТабДок.Показать("Чек оплаты штрафа");

КонецПроцедуры

&НаСервере
Процедура Печать(ТабДок, ССылкаНаДокумент)

	Макет = Документы.ОплатаШтрафов.ПолучитьМакет("ЧекОплатыШтрафа");
	ОбластьШапка = Макет.ПолучитьОбласть("Шапка");
	ОбластьШапкаТЧ = Макет.ПолучитьОбласть("ШапкаТЧ");
	ОбластьТЧ = Макет.ПолучитьОбласть("ТЧ");
	ОбластьПодвал = Макет.ПолучитьОбласть("Подвал");

	ОбластьШапка.Параметры.НомерДок = ССылкаНаДокумент.Номер;
	ОбластьШапка.Параметры.Дата = Формат(ССылкаНаДокумент.Дата,"ДЛФ=D");
```

```
ТабДок.Вывести(ОбластьШапка);


ТабДок.Вывести(ОбластьШапкаТЧ);


Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|
ОплатаШтрафовШтрафы.НомерСтроки
КАК НомерСтроки,
|	ОплатаШтрафовШтрафы.Книга
КАК Книга,
|
ОплатаШтрафовШтрафы.Количество КАК
Количество,
|
ОплатаШтрафовШтрафы.СуммаШтрафа
КАК Сумма
|ИЗ
|
Документ.ОплатаШтрафов.Штрафы КАК
ОплатаШтрафовШтрафы
|ГДЕ
|	ОплатаШтрафовШтрафы.Ссылка =
&ССылкаНаДокумент";
Запрос.УстановитьПараметр("ССылкаНаДо
кумент", ССылкаНаДокумент);

РезультатЗапроса = Запрос.Выполнить();

Выборка = РезультатЗапроса.Выбрать();
СуммаКоличство = 0;
СуммаДока = 0;
Пока Выборка.Следующий() Цикл

	ОбластьТЧ.Параметры.Номер =
Выборка.НомерСтроки;

ОбластьТЧ.Параметры.Наименование =
Выборка.Книга;
	ОбластьТЧ.Параметры.Количество
= Выборка.Количество;
	ОбластьТЧ.Параметры.Сумма =
Выборка.Сумма;
	СуммаДока = СуммаДока+
Выборка.Сумма;
	СуммаКоличство =
СуммаКоличство+ Выборка.Количество;
	ТабДок.Вывести(ОбластьТЧ);

КонецЦикла;

Ответственный =
ССылкаНаДокумент.Ответственный.Фамилия + " "
+ССылкаНаДокумент.Ответственный.Наименовани
е + " "
+ССылкаНаДокумент.Ответственный.Отчество;
ОбластьПодвал.Параметры.КолНаимен =
СуммаКоличство;
ОбластьПодвал.Параметры.Ответственный
= Ответственный;
ОбластьПодвал.Параметры.СуммаВсего =
СуммаДока;
ОбластьПодвал.Параметры.Итого =
СуммаДока;


ТабДок.Вывести(ОбластьПодвал);
КонецПроцедуры
```

Код модуля объекта «ВыкупУтерянной Книги»

```
&НаСервере
Процедура ОбработкаПроведения(Отказ, Режим)

// === Движения по регистрам ===
Движения.ИсторияВыдачКниг.Записывать
= Истина;
Движения.ОстаткиКниг.Записывать =
Истина; // Если нужно списывать с остатков при
выкупе
```

```
        Для Каждого ТекСтрокаКниги Из Книги
Цикл

            // === Списание книги с читателя
(всегда, так как только утеря) ===


        СписатьКнигуСЧитателя(ТекСтрокаКниги.
Книга, Читатель, ТекСтрокаКниги.Количество,
Отказ);

            // === Движения по регистру
бухгалтерии ===

        Движения.РегистрБухУчет.Записывать =
Истина;

            // Проводка по получению денег от
читателя
            ДвижениеБУ =
Движения.РегистрБухУчет.Добавить();
            ДвижениеБУ.Период = Дата;
            ДвижениеБУ.СчетДт =
ПланыСчетов.БухгалтерскийУчет.РасчетныйСчет;
            ДвижениеБУ.СчетКт =
ПланыСчетов.БухгалтерскийУчет.Выручка;

            ДвижениеБУ.Сумма =
ТекСтрокаКниги.Сумма;



            // Проводка по списанию
себестоимости книги
            ДвижениеБУ =
Движения.РегистрБухУчет.Добавить();
            ДвижениеБУ.Период = Дата;
            ДвижениеБУ.СчетДт =
ПланыСчетов.БухгалтерскийУчет.Собестоимость;
            ДвижениеБУ.СчетКт =
ПланыСчетов.БухгалтерскийУчет.Товары;

            ДвижениеБУ.Сумма =
ТекСтрокаКниги.Сумма;

            // Движение по регистру
взаиморасчетов

        Движения.Взаиморасчеты.Записывать =
Истина;
            ДвижениеВР =
Движения.Взаиморасчеты.Добавить();
            ДвижениеВР.ВидДвижения =
ВидДвиженияНакопления.Приход;
            ДвижениеВР.Период = Дата;
            ДвижениеВР.Читатель = Читатель;
            ДвижениеВР.Сумма =
ТекСтрокаКниги.Сумма;



        КонецЦикла;



        Движения.СредняяСобестоимостьТоваров.
Записывать = Истина;
        Движения.ПродажаКниг.Записывать =
Истина;

        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |
        СредняяСобестоимостьТоваровОстатки.Кн
ига КАК Книга,
        |
        СредняяСобестоимостьТоваровОстатки.Су
ммаОстаток КАК Сумма,
        |
        СредняяСобестоимостьТоваровОстатки.Ко
личествоОстаток КАК Количество
        |ИЗ
```

```
        |
        РегистрНакопления.СредняяСобестоимостьТоваров.Остатки(
        |
        &МоментВремени,
        |                                   Книга В
        |
        (ВЫБРАТЬ
        |
        ВыкупУтерянногоКнигиКниги.Книга КАК Книга
        |                                   ИЗ
        |
        Документ.ВыкупУтерянногоКниги.Книги КАК ВыкупУтерянногоКнигиКниги
        |                                   ГДЕ
        |
        ВыкупУтерянногоКнигиКниги.Ссылка = &Ссылка)) КАК СредняяСобестоимостьТоваровОстатки";

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

        РезультатЗапроса = Запрос.Выполнить();

        ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();

        СуммаСобестоимости = 0;

        Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

                Если ВыборкаДетальныеЗаписи.Количество <> 0 тогда

                        СобестоимостьЕдиницы = ВыборкаДетальныеЗаписи.Сумма/ВыборкаДетальныеЗаписи.Количество;

                Иначе

                СобестоимостьЕдиницы=0;
                КонецЕсли;

                Движение = Движения.СредняяСобестоимостьТоваров.Добавить();
                Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
                Движение.Период = Дата;

                СтрокаТЧ = Книги.Найти(ВыборкаДетальныеЗаписи.Книга,"Книга");
                Движение.Книга = ВыборкаДетальныеЗаписи.Книга;
                Движение.Количество = СтрокаТЧ.Количество;
                СебеСтоимостьСписание = СобестоимостьЕдиницы*СтрокаТЧ.Количество;
                Движение.Сумма = СебеСтоимостьСписание;
                //движения по регистру ПродажаКниг
                Движение = Движения.ПродажаКниг.Добавить();
                Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
                Движение.Период = Дата;
                Движение.Книга = ВыборкаДетальныеЗаписи.Книга;
                Движение.Читатель = Читатель;
                Движение.Количество = СтрокаТЧ.Количество;
                Движение.Сумма = СтрокаТЧ.Сумма;
                Движение.Собестоимость =СебеСтоимостьСписание;
```

```
        СуммаСобестоимости=СуммаСобестоимости+СебеСтоимостьСписание;



        КонецЦикла;



        // === Запись в регистр "СрокВозвратаКниг" ===
        Движения.СрокВозвратаКниг.Записывать = Истина;
        Для Каждого СтрокаТЧ Из Книги Цикл

            // Проверяем, была ли книга ранее выдана этому читателю
            Запрос = Новый Запрос;
            Запрос.Текст =
            "ВЫБРАТЬ ПЕРВЫЕ 1
            |
СрокВозвратаКниг.ДокументВыдачи, // Добавили нужные поля
            |
СрокВозвратаКниг.ПланируемаяДатаВозврата,
            |
СрокВозвратаКниг.ВремяДействия,
            |
        СрокВозвратаКниг.Количество // Добавляем количество, чтобы проверить, есть ли еще невозвращенные книги
            |ИЗ
            |
РегистрСведений.СрокВозвратаКниг КАК СрокВозвратаКниг
            |ГДЕ
            |   СрокВозвратаКниг.Книга = &Книга
            |   И СрокВозвратаКниг.Читатель = &Читатель
            |           И СрокВозвратаКниг.ФактическаяДатаВозврата = ДАТАВРЕМЯ(1, 1, 1)
            |УПОРЯДОЧИТЬ ПО
            |   СрокВозвратаКниг.Период УБЫВ";


        Запрос.УстановитьПараметр("Книга", СтрокаТЧ.Книга);

        Запрос.УстановитьПараметр("Читатель", Читатель);

            Результат = Запрос.Выполнить();

            Если Не Результат.Пустой() Тогда // Если книга была выдана
                Выборка = Результат.Выбрать();
                Выборка.Следующий();

                // Создаем движение на основании найденной записи
                Движение = Движения.СрокВозвратаКниг.Добавить();


        ЗаполнитьДвижениеРегистра(Движение, СтрокаТЧ, Читатель, Выборка);

            КонецЕсли;
        КонецЦикла;

КонецПроцедуры

// === Процедура для списания книги с читателя ===

&НаСервере
Процедура СписатьКнигуСЧитателя(Книга, Читатель, Количество, Отказ)
```

```
        // Поиск записи о выдаче данной книги этому читателю
        ЗаписьВыдачи = НайтиЗаписьВыдачи(Книга, Читатель);

        Если ЗаписьВыдачи = Неопределено Тогда
            Сообщить("Не найдена запись о выдаче книги " + Книга + " читателю " + Читатель);
            Отказ = Истина;
            Возврат; // Прекращаем выполнение процедуры, если запись не найдена
        КонецЕсли;

        // Проверка количества
        Если ЗаписьВыдачи.Количество < Количество Тогда
            Сообщить("Попытка списать больше книг, чем было выдано. Доступно для списания: " + ЗаписьВыдачи.Количество);
            Отказ = Истина;
            Возврат;
        КонецЕсли;

        // Создание движения по регистру "ИсторияВыдачКниг"
        Движение = Движения.ИсторияВыдачКниг.Добавить();
        Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
        Движение.Период = Дата;
        Движение.Книга = Книга;
        Движение.Читатель = Читатель;
        Движение.Количество = Количество;

КонецПроцедуры

// === Функция для получения остатка книг ===

Функция ПолучитьОстаток(Книга)

        Результат = 0;
        Отбор = Новый Структура;
        Отбор.Вставить("Книги", Книга);
        НайденныеЗначения = РегистрыНакопления.ДоступныеКниги.Остатки(Дата,Отбор);
        Если НайденныеЗначения.Количество() > 0 Тогда
            Результат = НайденныеЗначения[0].Количество;
        КонецЕсли;
        Возврат Результат;

КонецФункции

// === Функция для поиска записи о выдаче ===

Функция НайтиЗаписьВыдачи(Книга, Читатель)

        Отбор = Новый Структура("Книга, Читатель", Книга, Читатель);
        // Ищем запись с нужной книгой и читателем
        НайденныеЗначения = РегистрыНакопления.ИсторияВыдачКниг.Остатки(Дата, Отбор);

        Если НайденныеЗначения.Количество() > 0 Тогда
            Возврат НайденныеЗначения[0];
        Иначе
            Возврат Неопределено;
        КонецЕсли;

КонецФункции


Процедура ПередЗаписью(Отказ, РежимЗаписи, РежимПроведения)

        СуммаДокумента=Книги.Итог("Сумма");

КонецПроцедуры
```

```
&НаСервере
Процедура
ЗаполнитьДвижениеРегистра(Движение, СтрокаТЧ,
Читатель, Выборка)

	Движение.Период = Дата;
	Движение.Книга = СтрокаТЧ.Книга;
	Движение.Читатель = Читатель;
	Движение.ВремяДействия =
Выборка.ВремяДействия;
	Движение.ДокументВыдачи =
Выборка.ДокументВыдачи;
	Движение.Количество =
СтрокаТЧ.Количество;
	Если
ЗначениеЗаполнено(Выборка.ПланируемаяДатаВоз
врата) Тогда

	Движение.ПланируемаяДатаВозврата =
Выборка.ПланируемаяДатаВозврата;
	КонецЕсли;
	Движение.ФактическаяДатаВозврата =
Дата;

	// Устанавливаем признак выкупа
	Движение.Выкуплена = Истина;

КонецПроцедуры
```

Код формы документа «ВыкупУтерянной Книги»

```
&НаКлиенте
Процедура КнигиЦенаПриИзменении(Элемент)

	СтрокаТабличнойЧасти =
Элементы.Книги.ТекущиеДанные;
	СтрокаТабличнойЧасти.Сумма =
СтрокаТабличнойЧасти.Количество*СтрокаТаблич
нойЧасти.Цена;
КонецПроцедуры

&НаКлиенте
Процедура КнигиСуммаПриИзменении(Элемент)

	СтрокаТабличнойЧасти =
Элементы.Книги.ТекущиеДанные;
	Если СтрокаТабличнойЧасти.Количество =
0 тогда
		СтрокаТабличнойЧасти.Цена = 0;
	Иначе
		СтрокаТабличнойЧасти.Цена  =
СтрокаТабличнойЧасти.Сумма/
СтрокаТабличнойЧасти.Количество;
	КонецЕсли;


КонецПроцедуры

&НаКлиенте
Процедура
КнигиКоличествоПриИзменении(Элемент)

	СтрокаТабличнойЧасти =
Элементы.Книги.ТекущиеДанные;
	СтрокаТабличнойЧасти.Сумма =
СтрокаТабличнойЧасти.Количество*СтрокаТаблич
нойЧасти.Цена;

КонецПроцедуры

&НаКлиенте
Процедура КнигиКнигаПриИзменении(Элемент)

	СтрокаТЧ =
Элементы.Книги.ТекущиеДанные;
	СтрокаТЧ.Цена=ОбщийМодульФункции.Ц
еныИзРегистраСведений(Объект.Дата,СтрокаТЧ.К
нига);

КонецПроцедуры
```

```
&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи)
	ТекущийПользователь = ПользователиИнформационнойБазы.ТекущийПользователь();

	// Поиск сотрудника по логину только если поле "Ответственный" не заполнено
	Если Не ЗначениеЗаполнено(ТекущийОбъект.Ответственный) Тогда
		НайденныйСотрудник = Справочники.Сотрудники.НайтиПоРеквизиту("Логин", ТекущийПользователь.Имя);

		Если НайденныйСотрудник <> Неопределено Тогда

	ТекущийОбъект.Ответственный = НайденныйСотрудник;
		Иначе
			Сообщить("Сотрудник не найден по логину текущего пользователя. Укажите ответственного вручную.");
		КонецЕсли;
	КонецЕсли;

КонецПроцедуры
```

Код печати «ТоварныйЧек»

```
&НаКлиенте
Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполненияКоманды)

	ТабДок = Новый ТабличныйДокумент;
	Печать (ТабДок, ПараметрКоманды);
	ТабДок.Показать("Товарный чек");

КонецПроцедуры

&НаСервере
Процедура Печать(ТабДок, ССылкаНаДокумент)

	Макет = Документы.ВыкупУтерянногоКниги.ПолучитьМакет("ТоварныйЧек");
	ОбластьШапка = Макет.ПолучитьОбласть("Шапка");
	ОбластьШапкаТЧ = Макет.ПолучитьОбласть("ШапкаТЧ");
	ОбластьТЧ = Макет.ПолучитьОбласть("ТЧ");
	ОбластьПодвал = Макет.ПолучитьОбласть("Подвал");

	ОбластьШапка.Параметры.НомерДок = ССылкаНаДокумент.Номер;
	ОбластьШапка.Параметры.Дата = Формат(ССылкаНаДокумент.Дата,"ДЛФ=D");
	ТабДок.Вывести(ОбластьШапка);


	ТабДок.Вывести(ОбластьШапкаТЧ);


	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|
	ВыкупУтерянногоКнигиКниги.НомерСтроки КАК НомерСтроки,
	|
	ВыкупУтерянногоКнигиКниги.Книга КАК Книга,
	|
	ВыкупУтерянногоКнигиКниги.Количество КАК Количество,
	|
	ВыкупУтерянногоКнигиКниги.Цена КАК Цена,
	|
	ВыкупУтерянногоКнигиКниги.Сумма КАК Сумма
```

```
|ИЗ
|
Документ.ВыкупУтерянногоКниги.Книги
КАК ВыкупУтерянногоКнигиКниги
|ГДЕ
|
ВыкупУтерянногоКнигиКниги.Ссылка =
&ССылкаНаДокумент";
Запрос.УстановитьПараметр("ССылкаНаДо
кумент", ССылкаНаДокумент);

РезультатЗапроса = Запрос.Выполнить();

Выборка = РезультатЗапроса.Выбрать();
СуммаКоличство = 0;
СуммаДока = 0;
Пока Выборка.Следующий() Цикл

ОбластьТЧ.Параметры.Номер =
Выборка.НомерСтроки;

ОбластьТЧ.Параметры.Наименование =
Выборка.Книга;
ОбластьТЧ.Параметры.Количество
= Выборка.Количество;
ОбластьТЧ.Параметры.Цена =
Выборка.Цена;
ОбластьТЧ.Параметры.Сумма =
Выборка.Сумма;
СуммаДока = СуммаДока+
Выборка.Сумма;
СуммаКоличство =
СуммаКоличство+ Выборка.Количество;
ТабДок.Вывести(ОбластьТЧ);

КонецЦикла;

ОбластьПодвал.Параметры.КолНаимен =
СуммаКоличство;
Ответственный =
ССылкаНаДокумент.Ответственный.Фамилия + " "
+ССылкаНаДокумент.Ответственный.Наименовани
е + " "
+ССылкаНаДокумент.Ответственный.Отчество;
ОбластьПодвал.Параметры.Ответственный
= Ответственный;
ОбластьПодвал.Параметры.СуммаВсего =
СуммаДока;
ОбластьПодвал.Параметры.Итого =
СуммаДока;

ТабДок.Вывести(ОбластьПодвал);
КонецПроцедуры
```

Код модуля объекта «ЧитательскийБилет»

```
&НаСервере
Процедура ОбработкаПроведения(Отказ,
РежимПроведения)

//Движения.РегистрБухУчет.Записывать =
Истина;
//Движение =
Движения.РегистрБухУчет.Добавить();
//Движение.Период = Дата;
//Движение.СчетДт =
ПланыСчетов.БухгалтерскийУчет.НайтиПоКоду("0
1.3"); // Расчетный счет (дебет)
//Движение.СчетКт =
ПланыСчетов.БухгалтерскийУчет.НайтиПоКоду("0
2.2"); // Выручка (кредит)
//Движение.Сумма = Цена; // Цена билета
из реквизита документа

ДатаОкончания =
ДатаОкончанияСрокаБилета;

// Запись в регистр сведений
Запись =
РегистрыСведений.СрокиДействияЧитательскихБи
летов.СоздатьМенеджерЗаписи();
```

```
    Запись.ЧитательскийБилет = Ссылка;
    Запись.ДатаОкончания = ДатаОкончания;
    Запись.ТипДействия =
Перечисления.ТипДействия.Выдача;
    Запись.Период = ТекущаяДата();
    Запись.Записать();

КонецПроцедуры

&НаСервере
Процедура ОбработкаОтменыПроведения(Отказ)

        //// Сторнирование движений
        //Для Каждого Движение Из
Движения.РегистрБухУчет Цикл
        //      НовоеДвижение =
Движения.РегистрБухУчет.Добавить();
        //      НовоеДвижение.Период =
Движение.Период;
        //      НовоеДвижение.СчетДт =
Движение.СчетКт;
        //      НовоеДвижение.СчетКт =
Движение.СчетДт;
        //      НовоеДвижение.Сумма =
Движение.Сумма;
        //КонецЦикла;

КонецПроцедуры
```

Код формы документа «ЧитательскийБилет»

```
&НаКлиенте
Процедура АбонементПриИзменении(Элемент)

        РассчитатьИУстановитьЦену();
        РассчитатьДатуОкончания();

КонецПроцедуры

&НаКлиенте
Процедура
ВозрастнойРейтингПриИзменении(Элемент)

        РассчитатьИУстановитьЦену();

КонецПроцедуры

&НаКлиенте
Процедура РассчитатьИУстановитьЦену()

        Абонемент = Объект.Абонемент;
        ВозрастнойРейтинг =
Объект.ВозрастнойРейтинг;
        Цена =
ОбщийМодульФункции.ПолучитьЦенуБилета(Або
немент, ВозрастнойРейтинг);
        Объект.Цена = Цена;

КонецПроцедуры

&НаКлиенте
Процедура РассчитатьДатуОкончания()

        РассчитатьДатуОкончанияНаСервере();

КонецПроцедуры

&НаСервере
Процедура РассчитатьДатуОкончанияНаСервере()

        ВыбранныйАбонемент =
Объект.Абонемент;

        Если ВыбранныйАбонемент =
Перечисления.СрокДействияАбониментов.Месяц_
1 Тогда
                КоличествоМесяцев = 1;
        ИначеЕсли ВыбранныйАбонемент =
Перечисления.СрокДействияАбониментов.Месяц_
3 Тогда
                КоличествоМесяцев = 3;
        ИначеЕсли ВыбранныйАбонемент =
Перечисления.СрокДействияАбониментов.Месяц_
6 Тогда
                КоличествоМесяцев = 6;
```

```
        ИначеЕсли ВыбранныйАбонемент =
Перечисления.СрокДействияАбониментов.Месяц_
12 Тогда
                КоличествоМесяцев = 12;
        Иначе
                КоличествоМесяцев = 0;
        КонецЕсли;

        Если КоличествоМесяцев > 0 Тогда
                ДатаОкончания =
ДобавитьМесяц(Объект.Дата,
КоличествоМесяцев); // Дата окончания = дата
регистрации + количество месяцев

        Объект.ДатаОкончанияСрокаБилета =
ДатаОкончания;
        Иначе

        Объект.ДатаОкончанияСрокаБилета = Null;
        КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура
ДатаРегистрацииБилетаПриИзменении(Элемент)

        РассчитатьДатуОкончания();

КонецПроцедуры

&НаСервере
Процедура ПриЗаписиНаСервере(Отказ,
ТекущийОбъект, ПараметрыЗаписи)

        //ДатаОкончания =
Объект.ДатаОкончанияСрокаБилета;
        //// Запись в регистр сведений
        //Запись =
РегистрыСведений.СрокиДействияЧитательскихБи
летов.СоздатьМенеджерЗаписи();
        //Запись.ЧитательскийБилет =
Объект.Ссылка;
        //Запись.ДатаОкончания = ДатаОкончания;
        //Запись.ТипДействия =
Перечисления.ТипДействия.Выдача;
        //Запись.Период = ТекущаяДата();
        //Запись.Записать();


КонецПроцедуры

&НаКлиенте
Процедура ДатаПриИзменении(Элемент)

        РассчитатьДатуОкончания();

КонецПроцедуры

&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ,
ТекущийОбъект, ПараметрыЗаписи)

        ПроверитьДопустимостьВыдачиБилетаНа
Сервере(Отказ);
        ТекущийПользователь =
ПользователиИнформационнойБазы.ТекущийПоль
зователь();

        // Поиск сотрудника по логину только если
поле "Ответственный" не заполнено
        Если Не
ЗначениеЗаполнено(ТекущийОбъект.Ответственны
й) Тогда
                НайденныйСотрудник =
Справочники.Сотрудники.НайтиПоРеквизиту("Лог
ин", ТекущийПользователь.Имя);

                Если НайденныйСотрудник <>
Неопределено Тогда

        ТекущийОбъект.Ответственный =
НайденныйСотрудник;
                Иначе
```

```
                Сообщить("Сотрудник не найден по логину текущего пользователя. Укажите ответственного вручную.");
            КонецЕсли;
        КонецЕсли;

КонецПроцедуры

&НаСервереБезКонтекста
Функция СколькоЛетТебе(ТекДата, ДатаРождения)

        Если ДатаРождения = Неопределено Тогда
            Возврат 0;
        КонецЕсли;

        Возраст = Год(ТекДата) - Год(ДатаРождения);
        Если Месяц(ТекДата) < Месяц(ДатаРождения) Или (Месяц(ТекДата) = Месяц(ДатаРождения) И День(ТекДата) < День(ДатаРождения)) Тогда
            Возраст = Возраст - 1;
        КонецЕсли;

        Возврат Возраст;

КонецФункции


&НаСервере
Функция ПроверитьДопустимостьВыдачиБилета(Читатель, Абонемент, ВозрастнойРейтинг) Экспорт

        Если Читатель.ДатаРождения = Неопределено Тогда
            Сообщить("Не указана дата рождения читателя.");
            Возврат Ложь;
        КонецЕсли;

        ВозрастЧитателя = СколькоЛетТебе(ТекущаяДата(), Читатель.ДатаРождения);

        Если ВозрастнойРейтинг = Перечисления.ВозрастнойРейтинг.БезОграничений Тогда
            Возврат Истина;
        ИначеЕсли ВозрастнойРейтинг = Перечисления.ВозрастнойРейтинг.ДляДетейОт6лет И ВозрастЧитателя >= 6 Тогда
            Возврат Истина;
        ИначеЕсли ВозрастнойРейтинг = Перечисления.ВозрастнойРейтинг.ДляДетейОт12Лет И ВозрастЧитателя >= 12 Тогда
            Возврат Истина;
        ИначеЕсли ВозрастнойРейтинг = Перечисления.ВозрастнойРейтинг.ДляПодростковОт16Лет И ВозрастЧитателя >= 16 Тогда
            Возврат Истина;
        ИначеЕсли ВозрастнойРейтинг = Перечисления.ВозрастнойРейтинг.ДляВзрослых И ВозрастЧитателя >= 18 Тогда
            Возврат Истина;
        Иначе
            Возврат Ложь;
        КонецЕсли;

КонецФункции

&НаСервере
Процедура ПроверитьДопустимостьВыдачиБилетаНаСервере(Отказ)

        Если Не ПроверитьДопустимостьВыдачиБилета(Объект.Читатель, Объект.Абонемент, Объект.ВозрастнойРейтинг) Тогда
            Сообщить("Выдача билета невозможна. Возраст читателя не соответствует возрастному рейтингу абонемента.");
```

```
        Отказ = Истина;
    КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура ПередЗаписью(Отказ,
ПараметрыЗаписи)

    ПроверитьДопустимостьВыдачиБилетаНа
Сервере(Отказ);

КонецПроцедуры
```

Код модуля объекта документа «ПродлениеЧитательскогоБилета»

```
Процедура
ОбработкаЗаполнения(ДанныеЗаполнения,
СтандартнаяОбработка)

    Если ТипЗнч(ДанныеЗаполнения) =
Тип("ДокументСсылка.ЧитательскийБилет") Тогда
        // Заполнение шапки
        Абонемент = неопределено;
        ВозрастнойРейтинг =
ДанныеЗаполнения.ВозрастнойРейтинг;
        Основание =
ДанныеЗаполнения.Ссылка;
        Цена = ДанныеЗаполнения.Цена;
        Читатель =
ДанныеЗаполнения.Читатель;
    КонецЕсли;

КонецПроцедуры

&НаСервере
Процедура ОбработкаПроведения(Отказ,
РежимПроведения)

    //Движение =
Движения.РегистрБухУчет.Добавить();
    //Движение.Период = Дата;
    //Движение.СчетДт =
ПланыСчетов.БухгалтерскийУчет.НайтиПоКоду("0
1.3"); // Расчетный счет (дебет)
    //Движение.СчетКт =
ПланыСчетов.БухгалтерскийУчет.НайтиПоКоду("0
2.2"); // Выручка (кредит)
    //Движение.Сумма = Цена; // Цена
продления из реквизита документа

КонецПроцедуры

&НаСервере
Процедура ОбработкаОтменыПроведения(Отказ)

    // Сторнирование движений
    Для Каждого Движение Из
Движения.РегистрБухУчет Цикл
        НовоеДвижение =
Движения.РегистрБухУчет.Добавить();
        НовоеДвижение.Период =
Движение.Период;
        НовоеДвижение.СчетДт =
Движение.СчетКт; // Меняем местами счета
        НовоеДвижение.СчетКт =
Движение.СчетДт;
        НовоеДвижение.Сумма =
Движение.Сумма;
    КонецЦикла;

КонецПроцедуры
```

Код формы документа «ПродлениеЧитательскогоБилета»

```
&НаСервере
Процедура ПриЗаписиНаСервере(Отказ,
ТекущийОбъект, ПараметрыЗаписи)

    ДатаОкончания =
Объект.ДатаОкончанияСрокаБилета;
    // Запись в регистр сведений
```

```
        Запись =
РегистрыСведений.СрокиДействияЧитательскихБи
летов.СоздатьМенеджерЗаписи();
        Запись.ЧитательскийБилет =
Объект.Основание;
        Запись.ДатаОкончания = ДатаОкончания;
        Запись.ТипДействия =
Перечисления.ТипДействия.Продление;
        Запись.Период = ТекущаяДата();
        Запись.Записать();

КонецПроцедуры

&НаКлиенте
Процедура АбонементПриИзменении(Элемент)

        РассчитатьИУстановитьЦену();
        РассчитатьДатуОкончания();

КонецПроцедуры


&НаКлиенте
Процедура
ВозрастнойРейтингПриИзменении(Элемент)

        РассчитатьИУстановитьЦену();

КонецПроцедуры

&НаКлиенте
Процедура
ДатаРегистрацииБилетаПриИзменении(Элемент)

        РассчитатьДатуОкончания();

КонецПроцедуры

&НаКлиенте
Процедура РассчитатьИУстановитьЦену()

        Абонемент = Объект.Абонемент;
        ВозрастнойРейтинг =
Объект.ВозрастнойРейтинг;
        Цена =
ОбщийМодульФункции.ПолучитьЦенуБилета(Або
немент, ВозрастнойРейтинг);
        Объект.Цена = Цена;

КонецПроцедуры

&НаКлиенте
Процедура РассчитатьДатуОкончания()

        РассчитатьДатуОкончанияНаСервере();

КонецПроцедуры

&НаСервере
Процедура РассчитатьДатуОкончанияНаСервере()

        ВыбранныйАбонемент =
Объект.Абонемент;

        Если ВыбранныйАбонемент =
Перечисления.СрокДействияАбонементов.Месяц_
1 Тогда
                КоличествоМесяцев = 1;
        ИначеЕсли ВыбранныйАбонемент =
Перечисления.СрокДействияАбонементов.Месяц_
3 Тогда
                КоличествоМесяцев = 3;
        ИначеЕсли ВыбранныйАбонемент =
Перечисления.СрокДействияАбонементов.Месяц_
6 Тогда
                КоличествоМесяцев = 6;
        ИначеЕсли ВыбранныйАбонемент =
Перечисления.СрокДействияАбонементов.Месяц_
12 Тогда
                КоличествоМесяцев = 12;
        Иначе
                КоличествоМесяцев = 0;
        КонецЕсли;
```

```
        Если КоличествоМесяцев > 0 Тогда
            ДатаОкончания =
ДобавитьМесяц(Объект.Дата,
КоличествоМесяцев); // Дата окончания = дата
регистрации + количество месяцев

        Объект.ДатаОкончанияСрокаБилета =
ДатаОкончания;
        Иначе

        Объект.ДатаОкончанияСрокаБилета = Null;
        КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура ДатаПриИзменении(Элемент)

        РассчитатьДатуОкончания();

КонецПроцедуры

&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ,
ТекущийОбъект, ПараметрыЗаписи)

        ПроверитьДопустимостьВыдачиБилетаНа
Сервере(Отказ);

        Если ТекущийОбъект.Основание <>
Неопределено Тогда
            ЧитательИзОснования =
ТекущийОбъект.Основание.ПолучитьОбъект().Чит
атель;
            Если ТекущийОбъект.Читатель <>
ЧитательИзОснования Тогда
                Сообщить("Нельзя менять
читателя при продлении билета!");
                Отказ = Истина; //
Запрещаем запись документа
            КонецЕсли;
        КонецЕсли;

        ТекущийПользователь =
ПользователиИнформационнойБазы.ТекущийПоль
зователь();

        // Поиск сотрудника по логину только если
поле "Ответственный" не заполнено
        Если Не
ЗначениеЗаполнено(ТекущийОбъект.Ответственны
й) Тогда
            НайденныйСотрудник =
Справочники.Сотрудники.НайтиПоРеквизиту("Лог
ин", ТекущийПользователь.Имя);

            Если НайденныйСотрудник <>
Неопределено Тогда

        ТекущийОбъект.Ответственный =
НайденныйСотрудник;
            Иначе
                Сообщить("Сотрудник не
найден по логину текущего пользователя. Укажите
ответственного вручную.");
            КонецЕсли;
        КонецЕсли;

КонецПроцедуры

&НаСервереБезКонтекста
Функция СколькоЛетТебе(ТекДата, ДатаРождения)

        Если ДатаРождения = Неопределено Тогда
            Возврат 0;
        КонецЕсли;

        Возраст = Год(ТекДата) -
Год(ДатаРождения);
        Если Месяц(ТекДата) <
Месяц(ДатаРождения) Или (Месяц(ТекДата) =
Месяц(ДатаРождения) И День(ТекДата) <
День(ДатаРождения)) Тогда
            Возраст = Возраст - 1;
        КонецЕсли;
```

```
    Возврат Возраст;

КонецФункции

&НаСервере
Функция
ПроверитьДопустимостьВыдачиБилета(Читатель,
Абонемент, ВозрастнойРейтинг) Экспорт

    Если Читатель.ДатаРождения =
Неопределено Тогда
        Сообщить("Не указана дата
рождения читателя.");
        Возврат Ложь;
    КонецЕсли;

    ВозрастЧитателя =
СколькоЛетТебе(ТекущаяДата(),
Читатель.ДатаРождения);

    Если ВозрастнойРейтинг =
Перечисления.ВозрастнойРейтинг.БезОграничений
Тогда
        Возврат Истина;
    ИначеЕсли ВозрастнойРейтинг =
Перечисления.ВозрастнойРейтинг.ДляДетейОт6лет
И ВозрастЧитателя >= 6 Тогда
        Возврат Истина;
    ИначеЕсли ВозрастнойРейтинг =
Перечисления.ВозрастнойРейтинг.ДляДетейОт12Л
ет И ВозрастЧитателя >= 12 Тогда
        Возврат Истина;
    ИначеЕсли ВозрастнойРейтинг =
Перечисления.ВозрастнойРейтинг.ДляПодростков
От16Лет И ВозрастЧитателя >= 16 Тогда
        Возврат Истина;
    ИначеЕсли ВозрастнойРейтинг =
Перечисления.ВозрастнойРейтинг.ДляВзрослых И
ВозрастЧитателя >= 18 Тогда
        Возврат Истина;
    Иначе
        Возврат Ложь;
    КонецЕсли;

КонецФункции

&НаСервере
Процедура
ПроверитьДопустимостьВыдачиБилетаНаСервере(
Отказ)

    Если Не
ПроверитьДопустимостьВыдачиБилета(Объект.Чит
атель, Объект.Абонемент,
Объект.ВозрастнойРейтинг) Тогда
        Сообщить("Выдача билета
невозможна. Возраст читателя не соответствует
возрастному рейтингу абонемента.");
        Отказ = Истина;
    КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура ПередЗаписью(Отказ,
ПараметрыЗаписи)

    ПроверитьДопустимостьВыдачиБилетаНа
Сервере(Отказ);

КонецПроцедуры

&НаСервере
Процедура ОснованиеПриИзмененииНаСервере()

    Если Объект.Основание = Неопределено
Тогда
        // Очищаем поля, если основание
не выбрано
        Объект.Абонемент =
Неопределено;
        Объект.ВозрастнойРейтинг =
Неопределено;
```

```
            Объект.Цена = 0;

        Объект.ДатаОкончанияСрокаБилета =
Неопределено;
            Объект.Читатель = Неопределено;
            Возврат;
        КонецЕсли;

        ЧитательскийБилет =
Объект.Основание.ПолучитьОбъект();

        Объект.Читатель =
ЧитательскийБилет.Читатель;
        Объект.Абонемент =
ЧитательскийБилет.Абонемент;
        Объект.ВозрастнойРейтинг =
ЧитательскийБилет.ВозрастнойРейтинг;
        Объект.Цена = ЧитательскийБилет.Цена;
        Объект.ДатаОкончанияСрокаБилета =
ЧитательскийБилет.ДатаОкончанияСрокаБилета;

        // Проверка читателя
        Если ЧитательскийБилет.Читатель <>
Объект.Читатель Тогда
            Сообщить("Читатель не
соответствует документу-основанию!");
            Объект.Читатель =
ЧитательскийБилет.Читатель;
        КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура ОснованиеПриИзменении(Элемент)

        ОснованиеПриИзмененииНаСервере();

КонецПроцедуры
```

Код формы обработки «ЗагрузитьАвторов»

```
&НаКлиенте
Процедура ПутьКФайлуНачалоВыбора(Элемент,
ДанныеВыбора, ВыборДобавлением,
СтандартнаяОбработка)

        Проводник = Новый
ДиалогВыбораФайла(РежимДиалогаВыбораФайла.
Открытие);
        Проводник.Заголовок = "Выберите файл";
        Если Объект.ФорматФайла = "XLSX"
Тогда
            Фильтр = "СписокАвторов
(*.xlsx)|*.xlsx";
        ИначеЕсли Объект.ФорматФайла = "TXT"
тогда
            Фильтр = "СписокАвторов
(*.txt)|*.txt";

        Иначе
            Возврат;
        КонецЕсли;
        Проводник.Фильтр = Фильтр;

        Оповещение = Новый
ОписаниеОповещения("ПослеВыбораФайла",
ЭтотОбъект);
        Проводник.Показать(Оповещение);


КонецПроцедуры

&НаКлиенте
Процедура
ПослеВыбораФайла(ВыбранныеФайлы,Дополнител
ьныеПараметры) Экспорт

        Если ВыбранныеФайлы = Неопределено
Тогда
            Возврат;
        КонецЕсли;
        Объект.ПутьКФайлу =
ВыбранныеФайлы[0];
```

```
КонецПроцедуры

&НаКлиенте
Процедура ПрочитатьФайл(Команда)

	Объект.ДанныеФайла.Очистить();

	Если Объект.ФорматФайла = "XLSX" Тогда
		ПрочитатьФайл_XLSX();
	ИначеЕсли Объект.ФорматФайла = "TXT" Тогда
		ПрочитатьФайл_TXT();
	КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура ПрочитатьФайл_TXT()

	ПоследовательноеЧтение = Истина;
	Если ПоследовательноеЧтение ТОгда
		Текст = Новый ЧтениеТекста;

	Текст.Открыть(Объект.ПутьКФайлу, КодировкаТекста.UTF8);

	ТекСтрока=Текст.ПрочитатьСтроку();
		Пока ТекСтрока <> Неопределено Цикл
			МассивСлов = СтрРазделить(ТекСтрока,";");
			Если МассивСлов.Количество()<5 Тогда
				Продолжить;
			КонецЕсли;

	НоваяСТрока=Объект.ДанныеФайла.Добавить();
			НоваяСтрока.Имя= МассивСлов[0];
			НоваяСтрока.Фамилия=МассивСлов[1];

	НоваяСтрока.ГодРождения=МассивСлов[2];

	НоваяСтрока.Национальность=МассивСлов[3];

	НоваяСтрока.ССылкаНаСайтАвтора=МассивСлов[4];

	ТекСтрока=Текст.ПрочитатьСтроку();
		КонецЦикла;
	Иначе
		Текст = Новый ТекстовыйДокумент;

	Текст.Прочитать(Объект.ПутьКФайлу);
		Для НомерСтроки=1 По Текст.КоличествоСтрок() Цикл
			ТекСтрока = Текст.ПолучитьСтроку(НомерСтроки);

	МассивСлов=СтрРазделить(ТекСтрока, ";");
			Если МассивСлов.Количество()< 5 Тогда
				Продолжить;
			КонецЕсли;

	НоваяСтрока=Объект.ДанныеФайла.Добавить();
			НоваяСтрока.Имя= МассивСлов[0];

	НоваяСтрока.Фамилия=МассивСлов[1];

	НоваяСтрока.ГодРождения=МассивСлов[2];
```

```
		НоваяСтрока.Национальность=МассивСлов[3];

		НоваяСтрока.ССылкаНаСайтАвтора=МассивСлов[4];
			КонецЦикла;
		КонецЕсли;

КонецПроцедуры

&НаСервере
Процедура ПрочитатьФайл_XLSX_НаСервере(АдресДанных)

		ТабДок = Новый ТабличныйДокумент;

		Данные = ПолучитьИзВременногоХранилища(АдресДанных);
		ПутьКФайлуНаСервере = ПолучитьИмяВременногоФайла("xlsx");
		Данные.Записать(ПутьКФайлуНаСервере);


		Попытка

		ТабДок.Прочитать(ПутьКФайлуНаСервере, СпособЧтенияЗначенийТабличногоДокумента.Значение);
		Исключение
			Сообщение = Новый СообщениеПользователю;
			Сообщение.Текст = "Не удалось прочитать указанный файл по причине: " + ОписаниеОшибки();
			Сообщение.Сообщить();
			Возврат;
		КонецПопытки;

		КоличествоСтрок = ТабДок.ВысотаТаблицы;
		Для НомерСТроки = 2 По КоличествоСтрок Цикл
			СтрокаДанных = Объект.ДанныеФайла.Добавить();
			СтрокаДанных.Имя = ТабДок.ПолучитьОбласть("R"+Формат(НомерСтроки,"ЧГ=0")+"C"+1).ТекущаяОбласть.Текст;
			СтрокаДанных.Фамилия = ТабДок.ПолучитьОбласть("R"+Формат(НомерСтроки,"ЧГ=0")+"C"+2).ТекущаяОбласть.Текст;
			СтрокаДанных.ГодРождения = ТабДок.ПолучитьОбласть("R"+Формат(НомерСтроки,"ЧГ=0")+"C"+3).ТекущаяОбласть.Текст;
			СтрокаДанных.Национальность = ТабДок.ПолучитьОбласть("R"+Формат(НомерСтроки,"ЧГ=0")+"C"+4).ТекущаяОбласть.Текст;

			СтрокаДанных.ССылкаНаСайтАвтора = ТабДок.ПолучитьОбласть("R"+Формат(НомерСтроки,"ЧГ=0")+"C"+5).ТекущаяОбласть.Текст;
		КонецЦикла;
КонецПроцедуры

&НаКлиенте
Процедура ПрочитатьФайл_XLSX()

		ДанныеФайла = Новый ДвоичныеДанные(Объект.ПутьКФайлу);
		АдресДанных = ПоместитьВоВременноеХранилище(ДанныеФайла);
		ПрочитатьФайл_XLSX_НаСервере(АдресДанных);

КонецПроцедуры

&НаСервере
Процедура ЗаписатьДанныеНаСервере()
```

```
        Для каждого СтрокаДанных из
Объект.ДанныеФайла Цикл
                НайденныйАвтор =
Справочники.Авторы.НайтиПоРеквизиту("Фамили
яАвтора", СтрокаДанных.Фамилия);
                Если НайденныйАвтор.Пустая()
Тогда
                        НайденныйАвтор =
Справочники.Авторы.СоздатьЭлемент();

        НайденныйАвтор.Наименование =
СтрокаДанных.Имя;

        НайденныйАвтор.ФамилияАвтора =
СтрокаДанных.Фамилия;
                        День =
Число(Лев(СтрокаДанных.ГодРождения, 2));
                        Месяц =
Число(Сред(СтрокаДанных.ГодРождения, 4, 2));
                        Год =
Число(Прав(СтрокаДанных.ГодРождения, 4));
                        ГодРожденияДата =
Дата(Год, Месяц, День);



        НайденныйАвтор.ГодРождения =
ГодРожденияДата;

        НайденныйАвтор.Национальность =
СтрокаДанных.Национальность;

        НайденныйАвтор.ССылкаНаСайтАвтора =
СтрокаДанных.ССылкаНаСайтАвтора;

        НайденныйАвтор.Записать();
                КонецЕсли;



        КонецЦикла;
        Сообщить("Добавление прошло успешно")

КонецПроцедуры
```

```
&НаКлиенте
Процедура ЗаписатьДанные(Команда)

        ЗаписатьДанныеНаСервере();

КонецПроцедуры
```

Код формы обработки «УстановкаЦенБилетов»

```
&НаКлиенте
Процедура ЗаполнитьТаблицу(Команда)

   ЗаполнитьТаблицуНаСервере();

КонецПроцедуры

&НаСервере
Процедура ЗаполнитьТаблицуНаСервере()

   Цены.Очистить();

   Для Каждого Абонемент Из
Перечисления.СрокДействияАбониментов Цикл
      Для Каждого ВозрастнойРейтинг Из
Перечисления.ВозрастнойРейтинг Цикл
        НоваяСтрока = Цены.Добавить();
        НоваяСтрока.Абонемент = Абонемент;
        НоваяСтрока.ВозрастнойРейтинг =
ВозрастнойРейтинг;

        // Попробуем получить текущую цену из
регистра сведений
        Запрос = Новый Запрос;
        Запрос.Текст =
          "ВЫБРАТЬ ПЕРВЫЕ 1
          |   ЦеныЧитательскихБилетов.Цена
          |ИЗ
          |
РегистрСведений.ЦеныЧитательскихБилетов КАК
ЦеныЧитательскихБилетов
          |ГДЕ
```

```
        |    ЦеныЧитательскихБилетов.Абонемент
= &Абонемент
        |    И
ЦеныЧитательскихБилетов.ВозрастнойРейтинг =
&ВозрастнойРейтинг
        |УПОРЯДОЧИТЬ ПО
        |    ЦеныЧитательскихБилетов.Период
УБЫВ";

    Запрос.УстановитьПараметр("Абонемент",
Абонемент);

Запрос.УстановитьПараметр("ВозрастнойРейтинг",
ВозрастнойРейтинг);

    Результат = Запрос.Выполнить();
    Выборка = Результат.Выбрать();

                    Если
Выборка.Следующий() Тогда
                        НоваяСтрока.Цена
= Выборка.Цена;
    Иначе
        НоваяСтрока.Цена = 0;
                    КонецЕсли;
КонецЦикла;
  КонецЦикла;

КонецПроцедуры

&НаКлиенте
Процедура ЗаписатьЦены(Команда)

  ЗаписатьЦеныНаСервере();

КонецПроцедуры

&НаСервере
Процедура ЗаписатьЦеныНаСервере()

    // Получаем текущую дату сервера
    ТекущаяДата = ТекущаяДата();

  Для Каждого Строка Из Цены Цикл
    Запись =
РегистрыСведений.ЦеныЧитательскихБилетов.Соз
датьМенеджерЗаписи();
    Запись.Абонемент = Строка.Абонемент;
    Запись.ВозрастнойРейтинг =
Строка.ВозрастнойРейтинг;
    Запись.Цена = Строка.Цена;

            // Устанавливаем период записи
            Запись.Период = ТекущаяДата;

    Запись.Записать();
  КонецЦикла;

  Сообщить("Цены успешно записаны.");

КонецПроцедуры
```

Код формы обработки «Виджеты»

```
&НаКлиенте
Перем ВременноеХранилищеФотографии;

&НаСервере
Процедура
ЗаполнитьТаблицуНаСервере(ВыбранныйМесяц)

    Если НЕ
ЗначениеЗаполнено(ВыбранныйМесяц) Тогда
        ВыбранныйМесяц =
ТекущаяДатаСеанса();
    КонецЕсли;
    НачалоПериода =
НачалоМесяца(ВыбранныйМесяц);
    КонецПериода =
КонецМесяца(ВыбранныйМесяц);
    // --- КОНЕЦ ИЗМЕНЕНИЙ ---

    Запрос = Новый Запрос;
    Запрос.Текст =
    "ВЫБРАТЬ ПЕРВЫЕ 5
```

```
        |   ПопулярностьКнигиОбороты.Книга,
        |   ПопулярностьКнигиОбороты.Период
КАК ПериодОборота, // Переименовал, чтобы не
путать с периодом запроса
        |
СУММА(ПопулярностьКнигиОбороты.Количество
Оборот) КАК КоличествоОборот
        |ИЗ
        |
РегистрНакопления.ПопулярностьКниги.Обороты(
&НачалоПериода, &КонецПериода, ДЕНЬ) КАК
ПопулярностьКнигиОбороты
        |ГДЕ
        |   ПопулярностьКнигиОбороты.Книга <>
ЗНАЧЕНИЕ(Справочник.Книги.ПустаяСсылка)
        |
        |СГРУППИРОВАТЬ ПО
        |   ПопулярностьКнигиОбороты.Книга,
        |   ПопулярностьКнигиОбороты.Период
        |
        |УПОРЯДОЧИТЬ ПО
        |
СУММА(ПопулярностьКнигиОбороты.Количество
Оборот) УБЫВ";

        Запрос.УстановитьПараметр("НачалоПери
ода", НачалоПериода);
        Запрос.УстановитьПараметр("КонецПерио
да", КонецПериода);
        Результат = Запрос.Выполнить();
        Выборка = Результат.Выбрать();
        Объект.Книги.Очистить();

        Пока Выборка.Следующий() Цикл
                НоваяСтрока =
Объект.Книги.Добавить();
                НоваяСтрока.Книга =
Выборка.Книга;
                НоваяСтрока.Период =
Выборка.ПериодОборота;
                НоваяСтрока.Количество =
Выборка.КоличествоОборот;
        КонецЦикла;

        ЗапросЗаказы = Новый Запрос;
        ЗапросЗаказы.Текст =
        "ВЫБРАТЬ ПЕРВЫЕ 10
        |       ЗакупкаКниг.Номер КАК Номер,
        |       ЗакупкаКниг.Дата КАК Дата,
        |       ЗакупкаКниг.Поставщик КАК
Поставщик,
        |       ЗакупкаКниг.Издатель КАК
Издатель,
        |       ЗакупкаКниг.СуммаДокумента
КАК СуммаДокумента,
        |       ЗакупкаКниг.Статус КАК Статус
        |ИЗ
        |       Документ.ЗакупкаКниг КАК
ЗакупкаКниг
        |ГДЕ
        |       ЗакупкаКниг.Статус <>
ЗНАЧЕНИЕ(Перечисление.СтатусЗакупки.Получен
)
        |
        |УПОРЯДОЧИТЬ ПО
        |       ЗакупкаКниг.Дата УБЫВ";

        РезультатЗаказы =
ЗапросЗаказы.Выполнить();
        ВыборкаЗаказы =
РезультатЗаказы.Выбрать();
        Объект.ОжидаемыеЗаказы.Очистить();

        Пока ВыборкаЗаказы.Следующий() Цикл
                НоваяСтрока =
Объект.ОжидаемыеЗаказы.Добавить();
                НоваяСтрока.Номер =
ВыборкаЗаказы.Номер;
                НоваяСтрока.Дата =
ВыборкаЗаказы.Дата;
                НоваяСтрока.Поставщик =
ВыборкаЗаказы.Поставщик;
                НоваяСтрока.Издатель =
ВыборкаЗаказы.Издатель;
```

```
            НоваяСтрока.СуммаДокумента =
ВыборкаЗаказы.СуммаДокумента;
            НоваяСтрока.Статус =
ВыборкаЗаказы.Статус;
        КонецЦикла;

КонецПроцедуры

&НаКлиенте
Процедура ЗаполнитьДиаграмму()
        ДиаграммаПопулярности.Очистить();

        Для Каждого СтрокаТаблицы Из
Объект.Книги Цикл
            Серия =
ДиаграммаПопулярности.УстановитьСерию(Строк
аТаблицы.Книга);
            Точка =
ДиаграммаПопулярности.УстановитьТочку(Строка
Таблицы.Период);
            Точка.Текст =
Формат(СтрокаТаблицы.Период,
"ДФ=dd.MM.yyyy");

        ДиаграммаПопулярности.УстановитьЗначе
ние(Точка, Серия, СтрокаТаблицы.Количество);
        КонецЦикла;

КонецПроцедуры

&НаКлиенте
Процедура ПриОткрытии(Отказ)
        ЭтотОбъект.Месяц= ТекущаяДата();
        ЗаполнитьТаблицуНаСервере(ЭтотОбъект.
Месяц);
        ЗаполнитьДиаграмму();
        ЗагрузкаПользователя();
        ДанныеУчереждения();
        ЭтотОбъект.Дата = ТекущаяДата();
КонецПроцедуры

&НаСервере
Процедура ЗагрузкаПользователя()

        ТекущийПользователь =
ПользователиИнформационнойБазы.ТекущийПоль
зователь();

        // Поиск сотрудника по логину
        Сотрудник =
Справочники.Сотрудники.НайтиПоРеквизиту("Лог
ин", ТекущийПользователь.Имя);

        Если Сотрудник <> Неопределено Тогда
            // Приветствие
            Элементы.Приветствие.Заголовок
= "Добро пожаловать в систему, " +
Сотрудник.Наименование + "!"; // Или другой текст
приветствия

            // ФИО и должность
            ФИО = Сотрудник.Фамилия + " " +
Сотрудник.Наименование + " " +
Сотрудник.Отчество;
            Должность =
Сотрудник.Должность;

        Элементы.ФИОДолжность.Заголовок =
ФИО + ", " + Должность;

            // Фотография
            Если
ЗначениеЗаполнено(Сотрудник.Фотография) Тогда

        ВременноеХранилищеФотографии =
Сотрудник.Фотография.Получить();  // Получаем
картинку из хранилища
                ФотографияСотрудника =
ПолучитьНавигационнуюССылку(Сотрудник.Ссыл
ка, "Фотография");
            КонецЕсли;
        Иначе
```

```
            Элементы.Приветствие.Заголовок
= "Добро пожаловать в систему!"; //  Приветствие
для пользователей без привязки к сотруднику
        КонецЕсли;

КонецПроцедуры

&НаСервере
Процедура ДанныеУчереждения()

        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ ПЕРВЫЕ 1
        |       Библиотека.ПолноеНаименование
КАК ПолноеНаименование,
        |       Библиотека.Адрес КАК Адрес,
        |       Библиотека.Телефон КАК
Телефон,
        |       Библиотека.Почта КАК Почта,
        |       Библиотека.Сайт КАК Сайт,
        |       Библиотека.ИНН КАК ИНН,
        |       Библиотека.КПП КАК КПП,
        |        Библиотека.Директор.Фамилия
КАК ДиректорФамилия,
        |   Библиотека.Директор.Наименование
КАК ДиректорИмя,
        |   Библиотека.Директор.Отчество КАК
ДиректорОтчество
        |ИЗ
        |       Справочник.Библиотека КАК
Библиотека
        |АВТОУПОРЯДОЧИВАНИЕ";
        Результат = Запрос.Выполнить();
        Выборка = Результат.Выбрать();

        Если Выборка.Следующий() Тогда
            Элементы.ПолноеНаименование.Заголовок
= Выборка.ПолноеНаименование;
            Элементы.Адрес.Заголовок =
"Адрес: " + Выборка.Адрес;
            Элементы.Телефон.Заголовок =
"Телефон: " + Выборка.Телефон;
            Элементы.Почта.Заголовок =
"Почта: " + Выборка.Почта;
            Элементы.Сайт.Заголовок = "Сайт:
" + Выборка.Сайт;
            ФИОДиректора =
Выборка.ДиректорФамилия + " " +
Выборка.ДиректорИмя + " " +
Выборка.ДиректорОтчество;
            Элементы.Директор.Заголовок =
"Руководитель учереждерения: " + ФИОДиректора;
        КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура МесяцПриИзменении(Элемент)
        ЗаполнитьТаблицуНаСервере(ЭтотОбъект.
Месяц);
        ЗаполнитьДиаграмму();
КонецПроцедуры

&НаКлиенте
Процедура Обновить(Команда)
        ЗаполнитьТаблицуНаСервере(ЭтотОбъект.
Месяц);
        ЗаполнитьДиаграмму();
КонецПроцедуры
```