Титульный лист

Задание лист 1

Задание лист 2

**Реферат**

Пояснительная записка: 186 c., 33 табл., 12 рис., 19 источников, 1 приложение.

НАСТОЛЬНОЕ ПРИЛОЖЕНИЕ, КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА, СЕРВЕР БАЗ ДАННЫХ, ДИЛЕРСКАЯ ДЕЯТЕЛЬНОСТЬ, АВТОМАТИЗИРОВАННЫЙ УЧЕТ

Цель проекта — разработка программного модуля управления дилерской деятельностью для существующей системы учета операций с ценными бумагами.

Средства разработки — среда программирования Microsoft Visual Studio 2008, язык программирования C#, СУБД Microsoft SQL Server 2008 R2.

Назначение разработанного модуля — автоматизация процесса учета сделок с ценными бумагами, заключаемых банком в собственных интересах.

Область применения — дилерская деятельность банка, бухгалтерский учет.

**Summary**

Explanatory memorandum: 186 pages, 33 tables, 12 figures, 19 sources, 1 appendix.

DESKTOP APPLICATION, CLIENT-SERVER ARCHITECTURE, DATABASE SERVER, DEALER ACTIVITY, AUTOMATED ACCOUNTING

Project aim: development of program module of dealer activity control which is to be integrated into existing securities operations back-office system.

Development instruments: integrated development environment Microsoft Visual Studio 2008, programming language C#, database management system Microsoft SQL Server 2008 R2.

Purpose of developed module: automation of accounting process of securities deals which are traded in bank’s own interest.

Use sphere: dealer activity of the bank and accounting.

**Содержание**

[Введение 10](#_Toc388912128)

[1 Постановка задачи 14](#_Toc388912129)

[2 Технико-экономическое обоснование темы проекта 16](#_Toc388912130)

[2.1 Обоснование актуальности темы проекта и необходимости разработки 16](#_Toc388912131)

[2.2 Обоснование выбора используемого программного обеспечения 20](#_Toc388912132)

[2.2.1 Семейство операционных систем Windows 20](#_Toc388912133)

[2.2.2 Платформа Microsoft .NET Framework 3.5 22](#_Toc388912134)

[2.2.3 Система управления базами данных Microsoft SQL Server 2008 R2 23](#_Toc388912135)

[2.2.4 Интегрированная среда разработки Microsoft Visual Studio 2008 24](#_Toc388912136)

[2.2.5 Язык программирования C# 25](#_Toc388912137)

[2.2.6 Набор компонентов пользовательского интерфейса DevExpress 9.2 26](#_Toc388912138)

[3 Проектирование программного модуля 27](#_Toc388912139)

[3.1 Архитектура системы 27](#_Toc388912140)

[3.1.1 Ядро системы 28](#_Toc388912141)

[3.1.2 Сервисы 30](#_Toc388912142)

[3.2 Анализ прикладной области и разработка требований 32](#_Toc388912143)

[3.2.1 Используемые термины и сокращения 33](#_Toc388912144)

[3.2.2 Описание требуемого бизнес-процесса 37](#_Toc388912145)

[3.2.3 Требования к сделкам 38](#_Toc388912146)

[3.2.4 Требования к учету сделок по методу FIFO 40](#_Toc388912147)

[3.2.5 Требования к применению торговли 43](#_Toc388912148)

[3.2.6 Требования к внутреннему учету 44](#_Toc388912149)

[3.2.7 Требования к закрытию бизнес-дня 45](#_Toc388912150)

[3.2.8 Требования к корпоративным событиям 45](#_Toc388912151)

[3.2.9 Требования к выгрузке в учетную систему GAAP 47](#_Toc388912152)

[3.2.10 Требования к выгрузке в учетную систему RAS 47](#_Toc388912153)

[3.2.11 Алгоритм GAAP FIFO 48](#_Toc388912154)

[3.2.12 Алгоритм генерации событий 54](#_Toc388912155)

[3.3 Структура программного модуля 57](#_Toc388912156)

[3.4 Проектирование структуры базы данных 59](#_Toc388912157)

[4 Разработка программного модуля 79](#_Toc388912158)

[4.1 Разработка базы данных 79](#_Toc388912159)

[4.2 Разработка пользовательского интерфейса 83](#_Toc388912160)

[5 Разработка программы и методики испытаний 87](#_Toc388912161)

[5.1 Разработка методики тестирования 89](#_Toc388912162)

[5.2 Разработка тестового стенда 91](#_Toc388912163)

[5.3 Разработка приемочного тест-плана 92](#_Toc388912164)

[6 Программная документация 96](#_Toc388912165)

[6.1 Руководство по сборке 96](#_Toc388912166)

[6.2 Руководство по развертыванию 98](#_Toc388912167)

[6.3 Руководство пользователя 100](#_Toc388912168)

[6.3.1 Батчи 100](#_Toc388912169)

[6.3.2 Ручное создание сделок 100](#_Toc388912170)

[6.3.3 Импорт сделок из трейдинговой системы 101](#_Toc388912171)

[6.3.4 Редактирование сделок 102](#_Toc388912172)

[6.3.5 Подтверждение и отмена подтверждения батча 103](#_Toc388912173)

[6.3.6 Удаление и отмена удаления батча 103](#_Toc388912174)

[6.3.7 Авторизация батча 104](#_Toc388912175)

[6.3.8 Просмотр сделок 105](#_Toc388912176)

[6.3.9 Корпоративные события 105](#_Toc388912177)

[6.3.10 Просмотр корпоративных событий 107](#_Toc388912178)

[6.3.11 Редактирование корпоративных событий 108](#_Toc388912179)

[6.3.12 Подтверждение и отмена подтверждения корпоративного события 111](#_Toc388912180)

[6.3.13 Авторизация корпоративных событий 112](#_Toc388912181)

[6.3.14 Применение торговли 112](#_Toc388912182)

[6.3.15 Проверка регистров внутреннего учета на отрицательные балансы 113](#_Toc388912183)

[6.3.16 Закрытие бизнес-дня 114](#_Toc388912184)

[6.3.17 Выгрузка в учетную систему GAAP 114](#_Toc388912185)

[6.3.18 Выгрузка в учетную систему RAS 115](#_Toc388912186)

[7 Экономическая часть 118](#_Toc388912187)

[7.1 План-график разработки 118](#_Toc388912188)

[7.2 Смета затрат на разработку 121](#_Toc388912189)

[7.2.1 Материальные затраты 121](#_Toc388912190)

[7.2.2 Затраты на оплату труда 121](#_Toc388912191)

[7.2.3 Амортизация основных фондов 122](#_Toc388912192)

[7.2.4 Прочие затраты 124](#_Toc388912193)

[7.2.5 Общие затраты 124](#_Toc388912194)

[7.2.6 Расчет цены для научно-исследовательской работы 125](#_Toc388912195)

[7.3 Расчет и выводы по эффективности предложений 126](#_Toc388912196)

[8 Безопасность и экологичность проекта 128](#_Toc388912197)

[8.1 Анализ опасных и вредных факторов на рабочем месте пользователя 129](#_Toc388912198)

[8.1.1 Параметры микроклимата 130](#_Toc388912199)

[8.1.2 Уровень шума 133](#_Toc388912200)

[8.1.3 Освещенность 134](#_Toc388912201)

[8.1.4 Возможность поражения электрическим током 135](#_Toc388912202)

[8.1.5 Психофизиологические факторы 138](#_Toc388912203)

[8.2 Обеспечение пожарной безопасности на рабочем месте 142](#_Toc388912204)

[8.3 Экологичность проекта 147](#_Toc388912205)

[Заключение 148](#_Toc388912206)

[Список использованных источников 150](#_Toc388912207)

[Приложение 152](#_Toc388912208)

[Скрипт создания БД 152](#_Toc388912209)

[Фрагменты исходных текстов модуля 175](#_Toc388912210)

[Листинг основных хранимых процедур 175](#_Toc388912211)

[Диаграммы классов компонентов пользовательского интерфейса 184](#_Toc388912212)

[RDLC-шаблоны отчетов 186](#_Toc388912213)

# Введение

В настоящее время многие организации, занимающиеся финансовой деятельностью, стремятся пустить имеющийся капитал в оборот для получения дополнительной прибыли. Возможными вариантами здесь являются как инвестиции в различные виды деятельности, так и приобретение и/или торговля ценными бумагами. При осуществлении инвестирования в ценные бумаги инвестор преследует достижение ряда целей: надежность, доходность, рост и ликвидность вложений. Кроме стандартных целей, у инвестора могут возникнуть и специфические цели инвестирования, например, обеспечение доступа к каким-либо видам ресурсов, достижение полного контроля над предприятием и т.п. Организации могут вести торговлю ценными бумагами для своих клиентов. выступая при этом посредниками — брокерами. Доходом в брокерской деятельности является комиссия, взимаемая с клиентов за проведенные операции. Однако более крупные финансовые организации, обладающие солидным собственным капиталом — например, банки — могут торговать бумагами в своих собственных интересах, выставляя публичные котировки на ценные бумаги; в этом случае речь идет о дилерской деятельности. Дилеры получают прибыль, как правило, за счет спекулятивных операций, т.е. за счет разницы между ценами покупки и продажи.

**Целью данного** **дипломного проекта** является доработка банковской автоматизированной системы учета операций с ценными бумагами. А именно — разработка модуля, добавляющего функциональность для управления дилерской деятельностью, осуществляемой банком. Таким образом, **практическая ценность разрабатываемого модуля** заключается в решении следующего перечня задач:

* оперативный учет сделок;
* оперативный учет корпоративных событий;
* оперативный учет движения денег и ценных бумаг;
* формирование выгрузок в системы бухгалтерского учета;
* формирование отчетности для сотрудников банка и регулирующих органов.

В последнее время спрос на разработки подобного рода непрерывно растет, так как ко многим людям приходит осознание высокой ликвидности ценных бумаг и, соответственно, возможности получения значительной прибыли от манипуляций с ними. Разумеется, уже существует достаточное количество программных продуктов, решающих схожие задачи. Однако значительная их часть не удовлетворяют требованиям заказчика. Одни обладают слишком широкой, избыточной функциональностью, и для их правильной настройки, а также для обучения персонала требуются чрезвычайно большие усилия. Другие, в свою очередь, имеют недостаточный функционал, не удовлетворяющий всем потребностям заказчика. Таким образом, учесть всю специфику деятельности проще и выгоднее в собственном программном продукте, построенном по гибкой архитектуре, обеспечивающей легкую расширяемость, что очень важно для подобного рода приложений. **Разрабатываемый модуль будет интегрирован в существующую банковскую систему.**

Дорабатываемый программный продукт организован по принципу клиент-серверной архитектуры. Хранение базы данных осуществляется на сервере компании. Для работы требуется установка клиентского приложения на рабочий компьютер.

При разработке данного проекта используются следующее программное обеспечение:

* операционная система Windows 7;
* платформа Microsoft .NET Framework 3.5;
* интегрированная среда разработки Microsoft Visual Studio 2008;
* система управления базами данных Microsoft SQL Server 2008 R2.

Данный документ, описывающий программный модуль и программный продукт (точки соприкосновения с программным модулем), содержит в себе следующие пункты:

1. Постановка задачи.
2. Технико-экономическое обоснование темы проекта.
3. Проектирование программного модуля.
4. Разработка программного модуля.
5. Разработка программы и методики испытаний.
6. Программная документация.
7. Экономическая часть.
8. Безопасность и экологичность проекта.

Далее приводится краткое содержание каждого из них.

В разделе «Постановка задачи» приведены краткие требования к разрабатываемому программному модулю.

В разделе «Технико-экономическое обоснование темы проекта» приводится обоснование преимущества разработки модуля перед использованием сторонних программных средств, а также обоснование примененного для разработки программного обеспечения.

В разделе «Проектирование программного модуля» показана существующая архитектура дорабатываемой системы и адаптированная к ней структура разрабатываемого модуля. Описываются высокоуровневые требования к системе, основанные на пожеланиях заказчика. Также в этом разделе расположен глоссарий терминов.

В разделе «Разработка программного модуля» приводится список разработанных хранимых процедур с указанием области их применения и компонентов пользовательского интерфейса клиентского приложения.

В разделе «Разработка программы и методики испытаний» производится разработка методов проверки соответствия программного продукта требованиям к нему. Описываются используемые методики тестирования, окружение, на котором проводились испытания, и приемочный тест-план, прохождение которого дает гарантию требуемого уровня качества.

В разделе «Программная документация» приводится содержимое разработанных программных документов: руководство по сборке — для программиста, руководство по развертыванию — для системного администратора, и руководство пользователя.

В разделе «Экономическая часть» производится расчет стоимости разработки и срока окупаемости модуля. Показана эффективность автоматизированного учета операций по сравнению с ручным. На основании расчетных данных сделан вывод о целесообразности ведения разработки.

В разделе «Безопасность и экологичность проекта» проанализированы условия труда пользователей программного продукта, выявлены вредные и опасные факторы труда, описаны меры для обеспечения пожарной безопасности.

В приложении приведен листинг наиболее значимых частей программного модуля.

# 1 Постановка задачи

Целью данного проекта является разработка программного модуля, дополняющего уже существующий программный продукт, представляющий собой банковскую систему бэк-офиса. Модуль позволит автоматизировать учет операций с ценными бумагами, совершенных в ходе дилерской деятельности, осуществляемой банком, на всех этапах.

Данные по большинству заключенных трейдерами сделок попадают в банковскую миддл-офисную систему, поэтому модуль должен позволять импортировать их из этой системы. Однако пользователь сможет завести сделки и вручную, и, конечно, редактировать и удалять их (в том числе и импортированные).

Учет финансового результата будет вестись по методу FIFO, то есть для исполнения операции продажи будут использованы ценные бумаги, купленные раньше всего.

Модуль должен информировать пользователей о наступивших корпоративных событиях, связанных с имеющимися ценными бумагами — купонные выплаты и амортизации для облигаций, дивиденды для акций.

Необходимо согласовать работу разрабатываемого модуля с банковскими регистрами внутреннего учета денежных средств и ценных бумаг: каждая заключенная сделка и выплата по корпоративным событиям повлияет на балансы соответствующих счетов. У пользователя должна быть возможность совершить ручную операцию ввода или вывода денежных средств или ценных бумаг.

Важной функцией модуля будет выгрузка данных по обработанным операциям и связанных бухгалтерских проводок в бухгалтерские учетные системы.

Графический интерфейс модуля не должен раздражать пользователей яркими, броскими, мерцающими или анимированными элементами, отвлекая их внимание от работы. Он обеспечит интуитивно понятный и удобный доступ к требующейся информации. Для комфорта пользователей приложения графический интерфейс модуля должен быть выполнен в единой стилистике с интерфейсом существующей бэк-офисной системы. Согласно корпоративным стандартам заказчика, интерфейс модуля (как и системы) будет англоязычным.

Необходимо распространить действие уже используемых в программном продукте механизмов и принципов на разрабатываемый модуль, например, таких, как ввод, изменение, удаление данных, проверка их корректности, импорт из других систем, сообщения о невозможности проведения запрашиваемых операций и прочих ошибках. Подобная унификация упростит как задачу разработки модуля, так и дальнейшей поддержки приложения, тем самым снизив финансовые затраты заказчика.

# 2 Технико-экономическое обоснование темы проекта

## 2.1 Обоснование актуальности темы проекта и необходимости разработки

Согласно Федеральному закону от 22.04.96 г. № 39-ФЗ «О рынке ценных бумаг» установлены виды профессиональной деятельности на рынке ценных бумаг:

* брокерская деятельность;
* дилерская деятельность;
* деятельность по управлению ценными бумагами;
* деятельность по определению взаимных обязательств (клиринг);
* депозитарная деятельность;
* деятельность по ведению реестра владельцев ценных бумаг;
* деятельность по организации торговли на рынке ценных бумаг.

Допускается совмещение следующих видов деятельности:

* брокерская деятельность, дилерская деятельность, деятельность по управлению ценными бумагами, депозитарная деятельность;
* клиринговая деятельность и депозитарная деятельность;
* деятельность по организации торговли и клиринговая деятельность.

Дилерской деятельностью признается совершение сделок купли-продажи ценных бумаг от своего имени и за свой счет путем публичного объявления цен покупки и/или продажи определенных ценных бумаг с обязательством покупки и/или продажи этих ценных бумаг по объявленным лицом, осуществляющим такую деятельность, ценам. Профессиональный участник рынка ценных бумаг, осуществляющий дилерскую деятельность, именуется дилером. Дилером может быть только юридическое лицо, являющееся коммерческой организацией, а также государственная корпорация, если для такой корпорации возможность осуществления дилерской деятельности установлена федеральным законом, на основании которого она создана.

Дилерская и брокерская деятельности похожи. В обоих случаях осуществляется покупка или продажа ценных бумаг. Однако брокер покупает ценные бумаги для клиента по его поручению, дилер же приобретает ценные бумаги для себя. Стоит отметить, что покупку ценных бумаг в собственных интересах могут осуществлять также и брокеры. Отличие и главное условие дилерской деятельности — это выставление публичных котировок на покупку и продажу какого-то круга ценных бумаг. Только при выполнении данного условия профессиональный участник может получить статус дилера.

Брокер за свои услуги получает комиссионные от клиентов, а дилер зарабатывает тем, что покупает ценные бумаги по одним ценам, а продает по другим, более высоким. Например, дилер объявляет котировки ценной бумаги: цена покупки — 202,00 р.; цена продажи — 205,00 р. Разница между ценой покупки и ценой продажи называется спрэдом. Чем меньше спрэд, тем выше ликвидность рынка, тем активнее ведутся операции на рынке. Дилер в данном примере, купив облигацию за 202,00 р. и продав ее по цене 205,00 р., заработал 3рубля. Сумма небольшая, но если объемы сделок велики и торги идут активно, то дилер получает приличный доход.

Деятельность на рынке ценных бумаг с течением времени приобретает все большую популярность в нашей стране, что отражено в росте количества сделок на биржевых рынках. Как следствие, для учета сделок требуются все большие временные и денежные затраты. Увеличивается риск ошибок при обработке операций, что больно ударит по репутации дилерской организации. Автоматизация учета сделок позволит решить данные проблемы.

В настоящее время существует огромное множество программ, способных вести автоматический учет сделок с ценными бумагами. У каждой из них есть свои преимущества и недостатки.

В качестве примера рассмотрим следующие приложения:

* «Дебет плюс» — бесплатный программный продукт, позволяющий вести учет сделок, но без возможности дополнения собственными элементами логики;
* «1С: Предприятие» — платный программный продукт, позволяющий вести учет сделок, с возможностью дополнения собственными элементами логики, запрограммированными на встроенном языке.

Из-за высокой специфичности деятельности продукт без возможности расширения совершенно непригоден для учета сделок с ценными бумагами.

В случае использования продукта «1С: Предприятие» расширяемость возможна, но для этого требуется команда разработчиков, специализирующихся на встроенном языке, которая потребует значительного финансирования. Также этот программный продукт является платным: требуются ежемесячные денежные отчисления фирме-поставщику, причем за каждую копию программы. Наконец, «1С: Предприятие» обладает избыточным функционалом, не используемым в рамках дилерской деятельности, а это значит, что немалых затрат займет обучение персонала по работе с системой.

Вышеперечисленных недостатков лишен собственный программный продукт:

* количество разработчиков, специализирующихся на платформе .NET, достаточно велико по сравнению с программистами, специализирующимися на встроенных языках, поэтому команда разработчиков сможет оперативно реагировать на все возникающие запросы заказчика по доработке и исправлению при среднем финансировании;
* в собственном продукте будут присутствовать только необходимые функциональные блоки, что позволит сократить издержки на обучение персонала;
* отсутствие денежных трат при установке программного продукта на большое количество рабочих станций.

Решающим фактором в пользу разработки собственного приложения управления дилерской деятельностью является то, что банк уже имеет систему управления операциями с ценными бумагами, используемую в течение многих лет и отлично себя зарекомендовавшую. Новое приложение должно тесным образом взаимодействовать с уже существующей системой. Интеграция сторонних программных продуктов в банковскую систему в лучшем случае потребует значительных усилий разработчиков, и, как следствие, дополнительных финансовых затрат заказчика. Из-за различий в поведении, логике и графическом интерфейсе будут испытывать неудобства пользователи как существующей системы, так и интегрированного приложения. В то же время банковская система, построенная по гибкой архитектуре, обладает возможностью легкого расширения за счет разработки дополнительных модулей.

Таким образом, в отличие от уже использования уже существующих аналогов, требованиям заказчика полностью удовлетворяет разработка собственного модуля.

## 2.2 Обоснование выбора используемого программного обеспечения

Для разработки приложения была выбрана платформа Microsoft .NET Framework 3.5 для клиентской части, Microsoft SQL Server 2008 R2 для серверной части, интегрированная среда разработки Microsoft Visual Studio 2008, язык программирования C#, а также набор компонентов пользовательского интерфейса DevExpress 9.2.

Используемые инструменты являются продуктами одной компании – Microsoft (кроме компонентов DevExpress), что гарантирует легкую связь инструментов при помощи стандартных технологий, а также бесперебойную работу приложения на всех компьютерах под управлением Windows XP/7, что является важнейшими факторами при проектировании приложения. В конечном счете, относительно совокупной стоимости, продукция корпорации Microsoft, для решения данного вида задач, привлекает своей дешевизной и доступностью.

### 2.2.1 Семейство операционных систем Windows

Наибольшее распространение по всему миру из операционных систем получили продукты из семейства Microsoft Windows.

В продуктах Windows XP/7 используется 32/64-разрядная компьютерная архитектура и полностью защищенная модель памяти, созданного на основе доказавшего свою надежность кода Windows NT, что делает операционные системы семейства Windows самыми надежными на сегодняшний день.

Также явным преимуществом является возможность получить доступ к рабочему компьютеру в любое время. С помощью функции удаленного управления рабочим столом можно создавать виртуальные сеансы и использовать рабочую станцию, получая к ней доступ с другого компьютера, работающего под управлением операционных системы Windows. Таким образом, даже если пользователь находится вне офиса, в его распоряжении всегда будут необходимые данные и приложения.

Windows позволяет обезопасить данные. Благодаря шифрованной файловой системе обеспечивается надежная защита от хакеров и высокая безопасность данных. Шифрование файлов не сказывается на функционировании системы и выполняется при помощи ключа, сгенерированного случайным образом.

Дополнительно существует возможность обезопасить рабочую станцию от возможных сбоев при работе. Аварийное восстановление системы — это функция средства архивации, позволяющая восстановить состояние системы и все файлы в системном разделе, когда проблемы или изменения в операционной системе становятся причиной нестабильной работы и сбоев при загрузке. Восстановление системы — возможность быстро и легко отменить нежелательные, существенные для функционирования системы изменения. Функция восстановления системы автоматически отслеживает и фиксирует важные изменения настроек системы на вашем компьютере. Если в результате изменения какого-либо параметра системы возникла проблема или сбой, существует возможность отменить данное изменение.

Управление уровнем доступа пользователей к ресурсам и данным, а также ко всей рабочей станции в целом является еще одним важным достоинством выбранных операционных систем. Управление доступом на уровне пользователей предусматривает ряд вариантов предоставления отдельным пользователям или группе пользователей возможностей совместной работы с файлами и папками. Вы можете определить, какие папки могут совместно использоваться, кто будет иметь доступ к этим папкам, каков уровень разрешений, предоставляемых каждому пользователю или группе.

Экономия денежных средств достигается благодаря общему доступу к подключению Интернета, что позволяет защитить компьютеры от несанкционированного доступа и атак хакеров, а также ограничить доступ пользователей к определенному виду ресурсов. Общий доступ к подключению Интернета позволяет осуществлять одновременный вход в Интернет с нескольких компьютеров через одно и то же широкополосное подключение или подключение удаленного доступа. Брандмауэр подключения к интернету обеспечивает защиту от атак хакеров и вторжения через Интернет.

Кроме описанных выше частных преимуществ, общая предпочтительность данного семейства операционных систем обусловлена также дешевизной, доступностью, простотой установки и эксплуатации.

### 2.2.2 Платформа Microsoft .NET Framework 3.5

Мультиязыковая платформа .NET Framework гарантирует стабильную работу приложения на всех рабочих станциях под управлением Windows XP/7 без установки, каких-либо дополнительных пакетов. Программы написанные для данной платформы сначала переводятся в единый для .NET промежуточный байт-код Common Intermediate Language(CIL). Затем код либо исполняется виртуальной машиной Common Language Runtime (CLR), либо транслируется в исполняемый код для конкретного целевого процессора. Использование виртуальной машины предпочтительно, так как избавляет разработчиков от необходимости заботиться об особенностях аппаратной части. В случае использования виртуальной машины CLR, встроенный в неё JIT-компилятор «на лету» (just-in-time) преобразует промежуточный байт-код в машинные коды нужного процессора. Современная технология динамической компиляции позволяет достигнуть высокого уровня быстродействия. Виртуальная машина CLR также сама заботится о базовой безопасности, управлении памятью и системе исключений, избавляя разработчика от части работы.

Объектные классы .NET, доступные для всех поддерживаемых языков программирования, содержатся в библиотеке Framework Class Library (FCL). В FCL входят классы Windows Forms, ADO.NET, ASP.NET, Language Integrated Query, Windows Presentation Foundation, Windows Communication Foundation и другие. ЯдроFCLназываетсяBase Class Library (BCL).

### 2.2.3 Система управления базами данных Microsoft SQL Server 2008 R2

В качестве системы управления реляционными базами данных для проекта выбрана SQL Server 2008 R2, которая в свою очередь, так же разработана корпорацией Microsoft, использующая язык запросов T-SQL, и имеющая высокую производительность при работе с промышленными базами.

SQL Server 2008 R2 помогает использовать последние достижения в области аппаратных технологий и обеспечивает высокий уровень масштабирования на основе стандартного оборудования с небольшой общей стоимостью владения. Операционные системы Windows в комбинации с SQL Server 2008 R2 поддерживает до 256 логических процессоров. Поддержка большего количества виртуальных машин на одном физическом узле ведет к снижению издержек и улучшению масштабируемости, и гибкости виртуальной инфраструктуры. Функции динамической миграции позволяет без прерывания работы перемещать виртуальную машину с одного сервера на другой. Все это обеспечивает максимальную отдачу от инвестиций в оборудование и более полную реализацию возможностей для консолидации.

Регулятор ресурсов SQL Server 2008 помогает обеспечивать прогнозируемую производительность на одинаковом уровне. С его помощью компания может управлять рабочими нагрузками SQL Server и потреблением системных ресурсов, задавая лимиты использования памяти и ЦП для отдельных приложений, а также поддерживать равномерную скорость выполнения параллельных нагрузок.

Защита ценной информации осуществляется с помощью гибкого и прозрачного механизма шифрования баз данных, файлов данных и файлов журналов. Функция шифрования данных в SQL Server 2008 позволяет защищать информацию от несанкционированного использования, не покидая интерфейса исходного приложения. Подсистема аудита SQL Server обеспечивает соблюдение нормативных требований и создает журнал аудита доступа к данным.

### 2.2.4 Интегрированная среда разработки Microsoft Visual Studio 2008

Интегрированная среда разработки Visual Studio 2008 включает в себя редактор кода, технологию автодополнения кода, отладчик и возможность подключения встраиваемых инструментов, что делает ее практически уникальной по сравнению с аналогами.

Microsoft Visual Studio 2008 помогает индивидуальным программистам и небольшим группам, создающим любые виды ПО, ускорить разработку приложений и создание пользовательских интерфейсов с принципиально новым уровнем удобства, повысить эффективность коллективной работы. Используя большое количество функций в Visual Studio 2008, разработчики могут решать сложные бизнес-задачи, используя основные Windows API, Microsoft Foundation Classes, .NET Framework или любой SDK, при одновременном сокращении совокупной стоимости процесса разработки. С помощью одного набора инструментов можно разрабатывать системные службы, драйверы устройств, утилиты, игры, настольное ПО, приложения для автоматизации направлений бизнеса и многое другое. Независимо от типа ПО, Visual Studio помогает создавать более удобные пользовательские интерфейсы, связывать приложения и бизнес-процессы и ускорять реализацию решений. Визуальные конструкторы Visual Studio 2008 позволяют создавать мощные и привлекательные приложения, основанные на WinForms — графической подсистеме .NET Framework. С помощью графического конструктора и отладчика для Windows Communication Foundation, входящих в Visual Studio 2008, разработчики могут легко связывать системы или приложения. Независимо от того, какие протоколы и транспорты используются, для программирования применяется общий API. Аналогичным образом инструменты для Windows Workflow Foundation позволяют моделировать в ПО реальные бизнес-процессы, предоставляя средства, с помощью которых несложно визуализировать, создавать, редактировать и отлаживать рабочие процессы с участием людей или устройств. Также Visual Studio 2008 позволяет создавать не только «типовые» приложения для Windows. Когда ведется разработка драйверов Windows, поддерживающих новое оборудование производители аппаратно-программного обеспечения могут писать программы, напрямую обращаясь к Win32 API, и использовать SDK, предоставляемые Microsoft. Это гарантирует совместимость и ускоряет реализацию решений.

### 2.2.5 Язык программирования C#

Объектно-ориентированный язык программирования C# является относительно молодым языком, который построен на улучшенной объектной модели, определенной в C++, в нем унифицирована система типов (т.е. тип рассматривается как объект), реализовано автоматическое управление памятью, облегчен процесс отладки приложения и пр., но при этом стал проще и удобнее для программиста, что позволяет программисту тратить время на написание программ гораздо эффективнее, чем в его предшественниках на платформе .NET. Многие из возможностей языка программирования C# были созданы для четырех различных целей:

* единая система типов и упрощение способов использования языком значений и ссылочных типов;
* разработка на основе компонентов, основанная на таких возможностях как XML комментарии, атрибуты, свойства, события и делегаты;
* практические преимущества основанные на уникальных возможностях языка C#, включая работу с безопасными указателями, контроль переполнения и т. д.;
* удобные языковые конструкции, такие как операторы foreach и using, повышающие производительность разработчика.

В декабре 2001 язык программирования C# был утвержден Европейской Ассоциацией производителей компьютеров (European Computer Manufacturer's Association (ECMA)) как стандартный (ECMA 334). Вскоре после этого стандарт C# был отправлен на рассмотрение в Международную организацию по стандартизации (ISO), где он также вскоре был стандартизован. Важной вехой в развитии нового языка программирования является создание стандарта C#, что несет за собой перспективу множества реализаций на различных платформах операционных систем. В самом деле, некоторое количество сторонних производителей компиляторов и исследователей реализовали стандарт и создали собственные версии компилятора C#.

### 2.2.6 Набор компонентов пользовательского интерфейса DevExpress 9.2

Набор компонентов пользовательского интерфейса – DevExpress 9.2 —содержит большой набор компонентов для создания интуитивно понятного пользовательского интерфейса и удобного представления информации, который также покрывает все потребности разработчика и имеет хорошую документацию и поддержку. В целом использование сторонних компонентов влечет некоторые дополнительные затраты, к примеру, необходимость обучения программистов использования компонентов в полной мере. Однако эти затраты перекрываются выгодой использования компонентов при создание большого программного продукта, и экономией ресурсов при сопровождении таких приложений. Кроме этого, различные пожелания заказчика, которые рано или поздно у него возникают, можно также реализовать гораздо быстрее и качественнее, что повышает доверие к компании-разработчику. Таким образом, выгода от использования сторонних компонентов практически всегда окупает их стоимость.

# 3 Проектирование программного модуля

## 3.1 Архитектура системы

Дорабатываемая система учета операций с ценными бумагами представляет собой большую систему, находящуюся в эксплуатации несколько лет, постоянно получающую обновления и расширения. Структура ее приведена на рисунке 3.1. Данная система успешно взаимодействует с другими банковскими системами, некоторые из которых также указаны на этом рисунке.



Рисунок 3.1 — Структура системы учета операций с ценными бумагами

Она состоит из двух основных частей — ядра и сервисов, обращающихся к ядру.

### 3.1.1 Ядро системы

Ядро системы построено по клиент-серверной архитектуре. Серверная часть представляет собой базу данных, находящуюся под управлением Microsoft SQL Server 2008 R2. Клиентская часть — .NET приложение, написанное на языке C#. Помимо стандартных .NET-библиотек, в нем применяются базовые core-библиотеки заказчика (используются во многих требующихся ему приложениях), а также ряд сторонних библиотек:

* BLToolkit — Business Logic Toolkit for .NET, представляет собой бесплатную библиотеку классов для реализации бизнес-логики на .NET;
* log4net — бесплатная библиотека классов, служащих для логирования сообщений;
* библиотеки компонентов пользовательского интерфейса компании DevExpress.

Для повышения быстродействия клиентской части значительная часть бизнес-логики (обработка данных и вычисления) реализована на стороне сервера в виде хранимых процедур и функций.

Основная задача клиентского приложения — обеспечение графического интерфейса для взаимодействия пользователя с базой данных. Он построен с помощью стандартных компонентов WinForms и набора пользовательских компонентов DevExpress 9.2.

К бизнес-логике клиентской части относятся реализованные правила обмена данными (импорт, экспорт) с другими системами посредством файлов.

Другой немаловажной целью клиента является построение отчетов. Разнообразны как набор возможных форматов (.rpt, .doc, .xls, .pdf), так и используемые для их построения технологии (Crystal Reports, MS Reporting, COM-модель, XSLT-шаблоны).

Клиентское приложение построено по классической трехслойной архитектуре. Каждый слой отражает логическую группировку классов в приложении. В нижней части иерархии — слой данных (англ. Data Layer, DL), или слой доступа к данным (англ. Data Access Layer, DAL), представляющий собой набор классов, обращающихся к базе данных для получения, добавления, обновления, удаления и обработки информации. Для этого используется технология ADO.NET. Средний уровень — бизнес-слой (англ. Business Layer, BL), или слой бизнес-логики (англ. Business Logic Layer, BLL). Это совокупность бизнес-сущностей, их наборов и бизнес-компонентов для их обработки. Его цель — организовать взаимодействие между слоем доступа к данным и следующим уровнем — слоем представления (англ. Presentation Layer, PL). Слой представления включает в себя классы, обеспечивающие непосредственный контакт с пользователем. В основном это компоненты пользовательского интерфейса.

Взаимодействие с другими системами осуществляется как прямым доступом к базам данных с использованием синонимов БД и связанных серверов (англ. linked servers), так и с помощью обмена особыми файлами выгрузок, называемых хэндоффами (от англ. to hand off — передавать). Файлы хэндоффов обычно являются текстовыми файлами, имеющими строго определенную разметку и структуру, которую система сможет интерпретировать в соответствии с разработанными правилами.

Для сведения частоты появления ошибок при вводе данных в системе используется особый механизм, именуемый «Maker-Checker». Его смысл в том, что любые изменения, вносимые одним пользователем, должны быть независимо проверены и одобрены другим пользователем. В противном случае эти изменения не будут учтены при обработке данных.

В целях безопасности клиентское приложение невозможно запустить напрямую. Необходимо использовать специальное банковское приложение, управляющую учетными записями пользователей, их правами на запуск различных банковских систем, на доступ к проведению различных операций в них и т.п. Это приложение обеспечения безопасности шифрует необходимые параметры и передает их требуемой программе, после чего осуществляет запуск. На рисунке 3.1 в целях упрощения она не указана.

Обширность системы увеличивает вероятность появления ошибок в старых компонентах при разработке новых ее частей, поэтому для помощи программисту имеются средства, позволяющие быстро выявить и исправить регрессионные ошибки. Для клиентского приложения написаны юнит-тесты (англ. unit testing — модульное тестирование), покрывающие базовую функциональностью классов клиентского кода. Они основаны на открытой среде юнит-тестирования NUnit. Также существует специальная утилита для автоматизированного функционального тестирования приложения, проверяющая более сложную логику.

В рамках дипломного проекта доработке будет подвергаться ядро системы, поэтому в дальнейшем под системой учета операции с ценными бумагами будет пониматься именно ядро, описанное в данном разделе.

### 3.1.2 Сервисы

Сервисы системы представляют собой службы Windows (англ. Windows services), написанные на языке C# платформы .NET. Как и в случае ядра, для реализации сервисов использовались core-библиотеки заказчика, BLToolkit и log4net. Для отладки разработаны консольные версии.

Большинство используемых сервисов автоматизируют обмен файлами (хэндоффов и не только) между рассматриваемой системой и прочими. Например, имеется сервис для автоматического формирования и передачи в клиентскую систему банка нескольких видов отчетов в особом формате, интерпретируемом этой системой. Существуют службы для импорта клиентских корпоративных событий из специальной системы их учета, для экспорта клиентских позиций и балансов в трейдинговую систему и прочие.

Работа сервисов гибко настраивается с помощью конфигурационных файлов. Регулируются как параметры, относящиеся непосредственно к задаче сервиса (время работы, условия срабатывания и прочие), так и уровень логирования (для служб подробное логирование особенно важно, так как это единственный способ получить информацию о сбоях в работе и выявить их причины), список адресов электронной почты для рассылки сообщений об ошибках и другое.

## 3.2 Анализ прикладной области и разработка требований

Анализ является первым этапом создания ПО, на котором уточняются, формализуются и документируются требования к системе. Фактически на этом этапе дается ответ на вопрос: «**Что должна делать будущая система?»**. По результатам обследования строится модель «**как есть»**: обобщенную логическую модель исходной предметной области, отображающую ее функциональную структуру, особенности основной деятельности и информационное пространство, в котором эта деятельность осуществляется.

Далее создают модель «**как надо**»: усовершенствованную обобщенную логическую модель, отображающую реорганизованную предметную область или ее часть, которая подлежит автоматизации. Эта стадия анализа содержит элементы проектирования. Очевидно, что конечная цель разработки ПО — это не моделирование, а получение работающих приложений (кода).

Для моделирования систем, в общем, и в структурном анализе в частности, используются три группы методов, иллюстрирующих:

1.Функции, которые система должна выполнять;

2.Отношения между данными;

3. Зависящее от времени поведение системы (аспекты реального времени).

Среди множества методов решения данных задач в методологиях структурного анализа наиболее часто и эффективно применяются:

* FDD (Functional Decomposition Diagrams) — диаграммы функциональной декомпозиции (например в нотации IDEF 0);
* DFD (Data Flow Diagrams) — диаграммы потоков данных совместно со словарями данных и спецификациями процессов (или миниспецификациями);
* ERD (Entity-Relationship Diagrams) — диаграммы «сущность – связь»;
* STD (State Transition Diagrams) —диаграммы переходов состояний.

Все они содержат графические и текстовые средства моделирования: первые — для удобства демонстрирования основных компонент модели, вторые — для обеспечения точного определения ее компонент и связей.

### 3.2.1 Используемые термины и сокращения

Дилерская деятельность является весьма специфичной областью бизнеса. Как и в любой отрасли, в ней выработались термины, характеризующие различные сущности и явления, связанные с бизнес-процессом. Необходимо привести их пояснение:

1. GAAP (Generally Accepted Accounting Principles) — национальные стандарты бухгалтерского учета. Чаще всего данная аббревиатура употребляется для обозначения US GAAP (United States GAAP). Далее будет использоваться именно такое значение.
2. RAS (Russian Accounting Standards) — российские стандарты бухгалтерского учета. Иногда называют Russian GAAP, но во избежание путаницы с вышеприведенным термином далее будет использоваться аббревиатура RAS.
3. Эмитент (англ. issuer) — организация, выпустившая ценные бумаги для развития и финансирования своей деятельности.
4. Облигация (англ. bond) — эмиссионная долговая ценная бумага, владелец которой имеет право получить от [эмитента](http://ru.wikipedia.org/wiki/%D0%AD%D0%BC%D0%B8%D1%82%D0%B5%D0%BD%D1%82) облигации в оговоренный срок ее [номинальную стоимость](http://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D0%BC%D0%B8%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D0%BE%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C) деньгами или в виде иного имущественного эквивалента. Т.е. покупка облигации — это выдача денег эмитенту в долг. Связанные термины:
   * Номинал (англ. nominal) — цена в валюте, определяемая при выпуске облигации.
   * Погашение (англ. maturity) — возвращение эмитентом первоначальной суммы долга (полностью или поэтапно).
   * Амортизируемая облигация (англ. amortized bond) — облигация, погашение которой происходит поэтапно. Частичное погашение называется амортизацией. Амортизация уменьшает номинал облигации.
   * Купон (англ. coupon) — проценты за долг.
   * Купонное погашение (англ. coupon maturity) — выплата эмитентом купонов держателям облигации.
5. Акция (англ. equity) — эмиссионная [ценная бумага](http://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B1%D1%83%D0%BC%D0%B0%D0%B3%D0%B0), закрепляющая права ее владельца (акционера) на получение [дивидендов](http://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B2%D0%B8%D0%B4%D0%B5%D0%BD%D0%B4), на участие в управлении акционерным обществом и на часть [имущества](http://ru.wikipedia.org/wiki/%D0%98%D0%BC%D1%83%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%BE), остающегося после его [ликвидации](http://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BA%D0%B2%D0%B8%D0%B4%D0%B0%D1%86%D0%B8%D1%8F_%D1%8E%D1%80%D0%B8%D0%B4%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B3%D0%BE_%D0%BB%D0%B8%D1%86%D0%B0). Связанные понятия:
   * Дивиденды (англ. dividends) — часть [прибыли](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%B1%D1%8B%D0%BB%D1%8C) [акционерного общества](http://ru.wikipedia.org/wiki/%D0%90%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B5%D1%80%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%BE), распределяемая между [акционерами](http://ru.wikipedia.org/wiki/%D0%90%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B5%D1%80).
6. Корпоративное событие (англ. corporate action, CA) — обобщенное название выплат дивидендов, купонных погашений, амортизаций и полных погашений по ценным бумагам, удерживаемым банком.
7. Бизнес-день (англ. business day) — день, принятый для удобства учета сделок с ценными бумагами. Его начало — 18:00 (по московскому времени) текущей даты, окончание — 12:00 даты, следующей за датой начала бизнес-дня. Выходные и праздничные дни исключаются.
8. Срочная сделка, форвард сделка (англ. forward deal), T+ сделка (англ. T+ deal) — сделка с отложенным исполнением: расчеты совершаются в некоторую дату после даты заключения.
9. Сделка РЕПО (англ. REPO deal, от repurchase agreement) — сделка покупки (продажи) [ценной бумаги](http://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B1%D1%83%D0%BC%D0%B0%D0%B3%D0%B0) с обязательством обратной продажи (покупки) через определенный срок по заранее определенной цене. Состоит из двух частей, часто называемых «ногами» (англ. legs). Обычно совершается с целью кредитования участников рынка деньгами либо ценными бумагами. Виды РЕПО:
   * Прямое РЕПО (англ. direct REPO) — продажа с обязательством покупки.
   * Обратное РЕПО (англ. reversed REPO) — покупка с обязательством продажи.
10. Портфель (англ. portfolio) — совокупность инвестиционных вложений [юридического](http://ru.wikipedia.org/wiki/%D0%AE%D1%80%D0%B8%D0%B4%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BB%D0%B8%D1%86%D0%BE) или [физического](http://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BB%D0%B8%D1%86%D0%BE) лица. Далее под портфелем будет иметься ввиду портфель ценных бумаг.
11. Портфель ценных бумаг (англ. securities portfolio) — совокупность [ценных бумаг](http://www.grandars.ru/student/finansy/cennaya-bumaga.html), принадлежащих юридическому или физическому лицу.
12. Категория портфеля (англ. portfolio category) — тип портфеля в зависимости от ряда признаков (цели приобретения ценных бумаг, их типом и т.п.). Далее будут употребляться следующие категории:
    * TD (trading) — торговый портфель, соответствует категории Held for trading классификации финансовых активов согласно GAAP. Ценные бумаги в торговый портфель приобретаются на короткий срок с целью их продажи с получением прибыли за счет колебаний цен.
    * REPO — портфель для заемных бумаг (бумаги, купленные в ходе обратного РЕПО).
    * AFS (available for sale) — портфель, соответствующий категории Available for sale классификации финансовых активов согласно GAAP. Ценные бумаги в AFS портфель приобретаются на значительный срок и, как правило, продаются очень редко.
13. Депозитарий (англ. depo location) — профессиональный участник рынка ценных бумаг, осуществляющий услуги по хранению [сертификатов ценных бумаг](http://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%80%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82_%D1%86%D0%B5%D0%BD%D0%BD%D1%8B%D1%85_%D0%B1%D1%83%D0%BC%D0%B0%D0%B3&action=edit&redlink=1) и по учёту перехода [прав собственности](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B0%D0%B2%D0%BE_%D1%81%D0%BE%D0%B1%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D0%B8) на [ценные бумаги](http://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D1%83%D0%BC%D0%B0%D0%B3%D0%B8).
14. Счет депо (англ. depo account) — совокупность записей в регистрах [депозитария](http://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%B0%D1%80%D0%BD%D0%B0%D1%8F_%D0%B4%D0%B5%D1%8F%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C), предназначенная для учёта [ценных бумаг](http://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D1%83%D0%BC%D0%B0%D0%B3%D0%B8).
15. Позиция (англ. position) — наличие остатков по ценным бумагам.
16. Длинная позиция (англ. long position) — позиция, образованная покупкой.
17. Короткая продажа (англ. short sell):
    * В терминах GAAP — продажа ценных бумаг при их отсутствии.
    * В терминах RAS — продажа заемных ценных бумаг из РЕПО портфеля при отсутствии собственных бумаг в торговом портфеле.
18. Короткая позиция (англ. short position) — позиция, образованная короткой продажей.
19. Откупка (англ. buyback) — операция, закрывающая короткую позицию.

Помимо них, существует ряд определений для сущностей и процессов, используемых в дорабатываемой системе.

1. Батч (англ. batch — партия, пакет) — коллекция сделок, загруженных за одну итерацию, предназначена для наглядного разделения добавленных сделок из общего их числа.
2. Применение торговли (англ. apply trading) — процедура учета заведенных за день сделок в системе.
3. CBD (англ. current business day — текущий бизнес-день) — текущий бизнес-день, заведенный в системе.
4. Хэндофф (от англ. to hand off — передавать) — файл выгрузки из системы, предназначенный для передачи данных в другую систему. Имеет строгую разметку (определенная структура, набор применяемых символов-разделителей и т.п.), которая интерпретируется системой при импорте.

Кроме того, примем следующие названия и сокращения для банковских систем, имеющих отношение к данному проекту:

1. Система учета операций с ценными бумагами, СУОЦБ — дорабатываемая автоматизированная система.
2. Трейдинговая система — сторонняя банковская система, в которой трейдеры заключают сделки и предварительно определяются категории портфелей сделок.
3. Учетная система GAAP, УС GAAP — сторонняя банковская система, являющаяся системой бухгалтерского учета согласно принципам GAAP. Обрабатывает сгенерированные СУОЦБ выгрузки с проводками.
4. Учетная система RAS, УС RAS — сторонняя банковская система, являющаяся системой бухгалтерского учета согласно принципам RAS. Самостоятельно генерирует проводки на основе выгрузок из СУОЦБ.

### 3.2.2 Описание требуемого бизнес-процесса

После внедрения разрабатываемого модуля пользователи системы будут совершать в течение дня следующую последовательность действий:

1. Ежедневно в 12:00 оператор создает новый бизнес-день, указывая при этом такие параметры, как курсы валют, ставка рефинансирования ЦБ России и т.п.
2. В течение дня происходит ручной ввод внебиржевых сделок, подтверждение корпоративных событий. При необходимости модифицируется информация о ценных бумагах, расписании корпоративных событий, портфелях и др.
3. Около 18:30, когда торги на бирже закрылись, осуществляется импорт биржевых сделок из трейдинговой системы.
4. После 18:30 происходит импорт биржевых цен на ценные бумаги и импорт котировальных листов.
5. После окончания ввода всех необходимых данных выполняется процедура применения торговли в СУОЦБ без проверки регистров внутреннего учета на отрицательные балансы.
6. Генерируются файлы выгрузки в УС GAAP и УС RAS для последующего экспорта.
7. Утром следующего дня в соответствии с отрицательными балансами осуществляется ввод операций с денежными средствами.
8. Выполняется процедура проверка регистров внутреннего учета на отрицательные балансы.
9. Бизнес-день отмечается как закрытый.
10. Формируется необходимая отчетность для регулирующих органов и сотрудников банка.

Далее описан ряд высокоуровневых требования к добавляемой функциональности, основанных на пожеланиях заказчика. На их основе были разработаны алгоритмы процессов, часть из которых приведена в конце раздела.

### 3.2.3 Требования к сделкам

Заключенные на бирже сделки могут быть исполнены из различных банковских портфелей. Поэтому они должны храниться в двух различных регистрах:

* регистр RAW сделок — отражает сделки в том виде, в котором они пришли с биржи;
* регистр SPLIT сделок — отражает сделки в том виде, в котором они будут исполнены в разрезе портфелей.

Сделки должны быть сформированы в коллекции, называемые батчами. Батчи должны иметь статусы, отражающие их состояние и доступные операции. Схема переходов статусов батчей приведена на рисунке 3.2. Там же указаны операции, осуществляемые над ними.



Рисунок 3.2 — Смена статусов батчей после совершения различных операций

Просмотр разрешен для RAW и SPLIT сделок вне зависимости от даты заведения. Редактирование возможно только для сделок, входящих в состав добавленных сегодня батчей, если те находятся в статусе «Input».

Необходимо разработать импорт биржевых сделок из трейдинговой системы. Он должен осуществляться не более одного раза в день, при этом подгружаются сделки с датой заключения, равной бизнес-дате в СУОЦБ. Импорт реализовывается посредством обращения к синонимам таблиц базы данных трейдинговой системы, находящейся на связанном сервере. Правила отображения сделок из таблиц трейдинговой системы на таблицы СУОЦБ отражены в спецификации заказчика.

В отличие от большинства других сущностей, заведенные батчи и сделки не должны удаляться при удалении заведенного бизнес-дня.

В дерево панелей приложения необходимо добавить новый узел — «Trade batches (P)». Новая панель должна иметь несколько вкладок.

На первой отображаются две таблицы: в верхней части — список батчей, заведенных сегодня, с возможностью их редактирования; в нижней — список RAW сделок, относящихся к выделенному в верхней таблице батчу. При двойном щелчке по батчу в верхней таблице должна появляться форма с возможностью редактирования сделок, входящих в этот батч.

Вторая и третья вкладки представляют собой графическое представление регистра RAW и SPLIT сделок соответственно с возможностью фильтрации по дате заключения.

Четвертая и пятая вкладки отвечают за ручные операции с денежными средствами и ценными бумагами.

### 3.2.4 Требования к учету сделок по методу FIFO

Учет финансового результата должен вестись по методу FIFO. Это означает, что для исполнения продажи будут использованы ценные бумаги, купленные раньше всего.

Каждая сделка покупки инициирует новую цепочку FIFO. При этом образуется новая позиция остатков с балансом, равным количеству купленных ценных бумаг.

Перед обработкой сделок остатки на конец предыдущего бизнес-дня должны быть продублированы на начало текущего.

Каждая сделка продажи ищет цепочки, образованные сделками по такой же ценной бумаге и попавшие в портфель той же категории, имеющие остаток, который продажа сможет использовать для своего исполнения. Цепочки перебираются в определенном порядке. Продажа уменьшает баланс остатка по каждой цепочке на максимально возможную величину. Для учета сделка продажи включается в каждую цепочку, из которой она исполнилась.

Остатки по полностью проданным цепочкам хранить не следует.

Таким образом, в ходе построения FIFO должны выполняться следующие операции:

* заведение новых цепочек;
* добавление операций к цепочкам;
* актуализация позиций активных цепочек.

Для реализации учета сделок по методу FIFO необходимы три таблицы:

* таблица, хранящая все цепочки;
* таблица, хранящая балансы цепочек на определенную дату;
* таблица, хранящая все сделки, включенные в цепочку.

Банку необходимо вести бухгалтерский учет по принципам как GAAP, так и RAS. Вследствие этого учет сделок в системе должен осуществляться двумя способами, соответствующим этим принципам. Соответствующие структуры данных называются GAAP FIFO и RAS FIFO.

Требования к GAAP FIFO:

1. Сделки обрабатываются в дату заключения.
2. Сделки РЕПО (как прямое, так и обратное) не учитываются.
3. Невозможность получить ошибку вследствие нехватки ценных бумаг для исполнения продажи: согласно принципам GAAP, в этом случае будет образована новая цепочка с отрицательным балансом, называемая короткой позицией. Образовавшая ее продажа именуется короткой продажей (в терминах GAAP).
4. Короткая позиция должна быть немедленно закрыта при следующей покупке ценных бумаг. Соответствующая операция называется откупкой.

Требования к RAS FIFO:

1. Сделки обрабатываются в реальную дату исполнения.
2. Сделки РЕПО (как прямое, так и обратное) учитываются.
3. При нехватке ценных бумаг для исполнения сделки продажи ил прямого РЕПО фиксируется ошибка, учет продолжить невозможно.
4. Продажа из торгового портфеля в случае нехватки ценных бумаг в имеющихся остатках может частично или полностью исполниться из заемных бумаг, т.е. из РЕПО портфеля. Такая продажа называется короткой продажей (в терминах RAS). Аналогично, прямое РЕПО из торгового портфеля может частично или полностью быть исполнено из РЕПО портфеля. В этом случае говорят о прямом РЕПО из обратного РЕПО.
5. В случае короткой продажи или прямого РЕПО из обратного ценные бумаги позже должны вернуться в РЕПО портфель для того, чтобы были исполнены обязательства по второй части сделки обратного РЕПО (т.е. 2-ая «нога» сделки обратного РЕПО). Подобные операции по возвращению ценных бумаг в РЕПО портфель называются откупками. Откупки могут быть совершены из портфелей трех основных категорий.
6. В случае, если для исполнения продажи из торгового портфеля не хватает ценных бумаг ни в торговом портфеле, ни заемных бумаг в РЕПО портфеле, то в качестве крайней меры допускается частичное или полное исполнение этой продажи из AFS портфеля. Такая операция называется продажей с AFS покрытием.
7. Некоторые виды откупок, меняющие реальные позиции в торговом и AFS портфелях, и продажи с AFS покрытием должны также менять и предварительные позиции в GAAP FIFO.

Для выполнения последнего требования к RAS FIFO следует, что оно должно вычисляться перед GAAP FIFO. На его основе необходимо в регистр SPLIT сделок добавлять корректирующие технические покупки и/или продажи, которые затем будут учтены в GAAP FIFO.

В связи с требованиями касательно выгрузки в УС RAS целесообразно вести RAS FIFO не на базе регистра SPLIT сделок, а на основе проводок блока BSDEAL (BSDEAL-сделок) данной выгрузки. Перед построением RAS FIFO необходимо согласно спецификации заказчика сформировать блок BSDEAL, а затем с его использованием вычислять RAS FIFO. Перед закрытием бизнес-дня сгенерированные BSDEAL-сделки будут просто переданы в выгрузку в УС RAS.

### 3.2.5 Требования к применению торговли

После того, как все необходимые данные о торговле за текущий бизнес-день введены и подтверждены, необходимо произвести их обработку. Для этого необходимо ввести процедуру, называемую применением торговли. В ходе ее выполнения следует совершить следующие действия:

* вычислить RAS FIFO за текущую дату;
* вычислить GAAP FIFO за текущую дату;
* построить проводки и остатки внутреннего учета (как денежных средств, так и ценных бумаг) за текущую дату. Проверку на отрицательные балансы счетов не совершать.

После выполнения применения торговли должны быть произведены сопоставления остатков по ценным бумагам в разных структурах данных:

* RAS FIFO и GAAP FIFO, при этом вследствие использования разной даты для учета (дата заключения сделки в GAAP FIFO, дата исполнения сделки в RAS FIFO) для корректной проверки необходимо принять во внимание заведенные Т+ сделки;
* RAS FIFO и внутренний регистр учета ценных бумаг.

При обнаружении расхождений необходимо вывести предупреждающее сообщение.

На главную панель приложения необходимо добавить кнопку, инициирующую выполнение данной операции. Следует расположить ее рядом с кнопкой проверки внутреннего учета на отрицательные остатки и флажком закрытия бизнес-дня.

### 3.2.6 Требования к внутреннему учету

Требуется расширить существующую функциональность. Помимо ручных операций с денежными средствами и ценными бумагами, на регистры внутреннего учета также должны влиять:

* частичные выплаты по платежным корпоративным события — на регистр учета денежных средств (при наступлении реальной даты оплаты);
* неплатежные корпоративные события — на регистр учета ценных бумаг (при наступлении реальной даты исполнения);
* сделки (из регистра RAW) — на регистры учета денежных средств (при наступлении реальной даты оплаты) и ценных бумаг (при наступлении реальной даты исполнения).

Необходимые счета внутреннего учета должны быть указаны в таблицах, хранящих соответствующие сущности.

Новые проводки внутреннего учета требуется реализовать в соответствии со спецификацией заказчика.

Панели для управления ручными операциями с ценными бумагами и денежными средствами должны быть перемещены в узел «Trade batches (P)».

Следует разделить вычисление остатков внутреннего учета и проверку на отрицательные балансы. Первое необходимо включить в механизм применения торговли, второе — выделить в отдельную процедуру. При этом должен быть реализован строгий порядок выполнения операций — сначала применение торговли, затем проверка негативных балансов. Лишь после этого может быть закрыт бизнес-день.

На главную панель приложения необходимо добавить кнопку, инициирующую выполнение данной операции. Следует расположить ее рядом с кнопкой применения торговли и флажком закрытия бизнес-дня.

### 3.2.7 Требования к закрытию бизнес-дня

Бизнес-день может быть закрыт лишь после применения торговли и проведения проверки регистров внутреннего учета на отрицательные остатки.

Если после совершения данных операций были внесены изменения в батчи, сделки и/или корпоративные события, то перед необходимо запретить возможность закрыть бизнес-день: сначала нужно повторить требуемые операции.

Следует запретить создание нового бизнес-дня, если не закрыт еще текущий бизнес-день.

На главную панель приложения должен быть добавлен флажок, отвечающий за закрытие дня. Его следует расположить рядом с кнопками применения торговли и проверки балансов внутреннего учета.

### 3.2.8 Требования к корпоративным событиям

Корпоративные события должны генерироваться при создании нового бизнес-дня.

Выплаты по платежным событиям (купоны, дивиденды и погашения номинала) могут приходить в несколько этапов по частям. Они должны учитываться во внутреннем регистре учета денежных средств.

Существуют неплатежные события, отражающиеся в изменении балансов счета депо. Так, полное погашение облигации инициирует ликвидацию всех позиций по данной ценной бумаге.

События должны храниться в регистре событий, состоящем из нескольких таблиц:

* общий регистр событий;
* регистр частичных выплат по платежному событию;
* регистр для связи счета депо и неплатежных событий.

При формировании событий следует рассматривать остатки во внутреннем учете.

События генерируются с пустыми реальными датами. Они заполняются пользователем.

События должны иметь статусы, отражающие их состояние и доступные операции. Схема переходов статусов корпоративных событий приведена на рисунке 3.3. Там же указаны операции, осуществляемые над ними.

Рисунок 3.3 — Смена статусов корпоративных событий после совершения различных операций

В дерево панелей приложения должен быть добавлен новый узел с именем «Events (P)», представляющий собой регистр корпоративных событий. Новая панель должна иметь 2 вкладки, отвечающие за платежные и неплатежные события. На каждой вкладке в верхней части необходимо поместить таблицу со списком событий и возможностью фильтрации по типу корпоративного события и дате, в нижней части — таблицу со списком частичных выплат (для вкладки платежных событий) либо со списком счетов депо (для вкладки неплатежных событий). Требуется реализовать возможность редактирования всех параметров событий согласно спецификации заказчика.

Помимо специального регистра, корпоративные события должны также создаваться в регистре SPLIT сделок (купоны — на основе остатков RAS FIFO, прочие события — на основе остатков GAAP FIFO) для дальнейшего учета в GAAP FIFO и/или выгрузке в УС GAAP.

### 3.2.9 Требования к выгрузке в учетную систему GAAP

Выгрузка в УС GAAP генерирует бухгалтерские проводки согласно принципам GAAP, используя в качестве основы данные сегодняшней торговли и результаты ее применения. Номер и тип счета, сторона бухгалтерского учета (дебит/кредит), валюта и сумма проводки зависят от вида сделки, вида рассматриваемого финансового показателя, категории портфеля, а также некоторых других показателей и должны быть реализованы согласно спецификации заказчика, равно как и преобразование подготовленных проводок в экспортируемый файл хэндоффа.

Помимо собственно проводок, необходимо сохранять ряд информации о выгрузке, такую как имя совершившего ее пользователя, дату и время совершения выгрузки и бизнес-день, за который совершается выгрузка.

На главную панель приложения в секции прочих выгрузок должна быть добавлена кнопка для совершения данной выгрузки и экспорта ее в текстовый файл. Также необходимо организовать возможность просмотра всех совершенных когда-либо выгрузок.

### 3.2.10 Требования к выгрузке в учетную систему RAS

Выгрузка в УС RAS, используя в качестве основы данные сегодняшней торговли и результаты ее применения, генерирует данные, которые, будучи переданными в УС RAS, интерпретируются ею и служат основой для автоматического создания бухгалтерских проводок согласно принципам RAS.

Выгрузка состоит из нескольких независимых блоков:

* BSINT — данные по ежедневным купонным накоплениям;
* BSAMRT — данные по амортизациям ценных бумаг;
* BSDEAL — данные о исполненных сделках и заключенных срочных сделках.

Набор данных, необходимых для экспорта в каждом блоке, равно как и принцип записи блоков в файл хэндоффа регламентируется спецификацией заказчика.

Необходимо сохранять ряд информации о выгрузке, такую как имя совершившего ее пользователя, дату и время совершения выгрузки и бизнес-день, за который совершается выгрузка.

На главную панель приложения в секции прочих выгрузок должна быть добавлена кнопка для совершения данной выгрузки и экспорта ее в текстовый файл. Дополнительно необходимо организовать возможность просмотра всех совершенных когда-либо выгрузок.

### 3.2.11 Алгоритм GAAP FIFO

На основе разработанных требований был составлен алгоритм вычисления GAAP FIFO:

1. Очистить построенное ранее FIFO за CBD.
2. Воспроизвести остатки с предыдущего бизнес-дня (позиции цепочек FIFO) на CBD.
3. Обработать в FIFO операции, увеличивающие балансы ценных бумаг.
   1. Выделить все покупки (в том числе технические) и ручные операции зачисления ценных бумаг, имеющие дату заключения (в случае ручных операций — дату оформления), равную CBD. Операции должны быть обработаны в определенном порядке:
      1. Сначала учитываются операции зачисления ценных бумаг, затем обычные покупки, в последнюю очередь — технические покупки.
      2. Среди операций с одинаковым типом приоритет имеют операции с наименьшим временем заключения сделок.
      3. Среди операций с одинаковым типом и временем заключения сделки сначала используется сделка с минимальной ценой.
   2. Выполнить следующие действия для каждой выбранной операции:
      1. Сформировать новую цепочку, указав следующую информацию:
         1. Уникальный идентификатор, присвоенный СУОЦБ.
         2. Дата образования цепочки, равная CBD.
         3. Ссылка на обрабатываемую операцию.
      2. Включить обрабатываемую операцию в цепочку, указав следующую информации:
         1. Идентификатор цепочки FIFO.
         2. Дата учета, равная CBD.
         3. Ссылка на обрабатываемую операцию.
         4. Тип учета, равный типу обрабатываемой операции.
         5. Количество из обрабатываемой операции.
      3. Инициализировать остаток для новой цепочки FIFO, указав следующую информацию:
         1. Идентификатор цепочки FIFO.
         2. Дата остатка, равная CBD.
         3. Баланс остатка, равный количеству из операции.
4. Проверить остатки на отрицательные значения и назначить для них откупки.
   1. Идентифицировать все отрицательные остатки за CBD. Они должны быть обработаны в определенном порядке:
      1. Приоритет имеют цепочки, образованные операциями с наименьшими датой и временем заключения сделок.
      2. Среди цепочек с одинаковыми датой и временем заключения сделок сначала закрывается цепочка, образованная сделкой с наибольшей ценой.
   2. Выполнить следующие шаги для каждого найденной отрицательной позиции:
      1. Выделить кандидаты в откупки — сделки, образовавшие новые цепочки FIFO в шаге 3, обладающие такой же комбинацией ценной бумаги и категории портфеля, как и соответствующая отрицательная позиция, и имеющие положительный остаток на CBD. Кандидаты в откупки должны выбираться в определенном порядке:
         1. Приоритет имеют кандидаты с наименьшими датой и временем заключения сделок.
         2. Среди сделок с одинаковыми датой и временем заключения сначала используется сделка с минимальной ценой.
      2. Повторять следующие шаги для кандидатов в откупки, пока отрицательная позиция не закрыта:
         1. Принять [Количество откупки] = MIN([Баланс остатка откупки за CBD]; -1 \* [Баланс негативной позиции на CBD]).
         2. Включить продажу в цепочку откупки, указав следующую информацию:
            1. Идентификатор цепочки FIFO — ссылка на цепочку, образованную сделкой-откупкой.
            2. Ссылка на операцию, образовавшую закрываемую отрицательную позицию.
            3. Дата учета, равная CBD.
            4. Тип учета ‘SS’.
            5. Количество, равное -1 \* [Количество откупки].
         3. Уменьшить баланс остатка откупки за CBD на [Количество откупки].
         4. Включить откупку в цепочку негативной позиции, указав следующую информацию:
            1. Идентификатор цепочки FIFO — ссылка на цепочку – негативную позицию.
            2. Ссылка на операцию-откупку.
            3. Дата учета, равная CBD.
            4. Тип учета ‘BB’.
            5. Количество, равное [Количество откупки].
         5. Увеличить баланс остатка негативной позиции за CBD на [Количество откупки].
5. Обработать в FIFO операции, уменьшающие балансы ценных бумаг.
   1. Выделить все продажи (в том числе технические), ручные операции списания ценных бумаг, полные погашения с датой заключения (датой оформления в случае ручных операций, датой исполнения в случае полных погашений), равной CBD. Они должны быть обработаны в определенном порядке:
      1. Сначала учитываются операции списания ценных бумаг, затем обычные продажи, далее технические продажи, и последними — полные погашения.
      2. Среди операций с одинаковым типом приоритет имеют операции с наименьшим временем заключения.
      3. Среди операций с одинаковым типом и временем заключения сделки сначала обрабатывается сделка с наименьшей ценой.
   2. Выполнить следующие шаги для каждой операции:
      1. Принять [Количество на продажу] = [Количество операции].
      2. Определить цепочки с положительными остатками за CBD, образованные сделками с такой же комбинацией ценной бумаги и категории портфеля, как и обрабатываемая операция. Цепочки должны выбираться в определенном порядке:
         1. Приоритет имеют цепочки с наименьшими датой и временем заключения сделок.
         2. Среди цепочек с одинаковыми датой и временем заключения сделок в первую очередь будет использована цепочка с наименьшей ценой сделки.
      3. Повторить следующие шаги для всех найденных цепочек, пока [Количество на продажу] > 0:
         1. Принять [Исполненное количество] = MIN([Количество на продажу], [Баланс найденной цепочки за CBD]).
         2. Включить обрабатываемую операцию в цепочку, указав следующую информацию:
            1. Идентификатор цепочки — ссылка на найденную цепочку.
            2. Ссылка на обрабатываемую операцию.
            3. Дата учета, равная CBD.
            4. Тип учета, равный типу обрабатываемой операции.
            5. Количество, равное -1 \* [Исполненное количество].
         3. Уменьшить баланс остатка найденной цепочки за CBD на [Исполненное количество].
         4. Уменьшить [Количество на продажу] на [Исполненное количество].
      4. Если [Количество на продажу] > 0, то создать отрицательную позицию.
         1. Сформировать новую цепочку, указав следующую информацию:
            1. Уникальный идентификатор, присвоенный СУОЦБ.
            2. Дата образования цепочки, равная CBD.
            3. Ссылка на обрабатываемую операцию.
         2. Включить обрабатываемую операцию в новую цепочку, указав следующую информацию:
            1. Идентификатор цепочки FIFO.
            2. Дата учета, равная CBD.
            3. Ссылка на обрабатываемую операцию.
            4. Тип учета, равный типу обрабатываемой операции.
            5. Количество, равное -1 \* [Количество продажи].
         3. Инициализировать остаток для новой цепочки FIFO, указав следующую информацию:
            1. Идентификатор цепочки FIFO.
            2. Дата остатка, равная CBD.
            3. Баланс остатка, равный -1 \* [Количество продажи].
6. Очистить позиции полностью проданных и закрытых цепочек (удалить остатки за CBD с нулевым балансом).

### 3.2.12 Алгоритм генерации событий

На основе разработанных требований был составлен алгоритм генерации событий:

1. Просмотреть расписание корпоративных событий:
   1. расписание купонных погашений;
   2. расписание амортизаций для амортизируемых ценных бумаг и дату полного погашения неамортизируемых ценных бумаг;
   3. расписание дивидендов.
2. Для ценных бумаг, по которым дата наступления корпоративного события равна CBD, совершить следующие действия:
   1. Создать события в регистре SPLIT сделок.
      1. Выделить остатки GAAP и RAS FIFO на даты среза наступающих событий.
      2. Для ценных бумаг, по которым наступают купонные погашения, сгенерировать события в регистре SPLIT сделок согласно выявленным остаткам в RAS FIFO.
      3. Для ценных бумаг, по которым наступают прочие события, сгенерировать события в регистре SPLIT сделок согласно выявленным остаткам в GAAP FIFO.
   2. Создать события в регистре событий.
      1. Выделить записи с балансами регистра внутреннего учета ценных бумаг на даты среза наступающих событий, для которых указан депозитарий.
      2. Добавить записи в основной регистр событий с указанием депозитария.
      3. Для платежных событий добавить записи в регистр частичных выплат с указанием полной суммы выплаты и платежного счета. Реальная дата оплаты должна быть пуста.
      4. Для неплатежных событий добавить записи в регистр счетов депо событий с указанием количества и счета депо.

Обобщенная блок-схема алгоритма создания событий приведена на рисунке 3.4.



Рисунок 3.4 — Обобщенная блок-схема алгоритма создания событий

## 3.3 Структура программного модуля

Под программным модулем в данном дипломном проекте подразумевается логически завершенный функционал, являющийся относительно независимым для программного продукта и содержащий новые для программного продукта функции.

Данный программный модуль интегрируется в существующую клиент-серверную архитектуру системы. Серверная часть модуля состоит из набора таблиц, хранящих данные, табличных ограничений, обеспечивающих целостность данных, хранимых процедур и функций, реализующих бизнес-логику по обработке информации и вычислению необходимых показателей. Эти объекты встраиваются в базу данных системы и взаимодействуют с уже существующими таблицами. Клиентская часть модуля представляет собой расширение графического интерфейса приложения за счет добавления новых пользовательских компонентов, отвечающих за работу с новыми функциями.

Структура программного модуля приведена на рисунке 3.5.



Рисунок 3.5 — Структура программного модуля

Клиентская часть программного модуля организована по тем же принципам, что и основное приложение. Имеются классы слоя доступа к данным, называемые менеджерами данных (англ. data managers), вызывающие хранимые процедуры базы данных с использованием технологии ADO.NET. Они унаследованы от базовых классов менеджеров данных, реализованных в core-библиотеке заказчика. Менеджеры данных в ходе своей работы возвращают классы-сущности (англ. entities) слоя бизнес-логики. Для них также существуют классы-предки в core-библиотеках. Кроме того, на этом же уровне расположены бизнес-компоненты, служащие для генерации файлов хэндоффов в учетные системы RAS и GAAP. Классы-сущности отображаются на элементах пользовательского интерфейса, относящихся к слою представления. Управляют ими классы-менеджеры пользовательского интерфейса (англ. UI managers, user interface managers). Как компоненты пользовательского интерфейса, так и соответствующие менеджеры также наследуют базовую функциональность от предков из core-библиотек.

Как и в прочих модулях системы, в данном модуле активно используется механизм «Maker-Checker», сводящий риск ошибок во вводимых данных к минимуму. Его суть в следующем. С объектами системы работают по крайней мере два пользователя. Первый пользователь (англ. maker) создает новые объекты, модифицирует уже существующие, удаляет нежелательные и ошибочные объекты. Эти изменения обязательно должны быть тщательно проверены и подтверждены другим пользователем (англ. checker) путем авторизации. Внесший исправления пользователь не может сам же авторизовать их. Пока изменения не были авторизованы, их можно быстро откатить. Неавторизованные объекты либо игнорируются при обработке данных, либо вообще запрещают начать эту обработку. Для реализации этого механизма в базе данных для каждого вида объектов используется пара таблиц: первая — с последними авторизованными параметрами сущностей, вторая (с суффиксом или префиксом PEND) хранит последнее изменения состояния объекта.

В данном программном модуле механизм «Maker-Checker» распространяется на наборы сделок (батчи), корпоративные события и частичные выплаты по платежным событиям.

## 3.4 Проектирование структуры базы данных

Для дополнения системы новым функционалом необходимо добавить в существующую базу данных ряд новых таблиц, которые будут активно взаимодействовать друг с другом и с уже существующими таблицами посредствам внешних ключей, что также обеспечит ссылочную целостность данных.

Ниже представлен перечень добавляемых таблиц и их описание.

* REG\_DEAL\_P\_RAW — таблица, содержащая все сделки в том виде, в каком они были заключены на бирже;
* REG\_DEAL\_P\_SPLIT — таблица, содержащая все сделки в том виде, в каком они исполняются в разрезе банковских портфелей;
* REG\_EVENT\_P — таблица, содержащая данные о корпоративных событиях;
* REG\_EVENT\_P\_PEND — таблица, содержащая данные о корпоративных событиях, которые были изменены / удалены, но еще не подтверждены другим пользователем;
* REG\_EVENT\_LOCATION\_P — таблица, отражающая связь между неплатежными событиями и депозитариями;
* REG\_EVENT\_PARTIAL\_PAYMENT\_P — таблица, содержащая данные о частичных выплатах по корпоративным событиям;
* REG\_EVENT\_PARTIAL\_PAYMENT\_P\_PEND — таблица, содержащая данные о частичных выплатах по корпоративным событиям, которые были изменены / удалены, но еще не подтверждены другим пользователем;
* GAAP.SCRIPTS\_KEPT\_P — таблица, содержащая все цепочки GAAP FIFO;
* GAAP.SCRIPTS\_HIST\_P — таблица, содержащая остатки по цепочкам GAAP FIFO;
* GAAP.SETTLEMENTS\_P — таблица, содержащая информацию об исполнении сделок из конкретных цепочек GAAP FIFO;
* RAS.BSDEAL\_P — таблица, содержащая сформированные BSDEAL-сделки, используемые в RAS FIFO и передаваемые в выгрузку в УС RAS;
* RAS.FIN\_SK\_P — таблица, содержащая все цепочки RAS FIFO;
* RAS.FIN\_SH\_P — таблица, содержащая остатки по цепочкам RAS FIFO;
* RAS.FIN\_SETTLEMENT\_P — таблица, содержащая информацию об исполнении сделок из конкретных цепочек RAS FIFO;
* GAAP.IBBS\_DISP — таблица, содержащая данные о выгрузках в УС GAAP;
* GAAP.IBBS\_HOFF — таблица, содержащая проводки, выгруженные в УС GAAP;
* RAS.CITI9\_DISP — таблица, содержащая данные о выгрузках в УС RAS;
* RAS.CITI9\_DISP\_BSINT — таблица, содержащая проводки блока BSINT выгрузок в УС RAS;
* RAS.CITI9\_DISP\_BSAMRT — таблица, содержащая проводки блока BSAMRT выгрузок в УС RAS;
* RAS.CITI9\_DISP\_BSDEAL — таблица, содержащая проводки блока BSDEAL выгрузок в УС RAS.

Спецификация полей таблиц базы данных приведена в таблицах 3.1-3.20. Столбец «Ключ» показывает, является ли поле первичным (PK) или внешним (FK) ключом таблицы.

Таблица 3.1 — Список полей таблицы REG\_DEAL\_P\_RAW

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| ID | int | Идентификатор | PK |
| Operation | nvarchar(2) | Тип сделки |  |
| BatchID | int | Идентификатор батча | FK |
| OrderNo | int | Порядковый номер сделки в разрезе батча |  |
| ScriptID | int | Идентификатор ценной бумаги | FK |
| TradingCenterID | int | Идентификатор торгового центра | FK |
| DealerID | int | Идентификатор дилера | FK |
| CpID | int | Идентификатор контрагента | FK |
| Quantity | int | Количество |  |
| TicketID | nvarchar(50) | Символьный идентификатор сделки |  |
| PricePcent | decimal(17,9) | Цена сделки в % |  |
| PriceCcy | decimal(17,9) | Цена сделки в валюте |  |
| CcyOfPrice | nchar(3) | Валюта, в которой указана цена |  |
| CcyOfPayment | nchar(3) | Валюта, которой производится оплата |  |
| Auction | bit | Признак аукциона |  |
| RepoID | int | Идентификатор параметров РЕПО сделки | FK |
| IsRepo | bit | Вычислеямое поле. Признак РЕПО сделки |  |
| CouponAmount | decimal(12,2) | Накопленный купонный доход в валюте ценной бумаги |  |
| DealValue | decimal(38,7) | Сумма сделки в валюте бумаги |  |
| Commission | decimal(12,2) | Комиссия биржи в валюте ценной бумаги |  |
| DealValueIncludingCouponRUB | decimal(38,7) | Сумма сделки и накопленный купонный доход, выраженные в рублях |  |
| CouponPaymentCcy | decimal(38,7) | Накопленный купонный доход в валюте оплаты |  |
| DealValuePaymentCcy | decimal(38,7) | Сумма сделки в валюте оплаты |  |
| CommissionPaymentCcy | decimal(38,7) | Комиссия биржи в валюте оплаты |  |
| TradeDT | datetime | Дата заключения сделки |  |

Продолжение таблицы 3.1

|  |  |  |  |
| --- | --- | --- | --- |
| TradeTime | datetime | Время заключения сделки |  |
| TcPlanPaymentDT | datetime | Планируемая дата оплаты комиссии биржи |  |
| TcRealPaymentDT | datetime | Реальная дата оплаты комиссии биржи |  |
| PlanSettlementDT | datetime | Планируемая дата исполнения сделки |  |
| RealSettlementDT | datetime | Реальная дата исполнения сделки |  |
| PlanPaymentDT | datetime | Планируемая дата оплаты |  |
| RealPaymentDT | datetime | Реальная дата оплаты |  |
| DepoReceiptProperties | nvarchar(50) | Название документа, подтверждающего исполнение сделки |  |
| PaymentOrderProperties | nvarchar(50) | Название документа, подтверждающего оплату сделки |  |
| Status | smallint | Статус сделки |  |
| ExchangeID | nvarchar(20) | Идентификатор сделки на бирже |  |
| DepoLocationID | int | Идентификатор депозитария | FK |
| LocationID | int | Идентификатор счета депо | FK |
| NostroAccountID | int | Идентификатор платежного счета | FK |
| CommissionAccountID | int | Идентификатор счета комиссии | FK |
| AirCashAccountID | int | Идентификатор искусственного денежного счета, необходимо для нужд внутреннего учета денежных средств | FK |
| AirSecuritiesAccountID | int | Идентификатор искусственного счета депо, необходимо для нужд внутреннего учета ценных бумаг | FK |
| SettleModeID | int | Идентификатор режима исполнения | FK |

Таблица 3.2 — Список полей таблицы REG\_DEAL\_P\_SPLIT

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| ID | int | Идентификатор | PK |
| RawID | int | Идентификатор RAW сделки | FK |
| Operation | nvarchar(3) | Тип сделки |  |
| BatchID | int | Идентификатор батча | FK |
| OrderNo | int | Порядковый номер сделки в разрезе батча |  |
| ScriptID | int | Идентификатор ценной бумаги | FK |
| TradingCenterID | int | Идентификатор торгового центра | FK |
| DealerID | int | Идентификатор дилера | FK |
| CpID | int | Идентификатор контрагента | FK |
| PortfolioID | int | Идентификатор портфеля | FK |
| Quantity | int | Количество |  |
| TicketID | nvarchar(50) | Символьный идентификатор сделки |  |
| PricePcent | decimal(17,9) | Цена сделки в % |  |
| PriceCcy | decimal(17,9) | Цена сделки в валюте |  |
| CcyOfPrice | nchar(3) | Валюта, в которой указана цена |  |
| CcyOfPayment | nchar(3) | Валюта, которой производится оплата |  |
| Auction | bit | Признак аукциона |  |
| RepoID | int | Идентификатор параметров РЕПО сделки | FK |
| IsRepo | bit | Вычислеямое поле. Признак РЕПО сделки |  |
| CouponAmount | decimal(12,2) | Накопленный купонный доход в валюте ценной бумаги |  |
| DealValue | decimal(38,7) | Сумма сделки в валюте бумаги |  |
| Commission | decimal(12,2) | Комиссия биржи в валюте ценной бумаги |  |
| DealValueIncludingCouponRUB | decimal(38,7) | Сумма сделки и накопленный купонный доход, выраженные в рублях |  |
| CouponPaymentCcy | decimal(38,7) | Накопленный купонный доход в валюте оплаты |  |
| DealValuePaymentCcy | decimal(38,7) | Сумма сделки в валюте оплаты |  |

Продолжение таблицы 3.2

|  |  |  |  |
| --- | --- | --- | --- |
| CommissionPaymentCcy | decimal(38,7) | Комиссия биржи в валюте оплаты |  |
| TradeDT | datetime | Дата заключения сделки |  |
| TradeTime | datetime | Время заключения сделки |  |
| TcPlanPaymentDT | datetime | Планируемая дата оплаты комиссии биржи |  |
| TcRealPaymentDT | datetime | Реальная дата оплаты комиссии биржи |  |
| PlanSettlementDT | datetime | Планируемая дата исполнения сделки |  |
| RealSettlementDT | datetime | Реальная дата исполнения сделки |  |
| PlanPaymentDT | datetime | Планируемая дата оплаты |  |
| RealPaymentDT | datetime | Реальная дата оплаты |  |
| DepoReceiptProperties | nvarchar(50) | Название документа, подтверждающего исполнение сделки |  |
| PaymentOrderProperties | nvarchar(50) | Название документа, подтверждающего оплату сделки |  |
| ParentDealID | int | Идентификатор родительской сделки | FK |
| EntryDT | datetime | Дата заведения сделки (бизнес-день) |  |
| Status | smallint | Статус сделки |  |
| IsTechnical | bit | Признак технической сделки |  |
| PortfolioCat | varchar(20) | Категория портфеля сделки |  |

Таблица 3.3 — Список полей таблицы REG\_EVENT\_P

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| ID | int | Идентификатор | PK |
| Type | nvarchar(3) | Тип корпоративного события |  |
| ScriptID | int | Идентификатор бумаги | FK |
| DepoLocationID | int | Идентификатор депозитария | FK |
| NostroAccountID | int | Идентификатор платежного счета | FK |
| PerPiece | decimal(15,9) | Выплата за единицу ценной бумаги |  |
| PlanDT | datetime | Планируемая дата события |  |
| RealDT | datetime | Реальная дата события |  |
| EntryDT | datetime | Дата заведения события (бизнес-день) |  |
| Properties | nvarchar(50) | Название документа, подтверждающего событие |  |
| EventNo | nvarchar(50) | Символьный идентификатор события |  |
| LastChangerName | nvarchar(50) | Имя пользователя, внесшего последние изменения |  |
| LastChangeDT | datetime | Дата и время внесения последних изменений |  |
| LastAuthName | nvarchar(50) | Имя пользователя, подтвердившего последние изменения |  |
| LastAuthDT | datetime | Дата и время подтверждения последних изменений |  |
| InputerName | nvarchar(50) | Имя пользователя, заведшего событие |  |
| InputDT | datetime | Дата и время заведения события (реальные) |  |
| Status | smallint | Статус события |  |
| IsPayment | bit | Флаг, указывающий, является ли событие платежным |  |
| PlanPaymentDT | datetime | Вычисляемое поле. Планируемая дата выплаты для платежных событий |  |
| PlanSettlementDT | datetime | Вычисляемое поле. Планируемая дата исполнения для неплатежных событий |  |
| RealSettlementDT | datetime | Вычисляемое поле. Реальная дата исполнения для неплатежных событий |  |

Продолжение таблицы 3.3

|  |  |  |  |
| --- | --- | --- | --- |
| DepoReceiptProperties | nvarchar(50) | Вычисляемое поле. Название документа, подтверждающего исполнение |  |

Таблица 3.4 — Список полей таблицы REG\_EVENT\_P\_PEND

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| ID | int | Идентификатор | PK |
| NostroAccountID | int | Идентификатор платежного счета | FK |
| RealDT | datetime | Реальная дата события |  |
| Properties | nvarchar(50) | Название документа, подтверждающего событие |  |
| LastChangerName | nvarchar(50) | Имя пользователя, внесшего последние изменения |  |
| LastChangeDT | datetime | Дата и время внесения последних изменений |  |
| LastAuthName | nvarchar(50) | Имя пользователя, подтвердившего последние изменения |  |
| LastAuthDT | datetime | Дата и время подтверждения последних изменений |  |
| InputerName | nvarchar(50) | Имя пользователя, заведшего событие |  |
| InputDT | datetime | Дата и время заведения события |  |
| Status | smallint | Статус неподтвержденного события |  |

Таблица 3.5 — Список полей таблицы REG\_EVENT\_LOCATION\_P

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| ID | int | Идентификатор | PK |
| EventID | int | Идентификатор события | FK |
| LocationID | int | Идентификатор счета депо | FK |
| AirSecuritiesAccountID | int | Идентификатор искусственного счета депо, необходимо для нужд внутреннего учета ценных бумаг | FK |
| Quantity | int | Количество |  |

Таблица 3.6 — Список полей таблицы REG\_EVENT\_PARTIAL\_PAYMENT\_P

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| ID | int | Идентификатор |  |
| EventID | int | Идентификатор события | FK |
| RealPaymentDT | datetime | Реальная дата частичной выплаты |  |
| Amount | decimal(28,7) | Сумма частичной выплаты |  |
| LastChangerName | nvarchar(50) | Имя пользователя, внесшего последние изменения |  |
| LastChangeDT | datetime | Дата и время внесения последних изменений |  |
| LastAuthName | nvarchar(50) | Имя пользователя, подтвердившего последние изменения |  |
| LastAuthDT | datetime | Дата и время подтверждения последних изменений |  |
| InputerName | nvarchar(50) | Имя пользователя, заведшего выплату |  |
| InputDT | datetime | Дата и время заведения выплаты |  |
| PaymentOrderProperties | nvarchar(50) | Название документа, подтверждающего выплату |  |

Таблица 3.7 — Список полей таблицы REG\_EVENT\_PARTIAL\_PAYMENT\_P\_PEND

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| ID | int | Идентификатор |  |
| EventID | int | Идентификатор события | FK |
| RealPaymentDT | datetime | Реальная дата частичной выплаты |  |
| Amount | decimal(28,7) | Сумма частичной выплаты |  |
| Action | tinyint | Статус неподтвержденной записи |  |
| LastChangerName | nvarchar(50) | Имя пользователя, внесшего последние изменения |  |
| LastChangeDT | datetime | Дата и время внесения последних изменений |  |
| LastAuthName | nvarchar(50) | Имя пользователя, подтвердившего последние изменения |  |
| LastAuthDT | datetime | Дата и время подтверждения последних изменений |  |
| InputerName | nvarchar(50) | Имя пользователя, заведшего выплату |  |
| InputDT | datetime | Дата и время заведения выплаты |  |
| PaymentOrderProperties | nvarchar(50) | Название документа, подтверждающего выплату |  |

Таблица 3.8 — Список полей таблицы GAAP.SCRIPTS\_KEPT\_P

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| ID | int | Идентификатор | PK |
| DealID | int | Идентификатор SPLIT сделки, образовавшей цепочку | FK |
| EntryDT | dateime | Дата образования цепочки |  |

Таблица 3.9 — Список полей таблицы GAAP.SCRIPTSHIST\_\_P

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| SKept\_ID | int | Идентификатор цепочки | PK,FK |
| DT | datetime | Дата остатков | PK |
| Quanttiy | int | Количество остатков |  |

Таблица 3.10 — Список полей таблицы GAAP.SETTLEMENTS\_P

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| SKept\_ID | int | Идентификатор цепочки | PK,FK |
| DealID | int | Идентификатор SPLIT сделки, относящейся к цепочке | PK,FK |
| Quantity | int | Количество сделки |  |
| Type | varchar(10) | Тип учета сделки |  |
| EntryDT | datetime | Дата учета сделки |  |

Таблица 3.11 — Список полей таблицы RAS.BSDEAL\_P

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| ID | int | Идентификатор | PK |
| DispatchDate | datetime | Дата генерации сделки |  |
| SplitDealID | int | Идентификатор SPLIT сделки | FK |
| LinkedDealID | int | Идентификатор связанной сделки | FK |
| ParentID | int | Идентификатор родительской сделки | FK |
| OrderNo | int | Порядковый номер сделки в пределах дня |  |
| dealNo | varchar(20) | Вычисляемое поле. Символьный идентификатор сделки в УС RAS |  |
| Operation | char(2) | Символьный код операции |  |
| ScriptID | int | Идентификатор ценной бумаги | FK |
| Quantity | int | Количество |  |
| PortfolioID | int | Идентификатор портфеля | FK |
| CpID | int | Идентификатор контрагента | FK |
| PortfolioCat | varchar(20) | Категория портфеля |  |
| SymbolsPortCat | char(2) | Код категории портфеля в УС RAS |  |
| BookDT | datetime | Дата заведения сделки |  |
| SettleDT | datetime | Планируемая дата исполнения сделки |  |
| AGGBookDT | datetime | Дата заключения сделки |  |
| AGGSettleDT | datetime | Реальная дата исполнения сделки |  |
| InOut | char(1) | Символьный признак типа операции (покупка / продажа) |  |
| ProductType | varchar(3) | Символьный код типа ценной бумаги |  |
| PricePcent | float | Цена сделки |  |
| FaceValue | decimal(15,2) | Сумма сделки без учета цены |  |
| PrincipalValue | decimal(15,2) | Сумма сделки с учетом цены |  |
| CouponAmount | decimal(12,2) | Накопленный купонный доход |  |
| MiscCost | decimal(12,2) | Комиссия биржи в валюте ценной бумаги |  |
| MiscCost5 | decimal(12,2) | Комиссия биржи в валюте счета |  |
| CashAcctNo | decimal(10,0) | Номер платежного счета |  |
| Depo | varchar(10) | Код депозитария |  |
| RepoID | int | Идентификатор параметров РЕПО сделки | FK |
| TradingCenterID | int | Идентификатор торгового центра | FK |
| IsCorrective | bit | Признак корректирующей проводки |  |
| IsTechnical | bit | Признак технической сделки |  |

Продолжение таблицы 3.11

|  |  |  |  |
| --- | --- | --- | --- |
| TechnicalLinkedDealID | int | Идентификатор связанной технической сделки | FK |
| SettleDT2 | datetime | Дата исполнения 2-ой ноги РЕПО сделки |  |
| TerminateFlag | nchar(1) | Символьный признак досрочного исполнения РЕПО |  |
| DRDealID | int | Идентификатор сделки прямого РЕПО из обратного | FK |

Таблица 3.12 — Список полей таблицы RAS.FIN\_SK\_P

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| ID | int | Идентификатор | PK |
| DealID | int | Идентификатор BSDEAL сделки, образовавшей цепочку | FK |
| CBR\_Portfolio | varchar(50) | Код категории портфеля в УС RAS |  |
| PurchaseDT | datetime | Дата образования цепочки |  |
| RepoID | int | Идентификатор параметров РЕПО сделки | FK |

Таблица 3.13 — Список полей таблицы RAS.FIN\_SH\_P

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| FinSkID | int | Идентификатор цепочки | PK |
| DT | datetime | Дата остатков | PK |
| Quantity | int | Общее количество остатков |  |
| RepoQuantity | int | Количество остатков, находящееся в прямом РЕПО |  |
| SHSQuantity | int | Количество остатков, находящееся в короткой продаже |  |

Таблица 3.14 — Список полей таблицы RAS.FIN\_SETTLEMENT\_P

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| FinSkID | int | Идентификатор цепочки | PK,FK |
| DealID | int | Идентификатор BSDEAL сделки, относящейся к цепочке | PK,FK |
| Quantity | int | Количество сделки |  |
| EntryDT | datetime | Дата учета сделки |  |
| Type | varchar(10) | Тип учета сделки |  |

Таблица 3.15 — Список полей таблицы GAAP.IBBS\_DISP

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| ID | int | Идентификатор | PK |
| ValueDT | datetime | Бизнес-день, за который совершается выгрузка | FK |
| DT | datetime | Дата и время выгрузки |  |
| UserName | varchar(30) | Имя пользователя, совершившего выгрузку |  |

Таблица 3.16 — Список полей таблицы GAAP.IBBS\_HOFF

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| DispID | int | Идентификатор выгрузки | FK |
| Acct | decimal(10,0) | Номер счета |  |
| AcctDescr | varchar(30) | Описание счета |  |
| Amount | decimal(17,4) | Сумма проводки |  |
| Ccy | char(3) | Валюта проводки |  |
| DR\_CR | bit | Дебит / кредит |  |
| Detail | varchar(50) | Описание проводки |  |
| PortfolioID | int | Идентификатор портфеля |  |
| SplitDealID | int | Идентификатор SPLIT сделки |  |
| ExchangeID | nvarchar(20) | Идентификатор сделки на бирже |  |
| RawDealID | int | Идентификатор RAW сделки |  |

Таблица 3.17 — Список полей таблицы RAS.CITI9\_DISP

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| ID | int | Идентификатор | PK |
| ValueDate | datetime | Бизнес-день, за который совершается выгрузка |  |
| DT | datetime | Дата и время выгрузки |  |
| UserName | varchar(30) | Имя пользователя, совершившего выгрузку |  |

Таблица 3.18 — Список полей таблицы RAS.CITI9\_DISP\_BSINT

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| DispID | int | Идентификатор выгрузки | FK |
| ISIN | varchar(20) | Символьный идентификатор ценной бумаги |  |
| ProductType | varchar(50) | Символьный код типа ценной бумаги |  |
| TBOOK | varchar(50) | Код категории портфеля остатка в УС RAS |  |
| VALUEDATE | datetime | Дата остатка |  |
| AI | decimal(28,7) | Общий накопленный купонный доход остатка |  |
| AI\_PER\_PIECE | decimal(33,12) | Накопленный купонный доход на единицу остатка |  |
| TRADE\_NO | varchar(20) | DealNo сделки, образовавшей остаток |  |

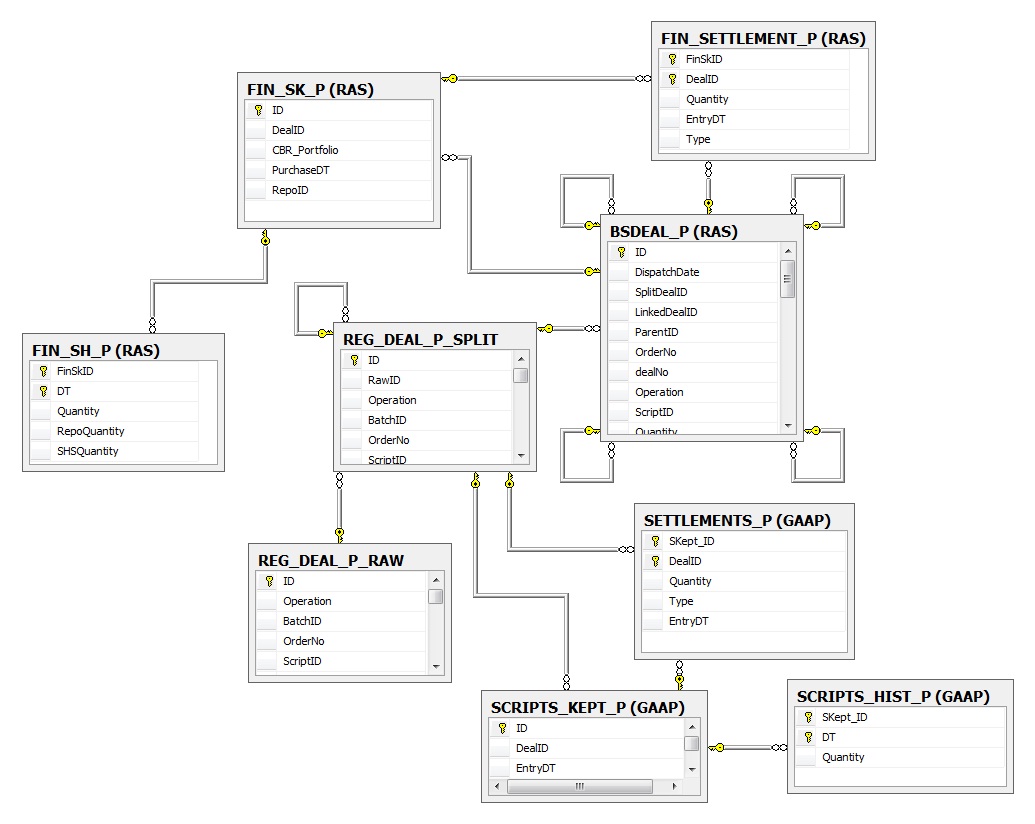
Таблица 3.19 — Список полей таблицы RAS.CITI9\_DISP\_BSAMRT

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| DispID | int | Идентификатор выгрузки | FK |
| ISIN | varchar(20) | Символьный идентификатор ценной бумаги |  |
| ProductType | varchar(50) | Символьный код типа ценной бумаги |  |
| TBOOK | varchar(50) | Код категории портфеля остатка в УС RAS |  |
| AMORT\_NO | int | Порядковый номер амортизации |  |
| VALUEDATE | datetime | Дата остатка |  |
| AMOUNT | decimal(28,8) | Общая сумма амортизации остатка |  |
| AMOUNT\_PER\_PIECE | decimal(20,8) | Сумма амортизации на единицу остатка |  |

Таблица 3.20 — Список полей таблицы RAS.CITI9\_DISP\_BSDEAL

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| DispID | int | Идентификатор выгрузки | FK |
| DispatchDate | datetime | Дата генерации сделки |  |
| DealNo | varchar(20) | Символьный идентификатор сделки в УС RAS |  |
| Operation | char(2) | Символьный код операции |  |
| SymbolsPortCat | char(2) | Код категории портфеля в УС RAS |  |
| BookDT | datetime | Дата заведения сделки |  |
| SettleDT | datetime | Дата исполнения сделки |  |
| InOut | char(1) | Символьный признак типа операции (покупка / продажа) |  |
| BaseNo | decimal(7,0) | Цифровой идентификатор контрагента |  |
| ProductType | varchar(3) | Символьный код типа ценной бумаги |  |
| ISIN | varchar(20) | Символьный идентификатор ценной бумаги |  |
| PricePcent | float | Цена сделки |  |
| FaceValue | decimal(15,2) | Сумма сделки без учета цены |  |
| PrincipalValue | decimal(15,2) | Сумма сделки с учетом цены |  |
| CouponAmount | decimal(12,2) | Накопленный купонный доход |  |
| MiscCost | decimal(12,2) | Комиссия биржи в валюте ценной бумаги |  |
| CashAcctNo | decimal(10,0) | Номер платежного счета |  |
| Depo | varchar(10) | Код депозитария |  |
| RepoID | int | Идентификатор параметров РЕПО сделки |  |
| LinkedDealNo | varchar(20) | DealNo связанной сделки |  |
| Quantity | int | Количество |  |
| ScriptID | int | Идентификатор ценной бумаги |  |
| PortfolioCat | varchar(20) | Категория портфеля |  |
| MiscCost5 | decimal(12,2) | Комиссия биржи в валюте счета |  |
| IsTechnical | bit | Признак технической сделки |  |
| TechnicalLinkedDealNo | varchar(20) | DealNo связанной технической сделки |  |
| ForwardDealNo | varchar(20) | DealNo срочной сделки |  |
| SettleDT2 | datetime | Дата исполнения 2-ой ноги РЕПО сделки |  |
| TerminateFlag | nchar(1) | Символьный признак досрочного исполнения РЕПО |  |
| DRDealNo | varchar(20) | DealNo сделки прямого РЕПО из обратного |  |

Схема базы данных, показывающая взаимосвязи добавляемых таблиц, представлена на рисунке 3.6.



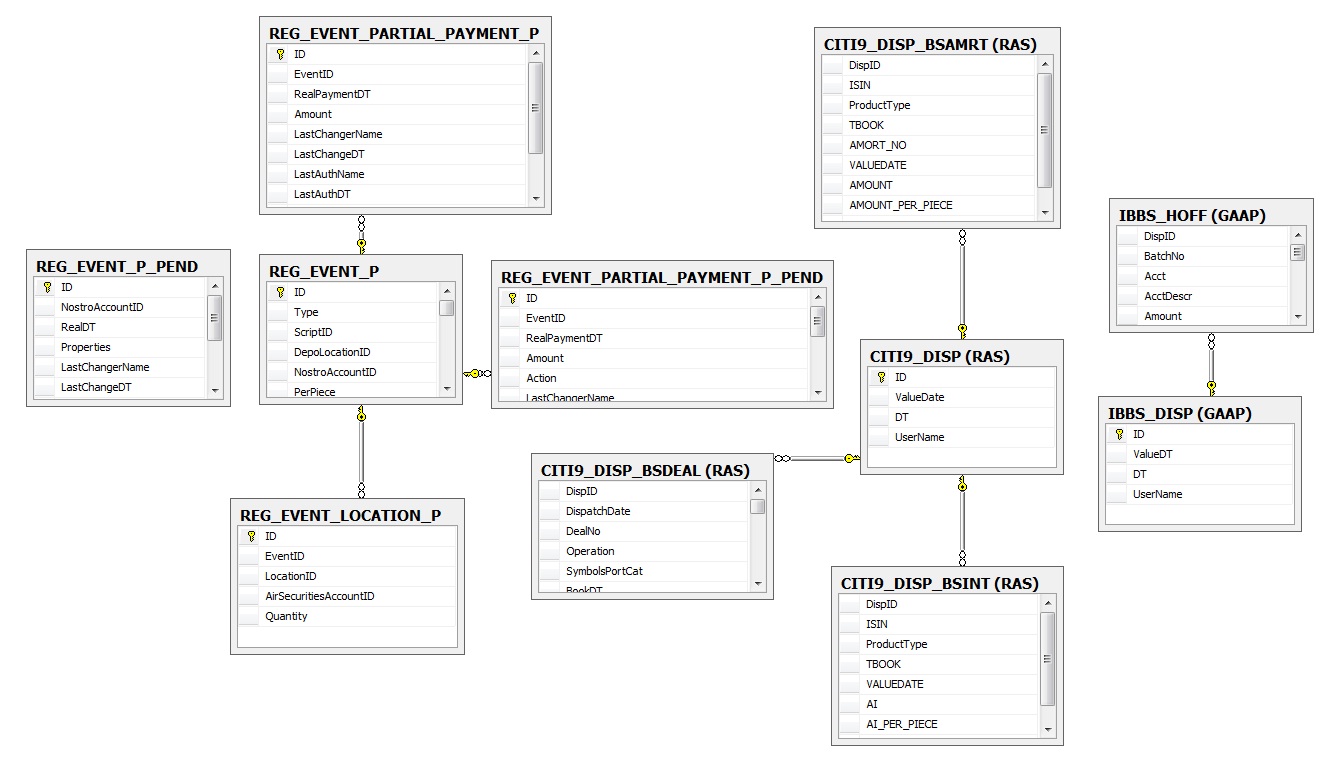


Рисунок 3.6 — Добавляемые таблицы и связи между ними

Для реализации разработанных требований, помимо создания новых таблиц, были добавлены новые поля в ряд уже существующих таблиц. Список таблиц приведен ниже.

* BOOK\_P\_CASH\_ENTR — регистр внутреннего учета, содержащий все проводки по денежным средствам;
* BOOK\_P\_SECURITY\_ENTR — регистр внутреннего учета, содержащий все проводки по ценным бумагам.
* REG\_SECURITY\_OPERATION\_P — таблица, содержащая данные о ручных операциях с ценными бумагами;
* REG\_SECURITY\_OPERATION\_P\_PEND — таблица, содержащая данные о ручных операциях с ценными бумагами, которые были изменены / удалены, но еще не подтверждены другим пользователем;
* DAILY — таблица, содержащая список бизнес-дней, заведенных в системе;
* PEND\_DAILY — таблица, содержащая список бизнес-дней, заведенных в системе, которые были изменены/удалены, но еще не подтверждены другим пользователем.

Спецификация полей таблиц базы данных приведена в таблицах 3.21-3.26. Столбец «Ключ» показывает, является ли поле первичным (PK) или внешним (FK) ключом таблицы.

Таблица 3.21 — Список добавляемых полей таблицы BOOK\_P\_CASH\_ENTR

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| DealID | int | Идентификатор сделки | FK |
| EventPartialPaymentID | int | Идентификатор частичной выплаты по платежному событию | FK |

Таблица 3.22 — Список добавляемых полей таблицы BOOK\_P\_SECURITY\_ENTR

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| DealID | int | Идентификатор сделки | FK |
| EventLocationID | int | Идентификатор связи неплатежного события со счетом депо | FK |

Таблица 3.23 — Список добавляемых полей таблицы REG\_SECURITY\_OPERATION\_P

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| SplitID | int | Идентификатор SPLIT сделки | FK |

Таблица 3.24 — Список добавляемых полей таблицы REG\_SECURITY\_OPERATION\_P\_PEND

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| SplitID | int | Идентификатор SPLIT сделки | FK |

Таблица 3.25 — Список добавляемых полей таблицы DAILY

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| NeedApplyP | bit | Признак необходимости проверки регистров внутреннего учета на отрицательные балансы |  |
| NeedApplyPNoOverdraft | bit | Признак необходимости применения торговли |  |
| ClosedP | bit | Признак закрытия дня |  |

Таблица 3.26 — Список добавляемых полей таблицы PEND\_DAILY

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | Описание | Ключ |
| NeedApplyP | bit | Признак необходимости проверки регистров внутреннего учета на отрицательные балансы |  |
| NeedApplyPNoOverdraft | bit | Признак необходимости применения торговли |  |
| ClosedP | bit | Признак закрытия дня |  |

# 4 Разработка программного модуля

## 4.1 Разработка базы данных

Для реализации требований, разработанных в п. 3.2, был создан ряд хранимых процедур. Чтобы облегчить дальнейшую поддержку написанного кода, было принято решение разбить большие хранимые процедуры (объемом более 500 строк) на несколько логически завершенных частей. При этом создается процедура-«обертка», выполняющая последовательный вызов дочерних процедур в нужном порядке, а также производящая требуемые предварительные вычисления.

Перечень наиболее важных разработанных хранимых процедур с описанием приведен ниже. Часть из них служит для выполнения обработки информации в базе данных, остальные — для взаимодействия с пользователем.

1. Манипуляции со сделками:
   1. GetProprietaryRawDealByID — выдает параметры RAW сделки по ее идентификатору.
   2. GetProprietaryRawDealsBatch — выводит список RAW сделок по идентификатору батча.
   3. GetProprietaryRawDealsByTradeDT — выводит список RAW сделок по дате заключения.
   4. GetProprietarySplitDealByID — выдает параметры SPLIT сделки по ее идентификатору.
   5. GetProprietarySplitDealsBatch — выводит список SPLIT сделок по идентификатору батча.
   6. GetProprietarySplitDealsByTradeDT — выводит список SPLIT сделок по дате заключения.
   7. InsertProprietaryDeal — добавляет новую сделку.
   8. UpdateProprietaryDeal — модифицирует параметров сделки.
   9. DeleteProprietaryDeal — удаляет сделку.
   10. AutoAssignPortfolios — автоматически назначает портфели SPLIT сделкам в батче.
2. Импорт сделок из трейдинговой системы:
   1. ImportProprietaryDealFIS — выполняет импорт сделок из трейдинговой системы, преобразуя данные в XML.
   2. ImportProprietaryDealXML — импортирует сделки, информация о которых содержится во входном XML.
3. Генерация корпоративных событий:
   1. DeleteEventProprietary — удаляет сгенерированные корпоративные события из регистра событий и сделок.
   2. GenerateEventProprietary — создает события с регистрах событий и сделок.
4. Манипуляции с корпоративными событиями:
   1. GetProprietaryEventByID — выдает список параметров события по его идентификатору.
   2. GetProprietaryEventList — выводит список корпоративных событий с применением фильтров по дате и типу.
   3. UpdateProprietaryEvent — выполняет модификацию события.
   4. CommitProprietaryEvent — переводит событие в подтвержденный статус.
   5. UncommitProprietaryEvent — отменяет подтверждение события.
   6. DropProprietaryEvent — отменяет модификацию события.
   7. AuthorizeProprietaryEvent — выполняет авторизацию события.
   8. GetEventPartialPaymentByID — выдает параметры выплаты по ее идентификатору.
   9. GetEventPartialPaymentList — выводит список выплат по событию.
   10. InsertEventPartialPayment — добавляет новую выплату.
   11. UpdateEventPartialPayment — обновляет параметры выплаты.
   12. DeleteEventPartialPayment — удаляет частичную выплату.
   13. AuthorizeEventPartialPayment — выполняет авторизацию частичной выплаты.
   14. GetProprietaryEventLocationList — выводит список счетов депо для неплатежного события.
5. Построение GAAP FIFO:
   1. GAAP.ComputeFIFO — вычисляет GAAP FIFO за определенную дату.
6. Построение RAS FIFO:
   1. RAS.GenerateBSDeal — генерирует BSDEAL-сделки за определенную дату.
   2. RAS.ComputeFIFO — вычисляет RAS FIFO за определенную дату.
7. Применение торговли:
   1. ApplyProprietaryTrading — выполняет полную процедуру применения торговли.
8. Проверка регистров внутреннего учета на отрицательные балансы:
   1. CheckProprietaryCashOverdrafts — выполняет проверку остатков по денежным средствам.
   2. CheckProprietarySecurityOverdrafts — выполняет проверку остатков по ценным бумагам.
9. Выгрузка в УС GAAP:
   1. GAAP.GenerateFlexHoffProprietary — генерирует выгрузку в базе данных.
   2. GAAP.GetFlexHoffProprietary — выводит список всех выгрузок.
   3. GAAP.GetFlexHoffProprietaryByDispID — выдает данные выгрузки по ее идентификатору.
10. Выгрузка в УС RAS:
    1. RAS.GenerateCiti9Handoff — генерирует выгрузку в базе данных (заполняет все таблицы, связанные с выгрузкой).
    2. RAS.GetCiti9Disp — выводит список всех выгрузок.
    3. RAS.GetCiti9HandoffData — выдает данные выгрузки по ее идентификатору.
11. Формирование отчетности:
    1. RptRasDisclosureReport — генерирует отчет об оборотах ценных для сотрудников банка, основанный на GAAP FIFO.
    2. RptGaapDisclosureReport — генерирует отчет об оборотах ценных для сотрудников банка, основанный на RAS FIFO.
    3. RptFinconMTM — генерирует отчет для регулирующих органов, основанный на RAS FIFO.

Кроме того, ряд существующих хранимых процедур был модифицирован для поддержки новых требований. Самые важные изменения коснулись процедур, связанных с построением проводок и остатков регистров внутреннего учета и отображением сгенерированных проводок:

1. Построение внутреннего учета:
   1. GenerateBookPCashEntr — формирует проводки в регистрах учета денежных средств.
   2. GenerateBookPCashHist — формирует остатки в регистрах учета денежных средств.
   3. GenerateBookPSecurityEntr — формирует проводки в регистрах учета ценных бумаг.
   4. GenerateBookPSecurityHist — формирует остатки в регистрах учета ценных бумаг.
2. Отображение проводок регистров внутреннего учета:
   1. GetProprietaryCashOperationList — возвращает список всех проводок по денежным средствам.
   2. GetProprietarySecurityOperationList — выдает список всех проводок по ценным бумагам.

## 4.2 Разработка пользовательского интерфейса

Для работы с новыми функциональными возможностями в клиентское приложение были добавлены две новые панели с заголовками «Trade batches (P)» и «Events (P)». Доступ к ним осуществляется либо через меню приложения, либо через дерево в левой части окна (см. рисунок 4.1).

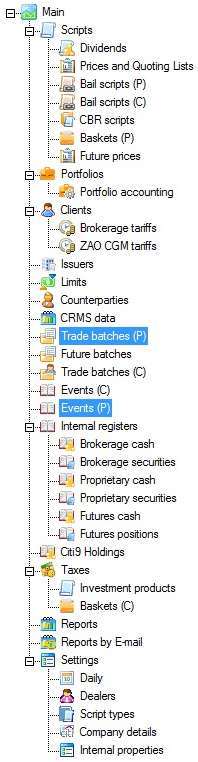


Рисунок 4.1 — Новые узлы в дереве панелей приложения

Для отображения новых элементов и их составляющих были разработаны классы компонентов пользовательского интерфейса. При этом активно использовались базовые классы как из core-библиотек заказчика, так и написанные непосредственно для данной системы.

Панель узла «Trade batches (P)» представлена классом ProprietaryTradingPanel, унаследованным от базового класса TabPanel. Эта панель состоит из 5 вкладок:

1. Вкладка «Batches» состоит из двух частей. В верхней ее половине таблица со списком батчей за текущий бизнес-день, в нижней — таблица со списком сделок, входящих в конкретный батч верхней таблицы. Соответствующий ей класс ProprietaryBatchesPanel унаследован от базового MainFormPanel.
2. Вкладка «Raw Deals» отображает список RAW сделок в виде таблицы. Возможно применений фильтров по дате заключения. Соответствующий ей класс ProprietaryRawDealsPanel унаследован от BaseFormPanel.
3. Вкладка «Split Deals» отображает список SPLIT сделок в виде таблицы. Возможно применений фильтров по дате заключения. Соответствующий ей класс SplitDealsPanel унаследован от BaseFormPanel.
4. Вкладка «Cash Operations» предназначена для работы с ручными операциями с денежными средствами. Данная панель была разработана в рамках предыдущих доработок системы и в ходе дипломного проекта была лишь перемещена в узел «Trade batches (P)».
5. Вкладка «Securities Operations» предназначена для работы с ручными операциями с ценными бумагами. Данная панель была разработана в рамках предыдущих доработок системы и в ходе дипломного проекта была лишь перемещена в узел «Trade batches (P)».

При выполнении на вкладке «Batches» команды создания нового батча либо открытия существующего появляется форма класса ProprietaryBatchForm, предназначенная для просмотра, создания, удаления и редактирования сделок, входящих в данный батч. Редактирование осуществляется посредством формы класса ProprietaryTransactionForm, унаследованной от предка BaseAuthorizableForm. Эта же форма используется для просмотра RAW и SPLIT сделок на соответствующих вкладках узла «Trade batches (P)».

Панель узла «Events (P)» представлена классом EventTabPanel, унаследованным от базового класса CoreTabPanel. Эта панель состоит из 2 вкладок:

1. Вкладка «Payment» (класс EventPaymentPanelEx) предназначена для работы с платежными событиями. В верхней ее части расположена таблица со списком корпоративных событий. Существует возможность применить фильтр по дате и типу события. В нижней части — таблица со списком частичных выплат, относящихся к конкретному событию из верхней таблицы.
2. Вкладка «Settlement» (класс EventSettlementPanelEx) предназначена для работы с неплатежными событиями. В верхней ее части расположена таблица со списком корпоративных событий. Существует возможность применить фильтр по дате и типу события. В нижней части — таблица со списком счетов депо, относящихся к конкретному событию из верхней таблицы.

Панели обеих вкладок имеют много общего, поэтому был создан базовый класс EventPanelEx, от которого они и унаследованы.

Редактирование параметров событий происходит на формах классов EventPaymentFormEx (для платежных событий) и EventSettlementFormEx (неплатежных).

Для просмотра и управления частичными выплатами используются два компонента — панель EventPartialPaymentPanel и форма EventPartialPayment.

Главная панель приложения (класс SobosPanel) была модифицирована. На нее были добавлены следующие элементы управления:

1. Кнопка «Apply today’s trading (P)», по нажатию на которую происходит запуск процедуры применения торговли.
2. Кнопка «Proprietary Overdraft Check», нажатие на которую выполняет проверку регистров внутреннего учета на отрицательные балансы.
3. Флажок «Proprietary trading is closed», отвечающий за закрытие бизнес-дня.
4. Кнопка «FlexCube handoff», нажатие на которую вызывает появление формы FlexHandoffParametersForm с настройками выгрузки в УС GAAP.
5. Кнопка, служащая для вызова формы FlexHandoffHistoryForm, отображающей список всех совершенных в прошлом выгрузок в УС GAAP с возможностью заново получить текстовый файл хэндоффа либо построить его в виде отчета (для этого используется стандартная форма отчетов класса ReportParameter).
6. Кнопка «Citi9 handoff», нажатие на которую вызывает появление формы Citi9TxtHandoffSettingForm с настройками выгрузки в УС RAS.
7. Кнопка, служащая для вызова формы Citi9HandoffHistoryForm, отображающей список всех совершенных в прошлом выгрузок в УС RAS с возможностью заново получить текстовый файл хэндоффа.

Новые отчеты («GAAP FIFO Disclosure report», «RAS FIFO Disclosure report», «Fincon MTM report») были добавлены в список панели узла «Reports». Построение отчетов осуществляется с помощью RDLC-шаблонов и COM-модели.

Диаграммы классов разработанных компонентов пользовательского интерфейса и пример RDLC-шаблона отчета приведены в приложении.

# 5 Разработка программы и методики испытаний

Одной из важнейших характеристик программного продукта является его качество.

С точки зрения стандарта ISO 9126 качество программного обеспечения определяется как совокупная характеристика программного обеспечения с учетом таких аспектов, как:

* надежность;
* сопровождаемость;
* удобство использования;
* эффективность;
* универсальность;
* функциональность.

Однако у каждого из участников процесса разработки программного обеспечения существует свой взгляд на это понятие, учитывающий дополнительные критерии помимо приведенных выше характеристик. Например, разработчику важна поддерживаемость кода, заказчику — безопасность, портируемость и др.

Механизмом обеспечения качества программного обеспечения, определения его корректности и полноты является тестирование программного обеспечения. Основные цели тестирования:

* убедиться, что программа соответствует требованиям, и продемонстрировать это заказчику;
* выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.

Важно отметить, что испытания проводятся не для того, чтобы полностью устранить все ошибки. Это невозможно, так как [тестирование программного обеспечения](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) не может доказать, что система, алгоритм или программа не содержит ошибок и дефектов и удовлетворяет определённому свойству. Это можно сделать лишь методами формальной верификации, т.е. строгим математическим аппаратом. В реальных условиях даже на сравнительно несложных программах это оказывается невозможно. Например, если рассматривать набор всевозможных путей исполнения, то условный оператор порождает два пути, N условий — 2N путей, цикл — бесконечное количество путей. Также невозможно протестировать на всех исходных данных даже небольшие процедуры.

Таким образом, тестирование служит для обеспечения некоторого заранее оговоренного уровня качества продукта. Это достигается за счет комплексного применения сразу нескольких типов и методик тестирования.

## 5.1 Разработка методики тестирования

В качестве основного метода тестирования было выбрано функциональное тестирование, поскольку оно основано на проверке реализованных функциональных [требований](http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%BC%D1%83_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8E), то есть способности программного обеспечения в определенных условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает программный продукт, какие задачи оно решает. Именно это наиболее важно для пользователей.

Для проверки сложной логики, такой, как вычисление GAAP и RAS FIFO, генерация проводок и остатков во внутреннем учете, совершение выгрузок в учетные системы RAS и GAAP, генерация событий было разработано 10 тестовых сценариев с наборами сделок, заводимых в течение нескольких дней, отражающих наиболее часто совершаемые банком операции. Для каждого сценария были рассчитаны ожидаемые значения упомянутых выше структур, и затем реальный результат сравнивался с ожидаемым.

Для прогона данных сценариев использовалась особая утилита, разработанная специально для автоматизированного тестирования дорабатываемой системы. Ее настройка осуществляется с помощью файла конфигурации. Утилита использует два входных XML-файла. Ее работа организована по следующему принципу. Тестовая утилита получает XML со входными данными сценария, распознает его и заносит информацию в базу данных. Затем выполняется специфичная последовательность вызовов хранимых процедур, уникальная для каждой тестируемой функциональности. После обработки данных происходит выгрузка реального результата в виде XML, который сравнивается с другим входным XML, содержащим ожидаемые результаты. По итогам выводится либо сообщение об успешном прохождении теста, либо список обнаруженных ошибок. Тестовая утилита получает входные XML не напрямую: необходимые данные оформляются в виде файла табличного процессора Microsoft Excel, откуда они автоматически выгружаются утилитой в виде XML и используются в ее работе. Такой подход обеспечивает быстрое, наглядное и удобное создание и поддержку тестовых сценариев.

Данная утилита была модифицирована для поддержки нового типа тестовых сценариев, связанных с тестированием разрабатываемого модуля. Автоматизация тестирования сложной логики помогла убедиться в том, что исправления ошибок в одном тестовом сценарии никак не сказались на результате других сценариев. Это позволило избавиться от множества рутинных сложных проверок и выиграть значительное количество времени для тестирования остальных требований. Обновление тестовой утилиты пригодится и для регрессионного тестирования при будущих доработках системы.

Проверка реализации остальных требований осуществлялась ручным тестированием. В качестве входных данных часто выступали тестовые сценарии, упомянутые выше.

Одновременно с ручным функциональным тестированием выполнялось также тестирование пользовательского интерфейса на наличие ошибок в выводимых сообщениях, в описаниях полей для ввода данных, удобства использования разработанных форм и т.п.

## 5.2 Разработка тестового стенда

Для повышения качества тестирования и отладки был разработан тестовый стенд, полностью повторяющем конфигурацию рабочих станций, на которых будут развернуты серверная и клиентская части программного продукта.

Описание аппаратного и программного обеспечения тестового стенда приведено ниже.

Аппаратное обеспечение серверной части:

* процессор: Intel Core i5-3470 (3.4 ГГц, 4 ядра);
* объем оперативной памяти: 8 ГБ;
* наличие сетевой карты и доступа к сети Интернет;
* объем свободного дискового пространства: 1 ТБ.

Программное обеспечение серверной части:

* операционная система: Microsoft Windows 2008 Server;
* СУБД: Microsoft SQL Server 2008 R2.

Аппаратное обеспечение клиентской части:

* процессор: Intel Core i5 760 (2.8 ГГц, 4 ядра);
* объем оперативной памяти: 4 ГБ;
* наличие сетевой карты и доступа к сети Интернет;
* объем свободного дискового пространства: 100 ГБ.

Программное обеспечение клиентской части:

* операционная система: Microsoft Windows 7;
* наличие предустановленного клиентского приложения.

## 5.3 Разработка приемочного тест-плана

Перед вводом модуля в эксплуатацию необходимо убедиться в правильности его работы и корректной реализации требований. Для этого был составлен приемочный тест-план, состоящий из тестовых сценариев, каждый из которых проверяет определенную функциональность. Успешное прохождение всех сценариев из тест-плана означает, что разработанный продукт соответствует оговоренным заказчиком уровню качества.

Фрагмент приемочного тест-плана, содержащий некоторые из подготовленных сценариев, представлен в таблице 5.1. В графе «Последовательность действий» для шагов, не относящихся к функциональности, тестируемой в данном сценарии, представлено недетализированное (высокоуровневое) описание.

Таблица 5.1 — Фрагмент приемочного тест-плана

|  |  |  |  |
| --- | --- | --- | --- |
| № сценария | Проверяемая функциональность | Последовательность действий | Ожидаемый результат |
| 1 | Создание сделки | 1. Перейти на вкладку «Trade batches (P)» / «Batches». 2. Вызвать команду создания батча. 3. В появившемся окне нового батча вызвать команду создания сделки. 4. В новом окне заполнить параметры сделки и нажать кнпоку ОК. 5. Закрыть окно батча. | Сделка добавлена в созданный батч. |
| 2 | Модификация сделки | 1. Перейти на вкладку «Trade batches (P)» / «Batches». 2. Вызвать команду создания батча. 3. В появившемся окне нового батча вызвать команду создания сделки. 4. В новом окне заполнить параметры сделки и нажать кнпоку ОК. 5. Выполнить команду изменения сделки. 6. В появившемся окне изменить параметры сделки и нажать кнопку ОК. | Сделка добавлена в созданный батч с последними сохраненными параметрами. |

Продолжение таблицы 5.1

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | Удаление сделки | 1. Перейти на вкладку «Trade batches (P)» / «Batches». 2. Вызвать команду создания батча. 3. В появившемся окне нового батча вызвать команду создания сделки. 4. В новом окне заполнить параметры сделки и нажать кнпоку ОК. 5. Выполнить команду удаления сделки. | Сделка удалена из созданного батча. |
| 4 | Переход статусов батча | 1. Перейти на вкладку «Trade batches (P)» / «Batches». 2. Вызвать команду создания батча. 3. Выполнить операцию подтверждения. 4. Выполнить операцию отмены подтверждения. 5. Выполнить операцию подтверждения. 6. Выполнить авторизацию от лица второго пользователя. 7. Выполнить команду сброса в исходное состояние. 8. Вызвать команду удаления батча. 9. Выполнить отмену удаления. | Батч добавлен в систему. В ходе выполнения операций его статус последовательно меняется: Input -> Committed -> Input -> Committed -> Authorized -> Input -> Deleted -> Input |
| 5 | Удаление батча | 1. Перейти на вкладку «Trade batches (P)» / «Batches». 2. Вызвать команду создания батча. 3. Вызвать команду удаления батча. 4. Выполнить авторизацию от лица второго пользователя. | Статус батча меняется с Input на Deleted, после авторизации батч исчезает и списка. |
| 6 | Импорт сделок из трейдинговой системы | 1. Перейти на вкладку «Trade batches (P)» / «Batches». 2. Вызвать команду импорта сделок из трейдинговой системы. | В систему добавлен новый батч. Параметры загруженных сделок соответствуют спецификации импорта. |

Продолжение таблицы 5.1

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | Просмотр списка RAW и SPLIT сделок со сменой фильтра | 1. Перейти на вкладку «Trade batches (P)» / «Batches». 2. Вызвать команду импорта сделок из трейдинговой системы. 3. Перейти на вкладку «Raw Deals». 4.Указать дату в фильтре, равную CBD+1. 5. Перейти на вкладку Split Deals». 6. Указать дату в фильтре, равную CBD+1. | В систему добавлен новый батч. Список RAW сделок отражает загруженные сделки, после смены фильтра он пуст. Список SPLIT сделок отражает загруженные сделки, после смены фильтра он пуст. |
| 8 | Смена статусов событий | 1. Изменить купонное расписание для некоторой ценной бумаги, указав в качестве даты среза CBD, даты выплаты - CBD+1. 2. Завести Т0 покупку по данной ценной бумаге. 3. Выполнить процедуру применения торговли. 4. Создать новый бизнес-день. 5. Перейти на вкладку «Events (P)» / «Payment» и выбрать корпоративное событие событие. 6. Вызвать команду подтверждения события. 7. Вызвать команду отмены подтверждения. 8. Выполнить модификацию параметров события. 9. Вызвать команду сброса изменений. 10. Вызвать команду подтверждения. 11. Выполнить авторизацию вторым пользователем. | Событие добавлено в систему. В ходе выполнения операций его статус последовательно меняется: Input -> Committed -> Pending -> Input -> Committed -> Authorized |

Продолжение таблицы 5.1

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | Соблюдение последовательности операций применения торговли и проверки регистров внутреннего учета на отрицательные балансы | 1. Завести любую сделку. 2. Перейти на главную панель приложения. 3. Нажать кнопку «Proprietary Overdraft check». | Выводится сообщение о том, что не была вызвана процедура применения торговли. |
| 10 | Соблюдение последовательности операций применения торговли и проверки регистров внутреннего учета на отрицательные балансы | 1. Завести любую сделку. 2. Перейти на главную панель приложения. 3. Нажать кнопку «Apply Trading (P)». 4. Дождаться окончания процедуры и в появившемся окне сообщения нажать кнопку ОК. 5. Нажать кнопку «Proprietary Overdraft check». | Выводится сообщение о результатах проведения проверки регистров внутреннего учета на отрицательные балансы. |

После прогона сценариев из тест-плана был обнаружен ряд ошибок. В ходе отладки были выявлены их причины, после чего ошибки были устранены.

По окончанию тестирования все сценарии из приведенного тест-плана, а также упомянутые в разделе 5.1 сценарии с наборами сделок успешно выполнялись.

# 6 Программная документация

## 6.1 Руководство по сборке

1. Перед сборкой необходимо задать версию клиентского приложения и версию базы данных. Применяется стандартная нумерация версий программного обеспечения, состоящая из четырех чисел, разделенных точкой: главный номер версии (major version number), вспомогательный номер версии (minor version number), номер сборки (build number), номер ревизии в системе контроля версий (revision number). Для работы системы должны совпадать версия базы данных и версия клиентского приложения. Чтобы изменить версию, совершите следующие действия:
   1. Откройте файл <SourceDirectory>\Sobos\Properties\AssemblyInfo.cs с помощью текстового редактора (например, Notepad), где <SourceDirectory> — каталог с исходными текстами программного продукта, и укажите необходимый номер версии в последних двух строках:

[assembly: AssemblyVersion("x.x.x.x")]

[assembly: AssemblyFileVersion("x.x.x.x")]

* 1. Откройте файл <SourceDirectory>\Database\AutoBuilder.sql с помощью текстового редактора и укажите необходимый номер версии в первой строке:

UPDATE dbo.GLOBALS SET DatabaseVersion = 'x.x.x.x'

1. Произведите автоматическую сборку программного продукта, включающую в себя:
   * + сборку решения Visual Studio;
     + конкатенацию SQL-скриптов, добавляющих и обновляющих таблицы базы данных, ограничения, хранимые процедуры, функции и т.п., в единый файл;
     + копирование скомпилированных файлов приложения и сконкатенированного файла базы данных в отдельный каталог.

Для этого:

* 1. Убедитесь, что на Вашем компьютере установлена корпоративная консольная утилита DatabaseCompiler. В противном случае обратитесь к системному администратору.
  2. Запустите файл сценария командной строки <SourceDirectory>\CreateBuild.cmd. При этом появится окно с исполняемыми командами и выводимыми сообщениями.
  3. Дождитесь окончания выполнения сценария и убедитесь, что сообщения об ошибках отсутствуют. В противном случае устраните их и запустите файл заново.
  4. Нажмите Enter для закрытия окна. Появившийся в текущей директории (<SourceDirectory>) подкаталог SobosBuild и есть необходимый каталог со сборкой.

1. Произведите архивацию содержимого созданного каталога сборки <SourceDirectory>\SobosBuild в файл build\_x\_x\_x\_x.zip, где “x” — соответствующее число из номера версии сборки. Для этого можно воспользоваться, например, свободно распространяемым архиватором 7-Zip.

## 6.2 Руководство по развертыванию

1. Распакуйте содержимое архива build\_x\_x\_x\_x.zip в произвольную директорию <BuildDirectory>. Для этого можно воспользоваться, например, свободно распространяемым архиватором 7-Zip.
2. Необходимо обновить рабочую базу данных до версии сборки. Для этого:
   1. Запустите Microsoft SQL Server 2008 R2 Management Studio.
   2. В появившемся окне Connect To Server введите данные, требуемые для подключения к серверу, и нажмите кнопку Connect.
   3. Откройте файл <BuildDirectory>\Database\01\_Database.sql со сценарием базы данных:
      1. Выберите пункт меню File->Open->File… либо нажмите комбинацию клавиш Ctrl+O.
      2. В появившемся диалоговом окне открытия файла выберите файл <BuildDirectory>\Database\01\_Database.sql и нажмите кнопку Open.
   4. Укажите имя рабочей базы данных в открытом сценарии:
      1. Выберите пункт меню Edit->Find and Replace->Quick Replace либо нажмите комбинацию клавиш Ctrl+H.
      2. В появившемся диалоговом окне поиска и замены в выпадающем списке Look in выберите пункт Current Document, в поле Find what введите значение Sobos1, а в поле Replace with укажите имя рабочей базы данных, после чего нажмите кнопку Replace All.
      3. Закройте диалоговые окна с результатами замены и с параметрами замены.
   5. Укажите режим выполнения скрипта с сохранением лог-файла, выбрав пункт меню Query->Results To->Results to File либо нажав комбинацию клавиш Ctrl+Shift+F.
   6. Запустите выполнение скрипта:
      1. Выберите пункт меню Query->Execute либо нажмите клавишу F5.
      2. В появившемся диалоговом окне сохранения лог-файла перейдите в каталог <BuildDirectory>, в выпадающем списке Save as type выберите пункт All files, введите в поле File name имя файла database\_log.txt и нажмите кнопку Save.
   7. Дождитесь окончания выполнения скрипта. Убедитесь, что в логе выполнения в Microsoft SQL Server Management Studio нет ошибок. В противном случае свяжитесь с разработчиком, отправив ему по электронной почте сохраненный лог-файл <BuildDirectory>\database\_log.txt.
   8. Закройте Microsoft SQL Server Management Studio.
3. Установите новую версию клиентского приложения. Для этого:
   1. Перейдите в каталог <ApplicationDirectory>, где развернуто клиентское приложение.
   2. Удалите все файлы из этого каталога.
   3. Скопируйте содержимое каталога <BuildDirectory>\Application в каталог <ApplicationDirectory>.

## 6.3 Руководство пользователя

### 6.3.1 Батчи

Все сделки в системе сгруппированы в коллекции, называемые батчами. Для просмотра списка батчей, заведенных за текущий бизнес-день, необходимо в дереве панелей приложения выбрать элемент «Trade batches (P)», затем перейти на вкладку «Batches». В верхней части вкладки отображается искомый список, в нижней — список всех сделок, включенных в выбранный батч.

Батчи могут иметь различные статусы, характеризующие их состояние и доступные для выполнения действия:

* «Input» — батч находится на стадии ввода и редактирования сделок;
* «Commited» — ввод и редактирование сделок закончены, второй пользователь может начинать работы по проверке внесенных изменений;
* «Authorized» — сделки в батче проверены на предмет наличия ошибок, авторизованы от лица второго пользователя и помечены для обработки;
* «Processed» — сделки в батче обработаны системой;
* «Deleted» — батч и входящие в него сделки отмечены для удаления из системы, требуется авторизация от лица второго пользователя.

### 6.3.2 Ручное создание сделок

Для ручного создания сделок выполните следующие действия:

1. Откройте вкладку с батчами («Trade batches (P)» / «Batches»).
2. Вызовите команду создания нового батча («Create») посредством главного меню, контекстного меню либо кнопки на панели инструментов. Появится окно текущего батча.
3. В окне текущего батча на вкладке «Today’s raw deals» вызовите команду добавления сделки («Create») посредством контекстного меню либо кнопки на панели инструментов. Появится окно для ввода параметров сделки.
4. Введите необходимые параметры сделки и нажмите кнопку «ОК» для закрытия окна. В случае незаполнения обязательного поля рядом с ним появится изображение восклицательного знака на красном фоне, сигнализирующее об ошибке.
5. Повторите шаги 3-4, пока не создадите все необходимые сделки.
6. В окне текущего батча на вкладке «Today’s split deals» вызовите команду автоматического назначения портфелей («Auto assign portfolio») посредством контекстного меню либо кнопки на панели инструментов. Появится сообщение с результатами назначения портфелей. Если по итогам для некоторых сделок портфель не был назначен, то портфель отсутствует в системе и его необходимо добавить, после чего вернуться к данному шагу.
7. Закройте окно текущего батча, нажав на кнопку «ОК».

По окончании работ в систему будет добавлен новый батч с типом «Manual».

### 6.3.3 Импорт сделок из трейдинговой системы

Для автоматической загрузки биржевых сделок из трейдинговой системы выполните следующие действия:

1. Откройте вкладку с батчами («Trade batches (P)» / «Batches»).
2. Вызовите команду импорта («Import MICEX deals from FIS») посредством главного меню, контекстного меню либо кнопки на панели инструментов.
3. Дождитесь завершения процедуры и нажмите кнопку «ОК» в окне сообщения с результатами импорта.

По окончании импорта в систему будет добавлен новый батч с типом «Imported».

Импорт сделок из трейдинговой системы разрешен лишь единожды в течение бизнес-дня. В случае попытки повторной загрузки выведется предупреждающее сообщение с текстом «MICEX deals have already been imported today».

### 6.3.4 Редактирование сделок

Редактировать сделки в батче может только тот пользователь, который завел его. Кроме того, батч должен находиться в статусе «Input».

Для модификации сделок выполните следующие действия:

1. Откройте вкладку с батчами («Trade batches (P)» / «Batches»).
2. Выберите требуемый батч.
3. Вызовите команду открытия батча («Open») посредством двойного клика, главного меню, контекстного меню либо кнопки на панели инструментов. Появится окно текущего батча.
4. В окне текущего батча на вкладке «Today’s raw deals» вызовите команду открытия сделки («Open») посредством двойного клика, контекстного меню либо кнопки на панели инструментов. Появится окно для ввода параметров сделки.
5. Внесите необходимые изменения в параметры сделки и нажмите кнопку «ОК» для закрытия окна. В случае некорректного заполнения поля либо пустого значения обязательного поля рядом с полем появится изображение восклицательного знака на красном фоне, сигнализирующее об ошибке. Для отмены внесенных изменений нажмите кнопку «Cancel».
6. Повторите шаги 4-5 для всех необходимых сделок.
7. Если в сделках были изменены ценная бумага либо категория портфеля, то необходимо заново выполнить назначение портфелей. В окне текущего батча на вкладке «Today’s split deals» вызовите команду автоматического назначения портфелей («Auto assign portfolio») посредством контекстного меню либо кнопки на панели инструментов. Появится сообщение с результатами назначения портфелей. Если по итогам для некоторых сделок портфель не был назначен, то портфель отсутствует в системе и его необходимо добавить, после чего вернуться к данному шагу.
8. Закройте окно текущего батча, нажав на кнопку «ОК».

### 6.3.5 Подтверждение и отмена подтверждения батча

Подтвердить батч после окончания редактирования может лишь тот пользователь, который завел его. Батч должен находиться в состоянии «Input».

Для перевода батча в подтвержденное состояние выполните следующие действия:

1. Откройте вкладку с батчами («Trade batches (P)» / «Batches»).
2. Выберите требуемый батч.
3. Вызовите команду подтверждения батча («Commit») посредством главного меню, контекстного меню либо кнопки на панели инструментов.

Батч сменит свое состояние на «Committed».

Если необходимо внести исправления в сделки данного батча, то нужно совершить отмену подтверждения. Совершить это действие может лишь тот пользователь, который завел батч. Батч должен находиться в состоянии «Committed».

Для отмены подтверждения батча выполните следующие действия:

1. Откройте вкладку с батчами («Trade batches (P)» / «Batches»).
2. Выберите требуемый батч.
3. Вызовите команду отмены подтверждения батча («Uncommit») посредством главного меню, контекстного меню либо кнопки на панели инструментов.

Батч сменит свое состояние на «Input».

### 6.3.6 Удаление и отмена удаления батча

Для того, чтобы пометить батч на удаление, выполните следующие действия:

1. Откройте вкладку с батчами («Trade batches (P)» / «Batches»).
2. Выберите требуемый батч.
3. Вызовите команду удаления батча («Delete») посредством главного меню, контекстного меню либо кнопки на панели инструментов.

Батч сменит свое состояние на «Deleted». Для окончательного удаления его из системы необходимо произвести авторизацию.

Если эта операция была совершена по ошибке, то необходимо отменить удаление. Совершить это действие может лишь тот пользователь, который пометил батч на удаление. Батч должен находиться в состоянии «Deleted».

Для отмены удаления выполните следующие действия:

1. Откройте вкладку с батчами («Trade batches (P)» / «Batches»).
2. Выберите требуемый батч.
3. Вызовите команду отмены подтверждения батча («Cancel deletion») посредством главного меню, контекстного меню либо кнопки на панели инструментов.

Батч вернется в прежнее состояние.

### 6.3.7 Авторизация батча

Авторизация служит для подтверждения изменений, внесенных пользователем. Пользователь, совершивший изменения, не может сам же авторизовать их. Батч должен находиться в состоянии «Committed» или «Deleted».

Для авторизации батча выполните следующие действия:

1. Откройте вкладку с батчами («Trade batches (P)» / «Batches»).
2. Выберите требуемый батч.
3. Вызовите команду авторизации батча («Authorize») посредством главного меню, контекстного меню либо кнопки на панели инструментов.

Если батч был в состоянии «Deleted», то он удалится из системы. В противном случае он сменит свое состояние на «Authorized».

### 6.3.8 Просмотр сделок

Для просмотра RAW сделок выполните следующие действия:

1. Откройте вкладку с RAW сделками («Trade batches (P)» / «Raw Deals»). Появится список сделок.
2. По умолчанию отображаются сделки с датой заключения, равной текущему бизнес-дню. Для изменения фильтра выберите необходимые даты в полях «Deals from» и «to», затем кликните мышкой в области списка.
3. Если необходимо просмотреть подробную информацию о сделке, вызовите команду открытия сделки («Open») посредством двойного клика, главного меню, контекстного меню либо кнопки на панели инструментов. Появится окно для просмотра параметров сделки. Для закрытия нажмите кнопку «ОК».

Для просмотра SPLIT сделок выполните следующие действия:

1. Откройте вкладку с SPLIT сделками («Trade batches (P)» / «Split Deals»). Появится список сделок.
2. По умолчанию отображаются сделки с датой заключения, равной текущему бизнес-дню. Для изменения фильтра выберите необходимые даты в полях «Deals from» и «to», затем кликните мышкой в области списка.
3. Если необходимо просмотреть подробную информацию о сделке, вызовите команду открытия сделки («Open») посредством двойного клика, главного меню, контекстного меню либо кнопки на панели инструментов. Появится окно для просмотра параметров сделки. Для закрытия нажмите кнопку «ОК».

### 6.3.9 Корпоративные события

Генерация корпоративных событий происходит при создании нового бизнес-дня.

Корпоративные события бывают двух видов: связанные с выплатами денежных средств и не связанные с ними (в этом случае меняются позиции счетов депо).

За операции над корпоративными событиями отвечает элемент «Events (P)» в дереве панелей. Эта панель имеет две вкладки — «Payment» и «Settlement», предназначенные для работы с платежными и неплатежными событиями соответственно.

Корпоративные события могут иметь различные статусы, характеризующие их состояние и доступные для выполнения действия:

* «Input» — событие не модифицировалось после генерации и доступно для редактирования;
* «Commited» — редактирование события закончено, второй пользователь может начинать работы по проверке внесенных изменений;
* «Authorized» — событие проверено на предмет наличия ошибок, авторизовано от лица второго пользователя и помечено для обработки;
* «Processed» — событие обработано системой.

Частичные выплаты могут иметь различные статусы, характеризующие их состояние и доступные для выполнения действия:

* «Active» — выплата авторизована и не имеет изменений;
* «PA To Create» — выплата была создана и еще не авторизована;
* «PA To Modify» — выплата была модифицирована и еще не авторизована;
* «PA To Delete» — выплата помечена для удаления.

Авторизация частичных выплат происходит совместно с авторизацией события. При этом выплаты из состояний «PA To Create» и «PA To Modify» переходят в «Active», а выплаты с состоянием «PA To Delete» удаляются из системы.

### 6.3.10 Просмотр корпоративных событий

Для просмотра платежных событий выполните следующие действия:

1. Откройте вкладку с платежными событиями («Events (P)» / «Payment»). Появятся две таблицы. Верхняя содержит список платежных корпоративных событий, нижняя — список частичных выплат по выбранному событию.
2. По умолчанию отображаются события всех видов с планируемой датой выплаты, равной текущему бизнес-дню. Для изменения фильтра выберите необходимые даты в полях «Events from» и «to», выберите интересующий тип события в выпадающем списке «Events type», затем нажмите кнопку «Apply». Если выбрать флажок «Show all unprocessed events», то фильтр по дате будет проигнорирован: будут отображаться события все события, кроме тех, что имеют статус «Processed».
3. Если необходимо просмотреть подробную информацию о событии, вызовите команду открытия события («Open») посредством двойного клика, главного меню, контекстного меню либо кнопки на панели инструментов. Появится окно для просмотра параметров события, в нижней части которого будет находиться список частичных выплат по данному событию.
4. Если необходимо просмотреть подробную информацию о частичной выплате, то вызовите команду открытия события («Open») посредством двойного клика, контекстного меню либо кнопки на панели инструментов. Появится окно для просмотра параметров частичной выплаты. Для закрытия нажмите кнопку «ОК».
5. Для закрытия окна с параметрами события нажмите кнопку «ОК».

Для просмотра неплатежных событий выполните следующие действия:

1. Откройте вкладку с неплатежными событиями («Events (P)» / «Settlement»). Появятся две таблицы. Верхняя содержит список неплатежных корпоративных событий, нижняя — список счетов депо по выбранному событию.
2. По умолчанию отображаются события с планируемой датой исполнения, равной текущему бизнес-дню. Для изменения фильтра выберите необходимые даты в полях «Events from» и «to», затем нажмите кнопку «Apply». Если выбрать флажок «Show all unprocessed events», то фильтр по дате будет проигнорирован: будут отображаться события все события, кроме тех, что имеют статус «Processed».
3. Если необходимо просмотреть подробную информацию о событии, вызовите команду открытия события («Open») посредством двойного клика, главного меню, контекстного меню либо кнопки на панели инструментов. Появится окно для просмотра параметров события. Для закрытия нажмите кнопку «ОК».

### 6.3.11 Редактирование корпоративных событий

Любой пользователь может редактировать события, для которых не указана реальная дата (выплаты или исполнения) либо реальная дата еще не наступила, и которые находятся в состоянии «Input», «Authorized» или «Processed». События в состоянии «Pending» может модифицировать только тот пользователь, который внес последние изменения. Кроме того, применяются упомянутые выше требования по дате.

Для редактирования события выполните следующие действия:

1. Откройте вкладку с событиями («Events (P)» / «Payment» или «Events (P)» / «Settlement»).
2. Выберите требуемое событие.
3. Вызовите команду открытия («Open») посредством двойного клика, главного меню, контекстного меню либо кнопки на панели инструментов. Появится окно для просмотра параметров события.
4. Если открыто платежное событие, то можно управлять списком частичных выплат (см. далее).
5. Внесите необходимые изменения в параметры события и нажмите кнопку «ОК» для закрытия окна. В случае некорректного заполнения поля либо пустого значения обязательного поля рядом с полем появится изображение восклицательного знака на красном фоне, сигнализирующее об ошибке. Для отмены внесенных изменений нажмите кнопку «Cancel».

После внесения изменений событие перейдет в состояние «Pending».

Частичные выплаты доступны для редактирования, если реальная дата выплаты не указана либо еще не прошла. Помимо этого, выплата должна находиться в состоянии «Active» — в этом случае ее может изменять пользователь, который имеет право на редактирование родительского события. Либо выплата находится в состоянии «PA To Create» или «PA To Modify», тогда модифицировать ее может тот, кто внес последние изменения.

Для управления частичными выплатами выполните следующие действия:

1. Откройте вкладку с платежными событиями («Events (P)» / «Payment»).
2. Выберите требуемое событие.
3. Вызовите команду открытия («Open») посредством двойного клика, главного меню, контекстного меню либо кнопки на панели инструментов. Появится окно для просмотра параметров события, в нижней части которого будет находиться список частичных выплат по данному событию.
4. Для добавления новой частичной выплаты вызовите команду создания («Create») посредством контекстного меню либо кнопки на панели инструментов. Появится окно для просмотра параметров частичной выплаты. Внесите необходимые изменения в параметры выплаты и нажмите кнопку «ОК» для закрытия окна (в списке появится новая частичная выплата с состоянием «PA To Create»). В случае некорректного заполнения поля либо пустого значения обязательного поля рядом с полем появится изображение восклицательного знака на красном фоне, сигнализирующее об ошибке. Для отмены нажмите кнопку «Cancel».
5. Для модификации частичной выплаты вызовите команду открытия («Open») посредством двойного клика, контекстного меню либо кнопки на панели инструментов. Появится окно для просмотра параметров частичной выплаты. Внесите необходимые изменения в параметры выплаты и нажмите кнопку «ОК» для закрытия окна (частичная выплата в списке сменит свое состояние на «PA To Modify», если она была активна, или останется в состоянии «PA To Create», если изменения вносились в еще не авторизованную недавно добавленную выплату). В случае некорректного заполнения поля либо пустого значения обязательного поля рядом с полем появится изображение восклицательного знака на красном фоне, сигнализирующее об ошибке. Для отмены внесенных изменений нажмите кнопку «Cancel».
6. Для удаления частичной в выплаты вызовите команду удаления («Delete») посредством двойного клика, контекстного меню либо кнопки на панели инструментов. Выплата сменит свое состояние на «Deleted».
7. Нажмите кнопку «ОК» для фиксации внесенных изменений в частичные выплаты либо «Cancel» для отмены. Окно с параметрами события закроется.

Событие перейдет в состояние «Pending».

Для авторизация выплат необходимо осуществить авторизацию родительского события.

Если модификация была совершена по ошибке, то необходимо отменить изменения. Совершить это действие может лишь тот пользователь, который внес их. Событие должно находиться в состоянии «Pending».

Для сброса изменений выполните следующие действия:

1. Откройте вкладку с событиями («Events (P)» / «Payment» или «Events (P)» / «Settlement»).
2. Выберите требуемое событие.
3. Вызовите команду сброса изменений («Drop changes») посредством главного меню, контекстного меню либо кнопки на панели инструментов.

Событие вернется в прежнее состояние.

### 6.3.12 Подтверждение и отмена подтверждения корпоративного события

Подтвердить событие, которое не подвергалось редактированию после своего появления, может любой пользователь. После редактирования совершить операцию подтверждения может лишь тот пользователь, который модифицировал его. Событие должно находиться в состоянии «Input» или «Pending».

Для перевода события в подтвержденное состояние выполните следующие действия:

1. Откройте вкладку с событиями («Events (P)» / «Payment» или «Events (P)» / «Settlement»).
2. Выберите требуемое событие.
3. Вызовите команду подтверждения корпоративного события («Commit») посредством главного меню, контекстного меню либо кнопки на панели инструментов.

Событие сменит свое состояние на «Committed».

Если необходимо внести исправления в данные события, то нужно совершить отмену подтверждения. Совершить это действие может лишь тот пользователь, который подтвердил его. Событие должно находиться в состоянии «Committed».

Для отмены подтверждения корпоративного события выполните следующие действия:

1. Откройте вкладку с событиями («Events (P)» / «Payment» или «Events (P)» / «Settlement»).
2. Выберите требуемое событие.
3. Вызовите команду отмены подтверждения корпоративного события («Uncommit») посредством главного меню, контекстного меню либо кнопки на панели инструментов.

Событие сменит свое состояние на «Pending».

### 6.3.13 Авторизация корпоративных событий

Авторизация служит для подтверждения изменений, внесенных пользователем. Пользователь, совершивший изменения, не может сам же авторизовать их. Событие должно находиться в состоянии «Committed».

Для авторизации корпоративного события выполните следующие действия:

1. Откройте вкладку с событиями («Events (P)» / «Payment» или «Events (P)» / «Settlement»).
2. Выберите требуемое событие.
3. Вызовите команду авторизации события («Authorize») посредством главного меню, контекстного меню либо кнопки на панели инструментов.

Событие сменит свое состояние на «Authorized».

### 6.3.14 Применение торговли

Для учета заведенных за день операция необходимо совершить процедуру, называемую применением торговли. В ходе ее выполнения строятся GAAP и RAS FIFO, формируются предварительные проводки и остатки внутреннего учета. Эта операция является обязательной и должна выполняться после окончания работ по изменению различных сущностей и авторизации всех изменений.

Для совершения этой процедуры выполните следующие действия:

1. Перейдите на главную панель приложения («Main»).
2. Нажмите на кнопку «Apply today’s trading (P)».
3. Если совершается попытка выполнить операцию применения торговли до того, как все необходимые сущности были авторизованы, то появится предупреждающее сообщение с соответствующим текстом.
4. Дождитесь окончания выполнения процедуры и нажмите кнопку «ОК» в сообщении с результатами.
5. По итогам применения торговли проводится сравнение остатков в GAAP и RAS FIFO, а также в RAS FIFO и регистрах внутреннего учета ценных бумаг. В случае появления расхождений появится предупреждающее окно с соответствующей информацией.

### 6.3.15 Проверка регистров внутреннего учета на отрицательные балансы

Перед закрытием бизнес-дня необходимо убедиться в отсутствии негативных балансов на счетах внутреннего учета. Для этого служит данная процедура. В ходе ее выполнения перестраиваются регистры внутреннего учета и выполняется проверка балансов. Эта операция является обязательной и должна выполняться после применения торговли.

Для совершения этой процедуры выполните следующие действия:

1. Перейдите на главную панель приложения («Main»).
2. Нажмите на кнопку «Proprietary Overdraft check».
3. Если совершается попытка выполнить эту операцию до применения торговли, то появится предупреждающее сообщение с соответствующим текстом.
4. Дождитесь окончания выполнения процедуры и нажмите кнопку «ОК» в сообщении с результатами. В случае отрицательных балансов на каких-либо счетах необходимо добавить ручные операции с денежными средствами или ценными бумагами.

### 6.3.16 Закрытие бизнес-дня

Данная операция проводится непосредственно перед созданием нового бизнес-дня и является обязательной. Ее можно совершить лишь после проверки регистров внутреннего учета на негативные балансы.

Для закрытия бизнес-дня выполните следующие действия:

1. Перейдите на главную панель приложения («Main»).
2. Нажмите на флажок «Proprietary trading is closed».
3. Если совершается попытка выполнить эту операцию до проверки регистров внутреннего учета на отрицательные остатки, то появится предупреждающее сообщение с соответствующим текстом.

### 6.3.17 Выгрузка в учетную систему GAAP

Выгрузка в бухгалтерскую учетную систему GAAP должна производиться после применения торговли, так как она основывается ее результатах.

Для совершения выгрузки без сохранения текстового файла выполните следующие действия:

1. Перейдите на главную панель приложения («Main»).
2. Нажмите на кнопку «FlexCube handoff». Появится окно с настройками выгрузки.
3. Введите необходимые параметры и нажмите кнопку «Generate».
4. Дождитесь окончания процедуры и нажмите кнопку «ОК» в окне сообщения с результатами. Появится окно с предложением отобразить данные выгрузки в виде отчета.
5. Если это необходимо, то в выпадающем списке выберите требуемый тип отчета и нажмите кнопку «Generate», в противном случае «Cancel».

Для совершения выгрузки с получением файла хэндоффа выполните следующие действия:

1. Перейдите на главную панель приложения («Main»).
2. Нажмите на кнопку с изображением блокнота справа от кнопки «FlexCube handoff». Появится диалоговое окно для выбора пути, по которому требуется сохранить файл хэндоффа.
3. Выберите требуемое назначение и нажмите кнопку «ОК». Появится окно с настройками выгрузки.
4. Введите необходимые параметры и нажмите кнопку «Generate».
5. Дождитесь окончания процедуры и нажмите кнопку «ОК» в окне сообщения с результатами. Появится окно с предложением отобразить данные выгрузки в виде отчета.
6. Если это необходимо, то в выпадающем списке выберите требуемый тип отчета и нажмите кнопку «Generate», в противном случае «Cancel».

Для построения уже совершенной выгрузки в виде отчета выполните следующие действия:

1. Перейдите на главную панель приложения («Main»).
2. Нажмите на кнопку с изображением желтой молнии справа от кнопки «FlexCube handoff». Появится окно со всеми совершенными выгрузками.
3. Выберите требуемую выгрузку.
4. Вызовите команду построения отчета («Make report») посредством двойного клика либо контекстного меню. Появится окно с настройками отчета.
5. В выпадающем списке выберите требуемый тип отчета, нажмите кнопку «Generate» и дождитесь окончания выполнения операции.

### 6.3.18 Выгрузка в учетную систему RAS

Выгрузка в бухгалтерскую учетную систему RAS должна производиться после применения торговли, так как она основывается ее результатах.

Для совершения выгрузки с получением файла хэндоффа выполните следующие действия:

1. Перейдите на главную панель приложения («Main»).
2. Нажмите на кнопку «Citi9 handoff». Появится окно для выбора даты, за которую совершается выгрузка
3. Выберите требуемую дату и нажмите кнопку «Generate».
4. Дождитесь окончания процедуры и нажмите кнопку «ОК» в окне сообщения с результатами.

Для совершения выгрузки с получением файла хэндоффа выполните следующие действия:

1. Перейдите на главную панель приложения («Main»).
2. Нажмите на кнопку с изображением блокнота справа от кнопки «Citi9 handoff». Появится диалоговое окно для выбора пути, по которому требуется сохранить файл хэндоффа.
3. Выберите требуемое назначение и нажмите кнопку «ОК». Появится окно для выбора даты, за которую совершается выгрузка
4. Выберите требуемую дату и нажмите кнопку «Generate».
5. Дождитесь окончания процедуры и нажмите кнопку «ОК» в окне сообщения с результатами.

Для построения файла хэндоффа уже совершенной выгрузки выполните следующие действия:

1. Перейдите на главную панель приложения («Main»).
2. Нажмите на кнопку с изображением желтой молнии справа от кнопки «Citi9 handoff». Появится окно со всеми совершенными выгрузками.
3. Выберите требуемую выгрузку.
4. Вызовите команду построения хэндоффа («Citi9 text handoff») посредством двойного клика либо контекстного меню. Появится диалоговое окно для выбора пути, по которому требуется сохранить файл хэндоффа.
5. Выберите требуемое назначение и нажмите кнопку «ОК».
6. Дождитесь окончания процедуры и нажмите кнопку «ОК» в окне сообщения с результатами.

# 7 Экономическая часть

## 7.1 План-график разработки

Разработка любого программного обеспечения состоит из нескольких стадий. Чтобы этот процесс происходил организованно, без лишних затрат труда, материальных средств и в минимальные сроки, необходимо эффективное планирование.

Основной задачей планирования является распределение по срокам и исполнителям этапов разработки программного продукта, а также определение общей продолжительности их проведения.

Для наглядного отображения плана выбрана модель, известная как ленточный график.

Ленточный график процесса (подготовки производства, разработки ПО) — это его графическая модель с указанием перечня и организационно-экономических характеристик всех работ, сроков и последовательности их исполнения, отражаемых совокупностью упорядоченных во времени горизонтальных линий.

Для построения ленточного графика необходимо сформировать перечень всех основных этапов и видов работ, которые должны быть выполнены в ходе разработки, с указанием их продолжительности. Важную роль играет логическое упорядочивание последовательности выполнения отдельных видов работ. В его основе лежит анализ смыслового содержания каждого вида работ и установление взаимосвязи между видами работ. Для каждого этапа и вида работ требуется указать список исполнителей.

Продолжительность работ определяется по формуле:

,

где

* Ti – трудоемкость работ, человеко-дни;
* ni – численность исполнителей, человек.

Соответственно, трудоемкость можно рассчитать на основании продолжительности работы и количества исполнителей:

Данные для построения ленточного графика представлены в табл. 7.1.

Таблица 7.1 —Данные для построения ленточного графика

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Наименование этапа работ | Исполнители | | Tn, дней | Ti, человеко-дней |
| Инженер-программист | Руководитель |
| 1 | Получение и согласование задания | **+** | **+** | 1 | 2 |
| 2 | Ознакомление с архитектурой системы | **+** |  | 2 | 2 |
| 3 | Ознакомление с предметной областью | **+** |  | 5 | 5 |
| 4 | Анализ и разработка требований | **+** |  | 6 | 6 |
| 5 | Согласование требований | **+** | **+** | 1 | 2 |
| 6 | Проектирование структуры модуля | **+** |  | 6 | 6 |
| 7 | Написание программного кода | **+** |  | 40 | 40 |
| 8 | Отладка и тестирование | **+** |  | 12 | 12 |
| 9 | Оформление программной документации | **+** |  | 5 | 5 |
| 10 | Оформление пояснительной записки | **+** |  | 10 | 10 |
| 11 | Сдача проекта | **+** | **+** | 1 | 2 |
| Итого: | | | | 89 | 92 |

Ленточный график разработки проекта, построенный по этим данным, приведен на рисунке 7.1.



Рисунок 7.1 — Ленточный график разработки проекта

## 7.2 Смета затрат на разработку

Затраты, образующие себестоимость продукции (работ, услуг), группируются в соответствии с их экономическим содержанием по следующим элементам:

* материальные затраты (за вычетом стоимости возвратных отходов);
* затраты на оплату труда;
* амортизация основных фондов;
* прочие затраты.

### 7.2.1 Материальные затраты

В материальные затраты данного проекта входят затраты на электроэнергию.

В ходе разработки из электрооборудования использовалась ПЭВМ.

Стоимость использованной электроэнергии рассчитывается по формуле:

где:

* P — мощность ПЭВМ, кВт;
* Tи — время использования ПЭВМ, ч;
* Цэл — стоимость электроэнергии, руб. / (кВт\*ч).

ПЭВМ имеет мощность 300 Вт, или 0,3 кВт. Время работы на ПЭВМ составляет 89 дней по 8 часов в день, т.е. 712 ч. Стоимость кВт\*ч составляет 3,31 руб.

Получаем следующую сумму материальных затрат:

### 7.2.2 Затраты на оплату труда

В разработке принимали участие двое — руководитель проекта и инженер-программист с месячными окладами 25000 рублей и 15000 рублей соответственно. Руководителем затрачено на разработку 3 дня, а инженером — 89 дней. Месячный фонд одного разработчика составляет 22 дня.

Основная заработная плата одного исполнителя определяется по формуле:

где:

* Т — тарифная ставка, руб. /день;
* Д — количество отработанных дней.

Получаем следующие значения заработной платы руководителя и инженера-программиста:

Общие затраты на оплату труда составляют:

### 7.2.3 Амортизация основных фондов

Под амортизационными отчислениями понимаются отчисления части стоимости [основных фондов](http://ru.wikipedia.org/wiki/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5_%D1%81%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B2%D0%B0) для возмещения их износа. Амортизационные отчисления включаются в [издержки производства](http://ru.wikipedia.org/wiki/%D0%98%D0%B7%D0%B4%D0%B5%D1%80%D0%B6%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%B0) или [обращения](http://ru.wikipedia.org/wiki/%D0%98%D0%B7%D0%B4%D0%B5%D1%80%D0%B6%D0%BA%D0%B8_%D0%BE%D0%B1%D1%80%D0%B0%D1%89%D0%B5%D0%BD%D0%B8%D1%8F). Производятся коммерческими организациями на основе установленных норм и балансовой стоимости [основных фондов](http://ru.wikipedia.org/wiki/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5_%D1%81%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B2%D0%B0), на которые начисляется амортизация.

Согласно статье 256 Налогового кодекса РФ, амортизируемым имуществом признается имущество со сроком полезного использования более 12 месяцев и первоначальной стоимостью более 40 000 рублей.

В ходе разработки проекта использовалась ПЭВМ.

Согласно Постановлению Правительства РФ от 1 января 2002 г. №1 "О Классификации основных средств, включаемых в амортизационные группы", ПЭВМ входит во 2-ую амортизационную группу (срок полезного использования от 2 до 3 лет включительно).

Балансовая стоимость ПЭВМ Фп = 27000 руб. Для разработки проекта потребовался ряд программных продуктов, часть из которых является платными. Использованное программное обеспечение приведено в табл. 7.2.

Таблица 7.2—Использованное программное обеспечение

|  |  |
| --- | --- |
| Программное обеспечение | Стоимость, руб. |
| Windows 7 Enterprise | 7500.00 |
| Visual Studio 2008Professional | 12000.00 |
| MSSQL Server 2008 R2 Express | бесплатно |

Балансовая стоимость использованной ПЭВМ с учетом стоимости программного обеспечения составляет 47000 рублей.

Таким образом, данная ПЭВМ относится к амортизируемому имуществу.

Сумма затрат на амортизацию оборудования за время его использования по теме научно-исследовательской работы рассчитывается по следующей формуле:

где:

* *Фп* – балансовая стоимость оборудования;
* *Ти* – время использования оборудования при проведении работ;
* *А* – норма амортизации;
* *Фэф* – годовой эффективный фонд времени работы оборудования, для односменной работы он составляет *Фэф =* 2007 ч*.*

Время работы на ПЭВМ составляет 89 дней по 8 часов в день, т.е.712 ч.

Приняв срок полезного использования, равный 3 годам, получим норму амортизации 33,3%.

Получаем следующее значение амортизационных отчислений:

### 7.2.4 Прочие затраты

Прочие затраты включают в себя страховые взносы и остальные прочие расходы.

Страховые взносы берутся в размере 30,2% от фонда оплаты труда:

Величина остальных прочих расходов берется от суммы прямых общих затрат в установленном размере. Для разработки программного обеспечения они составят 10 %.

Общие прямые затраты составят следующую сумму:

Воспользовавшись рассчитанным значением, получим следующую величину остальных прочих расходов:

Таким образом, прочие затраты составят:

### 7.2.5 Общие затраты

Общие затраты на разработку складываются из общих прямых затрат и прочих затрат:

Общие затраты составят:

Необходимые расходы сведены в таблице 7.3.

Таблица 7.3 — Смета затрат на разработку

|  |  |  |
| --- | --- | --- |
| Наименование калькуляционных статей расходов | Сумма, руб. | Удельный вес, % |
| Материальные затраты, Зм | 707,02 | 0,73 |
| Затраты на оплату труда, Фзп | 64090,91 | 65,81 |
| Амортизация оборудования, Анир | 5552,32 | 5,70 |
| Прочие расходы, Зпр | 27031,39 | 27,76 |
| Общие затраты, З | 97381,64 | 100 |

### 7.2.6 Расчет цены для научно-исследовательской работы

Цена для НИР рассчитывается как сумка общих затрат и закладываемой прибыли:

Прибыль, исходя из планируемой нормы доходности 30%, составит:

Таким образом, цена для НИР имеет следующее значение:

## 7.3 Расчет и выводы по эффективности предложений

Разрабатываемый программный модуль являются частью банковской системы по учету операций с ценными бумагами. Он позволит автоматизировать дилерскую деятельность, осуществляемую банком. Автоматизация процесса значительно сократит время, требуемое на ввод и обработку информации и на подготовку отчетности, позволяющей принимать решения по ведению торговли. Это, в свою очередь, приведет к экономии затрачиваемых денежных средств по сравнению с ручным учетом.

Для расчета величины сэкономленных средств сравним ежедневные временные затраты пользователей при ручном учете и с использованием данного модуля. Примем во внимание, что пользователи работают в парах: первый пользователь выполняет ввод новой информации и обновление существующих данных, второй — проверяет корректность добавлений и изменений и подтверждает их.

Результаты сравнения приведены в таблице 7.4.

Таблица 7.4 — Сравнение временных затрат пользователей

|  |  |  |  |
| --- | --- | --- | --- |
| Пользователь | Ежедневные временные затраты, ч | | Снижение временных затрат, ч |
| Ручной учет | Автоматизированный учет |
| Пользователь 1 | 4 | 1,5 | 2,5 |
| Пользователь 2 | 2,5 | 1 | 1,5 |
| Итого | 6,5 | 2,5 | 4 |

Автоматизация учета приведет к ежедневной экономии времени в 4 ч для пары пользователей. Приняв месячный фонд за 22 дня, получим 88 ч ежемесячно. Как правило, из-за большого количества сделок ежедневно работают 2 пары пользователей; в этом случае ежемесячная экономия составит уже 176 ч. Рассчитаем сэкономленные денежные средства при окладе оператора в 20000 руб.:

Зная сумму общих затрат на разработку, можно рассчитать срок окупаемости проекта:

Таким образом, разрабатываемый модуль окупится практически через полгода. С учетом того, что он интегрируется в банковскую систему, обладающей большими возможностями для дальнейшего расширения, разработка данного программного продукта экономически выгодна.

# 8 Безопасность и экологичность проекта

Использование программного продукта будет производится в помещении, имеющем размеры 10x 6м, с высотой потолка 3 м. Внутренний интерьер включает в себя 6 столов со стульями посередине помещения и по 2 стола около каждой стены, а также 2 шкафа с документами. Помещение оборудовано 8 персональными компьютерами HP с ЖК мониторами. Питание средств вычислительной техники осуществляется от трехфазной цепи переменного тока частотой 50 Гц напряжением 380/220В с глухозаземленной нейтралью. Помещение снабжено одним углекислотным огнетушителем типа ОУ-5.

Помещения, в которых эксплуатируется ПЭВМ, оказывают существенное влияние на безопасность пользователя, а потому должны соответствовать требованиям СанПиН 2.2.2/2.4.1340-03 "Гигиенические требования к персональным электронно-вычислительным машинам и организации работы". Ввод в эксплуатацию помещений, предназначенных для работы с ПЭВМ, должен осуществляться при обязательном участии представителей санэпидемнадзора (ГСЭН).

Для оптимизации условий труда пользователя с целью сохранения здоровья и работоспособности необходимо выявить опасные и вредные факторы применительно к конкретной обстановке, оценить значимость этих факторов и возможные неблагоприятные последствия на основе комплексного подхода и с учетом условий эксплуатации ПЭВМ и вида выполняемой работы.

## 8.1 Анализ опасных и вредных факторов на рабочем месте пользователя

На человека в процессе его трудовой деятельности могут воздействовать опасные (вызывающие травмы) и вредные (вызывающие заболевания) производственные факторы.

Вредный производственный фактор — производственный фактор, воздействие которого на работающего в определённых условиях приводит к заболеванию или снижению работоспособности.

Опасный производственный фактор — производственный фактор, воздействие которого на работающего в определённых условиях приводит к травме или другому внезапному ухудшению здоровья.

Вредный производственный фактор в зависимости от интенсивности и продолжительности воздействия может стать опасным.

ГОСТ 12.003-74 «ССБТ. Опасные и вредные производственные факторы. Классификация» приводит основанную на природе воздействия классификацию опасных и вредных производственных факторов.

* физические: неблагоприятные (повышенные или пониженные) значения физических свойств окружающей среды (температуры, влажности и подвижности воздуха, уровня шума, освещенности, электромагнитных полей, статического электричества, радиоактивного излучения и т.д.);
* химические: ряд веществ и соединений, которые могут оказывать разнообразные негативные воздействия на организм человека (токсические, раздражающие, сенсибилизирующие, канцерогенные, мутагенные вещества, а также влияющие на репродуктивную систему);
* биологические: наличие в воздухе рабочей зоны патогенных микроорганизмов и продуктов их жизнедеятельности;
* психофизиологические: физические (статические и динамические) и нервно-психические перегрузки (умственное перенапряжение, перенапряжение анализаторов, монотонность труда, эмоциональные перегрузки).

Пользователи ПЭВМ в основном сталкиваются с воздействием физических и психофизиологических опасных и вредных производственных факторов. Наиболее значительные из этих факторов, их нормы и методы уменьшения их влияния рассмотрены ниже.

### 8.1.1 Параметры микроклимата

По определению, приведенному в СанПиН 2.2.4.548-96 «Гигиенические требования к микроклимату производственных помещений», микроклимат производственных помещений определяется действующим на организм человека сочетанием температуры, влажности и скорости движения воздуха, а также температурой окружающей поверхности.

При благоприятных сочетаниях параметров микроклимата человек испытывает состояние теплового комфорта, что является важным условием высокой производительности труда и предупреждения заболеваний.

Значительное отклонение микроклимата рабочей зоны от оптимального может быть причиной ряда физических нарушений в организме работающего, привести к резкому снижению работоспособности и даже к профессиональным заболеваниям. Так, при температуре воздуха более 300С наступает нарушение терморегуляции, что может привести к перегреву организма, который сильно сказывается на состоянии нервной системы и работоспособности человека. Длительное и сильное воздействие низких температур может вызвать охлаждение организма, которое характеризуется снижением частоты сердечных сокращений и развитием процессов торможения в коре головного мозга, что также ведет к уменьшению работоспособности.

Влажность воздуха определяется содержанием в нем водяных паров. Физиологически оптимальной является относительная влажность в пределах 40...60%. Повышенная влажность в сочетании с низкими температурами оказывает значительное охлаждающее действие, а в сочетании с высокими — способствует перегреванию организма. Относительная влажность менее 25% также неблагоприятна для человека, так как приводит к высыханию слизистых оболочек и к снижению защитной деятельности мерцательного эпителия верхних дыхательных путей.

Подвижность воздуха также влияет на организм человека. Легкое движение воздуха при обычных температурах способствует хорошему самочувствию, сдувая обволакивающий человека насыщенный водяными парами и перегретый слой воздуха. Большая скорость движения воздуха, особенно в условиях низких температур, вызывает увеличение теплопотерь и ведет к сильному охлаждению организма.

Под оптимальными микроклиматическими условиями понимают такие сочетания параметров микроклимата, которые при длительном и систематическом воздействии на человека обеспечивают сохранение нормального функционального и теплового состояния организма без напряжения механизма терморегуляции. Они обеспечивают ощущение теплового комфорта и создают предпосылки для хорошей работоспособности.

К способам нормализации микроклимата рабочего места относятся:

* обеспечение правильного устройства рабочего места: площадь на одно рабочее место ПЭВМ должна составлять не менее 6,0 м2, а объем не менее 20,0 м3 (СанПиН 2.2.4.548-96);
* обеспечение рациональной вентиляции, отопления и кондиционирования;
* обеспечение тепловой изоляции оборудования.

Системы вентиляции, отопления и кондиционирования воздуха в помещениях с ПЭВМ должны быть выполнены в соответствии с требованиями СанПиН 2.2.4.548-96 «Гигиенические требования к микроклимату производственных помещений» и СНиП 41-01-2003 «Отопление, вентиляция и кондиционирование».

Оптимальные и допустимые параметры микроклимата, указанные в СанПиН 2.2.4.548-96 «Гигиенические требования к микроклимату производственных помещений», сведены в таблицы 8.1 и 8.2 соответственно. Данные приведены для категории работ Iа по уровню энергозатрат, т.к. работа оператора ПЭВМ относится именно к ней.

Таблица 8.1 — Оптимальные параметры микроклимата

|  |  |  |  |
| --- | --- | --- | --- |
| Период года | Температура воздуха, °C | Относительная влажность воздуха, % | Скорость движения воздуха, м/с |
| Холодный | 22-24 | 40-60 | 0,1 |
| Теплый | 23-25 | 40-60 | 0,1 |

Таблица 8.2 — Допустимые параметры микроклимата

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Период года | Температура воздуха, °C | Относительная влажность воздуха, % | Скорость движения воздуха, м/с | |
| Для диапазона температур воздуха ниже оптимальных величин, не более | Для диапазона температур воздуха выше оптимальных величин, не более |
| Холодный | 20-25 | 15-75 | 0,1 | 0,1 |
| Теплый | 21-28 | 15-75 | 0,1 | 0,2 |

Для поддержания оптимального микроклимата рабочее помещение снабжено кондиционерами. Температура воздуха находится в пределах 23-25 °С в теплое время года и в пределах 22-24 °С в холодное время года. Относительная влажность воздуха 40-60%, скорость движения воздуха не превышает 0,1 м/с. Геометрические размеры помещения (10 x 6 x 3 м) обеспечивают достаточную площадь (7,5 м2) и объем (22,5 м3) на одно рабочее место. Система кондиционирования, вентиляции и отопления соответствует требованиям СНиП 41-01-2003 «Отопление, вентиляция и кондиционирование»

Проведя сравнение реальных микроклиматических условий с требуемыми, можно сделать вывод, что условия труда соответствуют допустимым параметрам микроклимата.

### 8.1.2 Уровень шума

Всякое нежелательное для человека звуковое воздействие называется шумом. Основными источниками шума в рабочем помещении являются работающие приборы, ПЭВМ, внешние шумы, система вентиляции. Шум оказывает негативное влияние не только на орган слуха, но и на весь организм человека. Он нарушает работу нервной системы, влияет на параметры дыхания, пульса, способствует нарушению обмена веществ, возникновению сердечнососудистых заболеваний, гипертонической болезни. Шум усиливает действия других вредных факторов, повышает общую заболеваемость, снижает работоспособность человека, безопасность труда. В условиях интенсивного шума на выполнение работ требуется больше физических и нервно-психических усилий. Поэтому уровни шума на рабочих местах необходимо контролировать и принимать меры к устранению нарушений существующих норм и правил.

Допустимые уровни шума устанавливаются в зависимости от назначения помещений или территорий и времени суток и должны соответствовать требованиям СН 2.2.4/2.1.8.562-96 «Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки». В помещениях программистов и операторов ПЭВМ уровень звука не должен превышать 50 дБА.

Классификация способов и средств защиты от шума приведена в ГОСТ 12.1.029-80 «Средства и методы защиты от шума. Классификация».

В нашем случае рабочее место расположено в помещении, огражденном от повышенных уровней шума, то есть реализовано архитектурно-планировочное средство коллективной защиты (планировка зданий, размещение оборудования и рабочих мест, пути транспортных потоков, размещение шумных производств организовано таким образом, чтобы исключить превышение допустимых уровней шума). Уровень шума составляет 48 дБА, что не превышает установленных санитарных норм. Принятие дополнительных мер по снижению уровня шума не требуется.

### 8.1.3 Освещенность

Недостаточная освещенность рабочего места обычно связана с неправильным выбором и размещением осветительных приборов. Правильно спроектированное и выполненное освещение действует тонизирующе, создает хорошее настроение, улучшает протекание основных процессов высшей нервной деятельности.

Работа на ПЭВМ может осуществляться при наличии естественного и искусственного освещения. Величина коэффициента естественной освещенности должна соответствовать нормативным уровням по СНиП 23-05-95 «Естественное и искусственное освещение».Для III светового пояса КЕО должен быть не ниже 1,2% в зонах с устойчивым снежным покровом и не ниже 1.5% на остальной территории страны. Рекомендуется рабочие места по отношению к световым проёмам располагать так, чтобы естественный свет падал сбоку слева.

Гигиенические требования к освещенности могут быть сведены к следующим:

* спектральный состав света искусственного источников должен приближаться к солнечному;
* освещенность на поверхности рабочего стола в зоне размещения рабочего документа должна быть 300...500 лк;
* яркость светящихся поверхностей, находящихся в поле зрения (окна, светильники и др.) должна быть не более 200 кд/м2;
* должны быть обеспечены равномерность и устойчивость уровня освещенности: соотношение яркости между рабочими поверхностями не должно превышать 3:1...5:1, а между рабочими поверхностями и поверхностями стен и оборудования 10:1; коэффициент пульсации не должен превышать 5%, что должно обеспечиваться использованием газоразрядных ламп в светильниках общего и местного пользования.

Наименьший объект различения на экране монитора составляет 0,3...0,5 мм, что соответствует III разряду зрительной работы. Контраст объекта различения с фоном — средний, фон — темный, и, следовательно, номинальная освещенность должна составлять 300 лк (СанПиН 2.2.2 / 2.4.1340-03).

В рабочем помещении для освещения используются светильники серии ЛП036 с зеркализованными решетками. Параметры светильников и их расположение удовлетворяют перечисленным выше требованиям.

### 8.1.4 Возможность поражения электрическим током

Электроопасность является особо опасным фактором, потому что электрический ток невидим, без цвета и запаха. Воздействия электрического тока на организм человека носит разносторонний характер (биологическое, термическое, электролитическое воздействия). Наиболее часто поражение электрическим током имеет вид электрического удара. Тяжесть поражения электрическим током зависит от значения силы тока, электрического сопротивления тела человека, длительности протекания тока через человека, от рода и частоты тока, индивидуальных свойств. Наиболее опасен переменный ток с частотой 20-100 Гц.

Основными причинами электротравм при работе с ПЭВМ являются следующие:

* случайное прикосновение человека к токоведущим частям, находящимся под напряжением;
* прикосновение к металлическим нетоковедущим частям (корпусу, элементам), которые могут оказаться под напряжением случайно при повреждении изоляции.

Электробезопасность регламентируется ГОСТ 12.1.019-79 «ССБТ. Электробезопасность. Общие требования», согласно которому необходимой мерой обеспечения электробезопасности в сети с глухозаземленной нейтралью является зануление корпусов электроаппаратуры. Все зануления выполнены в соответствии с требованиями Правила устройства электроустановок (ПУЭ) от 01.01.03.

Опасность поражения электрическим током во многом зависит и от условий эксплуатации электроаппаратуры, характеризующих помещение. К ним относятся:

* сырость (относительная влажность воздуха превышает 75%);
* повышенная температура воздуха, превышающая +35°С;
* токопроводящий (без изолирующего покрытия) пол;
* токопроводящая пыль;
* химически активная или органическая среда (агрессивные пары, отложения или плесень, разрушающие изоляцию и токоведущие части);
* возможность одновременного прикосновения человека к имеющим соединение с землей металлоконструкциям зданий, механизмов и т.п. и металлическим элементам электроустройств, которые могут оказаться под напряжением при повреждении рабочей изоляции.

В рассматриваемом помещении поддерживается относительно постоянная температура и влажность воздуха за счет центрального водяного отопления и системы кондиционирования. Все токоопасные поверхности изолированы. В соответствии с ГОСТ 12.4.124-83 «ССБТ. Средства защиты от статического электричества» пол покрыт деревянным паркетом, что позволяет снизить величину статического электричества. Все электрические розетки имеют зануление. Таким образом, рассматриваемое помещение в отношении опасности поражения человека электрическим током классифицируется как помещения без повышенной опасности, в которых отсутствуют условия, создающие повышенную или особую опасность.

Основными техническими средствами защиты человека от поражения электрическим током, используемыми отдельно или в сочетании друг с другом, являются:

* защитное заземление;
* защитное зануление;
* защитное отключение;
* электрическое разделение сети;
* малое напряжение;
* электрозащитные средства;
* уравнивание потенциалов;
* двойная изоляция;
* предупредительная сигнализация;
* блокировка;
* знаки безопасности.

Для защиты пользователя ПЭВМ от поражения электрическим током в рассматриваемом помещении применяется двойная и усиленная изоляция, а также, согласно ГОСТ 12.1.019-79 «ССБТ. Электробезопасность. Общие требования», защитное зануление (провод ПЭВМ для подключения к питающей сети имеет дополнительную зануляющую жилу и евровилку). В комнате применяются евророзетки с клеммой зануления. При защитном занулении безопасность человека, касающегося корпуса поврежденной установки, обеспечивается за счет уменьшения времени воздействия опасного напряжения, действующего до момента срабатывания элемента защиты. Согласно ПУЭ-07 на вводе в помещение необходимо установить устройство защитного отключения с номинальным отключающим дифференциальным током не более 30 мА.

Основным организационным мероприятием, позволяющим повысить электробезопасность, является периодически проводимый инструктаж и обучение безопасным методам труда, а так же проверка знаний правил безопасности и инструкций.

### 8.1.5 Психофизиологические факторы

Психофизиологические факторы, воздействующие на пользователя, приводят к его физическим и нервно-психическим перегрузкам.

При работе с ПЭВМ человек подвержен психоэмоциональным перегрузкам, в частности — умственному перенапряжению, перенапряжению зрительных и слуховых анализаторов, монотонного труда, эмоциональных перегрузок. Все это может привести к заболеваниям нервной системы.

Характерной при работе с ПЭВМ является такая физическая перегрузка, как длительное статическое напряжение мышц при продолжительном сидении в одной и той же позе при необходимости постоянного наблюдения за экраном (напряжение мышц шеи, ухудшение мозгового кровообращения) и при наборе большого количества знаков за рабочую смену (перенапряжение мышц плечевого пояса и рук). Статическим перенапряжениям мышц способствуют неудовлетворительные эргономические параметры рабочего места и его компонентов: отсутствие возможности регулировки параметров стула, высоты рабочей поверхности стола, неудобное расположение устройств ввода-вывода ПЭВМ, отсутствие регламентированных перерывов, невыполнение специальных упражнений для снятия напряжения и расслабления мышечных групп плечевого пояса, рук, шеи, спины, улучшения кровообращения. Поэтому стоит уделить внимание эргономизации рабочего места пользователя.

При организации рабочего места пользователя ПЭВМ следует обеспечить соответствие конструкции всех элементов рабочего места и их взаимного расположения эргономическим требованиям с учетом характера выполняемой деятельности, комплектности технических средств, форм организации труда и основного рабочего положения пользователя. Эти вопросы нашли отражение в СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к ПЭВМ и организации работы».

Основными элементами рабочего места, оснащенного ПЭВМ, является рабочий стол, рабочий стул, экран дисплея и клавиатура.

Рабочий стол для ПЭВМ может быть любой конструкции, отвечающей современным требованиям эргономики с учетом характера выполняемой деятельности. Он должен обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей (дисплея, клавиатуры и др.), а также возможность выполнения трудовых операций в пределах досягаемости.

Поверхности рабочих столов с ПЭВМ должны быть цвета натуральной древесины, голубого, светло-зеленого или светло-серого цвета. Требуемые размеры рабочей поверхности стола определяются габаритами технических средств вычислительной техники и должны быть не менее 800 мм в ширину и глубину.

Высота рабочей поверхности стола должна регулироваться в пределах от 680 до 800 мм в зависимости от роста пользователя, а при отсутствии такой возможности составлять 725 мм. Стол пользователя должен иметь пространство для ног высотой не менее 600 мм, шириной – не менее 500 мм.

Рабочий стул (кресло) должен обеспечивать поддержание рациональной рабочей позы в зависимости от характера работы с учетом роста пользователя, а его конструкция должна обеспечивать возможность изменения позы пользователя с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения утомления.

Установка правильной высоты сиденья является первоочередной задачей при организации рабочего места, так как этот параметр определяет прочие пространственные параметры — высоты положения экрана, клавиатуры, поверхности для записей и пр. Одним из способов снижения утомления мышц во время сидения является создание опоры туловища на спинку сидения. Самым эффективным является упор в поясничной области, который поддерживает естественный изгиб позвоночника, и менее эффективным – упор в области лопаток.

Для выполнения указанных требований рабочий стул должен быть подъемно-поворотным, регулируемым по высоте и углам наклона сидения и спинки. Поверхность элементов стула должна быть полумягкой, с нескользящим, неэлектризующимся и воздухонепроницаемым покрытием, обеспечивающим легкую чистку от загрязнения.

Экран должен находиться от глаз пользователя на оптимальном расстоянии 600-700 мм с учетом размеров знаков и символов, но не ближе 500 мм.

Уровень глаз пользователя должен приходиться на центр экрана или на 2/3 его высоты. Линия взора должна быть перпендикулярна центру экрана, а оптимальное ее отклонение в вертикальной плоскости должно находиться в пределах +5, а допустимое – +10 градусов. Оптимальный обзор в горизонтальной плоскости от центральной оси экрана должен быть в пределах +30 градусов.

Для уменьшения негативных последствий работы с ПЭВМ следует выбирать рациональные режимы труда и отдыха, проводить комплексные оздоровительно-профилактические мероприятия, правильно организовывать рабочее место пользователя. Необходимо проводить регулярные психоэмоциональные разгрузки в виде аутотренинговых упражнений.

Учитывая изложенные выше требования к рабочему месту пользователя и в соответствии с СанПиН 2.2.2 / 2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы» был разработан примерный план организации рабочего места, представленный на рисунке 8.1.

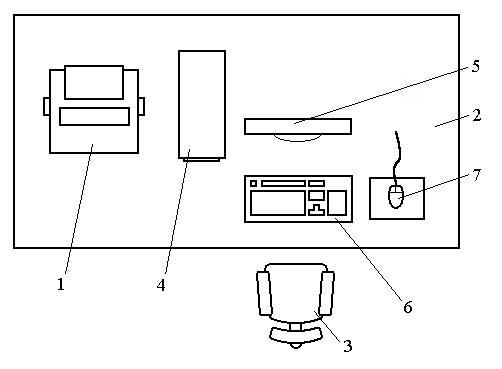


Рисунок 8.1 — Организация рабочего места пользователя ПЭВМ

Обозначения:

1 — принтер,

2 — рабочий стол,

3 — кресло,

4 — системный блок,

5 — монитор,

6 — клавиатура,

7 — мышка.

## 8.2 Обеспечение пожарной безопасности на рабочем месте

В рассматриваемом помещении находится много горючих предметов (бумага, столы, шкафы, изоляция токопроводящих элементов и др.). Пожар в комнате может привести к чрезвычайно неблагоприятным последствиям (потеря ценной информации, порча имущества, гибель людей и т.д.). Поэтому пожарная безопасность в данном помещении является очень актуальной.

Опасными факторами пожара, воздействующими на людей и материальные ценности, согласно ГОСТ 12.1.004-91 «ССБТ. Пожарная безопасность. Общие требования» являются:

* пламя и искры;
* повышенная температура окружающей среды;
* токсичные продукты горения и термического разложения;
* дым;
* пониженная концентрация кислорода.

К вторичным проявлениям опасных факторов пожара относятся:

* осколки и части разрушившихся аппаратов, конструкций;
* токсичные вещества и материалы, вышедшие из разрушенных аппаратов;
* электрический ток, возникший в результате выноса высокого напряжения на токопроводящие части конструкций, аппаратов;
* опасные факторы взрыва, происшедшего вследствие пожара;
* огнетушащие вещества.

Как известно, пожар может возникнуть при взаимодействии горючих веществ (деревянный стол, стул, бумага, изоляция электрических проводов, строительные материалы для акустической и эстетической отделки помещений, дверь, полы), окислителя (кислород) и источников зажигания.

Причины возникновения пожара можно разделить на причины электрического и неэлектрического характера.

К факторам возникновения пожара электрического характера относятся следующие:

* короткое замыкание;
* перегрузки в сети;
* большие переходные сопротивления;
* отсутствие устройств защиты от перегрузок по току и напряжению;
* перегретые элементы в электронных схемах ПЭВМ, устройств электропитания, кондиционирования воздуха;
* электрические искры и дуги, способные вызвать загорания горючих материалов.

К причинам неэлектрического характера относятся:

* неосторожное обращение с огнем (курение в не отведенном для этого месте);
* использование оборудования не по назначению;
* применение неисправного оборудования;
* оставление без присмотра различных отопительных приборов.

Противопожарная защита — это комплекс организационных и технических мероприятий, направленных на обеспечение безопасности людей, на предотвращение пожара, ограничение его распространения, а также на создание условий для успешного тушения пожара.

По пожарной опасности помещения с ПЭВМ согласно НПБ 105-03 «Определение категорий помещений, зданий и наружных установок по взрывопожарной и пожарной опасности» относятся к категории В (пожароопасные — характеризуются наличием горючих жидкостей с температурой вспышки паров выше 61°С; горючей пыли или волокон, нижний предел взрываемости которых более 65 г/м3 к объему воздуха; веществ, способных только гореть при взаимодействии с кислородом воздуха или друг с другом; твердых сгораемых веществ и материалов).

Пожароопасная категория помещения (В1-В4) зависит от величины его пожарной нагрузки Q [МДж], которая включает в себя различные сочетания горючих и трудногорючих жидкостей, твёрдых веществ и материалов:

(6.2.1)

Q = ∑Gi ∙ Qнi ,

где:

* Gi—масса i-гo материала пожарной нагрузки, кг;
* QHi— низшая теплота сгорания i-гo материала, МДж/кг.

Определение пожароопасной категории помещения (В1-В4) производится путём сравнения максимального значения удельной пожарной нагрузкиgпомещения с нормируемой величиной удельной пожарной нагрузки gн. Удельная пожарная нагрузка g[МДж/м2] определяется выражением из соотношения:

гдеS – площадь размещения пожарной нагрузки, м2 .

Пожарную нагрузку составляют горючие материалы: изоляционные материалы проводов и кабелей (G=60 кг, =39 МДж/кг), деревянные шкафы, столы, паркет (G= 469 кг, = 16,6 МДж/кг), , горючие элементы оргтехники (G=100 кг, = 14,4 МДж/кг), различная документация и литература (G= 110 кг, = 13,4 МДж/кг).

Пожарная нагрузка равна:

Удельная пожарная нагрузка равна:

Сравнив эту величину с нормируемой величиной пожарной нагрузки, видно, что рабочее помещение относится к категории В3 (gн = 181-1400 МДж/м2).

Исходя из норм пожарной безопасности для помещения с ПЭВМ площадью меньше 100 м2, требуется один углекислотный огнетушитель типа ОУ-5 или ОУ-8, с помощью которого можно тушить возгорания различных материалов и установок напряжением до 1000 В.

Мероприятия по устранению причин пожара подразделяются на:

* организационные (обучение противопожарным правилам, проведение бесед, лекций, инструктажей);
* эксплуатационные (предусматривают правильную эксплуатацию машин, оборудования, правильное содержание зданий, территории);
* технические (соблюдение противопожарных норм и правил);
* режимные (запрещение курения в неустановленных местах);

Одним из средств предупреждения (профилактики) пожаров является проведение инструктажей с персоналом.

Противопожарный инструктаж — доведение до работников организаций основных требований пожарной безопасности, изучение пожарной опасности технологических процессов производства, оборудования, средств противопожарной защиты и действий в случае возникновения пожара. Проводится со всеми работниками организации по утвержденным программам и в порядке, определяемом руководителем (собственником). При проведении инструктажей по пожарной безопасности следует учитывать специфику деятельности организации. По характеру и времени проведения противопожарный инструктаж подразделяется на:

* вводный;
* первичный на рабочем месте;
* повторный;
* внеплановый;
* целевой.

Для борьбы с пожаром в помещении, где при работе эксплуатируются несколько ПЭВМ, должна иметься противопожарная сигнализация, средства тушения огня, должен быть предусмотрен запасной выход и план эвакуации. Рубильники выключения тока должны находиться в легкодоступных местах. Также должна существовать возможность полного отключения электропитания одним рубильником. Установка самых простых и дешевых датчиков дыма в электрических распределительных шкафах позволяет предотвратить пожар ещё до возникновения возгорания.

Согласно требованиям Правил пожарной безопасности ППБ 01-03 на видных местах вывощены планы эвакуации людей при пожаре. Пути эвакуации освещены в соответствии с требованиями СНиП 23-05 «Естественное и искусственное освещение». Знаки обозначения мест выхода при эвакуации оформлены в соответствии с НПБ 104-03 «Системы оповещения и управления эвакуацией людей при пожарах в зданиях и сооружениях».

## 8.3 Экологичность проекта

Данный программный модуль проектируется на ПЭВМ и предназначен для работы на ПЭВМ. Известно, что ПЭВМ является источником вредных электромагнитных излучений и других неблагоприятных факторов, представляет некоторую опасность и вредность для людей, обслуживающих или же находящихся в непосредственной близости от нее. Однако разработанная в данном дипломном проекте автоматизация учета сделок с ценными бумагам, не оказывает вредного влияния на окружающую среду, так как в ПЭВМ не применяются комплектующие из радиоактивных и других опасных веществ, она не излучает в радиодиапазоне. В процессе разработки и эксплуатации нет выброса вредных веществ в атмосферу.

Важными требованиями по предотвращению загрязнения окружающей среды являются своевременная уборка отработанной бумаги от принтера в специальные мусорные контейнеры, препятствующие распространению её на прилегающей территории, а также правильная утилизация изношенных частей ПЭВМ. Немаловажным является и снижение электропотребления ПЭВМ путем перевода ее в режим ожидания или ее выключения, если ПЭВМ не используется.

Таким образом, данный проект полностью удовлетворяет требованиям по охране окружающей среды, предъявляемым к его проектированию и эксплуатации, то есть является экологически безопасным.

# Заключение

В ходе дипломного проектирования был разработан программный модуль ведения дилерской деятельности, интегрируемый в банковскую автоматизированную систему учета операций с ценными бумагами.

Была обоснована целесообразность собственной разработки и описаны ее преимущества перед готовыми решениями.

Были описаны преимущества собственной разработки по сравнению с готовыми решениями, обоснованы выбранные для использования программные средства.

Для решения задачи были составлены высокоуровневые требования к программному модулю, основанные на информации заказчика. Было проведено проектирование клиентской и серверной части модуля, отражающее реализацию данных требований с учетом существующей архитектуры системы.

На данную разработку распространены принципы, уже используемые в системе. Организовано взаимодействие с другими банковскими системами (как экспорт, так и импорт). Реализована поддержка механизма учета данных с минимизацией ошибок. Существует возможность построения отчетности для сотрудников банка и регулирующих органов.

Процесс испытания программного продукта происходил на тестовом стенде, воспроизводящем конфигурацию рабочих станций, на которых будут развернуты серверная и клиентская часть приложения в банке. Для упорядочивания и систематизации тестирования был создан приемочный тест-план, который опирается на функциональные требования к модулю и полностью покрывает их. В ходе испытаний обнаруженные ошибки и недочеты были устранены, и, таким образом, тестирование завершилось успешно.

Кроме того, была составлена программная документация: руководство по сборке для программиста, руководство по развертыванию для системного администратора, а также руководство пользователя, которое описывает требуемые последовательности действий для выполнения функциональных операций, добавленных в данном модуле.

Предварительно был подготовлен план выполнения работ, наглядно оформленный в виде ленточного графика. Была детально рассчитана себестоимость разработки приложения и срок окупаемости, на основе чего был сделан вывод о экономической выгоде данного решения.

С целью выявить опасные и вредные факторы на рабочем месте пользователей приложения был произведен анализ условий труда оператора ПК. Также была оценена экологичность дипломного проекта.

Таким образом, программный модуль полностью удовлетворяет предъявляемым к нему требованиям. По завершению разработки данное решение было успешно внедрено.

# Список использованных источников

1. Дипломное проектирование на кафедре ЭВМ: методические указания / Рязан. гос. радиотехн. ун-т; cост.: М. Б. Никифоров, Н. В. Чирков. – Рязань, 2011. 32 с. № 4448
2. Microsoft Developer Network: [Электронный ресурс]. URL: <http://msdn.microsoft.com/>
3. **Stack Overflow** — a question and answer site for professional and enthusiast programmers: [Электронный ресурс]. URL: <http://stackoverflow.com/>
4. Роланд Ф. Д. Основные концепции баз данных. М.: Вильямс, 2002. 253 с.
5. Фленов М. Е. Transact-SQL. — СПб.: БХВ-Петербург, 2006. — 576 с.:ил.
6. Приемы объектно-ориентированного проектирования. Пер. с англ. Издательство «Питер», 2011. 367 с.
7. Макконнелл С. Совершенный код / Пер. с англ. М.: Издательство "Русская редакция", 2012. 896 с.
8. Новиков Д.А. Управление проектами: организационные механизмы, М: ПМСОФТ, 2007. 378 с.
9. Кент Б. Экстремальное программирование. Пер. с англ. Издательство "Питер", 2002. 222 с.
10. Троелсен Э. Язык программирования C# 2008 и платформа .Net. Пер. с англ. М.: Издательство "Вильямс", 2010. 1344 с.
11. Койн М. Scrum. Гибкая разработка ПО. Пер. с англ. М.: Издательство "Вильямс", 2011. 12984 с.
12. Выполнение экономической части дипломного проекта: Методические указания / Рязан. гос. радиотехн. ун-т; Сост.: Л. В. Васина, Е. Н. Евдокимова, А. В. Рыжкова. Рязань, 2008. 36 с. № 4020
13. Налоговый Кодекс Российской Федерации.
14. Безопасность и экологичность проекта: Методические указания для дипломников / Ю. В. Зайцев, Н. В. Веселкин, С. И. Кордюков, А. Я. Агеев; Под ред. Ю. В. Зайцева. Рязан. гос. радиотехн. ун-т. Рязань, 2006. 24 с. № 3860
15. Обеспечение пожарной безопасности: Методические указания для дипломников / Рязан. гос. радиотехн. ун-т; Сост.: Н. В. Весёлкин, В. А. Крысанов. Рязань, 2011. 20 с. № 4467
16. Гигиенические требования к персональным электронно-вычислительным машинам и организации работы. СанПиН 2.2.2/2.4.1340-03
17. Гигиенические требования к микроклимату производственных помещений. СанПиН 2.2.4.548-96
18. Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки. СН 2.2.4/2.1.8.562-96
19. ГОСТ 12.003-74 «ССБТ. Опасные и вредные производственные факторы. Классификация»

# Приложение

## Скрипт создания БД

-----------------------------------------------------------------------------

-- REG\_DEAL\_P\_RAW creating

-----------------------------------------------------------------------------

IF NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW')

BEGIN

PRINT 'Creating Table [REG\_DEAL\_P\_RAW]'

CREATE TABLE dbo.[REG\_DEAL\_P\_RAW]

(

[ID] int not null IDENTITY(1,1),

[Operation] nvarchar(2) not null,

[BatchID] int not null,

[OrderNo] int not null,

[ScriptID] int not null,

[TradingCenterID] int null,

[DealerID] int null,

[CpID] int null,

[Quantity] int not null,

[TicketID] nvarchar(50) null,

[PricePcent] decimal(17, 9) null,

[PriceCcy] decimal(17, 9) null,

[CcyOfPrice] nchar(3) null,

[CcyOfPayment] nchar(3) null,

[Auction] bit not null CONSTRAINT [DF\_REG\_DEAL\_P\_RAW\_Auction] DEFAULT(0),

[RepoID] int null,

[IsRepo] as CAST(CASE WHEN [RepoID] > 0 THEN 1 ELSE 0 END as bit),

[CouponAmount] decimal(12, 2) not null,

[DealValue] decimal(38, 7) not null,

[Commission] decimal(12, 2) null,

[DealValueIncludingCouponRUB] decimal(38, 7) null,

[CouponPaymentCcy] decimal(38, 7) null,

[DealValuePaymentCcy] decimal(38, 7) null,

[CommissionPaymentCcy] decimal(38, 7) null,

[TradeDT] datetime not null,

[TradeTime] datetime not null,

[TcPlanPaymentDT] datetime null,

[TcRealPaymentDT] datetime null,

[PlanSettlementDT] datetime null,

[RealSettlementDT] datetime null,

[PlanPaymentDT] datetime null,

[RealPaymentDT] datetime null,

[DepoReceiptProperties] nvarchar(50) null,

[PaymentOrderProperties] nvarchar(50) null,

[Status] smallint not null CONSTRAINT [DF\_REG\_DEAL\_P\_RAW\_Status] DEFAULT (0),

[ExchangeID] nvarchar(20) null,

[DepoLocationID] int null,

[LocationID] int null,

[NostroAccountID] int null,

[CommissionAccountID] int null,

[AirCashAccountID] int null,

[AirSecuritiesAccountID] int null,

[SettleModeID] int null,

CONSTRAINT [PK\_REG\_DEAL\_P\_RAW] PRIMARY KEY CLUSTERED ([ID])

)

END

GO

-----------------------------------------------------------------------------

-- REG\_DEAL\_P\_SPLIT creating

-----------------------------------------------------------------------------

IF NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT')

BEGIN

PRINT 'Creating Table [REG\_DEAL\_P\_SPLIT]'

CREATE TABLE dbo.[REG\_DEAL\_P\_SPLIT]

(

[ID] int not null IDENTITY(1,1),

[RawID] int null, -- reference to [REG\_DEAL\_P\_RAW]

[Operation] nvarchar(3) not null,

[BatchID] int null,

[OrderNo] int null,

[ScriptID] int null,

[TradingCenterID] int null,

[DealerID] int null,

[CpID] int null,

[PortfolioID] int null,

[Quantity] int not null,

[TicketID] nvarchar(50) null,

[PricePcent] decimal(17, 9) null,

[PriceCcy] decimal(17, 9) null,

[CcyOfPrice] nchar(3) null,

[CcyOfPayment] nchar(3) null,

[Auction] bit not null CONSTRAINT [DF\_REG\_DEAL\_P\_SPLIT\_Auction] DEFAULT(0),

[RepoID] int null,

[IsRepo] as CAST(CASE WHEN [RepoID] > 0 THEN 1 ELSE 0 END as bit),

[CouponAmount] decimal(12, 2) not null,

[DealValue] decimal(38, 7) not null,

[Commission] decimal(12, 2) null,

[DealValueIncludingCouponRUB] decimal(38, 7) null,

[CouponPaymentCcy] decimal(38, 7) null,

[DealValuePaymentCcy] decimal(38, 7) null,

[CommissionPaymentCcy] decimal(38, 7) null,

[TradeDT] datetime not null,

[TradeTime] datetime not null,

[TcPlanPaymentDT] datetime null,

[TcRealPaymentDT] datetime null,

[PlanSettlementDT] datetime null,

[RealSettlementDT] datetime null,

[PlanPaymentDT] datetime null,

[RealPaymentDT] datetime null,

[DepoReceiptProperties] nvarchar(50) null,

[PaymentOrderProperties] nvarchar(50) null,

[ParentDealID] int null,

[EntryDT] datetime null,

[Status] smallint not null CONSTRAINT [PK\_REG\_DEAL\_P\_SPLIT\_Status] DEFAULT (0),

[IsTechnical] bit not null CONSTRAINT [DF\_REG\_DEAL\_P\_SPLIT\_IsTechnical] DEFAULT (0),

[PortfolioCat] varchar(20) null,

CONSTRAINT [PK\_REG\_DEAL\_P\_SPLIT] PRIMARY KEY CLUSTERED ([ID])

)

END

GO

-----------------------------------------------------------------------------

-- REG\_EVENT\_P creating

-----------------------------------------------------------------------------

IF NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'REG\_EVENT\_P')

BEGIN

PRINT 'Creating Table [REG\_EVENT\_P]'

CREATE TABLE dbo.[REG\_EVENT\_P]

(

[ID] int not null identity(1,1),

[Type] nvarchar(3) not null,

[ScriptID] int not null,

[DepoLocationID] int not null,

[NostroAccountID] int not null,

[PerPiece] decimal(15, 9) not null,

[PlanDT] datetime not null, -- Payment or Settlement plan date

[RealDT] datetime null, -- Payment or Settlement real date

[EntryDT] datetime null, -- date of event creation

[Properties] nvarchar(50) null, -- Payment order or Depo reciept properties

[EventNo] nvarchar(50) null,

[LastChangerName] nvarchar(50) null,

[LastChangeDT] datetime null,

[LastAuthName] nvarchar(50) null,

[LastAuthDT] datetime null,

[InputerName] nvarchar(50) null,

[InputDT] datetime null,

[Status] smallint null,

[IsPayment] bit not null CONSTRAINT [PK\_REG\_EVENT\_P\_IsPayment] DEFAULT (1),

[PlanPaymentDT] AS CASE [IsPayment] WHEN 1 THEN [PlanDT] END,

[PlanSettlementDT] AS CASE [IsPayment] WHEN 0 THEN [PlanDT] END,

[RealSettlementDT] AS CASE [IsPayment] WHEN 0 THEN [RealDT] END,

[DepoReceiptProperties] AS CASE [IsPayment] WHEN 0 THEN [Properties] END,

CONSTRAINT [PK\_REG\_EVENT\_P] PRIMARY KEY ([ID])

)

END

GO

-----------------------------------------------------------------------------

-- REG\_EVENT\_P\_PEND creating

-----------------------------------------------------------------------------

IF NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'REG\_EVENT\_P\_PEND')

BEGIN

PRINT 'Creating Table [REG\_EVENT\_P\_PEND]'

CREATE TABLE dbo.[REG\_EVENT\_P\_PEND]

(

[ID] int not null,

[NostroAccountID] int not null,

[RealDT] datetime null, -- Payment or Settlement real date

[Properties] nvarchar(50) null, -- Payment or Depo reciept order properties

[LastChangerName] nvarchar(50) null,

[LastChangeDT] datetime null,

[LastAuthName] nvarchar(50) null,

[LastAuthDT] datetime null,

[InputerName] nvarchar(50) null,

[InputDT] datetime null,

[Status] smallint null,

CONSTRAINT [PK\_REG\_EVENT\_P\_PEND] PRIMARY KEY ([ID])

)

END

GO

-----------------------------------------------------------------------------

-- REG\_EVENT\_LOCATION\_P creating

-----------------------------------------------------------------------------

IF NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'REG\_EVENT\_LOCATION\_P')

BEGIN

PRINT 'Creating Table [REG\_EVENT\_LOCATION\_P]'

CREATE TABLE dbo.[REG\_EVENT\_LOCATION\_P]

(

[ID] int not null identity(1,1),

[EventID] int not null,

[LocationID] int not null,

[AirSecuritiesAccountID] int null,

[Quantity] int not null,

CONSTRAINT [PK\_REG\_EVENT\_LOCATION\_P] PRIMARY KEY ([ID])

)

END

GO

-----------------------------------------------------------------------------

-- REG\_EVENT\_PARTIAL\_PAYMENT\_P creating

-----------------------------------------------------------------------------

IF NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'REG\_EVENT\_PARTIAL\_PAYMENT\_P')

BEGIN

PRINT 'Creating Table [REG\_EVENT\_PARTIAL\_PAYMENT\_P]'

CREATE TABLE dbo.[REG\_EVENT\_PARTIAL\_PAYMENT\_P]

(

[ID] int not null identity(1,1),

[EventID] int not null,

[RealPaymentDT] datetime null,

[Amount] decimal(28, 7) not null,

[LastChangerName] nvarchar(50) null,

[LastChangeDT] datetime null,

[LastAuthName] nvarchar(50) null,

[LastAuthDT] datetime null,

[InputerName] nvarchar(50) null,

[InputDT] datetime null,

[PaymentOrderProperties] nvarchar(50) null,

CONSTRAINT [PK\_REG\_EVENT\_PARTIAL\_PAYMENT\_P] PRIMARY KEY ([ID])

)

END

GO

-----------------------------------------------------------------------------

-- REG\_EVENT\_PARTIAL\_PAYMENT\_P\_PEND creating

-----------------------------------------------------------------------------

IF NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'REG\_EVENT\_PARTIAL\_PAYMENT\_P\_PEND')

BEGIN

PRINT 'Creating Table [REG\_EVENT\_PARTIAL\_PAYMENT\_P\_PEND]'

CREATE TABLE dbo.[REG\_EVENT\_PARTIAL\_PAYMENT\_P\_PEND]

(

[ID] int not null,

[EventID] int not null,

[RealPaymentDT] datetime null,

[Amount] decimal(28, 7) not null,

[Action] tinyint not null,

[LastChangerName] nvarchar(50) null,

[LastChangeDT] datetime null,

[LastAuthName] nvarchar(50) null,

[LastAuthDT] datetime null,

[InputerName] nvarchar(50) null,

[InputDT] datetime null,

[PaymentOrderProperties] nvarchar(50) null

CONSTRAINT [PK\_REG\_EVENT\_PARTIAL\_PAYMENT\_P\_PEND] PRIMARY KEY ([ID])

)

END

GO

-----------------------------------------------------------------------------

-- GAAP.SCRIPTS\_KEPT\_P creating

-----------------------------------------------------------------------------

IF NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'SCRIPTS\_KEPT\_P' AND TABLE\_SCHEMA = 'GAAP')

BEGIN

PRINT 'Creating Table GAAP.[SCRIPTS\_KEPT\_P]'

CREATE TABLE [GAAP].[SCRIPTS\_KEPT\_P]

(

[ID] int not null IDENTITY(1,1),

[DealID] int not null,

[EntryDT] datetime not null,

CONSTRAINT [PK\_SCRIPTS\_KEPT\_P] PRIMARY KEY CLUSTERED ([ID])

)

END

GO

-----------------------------------------------------------------------------

-- GAAP.SCRIPTS\_HIST\_P creating

-----------------------------------------------------------------------------

IF NOT EXISTS(SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'SCRIPTS\_HIST\_P' AND TABLE\_SCHEMA = 'GAAP')

BEGIN

PRINT 'Creating Table GAAP.[SCRIPTS\_HIST\_P]'

CREATE TABLE [GAAP].[SCRIPTS\_HIST\_P]

(

[SKept\_ID] int not null,

[DT] datetime not null,

[Quantity] int not null,

CONSTRAINT [PK\_SCRIPTS\_HIST\_P] PRIMARY KEY CLUSTERED ([SKept\_ID], [DT])

)

END

GO

-----------------------------------------------------------------------------

-- GAAP.SETTLEMENTS\_P creating

-----------------------------------------------------------------------------

IF NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'SETTLEMENTS\_P' AND TABLE\_SCHEMA = 'GAAP')

BEGIN

PRINT 'Creating Table GAAP.[SETTLEMENTS\_P]'

CREATE TABLE [GAAP].[SETTLEMENTS\_P]

(

[SKept\_ID] int not null,

[DealID] int not null,

[Quantity] int not null,

[Type] varchar(10) not null,

[EntryDT] datetime not null,

CONSTRAINT [PK\_SETTLEMENTS\_P] PRIMARY KEY CLUSTERED ([SKept\_ID], [DealID])

)

END

GO

-------------------------------------------------

-- RAS.BSDEAL\_P creating

-------------------------------------------------

IF NOT EXISTS (SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'BSDEAL\_P')

BEGIN

PRINT 'Creating table BSDEAL\_P'

CREATE TABLE [RAS].[BSDEAL\_P]

(

[ID] int IDENTITY(1,1) NOT NULL,

[DispatchDate] datetime NOT NULL,

[SplitDealID] int NULL,

[LinkedDealID] int NULL,

[ParentID] int NULL,

[OrderNo] int NOT NULL,

[dealNo] as cast(

Convert(char(4), DatePart(Year, DispatchDate)) -- YYYY

+

Replicate('0', 3 - len(Convert(varchar(3), DatePart(dy, DispatchDate)))) + Convert(varchar(3), DatePart(dy, DispatchDate)) -- JJJ

+

RIGHT('0000' + CAST([OrderNo] as varchar), 4)

as varchar(20)

),

[Operation] char(2) NOT NULL,

[ScriptID] int NOT NULL,

[Quantity] int NOT NULL,

[PortfolioID] int NULL,

[CpID] int NULL,

[PortfolioCat] varchar(20) NOT NULL,

[SymbolsPortCat] char(2) NOT NULL,

[BookDT] datetime NOT NULL,

[SettleDT] datetime NOT NULL,

[AGGBookDT] datetime NULL,

[AGGSettleDT] datetime NULL,

[InOut] char(1) NOT NULL,

[ProductType] varchar(3) NOT NULL,

[PricePcent] float NOT NULL,

[FaceValue] decimal(15, 2) NULL,

[PrincipalValue] decimal(15, 2) NULL,

[CouponAmount] decimal(12, 2) NOT NULL,

[MiscCost] decimal(12, 2) NOT NULL,

[MiscCost5] decimal(12, 2) NULL,

[CashAcctNo] decimal(10, 0) NULL,

[Depo] varchar(10) NOT NULL,

[RepoID] int NULL,

[TradingCenterID] int NULL,

[IsCorrective] bit NOT NULL CONSTRAINT [DF\_BSDEAL\_P\_IsCorrective] DEFAULT (0),

[IsTechnical] bit NOT NULL CONSTRAINT [DF\_BSDEAL\_P\_IsTechnical] DEFAULT (0),

[TechnicalLinkedDealID] int NULL,

[SettleDT2] datetime NULL,

[TerminateFlag] nchar(1) NULL,

[DRDealID] int NULL,

CONSTRAINT [PK\_BSDEAL\_P] PRIMARY KEY ([ID])

) ON [PRIMARY]

END

GO

-------------------------------------------------

-- RAS.FIN\_SK\_P creating

-------------------------------------------------

IF NOT EXISTS (SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'FIN\_SK\_P')

BEGIN

PRINT 'Creating table RAS.FIN\_SK\_P'

CREATE TABLE [RAS].[FIN\_SK\_P]

(

[ID] int IDENTITY(1,1) NOT NULL,

[DealID] int NULL,

[CBR\_Portfolio] varchar(50) NOT NULL,

[PurchaseDT] datetime NOT NULL,

[RepoID] int NULL,

CONSTRAINT [PK\_FIN\_SK\_P] PRIMARY KEY ([ID])

) ON [PRIMARY]

END

GO

-------------------------------------------------

-- RAS.FIN\_SH\_P creating

-------------------------------------------------

IF NOT EXISTS (SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'FIN\_SH\_P')

BEGIN

PRINT 'Creating table RAS.FIN\_SH\_P'

CREATE TABLE [RAS].[FIN\_SH\_P]

(

[FinSkID] int NOT NULL,

[DT] datetime NOT NULL,

[Quantity] int NOT NULL,

[RepoQuantity] int NULL CONSTRAINT [DF\_FIN\_SH\_RepoQuantity] DEFAULT (0),

[SHSQuantity] int NOT NULL CONSTRAINT [DF\_FIN\_SH\_SHSQuantity] DEFAULT (0),

CONSTRAINT [PK\_FIN\_SH\_P] PRIMARY KEY ([DT], [FinSkID])

) ON [PRIMARY]

END

GO

-----------------------------------------------------------------------------

-- RAS.FIN\_SETTLEMENT\_P creating

-----------------------------------------------------------------------------

IF NOT EXISTS (SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'FIN\_SETTLEMENT\_P')

BEGIN

PRINT 'Creating Table RAS.FIN\_SETTLEMENT\_P'

CREATE TABLE [RAS].[FIN\_SETTLEMENT\_P] (

[FinSkID] int NOT NULL,

[DealID] int NOT NULL,

[Quantity] int NOT NULL,

[EntryDT] datetime NOT NULL,

[Type] char(3) NULL,

CONSTRAINT [PK\_FIN\_SETTLEMENT\_P] PRIMARY KEY CLUSTERED ([FinSkID], [DealID])

) ON [PRIMARY]

END

GO

-------------------------------------------------

-- GAAP.IBBS\_DISP creating

-------------------------------------------------

IF NOT EXISTS (SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'IBBS\_DISP')

BEGIN

PRINT 'Creating table GAAP.IBBS\_DISP'

CREATE TABLE [GAAP].[IBBS\_DISP]

(

[ID] [int] IDENTITY(2000,1) NOT NULL,

[ValueDT] [datetime] NOT NULL,

[DT] [datetime] NOT NULL,

[UserName] [dbo].[t\_uname] NOT NULL

) ON [PRIMARY]

END

GO

-------------------------------------------------

-- GAAP.IBBS\_HOFF creating

-------------------------------------------------

IF NOT EXISTS (SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'IBBS\_HOFF')

BEGIN

PRINT 'Creating table GAAP.IBBS\_HOFF'

CREATE TABLE [GAAP].[IBBS\_HOFF]

(

[DispID] [int] NOT NULL,

[BatchNo] [int] NOT NULL CONSTRAINT [DF\_IBBS\_HOFF\_BatchNo] DEFAULT (0),

[Acct] [decimal](10, 0) NULL,

[AcctDescr] [varchar](30) NOT NULL CONSTRAINT [DF\_IBBS\_HOFF\_AcctDescr] DEFAULT (' '),

[Amount] [decimal](17, 4) NULL,

[Ccy] [char](3) NOT NULL,

[DR\_CR] [tinyint] NOT NULL,

[Detail] [varchar](50) NOT NULL CONSTRAINT [DF\_IBBS\_HOFF\_Detail] DEFAULT (' '),

[PortfolioID] [int] NULL CONSTRAINT [DF\_IBBS\_HOFF\_PortfolioID] DEFAULT (0),

[SplitDealID] int NULL,

[ExchangeID] nvarchar(20),

[RawDealID] int NULL

) ON [PRIMARY]

END

GO

-------------------------------------------------

-- RAS.CITI9\_DISP creating

-------------------------------------------------

IF NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'CITI9\_DISP')

BEGIN

PRINT 'Creating table CITI9\_DISP'

CREATE TABLE [RAS].[CITI9\_DISP](

[ID] [int] IDENTITY(1,1) NOT NULL,

[ValueDate] [datetime] NOT NULL,

[DT] [datetime] NOT NULL,

[UserName] [varchar](30) NOT NULL

) ON [PRIMARY]

END

GO

-------------------------------------------------

-- RAS.CITI9\_DISP\_BSAMRT creating

-------------------------------------------------

IF NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'CITI9\_DISP\_BSAMRT')

BEGIN

PRINT 'Creating table RAS.CITI9\_DISP\_BSAMRT'

CREATE TABLE [RAS].[CITI9\_DISP\_BSAMRT](

[DispID] [int] NOT NULL,

[ISIN] [varchar](20) NOT NULL,

[ProductType] [varchar](50) NOT NULL,

[TBOOK] [varchar](50) NOT NULL,

[AMORT\_NO] [int] NULL,

[VALUEDATE] [datetime] NULL,

[AMOUNT] [decimal](28, 8) NULL,

[AMOUNT\_PER\_PIECE] [decimal](20, 8) NULL,

[SETTLEMENT\_ACCOUNT] [decimal](10, 0) NULL

) ON [PRIMARY]

END

GO

-------------------------------------------------

-- RAS.CITI9\_DISP\_BSINT creating

-------------------------------------------------

IF NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'CITI9\_DISP\_BSINT')

BEGIN

PRINT 'Creating table RAS.CITI9\_DISP\_BSINT'

CREATE TABLE [RAS].[CITI9\_DISP\_BSINT](

[DispID] [int] NOT NULL,

[ISIN] [varchar](20) NOT NULL,

[ProductType] [varchar](50) NOT NULL,

[TBOOK] [varchar](50) NOT NULL,

[VALUEDATE] [datetime] NOT NULL,

[AI] [decimal](28, 7) NOT NULL,

[AI\_PER\_PIECE] [decimal](33, 12) NOT NULL,

[TRADE\_NO] [varchar](20) NOT NULL

) ON [PRIMARY]

END

GO

-------------------------------------------------

-- RAS.CITI9\_DISP\_BSDEAL creating

-------------------------------------------------

IF NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'CITI9\_DISP\_BSDEAL')

BEGIN

PRINT 'Creating table RAS.CITI9\_DISP\_BSDEAL'

CREATE TABLE [RAS].[CITI9\_DISP\_BSDEAL](

[DispID] [int] NOT NULL,

[DispatchDate] [datetime] NOT NULL,

[DealNo] [varchar](20) NOT NULL,

[Operation] [char](2) NOT NULL,

[SymbolsPortCat] [char](2) NOT NULL,

[BookDT] [datetime] NOT NULL,

[SettleDT] [datetime] NOT NULL,

[InOut] [char](1) NOT NULL,

[BaseNo] [decimal](7, 0) NULL,

[ProductType] [varchar](3) NOT NULL,

[ISIN] [varchar](20) NULL,

[PricePcent] [float] NOT NULL,

[FaceValue] [decimal](15, 2) NOT NULL,

[PrincipalValue] [decimal](15, 2) NOT NULL,

[CouponAmount] [decimal](12, 2) NOT NULL,

[MiscCost] [decimal](12, 2) NOT NULL,

[CashAcctNo] [decimal](10, 0) NULL,

[Depo] [varchar](10) NOT NULL,

[RepoID] [int] NOT NULL,

[LinkedDealNo] [varchar](20) NOT NULL,

[Quantity] [int] NOT NULL,

[ScriptID] [int] NOT NULL,

[PortfolioCat] [varchar](20) NOT NULL,

[MiscCost5] [decimal](12, 2) NULL,

[IsTechnical] [char](1) NULL,

[TechnicalLinkedDealNo] [varchar](20) NULL,

[ForwardDealNo] [varchar](20) NULL,

[SettleDT2] [datetime] NULL,

[TerminateFlag] [nchar](1) NULL,

[DRDealNo] [nvarchar](20) NULL

) ON [PRIMARY]

END

GO

-----------------------------------------------------------------------------

-- REG\_DEAL\_P\_RAW constraints

-----------------------------------------------------------------------------

-- ScriptID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND COLUMN\_NAME = 'ScriptID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_RAW\_ScriptID')

BEGIN

ALTER TABLE dbo.[REG\_DEAL\_P\_RAW] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_DEAL\_P\_RAW\_ScriptID]

FOREIGN KEY ([ScriptID]) REFERENCES dbo.[SCRIPTS] ([ID])

END

GO

-- TradingCenterID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND COLUMN\_NAME = 'TradingCenterID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_RAW\_TradingCenterID')

BEGIN

ALTER TABLE dbo.[REG\_DEAL\_P\_RAW] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_DEAL\_P\_RAW\_TradingCenterID]

FOREIGN KEY ([TradingCenterID]) REFERENCES dbo.[DIC\_TRADING\_CENTER] ([ID])

END

GO

-- DealerID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND COLUMN\_NAME = 'DealerID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_RAW\_DealerID')

BEGIN

ALTER TABLE dbo.[REG\_DEAL\_P\_RAW] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_DEAL\_P\_RAW\_DealerID]

FOREIGN KEY ([DealerID]) REFERENCES dbo.[DIC\_DEALER] ([ID])

END

GO

-- CpID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND COLUMN\_NAME = 'CpID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_RAW\_CpID')

BEGIN

ALTER TABLE dbo.[REG\_DEAL\_P\_RAW] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_DEAL\_P\_RAW\_CpID]

FOREIGN KEY ([CpID]) REFERENCES dbo.[CP] ([ID])

END

GO

-- RepoID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND COLUMN\_NAME = 'RepoID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_RAW\_RepoID')

BEGIN

ALTER TABLE dbo.[REG\_DEAL\_P\_RAW] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_DEAL\_P\_RAW\_RepoID]

FOREIGN KEY ([RepoID]) REFERENCES dbo.[REPO\_P] ([ID])

END

GO

-- BatchID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND COLUMN\_NAME = 'BatchID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_RAW\_BatchID')

BEGIN

ALTER TABLE dbo.[REG\_DEAL\_P\_RAW] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_DEAL\_P\_RAW\_BatchID]

FOREIGN KEY ([BatchID]) REFERENCES dbo.[BATCHES] ([ID])

END

GO

IF EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND COLUMN\_NAME = 'DepoLocationID')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_RAW\_DepoLocationID')

BEGIN

ALTER TABLE dbo.REG\_DEAL\_P\_RAW WITH NOCHECK

ADD CONSTRAINT FK\_REG\_DEAL\_P\_RAW\_DepoLocationID

FOREIGN KEY(DepoLocationID) REFERENCES dbo.DIC\_DEPO\_LOCATION (ID)

END

GO

IF EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND COLUMN\_NAME = 'LocationID')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_RAW\_LocationID')

BEGIN

ALTER TABLE dbo.REG\_DEAL\_P\_RAW WITH NOCHECK

ADD CONSTRAINT FK\_REG\_DEAL\_P\_RAW\_LocationID

FOREIGN KEY(LocationID) REFERENCES dbo.DIC\_SECURITY\_ACCOUNT (ID)

END

GO

IF EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND COLUMN\_NAME = 'NostroAccountID')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_RAW\_NostroAccountID')

BEGIN

ALTER TABLE dbo.REG\_DEAL\_P\_RAW WITH NOCHECK

ADD CONSTRAINT FK\_REG\_DEAL\_P\_RAW\_NostroAccountID

FOREIGN KEY(NostroAccountID) REFERENCES dbo.DIC\_CASH\_ACCOUNT (ID)

END

GO

IF EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND COLUMN\_NAME = 'CommissionAccountID')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_RAW\_CommissionAccountID')

BEGIN

ALTER TABLE dbo.REG\_DEAL\_P\_RAW WITH NOCHECK

ADD CONSTRAINT FK\_REG\_DEAL\_P\_RAW\_CommissionAccountID

FOREIGN KEY(CommissionAccountID) REFERENCES dbo.DIC\_CASH\_ACCOUNT (ID)

END

GO

IF EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND COLUMN\_NAME = 'AirCashAccountID')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_RAW\_AirCashAccountID')

BEGIN

ALTER TABLE dbo.REG\_DEAL\_P\_RAW WITH NOCHECK

ADD CONSTRAINT FK\_REG\_DEAL\_P\_RAW\_AirCashAccountID

FOREIGN KEY(AirCashAccountID) REFERENCES dbo.DIC\_CASH\_ACCOUNT (ID)

END

GO

IF EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND COLUMN\_NAME = 'AirSecuritiesAccountID')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_RAW\_AirSecuritiesAccountID')

BEGIN

ALTER TABLE dbo.REG\_DEAL\_P\_RAW WITH NOCHECK

ADD CONSTRAINT FK\_REG\_DEAL\_P\_RAW\_AirSecuritiesAccountID

FOREIGN KEY(AirSecuritiesAccountID) REFERENCES dbo.DIC\_SECURITY\_ACCOUNT (ID)

END

GO

IF EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND COLUMN\_NAME = 'SettleModeID')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_RAW' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_RAW\_SettleModeID')

BEGIN

ALTER TABLE dbo.REG\_DEAL\_P\_RAW WITH NOCHECK

ADD CONSTRAINT FK\_REG\_DEAL\_P\_RAW\_SettleModeID

FOREIGN KEY(SettleModeID) REFERENCES dbo.DIC\_SETTLE\_MODE (ID)

END

GO

-----------------------------------------------------------------------------

-- REG\_DEAL\_P\_SPLIT constraints

-----------------------------------------------------------------------------

-- RawID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND COLUMN\_NAME = 'RawID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_SPLIT\_RawID')

BEGIN

ALTER TABLE dbo.[REG\_DEAL\_P\_SPLIT] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_DEAL\_P\_SPLIT\_RawID]

FOREIGN KEY ([RawID]) REFERENCES dbo.[REG\_DEAL\_P\_RAW] ([ID]) ON DELETE CASCADE

END

GO

-- ScriptID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND COLUMN\_NAME = 'ScriptID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_SPLIT\_ScriptID')

BEGIN

ALTER TABLE dbo.[REG\_DEAL\_P\_SPLIT] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_DEAL\_P\_SPLIT\_ScriptID]

FOREIGN KEY ([ScriptID]) REFERENCES dbo.[SCRIPTS] ([ID])

END

GO

-- TradingCenterID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND COLUMN\_NAME = 'TradingCenterID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_SPLIT\_TradingCenterID')

BEGIN

ALTER TABLE dbo.[REG\_DEAL\_P\_SPLIT] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_DEAL\_P\_SPLIT\_TradingCenterID]

FOREIGN KEY ([TradingCenterID]) REFERENCES dbo.[DIC\_TRADING\_CENTER] ([ID])

END

GO

-- DealerID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND COLUMN\_NAME = 'DealerID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_SPLIT\_DealerID')

BEGIN

ALTER TABLE dbo.[REG\_DEAL\_P\_SPLIT] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_DEAL\_P\_SPLIT\_DealerID]

FOREIGN KEY ([DealerID]) REFERENCES dbo.[DIC\_DEALER] ([ID])

END

GO

-- CpID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND COLUMN\_NAME = 'CpID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_SPLIT\_CpID')

BEGIN

ALTER TABLE dbo.[REG\_DEAL\_P\_SPLIT] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_DEAL\_P\_SPLIT\_CpID]

FOREIGN KEY ([CpID]) REFERENCES dbo.[CP] ([ID])

END

GO

-- RepoID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND COLUMN\_NAME = 'RepoID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_SPLIT\_RepoID')

BEGIN

ALTER TABLE dbo.[REG\_DEAL\_P\_SPLIT] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_DEAL\_P\_SPLIT\_RepoID]

FOREIGN KEY ([RepoID]) REFERENCES dbo.[REPO\_P] ([ID])

END

GO

-- PortfolioID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND COLUMN\_NAME = 'PortfolioID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_SPLIT\_PortfolioID')

BEGIN

ALTER TABLE dbo.[REG\_DEAL\_P\_SPLIT] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_DEAL\_P\_SPLIT\_PortfolioID]

FOREIGN KEY ([PortfolioID]) REFERENCES dbo.[PORTFOLIOS] ([ID])

END

GO

-- BatchID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND COLUMN\_NAME = 'BatchID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_SPLIT\_BatchID')

BEGIN

ALTER TABLE dbo.[REG\_DEAL\_P\_SPLIT] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_DEAL\_P\_SPLIT\_BatchID]

FOREIGN KEY ([BatchID]) REFERENCES dbo.[BATCHES] ([ID])

END

GO

-- ParentDealID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND COLUMN\_NAME = 'ParentDealID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_DEAL\_P\_SPLIT' AND CONSTRAINT\_NAME = 'FK\_REG\_DEAL\_P\_SPLIT\_ParentDealID')

BEGIN

ALTER TABLE dbo.[REG\_DEAL\_P\_SPLIT] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_DEAL\_P\_SPLIT\_ParentDealID]

FOREIGN KEY ([ParentDealID]) REFERENCES dbo.[REG\_DEAL\_P\_SPLIT] ([ID])

END

GO

-----------------------------------------------------------------------------

-- REG\_EVENT\_P constraints

-----------------------------------------------------------------------------

-- ScriptID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_EVENT\_P' AND COLUMN\_NAME = 'ScriptID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_EVENT\_P' AND CONSTRAINT\_NAME = 'FK\_REG\_EVENT\_P\_ScriptID')

BEGIN

ALTER TABLE dbo.[REG\_EVENT\_P] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_EVENT\_P\_ScriptID]

FOREIGN KEY ([ScriptID]) REFERENCES dbo.[SCRIPTS] ([ID])

END

GO

-- DepoLocationID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_EVENT\_P' AND COLUMN\_NAME = 'DepoLocationID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_EVENT\_P' AND CONSTRAINT\_NAME = 'FK\_REG\_EVENT\_P\_DepoLocationID')

BEGIN

ALTER TABLE dbo.[REG\_EVENT\_P] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_EVENT\_P\_DepoLocationID]

FOREIGN KEY ([DepoLocationID]) REFERENCES dbo.[DIC\_DEPO\_LOCATION] ([ID])

END

GO

-- NostroAccountID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_EVENT\_P' AND COLUMN\_NAME = 'NostroAccountID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_EVENT\_P' AND CONSTRAINT\_NAME = 'FK\_REG\_EVENT\_P\_NostroAccountID')

BEGIN

ALTER TABLE dbo.[REG\_EVENT\_P] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_EVENT\_P\_NostroAccountID]

FOREIGN KEY ([NostroAccountID]) REFERENCES dbo.[DIC\_CASH\_ACCOUNT] ([ID])

END

GO

-----------------------------------------------------------------------------

-- REG\_EVENT\_P\_PEND constraints

-----------------------------------------------------------------------------

-- NostroAccountID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_EVENT\_P\_PEND' AND COLUMN\_NAME = 'NostroAccountID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_EVENT\_P\_PEND' AND CONSTRAINT\_NAME = 'FK\_REG\_EVENT\_P\_PEND\_NostroAccountID')

BEGIN

ALTER TABLE dbo.[REG\_EVENT\_P\_PEND] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_EVENT\_P\_PEND\_NostroAccountID]

FOREIGN KEY ([NostroAccountID]) REFERENCES dbo.[DIC\_CASH\_ACCOUNT] ([ID])

END

GO

---------------------------------------------------------------------------------------------------------------------------------

-- REG\_EVENT\_LOCATION\_P constraints

---------------------------------------------------------------------------------------------------------------------------------

-- EventID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_EVENT\_LOCATION\_P' AND COLUMN\_NAME = 'EventID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_EVENT\_LOCATION\_P' AND CONSTRAINT\_NAME = 'FK\_REG\_EVENT\_LOCATION\_P\_EventID')

BEGIN

ALTER TABLE dbo.[REG\_EVENT\_LOCATION\_P] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_EVENT\_LOCATION\_P\_EventID]

FOREIGN KEY ([EventID]) REFERENCES dbo.[REG\_EVENT\_P] ([ID]) ON DELETE CASCADE

END

GO

-- LocationID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_EVENT\_LOCATION\_P' AND COLUMN\_NAME = 'LocationID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_EVENT\_LOCATION\_P' AND CONSTRAINT\_NAME = 'FK\_REG\_EVENT\_LOCATION\_P\_LocationID')

BEGIN

ALTER TABLE dbo.[REG\_EVENT\_LOCATION\_P] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_EVENT\_LOCATION\_P\_LocationID]

FOREIGN KEY ([LocationID]) REFERENCES dbo.[DIC\_SECURITY\_ACCOUNT] ([ID])

END

GO

-- AirCashAccountID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_EVENT\_LOCATION\_P' AND COLUMN\_NAME = 'AirCashAccountID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_EVENT\_LOCATION\_P' AND CONSTRAINT\_NAME = 'FK\_REG\_EVENT\_LOCATION\_P\_AirCashAccountID')

BEGIN

ALTER TABLE dbo.[REG\_EVENT\_LOCATION\_P] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_EVENT\_LOCATION\_P\_AirCashAccountID]

FOREIGN KEY ([AirCashAccountID]) REFERENCES dbo.[DIC\_CASH\_ACCOUNT] ([ID])

END

GO

-- AirSecuritiesAccountID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_EVENT\_LOCATION\_P' AND COLUMN\_NAME = 'AirSecuritiesAccountID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_EVENT\_LOCATION\_P' AND CONSTRAINT\_NAME = 'FK\_REG\_EVENT\_LOCATION\_P\_AirSecuritiesAccountID')

BEGIN

ALTER TABLE dbo.[REG\_EVENT\_LOCATION\_P] WITH NOCHECK

ADD CONSTRAINT [FK\_REG\_EVENT\_LOCATION\_P\_AirSecuritiesAccountID]

FOREIGN KEY ([AirSecuritiesAccountID]) REFERENCES dbo.[DIC\_SECURITY\_ACCOUNT] ([ID])

END

GO

-------------------------------------------------

-- REG\_EVENT\_PARTIAL\_PAYMENT\_P constraints

-------------------------------------------------

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_EVENT\_PARTIAL\_PAYMENT\_P'

AND COLUMN\_NAME = 'EventID')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_EVENT\_PARTIAL\_PAYMENT\_P'

AND CONSTRAINT\_NAME = 'FK\_REG\_EVENT\_PARTIAL\_PAYMENT\_P\_EventID')

BEGIN

ALTER TABLE dbo.REG\_EVENT\_PARTIAL\_PAYMENT\_P WITH CHECK

ADD CONSTRAINT FK\_REG\_EVENT\_PARTIAL\_PAYMENT\_P\_EventID

FOREIGN KEY (EventID) REFERENCES dbo.REG\_EVENT\_P (ID) ON DELETE CASCADE

END

GO

-------------------------------------------------

-- REG\_EVENT\_PARTIAL\_PAYMENT\_P\_PEND constraints

-------------------------------------------------

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'REG\_EVENT\_PARTIAL\_PAYMENT\_P\_PEND'

AND COLUMN\_NAME = 'EventID')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'REG\_EVENT\_PARTIAL\_PAYMENT\_P\_PEND'

AND CONSTRAINT\_NAME = 'FK\_REG\_EVENT\_PARTIAL\_PAYMENT\_P\_PEND\_EventID')

BEGIN

ALTER TABLE dbo.REG\_EVENT\_PARTIAL\_PAYMENT\_P\_PEND WITH CHECK

ADD CONSTRAINT FK\_REG\_EVENT\_PARTIAL\_PAYMENT\_P\_PEND\_EventID

FOREIGN KEY (EventID) REFERENCES dbo.REG\_EVENT\_P (ID) ON DELETE CASCADE

END

GO

-----------------------------------------------------------------------------

-- GAAP.SCRIPTS\_KEPT\_P constraints

-----------------------------------------------------------------------------

-- DealID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'SCRIPTS\_KEPT\_P' AND TABLE\_SCHEMA = 'GAAP' AND COLUMN\_NAME = 'DealID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'SCRIPTS\_KEPT\_P' AND TABLE\_SCHEMA = 'GAAP'

AND CONSTRAINT\_NAME = 'FK\_SCRIPTS\_KEPT\_P\_DealID')

BEGIN

ALTER TABLE GAAP.[SCRIPTS\_KEPT\_P] WITH NOCHECK

ADD CONSTRAINT [FK\_SCRIPTS\_KEPT\_P\_DealID]

FOREIGN KEY ([DealID]) REFERENCES dbo.[REG\_DEAL\_P\_SPLIT] ([ID])

END

GO

-----------------------------------------------------------------------------

-- GAAP.SCRIPTS\_HIST\_P constraints

-----------------------------------------------------------------------------

-- SKept\_ID

IF EXISTS(SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'SCRIPTS\_HIST\_P' AND TABLE\_SCHEMA = 'GAAP' AND COLUMN\_NAME = 'SKept\_ID')

AND NOT EXISTS(SELECT \* FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'SCRIPTS\_HIST\_P' AND TABLE\_SCHEMA = 'GAAP' AND CONSTRAINT\_NAME = 'FK\_SCRIPTS\_HIST\_P\_SKept\_ID')

BEGIN

ALTER TABLE GAAP.[SCRIPTS\_HIST\_P] WITH NOCHECK

ADD CONSTRAINT [FK\_SCRIPTS\_HIST\_P\_SKept\_ID]

FOREIGN KEY ([SKept\_ID]) REFERENCES GAAP.[SCRIPTS\_KEPT\_P] ([ID]) ON DELETE CASCADE

END

GO

-----------------------------------------------------------------------------

-- GAAP.SETTLEMENTS\_P constraints

-----------------------------------------------------------------------------

-- SKept\_ID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'SETTLEMENTS\_P' AND TABLE\_SCHEMA = 'GAAP' AND COLUMN\_NAME = 'SKept\_ID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'SETTLEMENTS\_P' AND TABLE\_SCHEMA = 'GAAP' AND CONSTRAINT\_NAME = 'FK\_SETTLEMENTS\_P\_SKept\_ID')

BEGIN

ALTER TABLE GAAP.[SETTLEMENTS\_P] WITH NOCHECK

ADD CONSTRAINT [FK\_SETTLEMENTS\_P\_SKept\_ID]

FOREIGN KEY ([SKept\_ID]) REFERENCES GAAP.[SCRIPTS\_KEPT\_P] ([ID]) ON DELETE CASCADE

END

GO

-- DealID

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'SETTLEMENTS\_P' AND TABLE\_SCHEMA = 'GAAP' AND COLUMN\_NAME = 'DealID')

AND NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'SETTLEMENTS\_P' AND TABLE\_SCHEMA = 'GAAP'

AND CONSTRAINT\_NAME = 'FK\_SETTLEMENTS\_P\_DealID')

BEGIN

ALTER TABLE GAAP.[SETTLEMENTS\_P] WITH NOCHECK

ADD CONSTRAINT [FK\_SETTLEMENTS\_P\_DealID]

FOREIGN KEY ([DealID]) REFERENCES dbo.[REG\_DEAL\_P\_SPLIT] ([ID])

END

GO

--------------------------------------------------------------------

-- BSDEAL\_P FOREIGN KEYS

--------------------------------------------------------------------

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'BSDEAL\_P' AND TABLE\_SCHEMA = 'RAS')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'BSDEAL\_P' AND CONSTRAINT\_NAME = 'FK\_BSDEAL\_P\_SplitDealID')

BEGIN

ALTER TABLE [RAS].[BSDEAL\_P] WITH NOCHECK ADD CONSTRAINT [FK\_BSDEAL\_P\_SplitDealID]

FOREIGN KEY ([SplitDealID]) REFERENCES [dbo].[REG\_DEAL\_P\_SPLIT] ([ID]) ON DELETE NO ACTION

END

GO

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'BSDEAL\_P' AND TABLE\_SCHEMA = 'RAS')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'BSDEAL\_P' AND CONSTRAINT\_NAME = 'FK\_BSDEAL\_P\_LinkedDealID')

BEGIN

ALTER TABLE [RAS].[BSDEAL\_P] WITH NOCHECK ADD CONSTRAINT [FK\_BSDEAL\_P\_LinkedDealID]

FOREIGN KEY ([LinkedDealID]) REFERENCES [RAS].[BSDEAL\_P] ([ID]) ON DELETE NO ACTION

END

GO

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'BSDEAL\_P' AND TABLE\_SCHEMA = 'RAS')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'BSDEAL\_P' AND CONSTRAINT\_NAME = 'FK\_BSDEAL\_P\_PortfolioID')

BEGIN

ALTER TABLE [RAS].[BSDEAL\_P] WITH NOCHECK ADD CONSTRAINT [FK\_BSDEAL\_P\_PortfolioID]

FOREIGN KEY ([PortfolioID]) REFERENCES [dbo].[PORTFOLIOS] ([ID]) ON DELETE NO ACTION

END

GO

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'BSDEAL\_P' AND TABLE\_SCHEMA = 'RAS')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'BSDEAL\_P' AND CONSTRAINT\_NAME = 'FK\_BSDEAL\_P\_CpID')

BEGIN

ALTER TABLE [RAS].[BSDEAL\_P] WITH NOCHECK ADD CONSTRAINT [FK\_BSDEAL\_P\_CpID]

FOREIGN KEY ([CpID]) REFERENCES [dbo].[CP] ([ID]) ON DELETE NO ACTION

END

GO

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'BSDEAL\_P' AND TABLE\_SCHEMA = 'RAS')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'BSDEAL\_P' AND CONSTRAINT\_NAME = 'FK\_BSDEAL\_P\_ScriptID')

BEGIN

ALTER TABLE [RAS].[BSDEAL\_P] WITH NOCHECK ADD CONSTRAINT [FK\_BSDEAL\_P\_ScriptID]

FOREIGN KEY ([ScriptID]) REFERENCES [dbo].[SCRIPTS] ([ID]) ON DELETE NO ACTION

END

GO

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'BSDEAL\_P' AND TABLE\_SCHEMA = 'RAS')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'BSDEAL\_P' AND CONSTRAINT\_NAME = 'FK\_BSDEAL\_P\_TradingCenterID')

BEGIN

ALTER TABLE [RAS].[BSDEAL\_P] WITH NOCHECK ADD CONSTRAINT [FK\_BSDEAL\_P\_TradingCenterID]

FOREIGN KEY ([TradingCenterID]) REFERENCES [dbo].[DIC\_TRADING\_CENTER] ([ID]) ON DELETE NO ACTION

END

GO

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'BSDEAL\_P' AND TABLE\_SCHEMA = 'RAS')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'BSDEAL\_P' AND CONSTRAINT\_NAME = 'FK\_BSDEAL\_P\_ParentID')

BEGIN

ALTER TABLE [RAS].[BSDEAL\_P] WITH NOCHECK ADD CONSTRAINT [FK\_BSDEAL\_P\_ParentID]

FOREIGN KEY ([ParentID]) REFERENCES [RAS].[BSDEAL\_P] ([ID]) ON DELETE NO ACTION

END

GO

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'BSDEAL\_P' AND TABLE\_SCHEMA = 'RAS')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'BSDEAL\_P' AND CONSTRAINT\_NAME = 'FK\_BSDEAL\_P\_RepoID')

BEGIN

ALTER TABLE [RAS].[BSDEAL\_P] WITH NOCHECK ADD CONSTRAINT [FK\_BSDEAL\_P\_RepoID]

FOREIGN KEY ([RepoID]) REFERENCES [dbo].[REPO\_P] ([ID]) ON DELETE NO ACTION

END

GO

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'BSDEAL\_P' AND TABLE\_SCHEMA = 'RAS')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'BSDEAL\_P' AND CONSTRAINT\_NAME = 'FK\_BSDEAL\_P\_TechnicalLinkedDealID')

BEGIN

ALTER TABLE [RAS].[BSDEAL\_P] WITH NOCHECK ADD CONSTRAINT [FK\_BSDEAL\_P\_TechnicalLinkedDealID]

FOREIGN KEY ([TechnicalLinkedDealID]) REFERENCES [RAS].[BSDEAL\_P] ([ID]) ON DELETE NO ACTION

END

GO

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'BSDEAL\_P' AND TABLE\_SCHEMA = 'RAS')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'BSDEAL\_P' AND CONSTRAINT\_NAME = 'FK\_BSDEAL\_P\_DRDealID')

BEGIN

ALTER TABLE [RAS].[BSDEAL\_P] WITH NOCHECK ADD CONSTRAINT [FK\_BSDEAL\_P\_DRDealID]

FOREIGN KEY ([DRDealID]) REFERENCES [RAS].[BSDEAL\_P] ([ID]) ON DELETE NO ACTION

END

GO

--------------------------------------------------------------------

-- FIN\_SK\_P FOREIGN KEYS

--------------------------------------------------------------------

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'FIN\_SK\_P' AND TABLE\_SCHEMA = 'RAS')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'FIN\_SK\_P' AND CONSTRAINT\_NAME = 'FK\_FIN\_SK\_P\_DealID')

BEGIN

ALTER TABLE [RAS].[FIN\_SK\_P] WITH NOCHECK ADD CONSTRAINT [FK\_FIN\_SK\_P\_DealID]

FOREIGN KEY ([DealID]) REFERENCES [RAS].[BSDEAL\_P] ([ID]) ON DELETE CASCADE

END

GO

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'FIN\_SK\_P' AND TABLE\_SCHEMA = 'RAS')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'FIN\_SK\_P' AND CONSTRAINT\_NAME = 'FK\_FIN\_SK\_P\_RepoID')

BEGIN

ALTER TABLE [RAS].[FIN\_SK\_P] WITH NOCHECK ADD CONSTRAINT [FK\_FIN\_SK\_P\_RepoID]

FOREIGN KEY ([RepoID]) REFERENCES [dbo].[REPO\_P] ([ID]) ON DELETE NO ACTION

END

GO

--------------------------------------------------------------------

-- FIN\_SH\_P FOREIGN KEYS

--------------------------------------------------------------------

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'FIN\_SH\_P' AND TABLE\_SCHEMA = 'RAS')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'FIN\_SH\_P' AND CONSTRAINT\_NAME = 'FK\_FIN\_SH\_P\_FinSkID')

BEGIN

ALTER TABLE [RAS].[FIN\_SH\_P] WITH NOCHECK ADD CONSTRAINT [FK\_FIN\_SH\_P\_FinSkID]

FOREIGN KEY ([FinSkID]) REFERENCES [RAS].[FIN\_SK\_P] ([ID]) ON DELETE CASCADE

END

GO

--------------------------------------------------------------------

-- FIN\_SETTLEMENT\_P FOREIGN KEYS

--------------------------------------------------------------------

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'FIN\_SETTLEMENT\_P' AND TABLE\_SCHEMA = 'RAS')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'FIN\_SETTLEMENT\_P' AND CONSTRAINT\_NAME = 'FK\_FIN\_SETTLEMENT\_P\_FinSkID')

BEGIN

ALTER TABLE [RAS].[FIN\_SETTLEMENT\_P] WITH NOCHECK ADD CONSTRAINT [FK\_FIN\_SETTLEMENT\_P\_FinSkID]

FOREIGN KEY ([FinSkID]) REFERENCES [RAS].[FIN\_SK\_P] ([ID]) ON DELETE CASCADE

END

GO

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'FIN\_SETTLEMENT\_P' AND TABLE\_SCHEMA = 'RAS')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'FIN\_SETTLEMENT\_P' AND CONSTRAINT\_NAME = 'FK\_FIN\_SETTLEMENT\_P\_DealID')

BEGIN

ALTER TABLE [RAS].[FIN\_SETTLEMENT\_P] WITH NOCHECK ADD CONSTRAINT [FK\_FIN\_SETTLEMENT\_P\_DealID]

FOREIGN KEY ([DealID]) REFERENCES [RAS].[BSDEAL\_P] ([ID]) ON DELETE NO ACTION

END

GO

-------------------------------------------------

-- GAAP.IBBS\_DISP constraints

-------------------------------------------------

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'IBBS\_DISP' AND COLUMN\_NAME = 'ValueDT')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'IBBS\_DISP'

AND CONSTRAINT\_NAME = 'FK\_IBBS\_DISP\_ValueDT')

BEGIN

PRINT 'Create FK constraint [IBBS\_DISP].[FK\_IBBS\_DISP\_ValueDT]'

ALTER TABLE [GAAP].[IBBS\_DISP] WITH NOCHECK ADD CONSTRAINT [FK\_IBBS\_DISP\_ValueDT]

FOREIGN KEY ([ValueDT]) REFERENCES [dbo].[DAILY] ([DT]) ON DELETE NO ACTION

END

GO

-------------------------------------------------

-- GAAP.IBBS\_HOFF constraints

-------------------------------------------------

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'IBBS\_HOFF' AND COLUMN\_NAME = 'DispID')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'IBBS\_HOFF'

AND CONSTRAINT\_NAME = 'FK\_IBBS\_HOFF\_DispID')

BEGIN

PRINT 'Create FK constraint [IBBS\_HOFF].[FK\_IBBS\_HOFF\_DispID]'

ALTER TABLE [GAAP].[IBBS\_HOFF] WITH NOCHECK ADD CONSTRAINT [FK\_IBBS\_HOFF\_DispID]

FOREIGN KEY ([DispID]) REFERENCES [GAAP].[IBBS\_DISP] ([ID]) ON DELETE NO ACTION

END

GO

-------------------------------------------------

-- RAS.CITI9\_DISP constraints

-------------------------------------------------

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'CITI9\_DISP' AND COLUMN\_NAME = 'ID')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'CITI9\_DISP' AND CONSTRAINT\_NAME = 'PK\_CITI9\_DISP')

BEGIN

ALTER TABLE [RAS].[CITI9\_DISP]

ADD CONSTRAINT [PK\_CITI9\_DISP]

PRIMARY KEY CLUSTERED ([ID])

END

GO

-------------------------------------------------

-- RAS.CITI9\_DISP\_BSAMRT constraints

-------------------------------------------------

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'CITI9\_DISP\_BSAMRT' AND COLUMN\_NAME = 'DispID')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'CITI9\_DISP\_BSAMRT' AND CONSTRAINT\_NAME = 'FK\_CITI9\_DISP\_BSAMRT\_DispID')

BEGIN

ALTER TABLE [RAS].[CITI9\_DISP\_BSAMRT] WITH CHECK

ADD CONSTRAINT [FK\_CITI9\_DISP\_BSAMRT\_DispID]

FOREIGN KEY([DispID]) REFERENCES [RAS].[CITI9\_DISP] ([ID])

END

GO

-------------------------------------------------

-- RAS.CITI9\_DISP\_BSINT constraints

-------------------------------------------------

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'CITI9\_DISP\_BSINT' AND COLUMN\_NAME = 'DispID')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'CITI9\_DISP\_BSINT' AND CONSTRAINT\_NAME = 'FK\_CITI9\_DISP\_BSINT\_DispID')

BEGIN

ALTER TABLE [RAS].[CITI9\_DISP\_BSINT] WITH CHECK

ADD CONSTRAINT [FK\_CITI9\_DISP\_BSINT\_DispID]

FOREIGN KEY([DispID]) REFERENCES [RAS].[CITI9\_DISP] ([ID])

END

GO

-------------------------------------------------

-- RAS.CITI9\_DISP\_BSDEAL constraints

-------------------------------------------------

IF EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'CITI9\_DISP\_BSDEAL' AND COLUMN\_NAME = 'DispID')

AND NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS WHERE TABLE\_NAME = 'CITI9\_DISP\_BSDEAL' AND CONSTRAINT\_NAME = 'FK\_CITI9\_DISP\_BSDEAL\_DispID')

BEGIN

ALTER TABLE [RAS].[CITI9\_DISP\_BSDEAL] WITH CHECK

ADD CONSTRAINT [FK\_CITI9\_DISP\_BSDEAL\_DispID]

FOREIGN KEY([DispID]) REFERENCES [RAS].[CITI9\_DISP] ([ID])

END

GO

## Фрагменты исходных текстов модуля

### Листинг основных хранимых процедур

-------------------------------------------------

-- GAAP.ComputeFIFO

-------------------------------------------------

IF NOT EXISTS (SELECT 1 FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[GAAP].[ComputeFIFO]') AND type in (N'P', N'PC'))

BEGIN

PRINT 'Creating procedure GAAP.[ComputeFIFO]'

EXEC ('CREATE PROCEDURE [GAAP].[ComputeFIFO] AS RAISERROR(''Not implemented yet.'', 16, 3);')

END

GO

PRINT 'Altering procedure GAAP.[ComputeFIFO]'

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE GAAP.[ComputeFIFO]

@dt datetime,

@errorStr varchar(255) = null OUTPUT,

@hist xml = null OUTPUT

AS

BEGIN

SET NOCOUNT ON

SET ANSI\_WARNINGS ON

SET ARITHABORT ON

SET XACT\_ABORT ON

SET CURSOR\_CLOSE\_ON\_COMMIT ON

DECLARE @result int,

@lastbd datetime,

@identity int,

@\_ID int,

@CancelledStatus int = -1,

@DealIdentity int,

@OperationID int

-- clear existing FIFO

EXEC GAAP.ClearFIFO @dt

CREATE TABLE #skept

(

[SKept\_ID] int not null,

[ScriptID] int not null,

[PortfolioID] int null,

[PricePcent] decimal(15,8) not null,

[PurchaseDT] datetime not null,

[PurchaseTime] time not null,

[SettleDT] datetime null,

[BatchID] int null,

[OrderNo] int null,

[ClientID] int null,

[EntryDT] datetime null,

[DealID] int null,

[Quantity] int not null default(0),

[New] bit not null default(1), -- new records indication

PRIMARY KEY ([SKept\_ID])

)

CREATE TABLE #settlements

(

[skID] int not null,

[DealID] int not null,

[Quantity] int not null,

[BookDT] datetime not null,

[SettleDT] datetime not null,

[BatchNo] int null,

[OrderNo] int null,

[EntryDate] datetime null,

[Type] varchar(10) null,

PRIMARY KEY ([skID], [DealID])

)

CREATE TABLE #manual\_operations

(

[ID] int null, -- id of operation in REG\_DEAL\_P\_SPLIT, but it will be assigned only if @write = 1

[OperationID] int not null,

[ScriptID] int not null,

[Quantity] int not null,

[PortfolioID] int not null,

[Price] decimal(14,6) not null,

[Operation] nvarchar(3) collate database\_default not null,

[BookDT] datetime not null,

[BookTime] time not null,

[SettleDT] datetime not null,

[\_ID] int not null identity,

PRIMARY KEY ([\_ID])

)

SET @result = 0

SET @errorStr = ''

SELECT @lastbd = MAX([DT]) FROM dbo.[DAILY] WHERE [DT] < @dt

INSERT INTO #skept

( [SKept\_ID], [ScriptID], [PortfolioID], [PricePcent],

[PurchaseDT], [PurchaseTime], [SettleDT],

[BatchID], [OrderNo], [ClientID], [EntryDT], [DealID],

[Quantity], [New]

)

SELECT

SH.[SKept\_ID], SK.[ScriptID], SK.[PortfolioID], SK.[PricePcent],

SK.[PurchaseDT], CONVERT(time, L.TradeTime) AS [PurchaseTime], SK.[SettleDT],

SK.[BatchID], SK.[OrderNo], SK.[ClientID], SK.[EntryDT], SK.[DealID],

SH.[Quantity], 0

FROM GAAP.[SCRIPTS\_KEPT] SK

INNER JOIN dbo.[REG\_DEAL\_P\_SPLIT] L ON L.[ID] = SK.[DealID]

INNER JOIN GAAP.[SCRIPTS\_HIST] SH ON (SK.[ID] = SH.[SKept\_ID])

WHERE SH.[DT] = @lastbd

SET @identity = ISNULL((SELECT MAX([SKept\_ID]) FROM #skept), 0) -- all new records will have greater id value

SET @DealIdentity = ISNULL((SELECT MAX([ID]) FROM dbo.[AGG\_TRADING\_LOG]), 0) -- all new records will have greater id value

--------------------------------------------------------------------

-- insert manual security operations

--------------------------------------------------------------------

insert into #manual\_operations

(

[OperationID],

[ScriptID],

[Quantity],

[Price],

[PortfolioID],

[Operation],

[BookDT],

[BookTime],

[SettleDT]

)

SELECT

OP.[ID], OP.[ScriptID], OP.[Quantity], OP.[PricePcent], OP.[PortfolioID], DSO.[ShortName],

OP.[TradeDT], CONVERT(time, OP.[TradeTime]), OP.[RealSettlementDT]

FROM dbo.[REG\_SECURITY\_OPERATION\_P] OP

INNER JOIN dbo.[DIC\_SECURITY\_OPERATION] DSO ON OP.[OperationID] = DSO.[ID]

WHERE OP.[TradeDT] = @dt

AND OP.[Status] != @CancelledStatus

AND DSO.[ShortName] in ('SI', 'SW')

AND DSO.[Proprietary] = 1

AND NOT EXISTS(SELECT 1 FROM dbo.[REG\_SECURITY\_OPERATION\_P\_PEND] WHERE [ID] = OP.[ID])

IF @@ROWCOUNT > 0

BEGIN

-- insert operations into SPLIT log to account in FIFO

SET @\_ID = -1

WHILE 1=1

BEGIN

SELECT TOP 1 @\_ID = [\_ID], @OperationID = [OperationID]

FROM #manual\_operations

WHERE [\_ID] > @\_ID

ORDER BY [\_ID]

IF @@ROWCOUNT = 0 BREAK

INSERT INTO [dbo].[AGG\_TRADING\_LOG]

(

[PortfolioID], [ScriptID], [Quantity], [Price], [Operation],

[SymbolsIbbs], [Auction], [BookDT], [SettleDT], [EntryDate],

[RepoID], [CpID]

)

SELECT

[PortfolioID], [ScriptID], [Quantity], [Price], [Operation],

0, 'S', [BookDT], [SettleDT], @dt,

null, null

FROM #manual\_operations

WHERE [\_ID] = @\_ID

SET @DealIdentity = @@IDENTITY

UPDATE dbo.[REG\_SECURITY\_OPERATION\_P]

SET

[SplitID] = @DealIdentity

WHERE [ID] = @OperationID

UPDATE #manual\_operations

SET [ID] = @DealIdentity

WHERE [\_ID] = @\_ID

END

END

--------------------------------------------------------------------

-- Buys

--------------------------------------------------------------------

EXEC GAAP.[ComputeFIFO\_Buy] @dt = @dt, @Technical = 0, @identity = @identity OUTPUT

--------------------------------------------------------------------

-- Sales

--------------------------------------------------------------------

EXEC GAAP.[ComputeFIFO\_Sell] @dt = @dt, @Technical = 0, @Maturity = 0, @identity = @identity OUTPUT

--------------------------------------------------------------------

-- Technical buys

--------------------------------------------------------------------

EXEC GAAP.[ComputeFIFO\_Buy] @dt = @dt, @Technical = 1, @identity = @identity OUTPUT

--------------------------------------------------------------------

-- Technical Sales

--------------------------------------------------------------------

EXEC GAAP.[ComputeFIFO\_Sell] @dt = @dt, @Technical = 1, @Maturity = 0, @identity = @identity OUTPUT

--------------------------------------------------------------------

-- Maturities

--------------------------------------------------------------------

EXEC GAAP.[ComputeFIFO\_Sell] @dt = @dt, @Technical = 0, @Maturity = 1, @identity = @identity OUTPUT

--------------------------------------------------------------------

-- Get new state of things

--------------------------------------------------------------------

-- TODO: investigate and fix this code

INSERT INTO GAAP.[SCRIPTS\_HIST]

([SKept\_ID], [Quantity], [CV], [DT],

[CvCf], [DailyCoupon], [DailyDiscount]

)

SELECT SK.SKept\_ID, Sum(SK.Quantity), 0, @dt,

0, 0, 0

FROM #skept SK

INNER JOIN dbo.[SCRIPTS] S ON (SK.ScriptID = S.ID)

WHERE S.MaturityDT > @dt or S.MaturityDT is null

GROUP BY SK.SKept\_ID

-- delete totally sold entries that have no pending settlement

DELETE SH

FROM GAAP.SCRIPTS\_HIST\_P SH

JOIN GAAP.SCRIPTS\_KEPT\_P SK ON SH.SKept\_ID = SK.ID

LEFT JOIN GAAP.SETTLEMENTS S ON SH.SKept\_ID = S.skID AND S.BookDT > SH.DT

WHERE SH.Quantity = 0

AND S.skID is null

--------------------------------------------------------------------

-- return xml

--------------------------------------------------------------------

SET @hist = cast((

SELECT T.[HIST] AS [\*]

FROM

(

SELECT (

SELECT

[SKept\_ID] AS [@ID],

[New] AS [@IsNew],

[ScriptID],

[PortfolioID],

[PricePcent],

[PurchaseDT],

[SettleDT],

[BatchID],

[OrderNo],

[ClientID],

[EntryDT],

[DealID],

[Quantity] AS [SCRIPTS\_HIST/Quantity],

(

SELECT

[DealID],

[Quantity],

[BookDT],

[SettleDT],

[BatchNo],

[OrderNo],

[EntryDate],

rtrim([Type]) AS [Type]

FROM #settlements

WHERE [skID] = sk.[SKept\_ID]

FOR XML PATH('SETTLEMENTS'), TYPE

)

FROM #skept sk

FOR XML PATH('SCRIPTS\_KEPT'), TYPE

) AS [HIST]

) T

FOR XML PATH(''), ROOT ('root')

) AS xml)

end\_l:

RETURN @result

END

GO

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

-- RAS.GenerateBSDeal

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

IF NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.ROUTINES WHERE ROUTINE\_NAME = 'GenerateBSDeal' AND ROUTINE\_TYPE = 'PROCEDURE'

AND ROUTINE\_SCHEMA = 'RAS')

BEGIN

PRINT 'Creating procedure RAS.[GenerateBSDeal]'

EXEC ('CREATE PROCEDURE [RAS].[GenerateBSDeal] AS RAISERROR(''Not implemented yet.'', 16, 3);')

END

PRINT 'Altering procedure RAS.[GenerateBSDeal]'

GO

-----------------------------------------------------------------------------

-- RAS.GenerateBSDeal creating

-----------------------------------------------------------------------------

ALTER PROCEDURE RAS.[GenerateBSDeal]

@dt datetime

AS

BEGIN

IF NOT EXISTS (SELECT DT FROM dbo.DAILY WHERE DT = @dt AND DumpIbbsEod = 1)

GOTO END\_L

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

BEGIN TRAN

-- Delete old BSDEAL and FIN FIFO

EXEC RAS.ClearFifoAndBSDEAL @dt

-- buys & sells, Direct and Reverse REPO (both legs) & Defautls

EXEC RAS.GenerateBSDealRegular @dt

-- coupon maturity

EXEC RAS.GenerateBSDealCoupon @dt

-- amortization & full script maturity

EXEC RAS.GenerateBSDealMaturity @dt

COMMIT TRAN

END\_L:

RETURN 0

END

GO

-------------------------------------------------

-- RAS.GenerateCiti9Handoff

-------------------------------------------------

IF NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.ROUTINES

WHERE ROUTINE\_NAME = 'GenerateCiti9Handoff' AND ROUTINE\_TYPE = 'PROCEDURE' AND ROUTINE\_SCHEMA = 'RAS')

BEGIN

PRINT 'Creating procedure RAS.GenerateCiti9Handoff'

EXEC ('CREATE PROCEDURE [RAS].[GenerateCiti9Handoff] AS RAISERROR(''Not implemented yet.'', 16, 3);')

END

PRINT 'Altering of procedure RAS.GenerateCiti9Handoff'

GO

-------------------------------------------------

-- RAS.GenerateCiti9Handoff creating

-------------------------------------------------

ALTER PROCEDURE RAS.GenerateCiti9Handoff

@ValueDate [datetime],

@UserName [varchar](30)

AS

DECLARE @DispatchID [int]

INSERT [RAS].[CITI9\_DISP] ([ValueDate],[DT],[UserName])

VALUES (@ValueDate, GetDate(),@UserName)

SET @DispatchID = SCOPE\_IDENTITY()

EXEC RAS.[DispBSDEAL] @ValueDate, @DispatchID

EXEC RAS.[DispBSINT] @ValueDate, @DispatchID

EXEC RAS.[DispBSAMRT] @ValueDate, @DispatchID

SELECT @DispatchID as result

GO

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

-- RAS.DispBSDEAL

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

IF NOT EXISTS(SELECT 1 FROM INFORMATION\_SCHEMA.ROUTINES WHERE ROUTINE\_NAME = 'DispBSDEAL' AND ROUTINE\_TYPE = 'PROCEDURE'

AND ROUTINE\_SCHEMA = 'RAS')

BEGIN

PRINT 'Creating procedure RAS.[DispBSDEAL]'

EXEC ('CREATE PROCEDURE [RAS].[DispBSDEAL] AS RAISERROR(''Not implemented yet.'', 16, 3);')

END

PRINT 'Altering procedure RAS.[DispBSDEAL]'

GO

-----------------------------------------------------------------------------

-- RAS.DispBSDEAL creating

-----------------------------------------------------------------------------

ALTER PROCEDURE RAS.DispBSDEAL

@dt datetime,

@DispID int

AS

INSERT [RAS].[CITI9\_DISP\_BSDEAL]

(

[DispID], [DispatchDate], [DealNo], [Operation], [SymbolsPortCat], -- 1

[BookDT], [SettleDT], [InOut], [BaseNo], [ProductType], -- 2

[ISIN], [PricePcent], [FaceValue], [PrincipalValue], [CouponAmount], -- 3

[MiscCost], [CashAcctNo], [Depo], [DepoType], [RepoID], -- 4

[LinkedDealNo], [Quantity], [ScriptID], [PortfolioCat], [MiscCost5], -- 5

[LDContractNo], [FuturesFlag], [FuturesCode], [IsTechnical], [TechnicalLinkedDealNo], -- 6

[ForwardDealNo], [CouponSettleAcct], [SettleDT2], [TerminateFlag], [DRDealNo] -- 7

)

SELECT

-- 1 -----------------------------

@DispID,

B.[DispatchDate],

B.[dealNo],

B.[Operation],

B.[SymbolsPortCat],

-- 2 -----------------------------

B.[BookDT],

B.[SettleDT],

B.[InOut],

B.[BaseNo],

B.[ProductType],

-- 3 -----------------------------

B.[ISIN],

B.[PricePcent],

B.[FaceValue],

B.[PrincipalValue],

B.[CouponAmount],

-- 4 -----------------------------

B.[MiscCost],

B.[CashAcctNo],

B.[Depo],

B.[DepoType],

B.[RepoID],

-- 5 -----------------------------

B.[LinkedDealNo],

B.[Quantity],

B.[ScriptID],

B.[PortfolioCat],

B.[MiscCost5],

-- 6 -----------------------------

B.[LDContractNo],

ISNULL(B.[FuturesFlag], ' ') AS [FuturesFlag],

B.[FuturesCode],

CASE

WHEN B.[IsTechnical] = 1 AND B.[Operation] = '40' THEN 'X'

WHEN B.[IsTechnical] = 1 THEN 'Y'

ELSE ' '

END AS [IsTechnical],

[TechnicalLinkedDealNo],

-- 7 -----------------------------

CASE

WHEN b.[Operation] like '4%' OR b.[Operation] like '8%' OR b.Operation IN ('50', '55', '91')

THEN (SELECT TOP 1 DealNo FROM [RAS].[BSDEAL\_P] WHERE [SplitDealID] = B.[SplitDealID] AND [Operation] like '3%')

END AS [ForwardDealNo],

CASE

WHEN B.[Operation] IN ('K5', 'L5', 'F5') THEN dbo.fn\_GetCpAccount(BS.[CpID], 'DR\_Coupon', S.[Ccy])

WHEN B.[Operation] IN ('D5', 'E5') THEN dbo.fn\_GetCpAccount(BS.[CpID], 'RR\_Coupon', S.[Ccy])

END AS [CouponSettleAcct],

BS.[SettleDT2],

ISNULL(BS.[TerminateFlag], ' ') AS [TerminateFlag],

DR.[dealNo] AS [DRDealNo]

FROM [RAS].[BSDEAL] B

INNER JOIN [RAS].[BSDEAL\_P] BS ON B.[ID] = BS.[ID]

INNER JOIN dbo.[SCRIPTS] S ON BS.[ScriptID] = S.[ID]

LEFT JOIN [RAS].[BSDEAL\_P] DR ON DR.[ID] = BS.[DRDealID]

WHERE B.[DispatchDate] = @dt

ORDER BY B.[dealNo]

GO

## Диаграммы классов компонентов пользовательского интерфейса

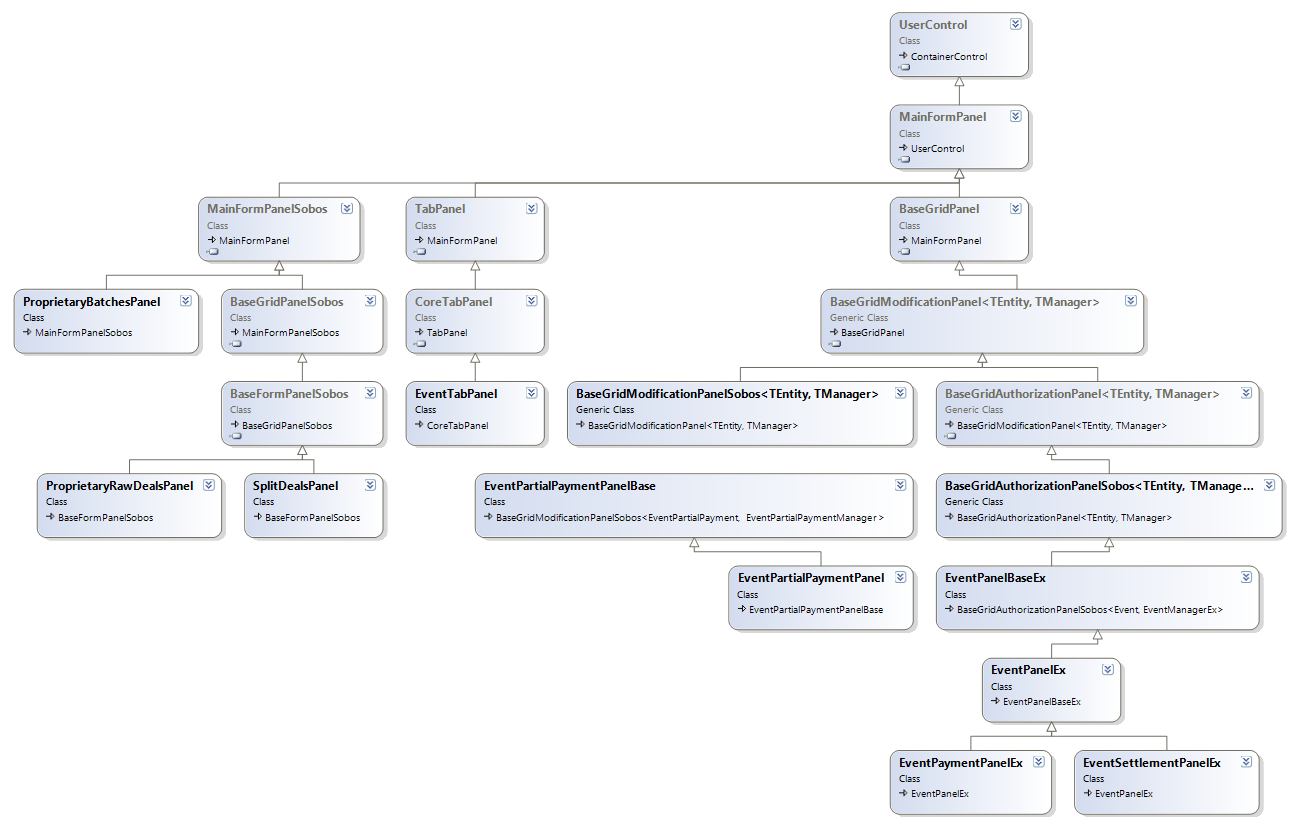


Рисунок П.1 — Диаграмма классов панелей

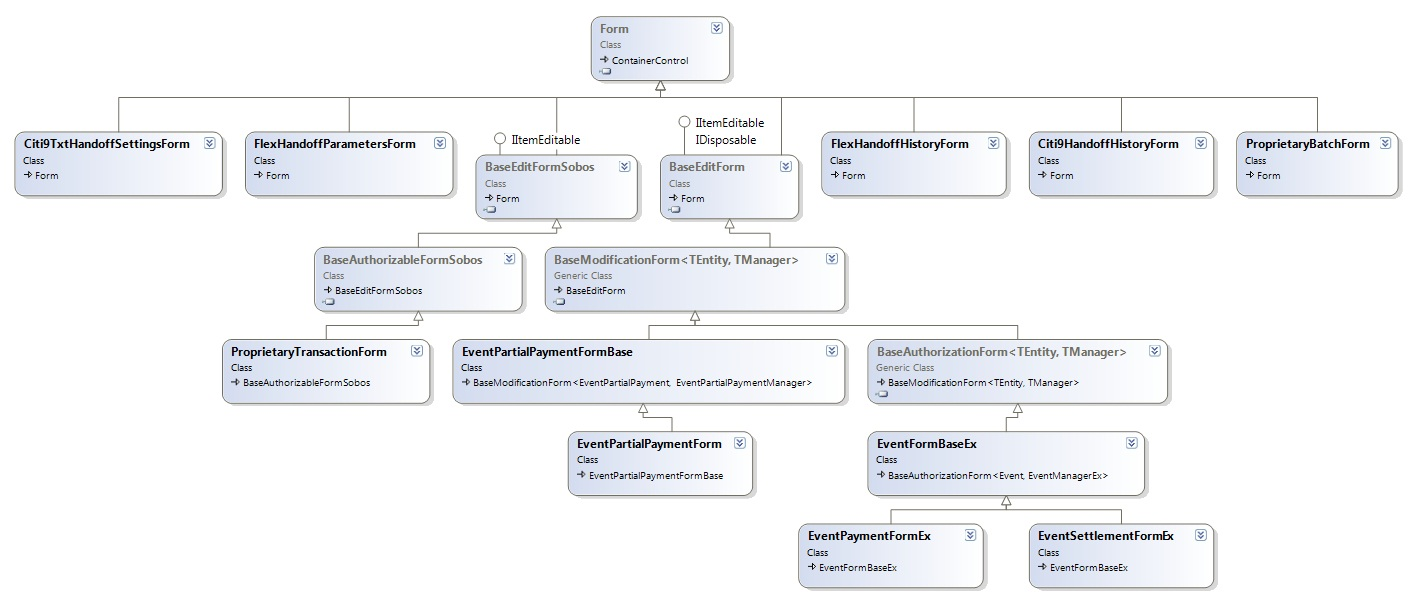


Рисунок П.2 — Диаграмма классов форм

## RDLC-шаблоны отчетов

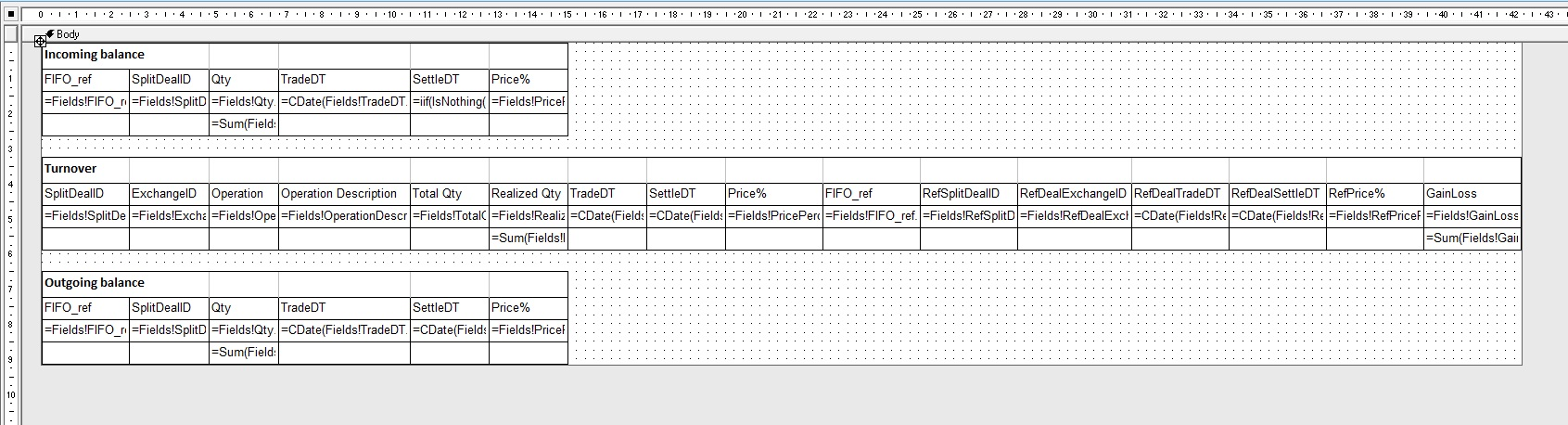


Рисунок П.3 — Шаблон отчета GAAP Disclosure report

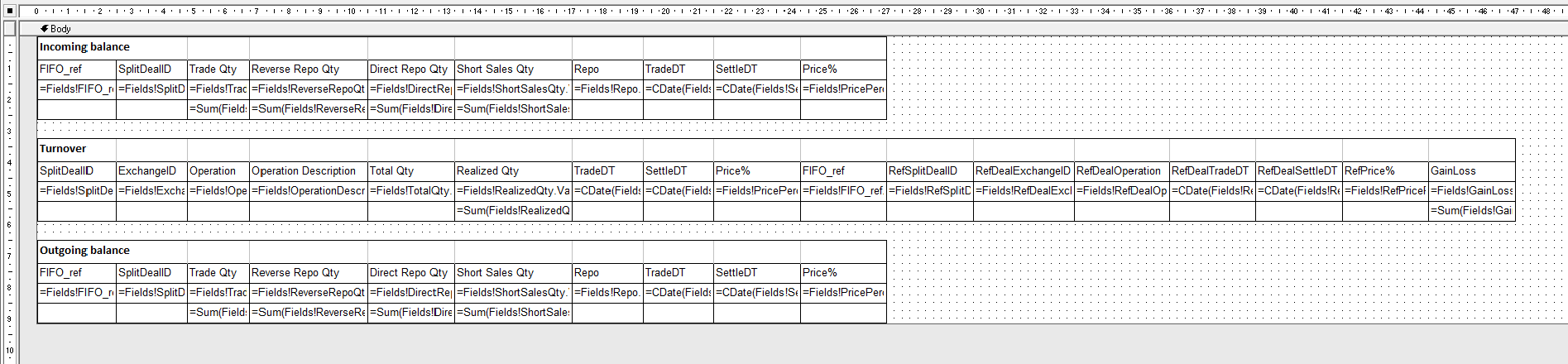


Рисунок П.4 — Шаблон отчета RAS Disclosure report