МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ПРАКТИЧЕСКИМ РАБОТАМ

«Разработка и администрирование в 1С, ERP»

Направление подготовки 09.03.04 «Программная инженерия»
Направленность (профиль) «Разработка и сопровождение программного обеспечения»
Квалификация выпускника: бакалавр

Ставрополь
2026

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Настоящие методические указания предназначены для выполнения лабораторных работ по дисциплине «Разработка и администрирование в 1С, ERP».

Цель дисциплины: сформировать у студентов системные компетенции в области проектирования, разработки, интеграции и администрирования корпоративных систем класса ERP на платформе «1С:Предприятие 8», включая понимание архитектурных особенностей платформы, объектной модели, языка запросов, механизмов транзакционности и интерфейсов интеграции с внешними системами (REST/SOAP/gRPC, БД, очереди сообщений), а также методов оптимизации производительности и администрирования промышленных конфигураций.

Задачи дисциплины:

1. Изучить архитектуру платформы «1С:Предприятие» как событийно-ориентированной (в отличие от классических ООП-языков) и её место в экосистеме корпоративных систем.

2. Освоить объектную модель 1С: справочники, документы, регистры (накопления, сведения, бухгалтерские, расчета) как аналог многомерных моделей данных (звезда/снежинка).

3. Научиться писать запросы на языке запросов 1С (аналог SQL с многомерной семантикой) с оптимизацией временных таблиц и индексов.

4. Сформировать навыки разработки управляемых форм и командного интерфейса как альтернативы React/Angular на клиенте.

5. Освоить методы интеграции 1С с внешними системами: HTTP-сервисы, REST/SOAP, вызов gRPC, обмен сообщениями через RabbitMQ/Kafka.

6. Изучить механизмы транзакционности, блокировок, отказоустойчивости и производительности в контексте ERP-систем.

7. Получить практические навыки администрирования: кластеризация, обновления, резервное копирование, мониторинг (в том числе через Prometheus/Grafana).

Данная методические указания включают 7 лабораторных работ, которые постепенно проводят студента от базовых принципов работы с метаданными и объектной моделью 1С до создания комплексного ERP-расширения с интеграцией, мониторингом и элементами администрирования. Каждая работа содержит чётко сформулированные цель и задачи, теоретическую часть, детальную методику выполнения с примерами, индивидуальные варианты заданий, контрольные вопросы, требования к отчёту и критерии оценивания.

Лабораторная работа №1.
Знакомство с платформой 1С: объектная модель и первый документ

Цель работы: освоить интерфейс конфигуратора (режим «Конфигуратор» и «Предприятие»), создать простейший справочник и документ, настроить движения документа по регистру накопления, получить навык построения простого отчёта.

Задачи:

- создать новую информационную базу в файловом режиме;
- создать справочник «Номенклатура»;
- создать документ «Приход товаров» с табличной частью;
- создать регистр накопления «Остатки товаров» (измерения, ресурс);
- реализовать программное формирование движений документа по регистру;
- создать простой отчёт по остаткам с использованием построителя отчёта или языка запросов.

Теоретическая часть

Платформа «1С:Предприятие 8» работает в событийно-ориентированной парадигме на уровне метаданных: разработчик создаёт не классы, а объекты метаданных (справочники, документы, регистры), которые автоматически получают набор стандартных событий, таких как ПриЗаписи, ПередУдалением или ОбработкаПроведения. Для начала работы необходимо создать новую информационную базу в файловом режиме — при запуске 1С нужно выбрать «Добавить» → «Создание новой информационной базы» → «Файловая» и указать каталог, например C:\1C_Base\LR1. Первым объектом обычно становится справочник «Номенклатура»: в конфигураторе в дереве метаданных нужно добавить справочник, задать имя и синоним, на закладке «Данные» оставить стандартный реквизит Наименование (тип Строка, длина

100), а на закладке «Прочее» установить автозаполнение кода. Запись справочника выполняется клавишей F7.

Методика выполнения работы

1. **Создание информационной базы**

   1.1. Установите инструменты нагрузочного тестирования:

   – Запустите «1С:Предприятие» → «Добавить» → «Создание новой информационной базы» → «Файловая» → Укажите каталог (например, C:\1C_Base\LR1).

2. **Создание справочника «Номенклатура»**

   2.1. Откройте Конфигуратор → дерево метаданных → правой кнопкой по «Справочники» → «Добавить».

   2.2. Имя: «Номенклатура», синоним: «Номенклатура».

   2.3. На закладке «Данные»: стандартный реквизит Наименование (тип Строка, длина 100).

   2.4. На закладке «Прочее»: установите «Код автозаполнение» = «Автонумерация».

   2.5. Запишите справочник (F7).

3. **Создание документа «Приход товаров» с табличной частью**

   3.1. Добавить документ: «Документы» → «Добавить». Имя = «ПриходТоваров», синоним = «Приход товаров».

   3.2. На закладке «Данные» создать **табличную часть** с именем «Товары». В табличной части добавить реквизиты:

   – Номенклатура (тип СправочникСсылка.Номенклатура)
   – Количество (тип Число, длина 10, точность 3)

   3.3. Записать документ.

4. **Создание регистра накопления «Остатки товаров»**

4.1. Добавить регистр накопления: «Регистры накопления» → «Добавить». Имя = «ОстаткиТоваров», синоним = «Остатки товаров».

4.2. Измерения: Номенклатура (тип СправочникСсылка.Номенклатура).

4.3. Ресурсы: Количество (тип Число, длина 10, точность 3, агрегатная функция = «Остаток»).

4.4. На закладке «Данные»: вид регистра – «Остатки» (для остаточного учёта).

4.5. Записать регистр.

5. **Реализация движений документа**

5.1. В документе «ПриходТоваров» перейти на закладку «Движения».

5.2. Нажать «Конструктор движений» → выбрать регистр «ОстаткиТоваров».

5.3. В сгенерированной процедуре ОбработкаПроведения написать (или оставить как есть):

```
Процедура ОбработкаПроведения(Отказ, Режим)
    Движения.ОстаткиТоваров.Записать = Истина;
    Для Каждого СтрокаТЧ Из Товары Цикл
        Движение = Движения.ОстаткиТоваров.Добавить();
        Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
        Движение.Номенклатура = СтрокаТЧ.Номенклатура;
        Движение.Количество = СтрокаТЧ.Количество;
    КонецЦикла;
КонецПроцедуры
```

6. **Создание отчёта по остаткам (через построитель)**

6.1. Добавить новый объект «Отчёт», имя «ОтчётОстаткиТоваров».

6.2. На закладке «Данные»: нажать «Открыть конструктор запроса» → выбрать регистр «ОстаткиТоваров.Остатки».

6.3. Выбрать поля: Номенклатура, КоличествоОстаток.

6.4. Сгенерировать форму отчёта с результатом.

6.5. В форме отчёта разместить элемент «ПолеТабличногоДокумента» и в событии ПриОткрытии выполнить: ПостроительОтчета.Выполнить().

Индивидуальные задания

1. Задания для индивидуального выполнения (15 вариантов). Каждый вариант отличается типом документа, ресурсом и/или дополнительным измерением.

Общее для всех вариантов:

- Создать документ с табличной частью, содержащей ссылку на указанный(е) справочник(и) и числовой ресурс.
- Провести документ по регистру накопления (вид регистра – Остатки, или Обороты если указано).
- Создать отчёт, показывающий остатки/обороты по всем измерениям.

Вариант	**Справочник (кроме Номенклатуры)**	**Документ**	**Регистр (измерения, ресурс)**	**Особенность**
1	Склады	Приход на склад	Склады, Номенклатура; Количество	-
2	Контрагенты	Поступление услуг	Контрагент, Услуга (новый справочник); Сумма	Создать справочник «Услуги»
3	Материалы	Приход материалов	Склад, Материал; Количество	Создать справочник «Материалы»
4	Подразделения	Приход ОС	Подразделение, ОС (новый); Количество	ОС – отдельный справочник
5	Контрагенты, Договоры	Приход денег	Контрагент; Сумма	Измерение – Контрагент
6	Сотрудники	Начисление зарплаты	Сотрудник; Сумма	Регистр – оборотов
7	Номенклатура, Склады	Перемещение товаров	СкладОтпр, СкладПолуч, Номенклатура; Количество	Два измерения-склада

8	Проекты	Приход товаров с проектом	Проект, Номенклатура; Количество	-
9	Характеристики номенклатуры	Приход товаров с характеристиками	Номенклатура, Характеристика; Количество	Создать справочник «Характеристики»
10	Серии	Приход по сериям	Номенклатура, Серия; Количество	Серии – отдельный справочник
11	Единицы измерения	Приход в разных единицах	Номенклатура, Единица; Количество	Ресурс – Количество (с точностью)
12	Транспортные средства	Приход топлива	ТС (спр.), Номенклатура; Литры	Ресурс – Литры (тип Число)
13	Группы финансового учёта	Приход с ФГУ	ФГУ, Номенклатура; Сумма	Справочник «ФГУ»
14	Места хранения (адресные)	Приход с адресом	МестоХранения, Номенклатура; Количество	Создать иерархический справочник
15	Заказчики (физ. лица)	Приход по заказам	Заказчик, Номенклатура; Количество	-

Требования к отчёту

Отчёт должен быть оформлен в электронном виде (PDF или DOCX) и содержать:

- титульный лист (ФИО, группа, вариант, номер работы);
- цель работы и задачи (скопировать из методички);
- краткое теоретическое обоснование (основные понятия платформы, объектная модель);
- практическая часть: скриншоты созданной конфигурации (дерево метаданных, форма справочника, структура регистра), листинг (текст) процедуры ОбработкаПроведения, скриншот проведённого документа и сформированных движений;
- результаты выполнения (скриншот отчёта с введёнными тестовыми данными);
- анализ результатов (что получилось, какие сложности возникли);

- ответы на контрольные вопросы (10 вопросов, письменно);
- вывод (освоенные навыки: создание справочника, документа, регистра, движений, отчёта);

Контрольные вопросы

1. Какие режимы запуска существуют в платформе 1С? Для чего нужен режим «Конфигуратор»?
2. Что такое объект метаданных? Приведите три примера.
3. Чем отличается справочник от документа?
4. Что такое табличная часть и зачем она нужна в документе?
5. Какие виды регистров накопления вы знаете? Объясните разницу между «Остатками» и «Обороты».
6. Что такое «измерение» и «ресурс» регистра?
7. Как связаны движения документа и регистр накопления?
8. Как программно добавить запись в набор движений? Как указать вид движения (приход/расход)?
9. Каким способом можно создать отчёт в 1С без глубокого знания запросов?
10. Что будет, если провести документ повторно? Как платформа обрабатывает старые движения?

Критерии оценивания

Оценка	Критерии
Отлично	Работа выполнена полностью, без ошибок. Конфигурация создана самостоятельно, все объекты названы в соответствии с заданием. Процедура проведения написана грамотно, с обработкой возможных ошибок (например, пустая табличная часть). Отчёт работает корректно и показывает данные.

	Реализовано дополнительное требование (вариативная часть): например, добавлен отбор по периоду в отчёт, или создан итог по ресурсу. Отчёт структурирован, содержит все скриншоты, диаграмму связей объектов, анализ. На защите студент уверенно отвечает на все вопросы.
Хорошо	Работа выполнена полностью, функционал работает. Есть незначительные замечания по оформлению кода (неиспользуемые переменные, несоблюдение отступов) или отчёта (отсутствует один скриншот). Вариативная часть не выполнена. На защите студент отвечает на большинство вопросов.
Удовлетворительно	Работа выполнена частично: основной функционал работает, но с ошибками (например, документ формирует движения только для первой строки табличной части). Отчёт неполный (отсутствует анализ или ответы на вопросы). Студент затрудняется при ответах на контрольные вопросы.
Неудовлетворительно	Работа не представлена. Ключевой функционал не работает (документ не проводится, регистр пуст, отчёт не выводит данные). Отчёт отсутствует или не соответствует требованиям (например, скопирован из сети без изменений).

Лабораторная работа №2.

Язык запросов 1С и оптимизация производительности

Цель работы: Освоить язык запросов 1С, научиться писать аналитические запросы с группировками и временными таблицами, выполнять замеры производительности и анализировать план запроса.

Задачи:

1. Создать и заполнить тестовую базу с регистрами «Остатки товаров» и/или «Обороты товаров».
2. Ввести несколько документов «Приход товаров» и «Расход товаров» для создания данных.
3. Написать запрос к виртуальной таблице **«Обороты»** регистра накопления с группировками по номенклатуре и периоду.
4. Реализовать расчёт средней цены товара с использованием **временных таблиц**.
5. Выполнить замер времени выполнения запроса с помощью ТекущаяУниверсальнаяДатаВМиллисекундах().
6. Проанализировать план запроса (через кнопку «Показать план запроса» в 1С).
7. *(Опционально)* Сравнить с аналогичным SQL-запросом к PostgreSQL (через внешнее подключение).

Теоретическая часть

Язык запросов 1С является SQL-подобным, но изначально ориентирован на многомерные регистры, предоставляя виртуальные таблицы — «Обороты», «Остатки», «ОстаткиИОбороты», «СрезПервых», «СрезПоследних». Эти таблицы уже содержат агрегированные данные с учётом периодичности, что отличает их от плоских таблиц SQL. Для написания запроса к виртуальной таблице «Обороты» регистра накопления, например «Продажи», указывается период и группировка по измерениям:

```
ВЫБРАТЬ ПродажиОбороты.Номенклатура, ПродажиОбороты.КоличествоОборот КАК Продано,
ПродажиОбороты.СуммаОборот КАК Выручка ИЗ РегистрНакопления.Продажи.Обороты(&НачПериод,
&КонПериод, День,) КАК ПродажиОбороты ГДЕ ПродажиОбороты.КоличествоОборот > 0
```

Временные таблицы создаются оператором ПОМЕСТИТЬ и позволяют разбить сложный запрос на этапы, избегая повторного чтения данных. Например, для вычисления средней цены товара за период сначала во временную таблицу ВТ_ИтогиПриходов помещаются суммы и количества из всех документов прихода, сгруппированные по номенклатуре:

```
ВЫБРАТЬ ПриходТовары.Номенклатура, СУММА(ПриходТовары.Количество * ПриходТовары.Цена) КАК
СуммаОбщая, СУММА(ПриходТовары.Количество) КАК КоличествоОбщее ПОМЕСТИТЬ ВТ_ИтогиПриходов ИЗ
Документ.ПриходТоваров.Товары КАК ПриходТовары ГДЕ ПриходТовары.Ссылка.Дата МЕЖДУ &ДатаНач И
&ДатаКон СГРУППИРОВАТЬ ПО ПриходТовары.Номенклатура
```

Затем основная выборка делит сумму на количество с приведением типа через ВЫРАЗИТЬ. Для замера производительности используется ТекущаяУниверсальнаяДатаВМиллисекундах(): замеры ставятся до и после выполнения запроса, разница даёт время в миллисекундах. План запроса анализируется через кнопку «Показать план запроса» в консоли запросов или отладчике — важно обращать внимание на наличие Table Scan (полное сканирование) и использование индексов. При сравнении с PostgreSQL можно создать аналогичную таблицу остатки_товаров и выполнить два параллельных обновления на уровне изоляции SERIALIZABLE, чтобы увидеть разницу в поведении (в PostgreSQL второй коммит вызовет ошибку сериализации, тогда как 1С в READ COMMITTED допустит оба изменения, что может привести к отрицательному остатку без явной блокировки).

Методика выполнения работы

1. Исходные данные (инфраструктура)

Предполагается, что у нас уже есть из ЛР №1:

- Справочник «Номенклатура».
- Регистр накопления «ОстаткиТоваров» (измерение: Номенклатура, ресурс: Количество).

- Документ «ПриходТоваров» (табличная часть: Номенклатура, Количество, Цена).
- Документ «РасходТоваров» (табличная часть: Номенклатура, Количество).

Добавим регистр «Продажи» (Обороты) для демонстрации виртуальной таблицы «Обороты».

2. Создание регистра оборотов «Продажи» (если отсутствует)

- Регистр накопления «Продажи», вид – «Обороты».
- Измерения: Номенклатура.
- Ресурсы: Количество, Сумма.
- Документ-регистратор: «РасходТоваров» (при проведении заполнять движения).

3. Запрос 1. Обороты по номенклатуре за январь 2025 г.

```
ВЫБРАТЬ
    ПродажиОбороты.Номенклатура,
    ПродажиОбороты.КоличествоОборот КАК Продано,
    ПродажиОбороты.СуммаОборот КАК Выручка
ИЗ
    РегистрНакопления.Продажи.Обороты(&НачПериод, &КонПериод, День,) КАК ПродажиОбороты
ГДЕ
    ПродажиОбороты.КоличествоОборот > 0
```

4. Запрос 2. Средняя цена товара за период с использованием временной таблицы

Задача: вычислить среднюю цену каждого товара на основе всех приходов.

```
ВЫБРАТЬ
    ПриходТовары.Номенклатура,
    СУММА(ПриходТовары.Количество * ПриходТовары.Цена) КАК СуммаОбщая,
    СУММА(ПриходТовары.Количество) КАК КоличествоОбщее
ПОМЕСТИТЬ ВТ_ИтогиПриходов
ИЗ
    Документ.ПриходТоваров.Товары КАК ПриходТовары
ГДЕ
    ПриходТовары.Ссылка.Дата МЕЖДУ &ДатаНач И &ДатаКон
СГРУППИРОВАТЬ ПО
    ПриходТовары.Номенклатура
;

ВЫБРАТЬ
    ВТ_ИтогиПриходов.Номенклатура,
    ВЫРАЗИТЬ(ВТ_ИтогиПриходов.СуммаОбщая / ВТ_ИтогиПриходов.КоличествоОбщее КАК ЧИСЛО(15, 2)) КАК Ср
едняяЦена
ИЗ
    ВТ_ИтогиПриходов КАК ВТ_ИтогиПриходов
```

5. Замер времени выполнения

```
Старт = ТекущаяУниверсальнаяДатаВМиллисекундах();
Запрос = Новый Запрос();
// ... текст запроса, установка параметров
Результат = Запрос.Выполнить();
Финиш = ТекущаяУниверсальнаяДатаВМиллисекундах();
Сообщить("Время выполнения (мс): " + (Финиш - Старт));
```

6. Анализ плана запроса

- В консоли запросов (или в отладчике) выполнить запрос.
- Нажать «Показать план запроса».
- Изучить: есть ли Table Scan (плохо), используется ли индекс по дате, какой порядок соединений.

Индивидуальные задания

1. Задания для индивидуального выполнения (15 вариантов).

Каждый вариант предполагает написание трёх запросов:

– Запрос к виртуальной таблице «Обороты» или «ОстаткиИОбороты».

– Запрос с временной таблицей для вычисления агрегата.

– Замер времени + анализ плана.

Исходные данные для всех вариантов: документы Приход и Расход за 3–6 месяцев.

Дополнительное задание (вариативная часть для оценки «Отлично»): Реализовать сравнение с PostgreSQL: выгрузить аналогичную таблицу фактов (например, через внешний источник или простой экспорт), написать эквивалентный SQL-запрос, замерить время выполнения в pgAdmin, сравнить с замером в 1С, сделать выводы.

Вариант	Измерения регистра	Дополнительное условие	Что вычисляем через временную таблицу	Особенность
1	Номенклатура	Только обороты > 100	Средняя цена по приходам	Период: текущий квартал
2	Номенклатура, Склад	Обороты по конкретному складу	Средний остаток на конец дня	Сортировка по убыванию остатка
3	Контрагент	Сумма > 5000	Максимальная сумма одной продажи	Группировка по месяцам
4	Номенклатура, Подразделение	Только расходные накладные	Коэффициент оборачиваемости (расход/приход)	Фильтр по подразделению
5	Номенклатура, Характеристика	Характеристика = "Красный"	Средний срок хранения (по датам прихода/расхода)	Использовать РАЗНОСТЬДАТ
6	Проект	Обороты по проекту А	Итого по проектам	Добавить УПОРЯДОЧИТЬ ПО
7	Номенклатура, Единица измерения	ЕИ = "шт"	Количество уникальных номенклатур	Использовать КОЛИЧЕСТВО РАЗЛИЧНЫХ
8	Сотрудник (ответственный)	Только приходы	Количество документов на сотрудника	Соединение с таблицей сотрудников
9	Номенклатура, Серия	Остатки по сериям	Доля (процент) каждой серии в общем количестве	Вычисление процента во временной таблице
10	Место хранения (адрес)	Адрес = "Стеллаж А"	Среднее количество на адрес	Подзапрос в ВЫРАЗИТЬ

11	Номенклатура, Вид цен (новый справочник)	Цена = закупочная	Минимальная и максимальная цена	Два ресурса во временной таблице
12	Группа номенклатуры (иерархия)	Товары из группы "Электроника"	Средняя цена по группе	Использовать В ИЕРАРХИИ
13	Договор контрагента	Обороты по договорам	Сумма с НДС (добавить ставку 20%)	Вычисляемое поле
14	Номенклатура, Тип упаковки	Разные упаковки	Пересчёт в базовую единицу (например, в кг)	Коэффициент из отдельного справочника
15	Произвольные измерения (по выбору)	Свой вариант из ЛР №1	Свой агрегат	Обосновать в отчёте необходимость временной таблицы

Требования к отчёту

Отчёт должен быть оформлен в текстовом редакторе (PDF) и содержать:

- **Титульный лист** (ФИО, группа, вариант, номер работы).
- **Цель и задачи работы**.
- **Краткое теоретическое обоснование** (специфика языка запросов 1С, виртуальные таблицы, временные таблицы).
- **Практическая часть**:
  - Схема регистров и документов, использованных в запросах.
  - Тексты всех трёх запросов с пояснениями.
  - Скриншот окна «План запроса» с анализом (например, «используется индекс по дате», «нет Table Scan»).
  - Листинг кода замера времени.
  - *(Опционально для сравнения с PostgreSQL)* SQL-запрос, скриншот замера.
- **Результаты выполнения** (таблица с результатами запроса — выборка из 5-10 строк).
- **Анализ результатов** (сравнение времени выполнения при разных условиях, какой запрос быстрее и почему).

- **Ответы на контрольные вопросы** (письменно).
- **Вывод** (какие приёмы оптимизации освоены).

Контрольные вопросы

1. Чем отличается виртуальная таблица «Остатки» от «Обороты»?
2. В каких случаях использование временной таблицы ускоряет запрос?
3. Как в 1С-запросе получить текущую дату и время?
4. Что делает оператор ПОМЕСТИТЬ? Можно ли создать индекс во временной таблице?
5. Как посмотреть план запроса в 1С и на что обращать внимание?
6. Какая разница между СГРУППИРОВАТЬ ПО и ИТОГИ ПО?
7. Зачем нужен оператор ВЫРАЗИТЬ? Когда его использование безвредно, а когда вредно?
8. Как выполнить запрос с параметрами через Запрос.УстановитьПараметр()?
9. Почему не рекомендуется в виртуальной таблице «Обороты» ставить измерение в условие ВЫБРАТЬ * без отбора по периоду?
10. Можно ли внутри одного запроса обратиться к двум разным виртуальным таблицам одного регистра? Будет ли это эффективно?

Критерии оценивания

Оценка	Критерии
Отлично	Работа выполнена полностью без ошибок. Все три запроса работают корректно и оптимизированы. Проведён анализ плана запроса с выводами. Реализована вариативная часть (сравнение с PostgreSQL или дополнительное задание на выбор преподавателя). Отчёт содержит все разделы, скриншоты, листинги. На защите — уверенные ответы на все вопросы.

Хорошо	Работа выполнена полностью. Запросы работают, но есть незначительные замечания (например, не оптимизирована временная таблица — отсутствует индекс). План запроса представлен, но анализ поверхностный. Вариативная часть не выполнена. Отчёт неполный (нет одного скриншота или листинга). На защите — ответы на большинство вопросов.
Удовлетворительно	Работа выполнена частично (например, выполнен только один запрос из трёх, или запрос работает с ошибками, или замер времени не сделан). Отчёт отсутствует или сильно урезан. Студент затрудняется ответить на контрольные вопросы.
Неудовлетворительно	Работа не представлена. Запросы не работают или выдают неверные данные. Отчёт не соответствует требованиям (скопирован из интернета без изменений). Ключевые элементы (временная таблица, анализ плана) отсутствуют.

Лабораторная работа №3.
Разработка управляемой формы и командного интерфейса

Цель работы: Реализовать интерактивную управляемую форму с командным интерфейсом, обработчиками событий и элементами динамического обновления данных.

Задачи:

- Создать управляемую форму списка документов «Приход товаров».
- Добавить на форму командную панель с кнопками «Провести» и «Отменить проведение».
- Реализовать программную бизнес-логику проведения и отмены проведения документа.
- Реализовать динамическое обновление табличной части формы при выборе номенклатуры (автозаполнение цены или остатка).
- Добавить условное оформление: выделение цветом проведённых документов в списке.
- Обеспечить интерактивную обратную связь с пользователем (сообщения об успехе/ошибке).

Теоретическая часть

Управляемые формы в 1С строятся на принципе декларативного описания структуры и привязки к реквизитам, в отличие от обычных форм, где код отвечает за отрисовку. Основными понятиями являются реквизиты формы (данные, отображаемые на форме, например Объект или Список), команды (действия пользователя в виде кнопок или ссылок) и элементы формы (поля, таблицы, кнопки, связанные с реквизитами и командами). Для создания управляемой формы списка документов «Приход товаров» в конфигураторе нужно открыть документ, на закладке «Формы» добавить «Форму списка», в дизайнере включить командную панель, добавить реквизит формы Список типа ДинамическийСписок с источником

данных Документ.ПриходТоваров, а затем разместить элемент «Таблица» на основе этого реквизита. Командный интерфейс формируется добавлением команд — например, КомандаПровести и КомандаОтменитьПроведение, — которые размещаются на командной панели таблицы. В модуле формы реализуются обработчики с директивой &НаСервере, поскольку проведение документа работает с базой данных. В процедуре ВыполнитьПроведение получается текущая строка списка через Список.ТекущаяСтрока, из неё извлекается объект документа через ПолучитьОбъект(), проверяется флаг Проведен, и если документ не проведён, вызывается ОбъектДокумента.Провести() и Записать(), после чего список обновляется методом Список.Обновить(). Ошибки перехватываются конструкцией Попытка...Исключение с выводом сообщения. Динамическое обновление табличной части формы документа реализуется через событие ПриИзменении у поля выбора номенклатуры. В обработчике на клиенте вызывается серверная процедура, которая выполняет запрос к регистру сведений или остатков, например для подстановки цены: ВЫБРАТЬ ЦеныНоменклатуры.Цена ИЗ РегистрСведений.ЦеныНоменклатуры.СрезПоследних(&Номенклатура,). Полученное значение присваивается соответствующей строке табличной части через обращение к Объект.Товары.Найти(...). Условное оформление настраивается в форме списка через кнопку «Условное оформление» на панели команд: добавляется новое условие, задаётся оформление (например, светло-зелёный фон), отбор по полю Проведен = Истина, и применяется к полю Список. В результате проведённые документы визуально выделяются, что улучшает пользовательский опыт.

Методика выполнения работы

1. Создание управляемой формы списка документов

1. В конфигураторе открыть документ «ПриходТоваров».
2. На закладке «Формы» → добавить «Форма списка».

3. В дизайнере форм:
    - Включить командную панель.
    - Добавить реквизит формы Список (тип: ДинамическийСписок), источник данных – документ.ПриходТоваров.
    - Добавить элемент формы «Таблица» на основе реквизита Список.

2. Добавление команд «Провести» и «Отменить проведение»

Шаг 1: в форме списка добавить две команды:

- Команда КомандаПровести (синоним «Провести»).
- Команда КомандаОтменитьПроведение (синоним «Отменить проведение»).

Шаг 2: разместить их на командной панели таблицы.

Шаг 3: реализовать обработчики (модуль формы):

```
&НаСервере
Процедура ВыполнитьПроведение(Команда)
    ЭлементыФормы = Элементы;
    СтрокаСписка = Список.ТекущаяСтрока;
    Если СтрокаСписка = Неопределено Тогда
        Сообщить("Не выбран документ для проведения");
        Возврат;
    КонецЕсли;

    ОбъектДокумента = СтрокаСписка.ПолучитьОбъект();
    Если ОбъектДокумента.Проведен Тогда
        Сообщить("Документ уже проведён");
        Возврат;
    КонецЕсли;

    Попытка
        ОбъектДокумента.Провести();
        ОбъектДокумента.Записать();
        Сообщить("Документ проведён успешно");
        Список.Обновить();  // обновить список
    Исключение
        Сообщить("Ошибка при проведении: " + ОписаниеОшибки());
    КонецПопытки;
КонецПроцедуры

&НаСервере
Процедура ВыполнитьОтменуПроведения(Команда)
    // аналогично, но ОбъектДокумента.ОтменитьПроведение()
    ...
КонецПроцедуры
```

Привязать действия команд к этим процедурам в свойствах команд.

3. Динамическое обновление табличной части

Допустим, на форме документа (не списка) есть табличная часть «Товары» с колонками: Номенклатура, Количество, Цена. Требуется: при выборе номенклатуры автоматически подставить цену из регистра сведений «ЦеныНоменклатуры».

В модуле формы документа:

```
&НаКлиенте
Процедура ТоварыНоменклатураПриИзменении(Элемент)
    // Элемент – это поле Номенклатура в табличной части
    Параметры = Новый Структура;
    Параметры.Вставить("Номенклатура", Элемент.ТекущиеДанные.Номенклатура);
    ПодставитьЦенуНаСервере(Параметры);
КонецПроцедуры

&НаСервере
Процедура ПодставитьЦенуНаСервере(Параметры)
    Запрос = Новый Запрос;
    Запрос.Текст =
        "ВЫБРАТЬ
        |    ЦеныНоменклатуры.Цена
        |ИЗ
        |    РегистрСведений.ЦеныНоменклатуры.СрезПоследних(&Номенклатура,) КАК ЦеныНоменклатуры";
    Запрос.УстановитьПараметр("Номенклатура", Параметры.Номенклатура);
    Результат = Запрос.Выполнить();
    Если Не Результат.Пустой() Тогда
        СтрокаТаблицы = Объект.Товары.Найти(Параметры.Номенклатура, "Номенклатура");
        Если СтрокаТаблицы <> Неопределено Тогда
            СтрокаТаблицы.Цена = Результат[0].Цена;
        КонецЕсли;
    КонецЕсли;
КонецПроцедуры
```

4. Условное оформление проведённых документов в списке

1. Открыть форму списка документа → «Условное оформление» (на панели команд).
2. Создать новое условие:
    - **Оформление**: цвет фона – светло-зелёный.
    - **Отбор**: Проведен = Истина.
3. Применить к полю Список (ко всей строке).

Индивидуальные задания

1. Задания для индивидуального выполнения (15 вариантов). Каждый вариант расширяет базовую задачу специфическим требованием.

Общее для всех вариантов:

- Создать управляемую форму списка (если документ уже был – использовать его).
- Добавить минимум две пользовательские команды (одна – «Провести»/«Отменить» или аналог).
- Реализовать динамическое обновление хотя бы одного реквизита табличной части.
- Настроить условное оформление (минимум одно правило).

Вариативная часть (для оценки «Отлично»):

- Добавить обработку выделенных строк (множественное проведение выбранных документов).
- Реализовать прогресс-бар при массовой операции.

Вариант	Тип документа (исходный)	Дополнительное требование к командам	Динамическое обновление	Условное оформление
1	Приход товаров	Кнопка «Копировать документ» (создать новый на основе текущего)	При выборе номенклатуры – подстановка последней цены покупки	Проведённые – зелёный, непроведённые – розовый
2	Расход товаров	Кнопка «Установить флаг "Отгружен"» (добавить реквизит документа)	При выборе склада – подстановка остатка в отдельное поле	Просроченные (дата > сегодня) – красный
3	Поступление услуг	Кнопка «Заполнить НДС» (вычислить 20% от суммы)	При выборе услуги – подстановка нормо-часов	Услуги с НДС – жёлтый фон
4	Приход материалов	Кнопка «Разнести по ячейкам» (показать диалог с местом хранения)	При выборе единицы измерения – пересчёт количества	Материалы с остатком < 10 – оранжевый

5	Перемещение товаров	Кнопка «Инвертировать направления» (склад-отправитель ↔ склад-получатель)	При выборе номенклатуры – показать остаток на складе-отправителе	Строки с нулевым количеством – серый
6	Начисление зарплаты	Кнопка «Рассчитать премию» (10% от оклада)	При выборе сотрудника – подстановка оклада из справочника	Сотрудники с премией > 5000 – жирный шрифт
7	Приход ОС	Кнопка «Начислить амортизацию» (показать диалог с периодом)	При выборе ОС – подстановка инвентарного номера	ОС со сроком службы > 5 лет – синий
8	Приход денег (платёжка)	Кнопка «Создать на основе авансового отчёта» (выбрать другой документ)	При выборе контрагента – подстановка банковских реквизитов	Проведённые – зелёный, удаленные (пометка удаления) – зачёркнутый
9	Приход товаров с сериями	Кнопка «Проверить серии» (наличие серии в справочнике)	При выборе серии – подстановка срока годности	Серии с истекшим сроком – красный
10	Отчёт о продажах (документ-отчёт)	Кнопка «Обновить данные отчёта» (перезапросить)	При выборе периода – автоматический перерасчёт	Ячейки с отрицательной суммой – красный
11	Заказ клиента	Кнопка «Резервировать товары» (создать движения в регистре резерва)	При выборе номенклатуры – показать свободный остаток	Заказы без резерва – серый
12	Счёт на оплату	Кнопка «Печать счёта» (вызвать печатную форму)	При выборе валюты – подстановка курса из регистра	Счета с суммой > 100000 – полужирный
13	Акт выполненных работ	Кнопка «Подписать ЭЦП» (имитация – сообщение)	При выборе подразделения – подстановка руководителя	Акты без подписи – курсив
14	Требование-накладная	Кнопка «Списать по лимиту» (сравнить с лимитом из регистра)	При выборе номенклатуры – подстановка лимита	Превышение лимита – красный фон
15	Возврат товаров	Кнопка «Провести сторно» (провести с отрицательными количествами)	При выборе исходного документа –	Строки возврата – фиолетовый

			подстановка его табличной части	

Требования к отчёту

Отчёт должен содержать следующие разделы:

- **Титульный лист**.
- **Цель и задачи работы**.
- **Краткое теоретическое обоснование** (управляемые формы, команды, события, условное оформление).
- **Практическая часть**:
  - Скриншот разработанной формы списка с командной панелью.
  - Листинг обработчиков команд (минимум два).
  - Листинг обработчика динамического обновления.
  - Скриншот настройки условного оформления.
- **Результаты выполнения**:
  - Демонстрация работы кнопок (проведение, отмена).
  - Демонстрация автозаполнения при выборе номенклатуры.
  - Демонстрация условного оформления (проведённые и непроведённые документы).
- **Анализ результатов** (с какими трудностями столкнулись, как решили проблемы синхронизации клиент-сервер).
- **Ответы на контрольные вопросы**.
- **Вывод** (освоенные навыки работы с управляемыми формами, командами, событиями, условным оформлением).

Контрольные вопросы

1. Что такое управляемая форма? Чем она отличается от обычной формы?
2. Какие способы размещения команд на форме вы знаете?

3. Зачем нужна директива &НаКлиенте и &НаСервере? Что произойдёт, если вызвать Объект.Провести() на клиенте?
4. Как обновить данные динамического списка после выполнения серверной операции?
5. Что такое реквизит формы и элемент формы? Приведите пример связи.
6. Как программно установить значение в определённую строку табличной части формы?
7. Как настроить условное оформление для строк списка на основе значения поля?
8. Что делает метод Элементы.Таблица.ТекущиеДанные?
9. Как отловить изменение значения в поле табличной части (например, выбор номенклатуры)?
10. Можно ли из обработчика команды на клиенте вызвать серверную процедуру с передачей структуры параметров? Как?

Критерии оценивания

Оценка	Критерии
Отлично	Работа выполнена полностью без ошибок. Создана управляемая форма списка, все команды работают корректно. Динамическое обновление табличной части реализовано грамотно (с соблюдением клиент-серверного разделения). Условное оформление настроено и отображается. Реализована вариативная часть (массовое проведение/прогресс-бар). Отчёт полный, с анализом и скриншотами. На защите — уверенные ответы на все вопросы.
Хорошо	Работа выполнена полностью. Основной функционал работает. Есть замечания: например, обработчик динамического обновления не вынесен на сервер (тормозит), или условное оформление работает не для всех строк. Вариативная часть не выполнена. Отчёт неполный

	(отсутствует один скриншот или листинг). На защите — ответы на большинство вопросов.
Удовлетворительно	Работа выполнена частично (например, только одна команда из двух, или динамическое обновление не работает, или условное оформление отсутствует). Отчёт урезан. Студент затрудняется при ответах на контрольные вопросы.
Неудовлетворительно	Работа не представлена. Форма не создана или не открывается. Команды не вызываются или вызывают ошибки. Условное оформление отсутствует. Отчёт не соответствует требованиям.

Лабораторная работа №4.

Интеграция 1С с внешним REST API (потребление)

Цель работы: Освоить механизм интеграции платформы 1С с внешними REST API: выполнение HTTP-запросов, парсинг JSON, обработка ответов и загрузка данных в регистр сведений.

Задачи:

- Развернуть локальный эмулятор внешнего REST API (Python + FastAPI) с эндпоинтом, возвращающим курсы валют (или другие данные по варианту).
- Создать в 1С **регистр сведений** «КурсыВалют» (периодический, измерение – Валюта, ресурс – Курс).
- Разработать внешнюю обработку (или общую форму) с кнопкой «Загрузить курсы».
- Реализовать в 1С:
  - Объект HTTPСоединение (или HTTPЗапрос).
  - Выполнение GET-запроса к эмулятору.
  - Чтение JSON-ответа через ЧтениеJSON.
  - Запись полученных курсов в регистр сведений.
- Добавить обработку ошибок: таймаут, недоступность сервера, некорректный JSON.
- Проверить корректность загрузки данных.

Теоретическая часть

Интеграция 1С с внешними REST API требует понимания механизмов HTTP-запросов и формата JSON. Платформа предоставляет объекты HTTPСоединение (для настройки прокси, таймаутов, аутентификации) и HTTPЗапрос (метод GET/POST, заголовки, тело). При потреблении внешнего API 1С выступает в роли клиента. Для лабораторной

работы сначала разворачивается локальный эмулятор на Python с использованием FastAPI, который имитирует внешний сервис курсов валют. Эмулятор запускается на http://127.0.0.1:8000 и предоставляет эндпоинт /rates?date=2025-01-15, возвращающий JSON вида {"date": "...", "base": "RUB", "rates": {"USD": 91.5, "EUR": 99.2, ...}}. В 1С создаётся периодический регистр сведений «КурсыВалют» с измерением Валюта (ссылка на справочник) и ресурсом Курс (число). Разрабатывается внешняя обработка или общая форма с реквизитом ДатаЗапроса и кнопкой «Загрузить курсы». В обработчике команды на сервере создаётся HTTPСоединение с указанием хоста и порта, таймаутом 30 секунд. Формируется HTTPЗапрос с методом GET и URL вида /rates?date=2025-01-15. После вызова метода HTTPСоединение.ВызватьHTTPМетод() проверяется код состояния ответа (200 — успех). Тело ответа получается как строка через Ответ.ПолучитьТелоКакСтроку(). Для парсинга JSON используется ЧтениеJSON и функция ПрочитатьJSON, которая преобразует JSON в структуры и массивы 1С. Ожидается, что в корне ответа находится структура с ключом rates, значение которого — ещё одна структура, где ключи — коды валют (например, "USD"), а значения — числовые курсы. Далее в цикле перебираются пары ключ-значение этой структуры. Для каждой валюты находится соответствующая ссылка в справочнике «Валюты» по коду (например, через Справочники.Валюты.НайтиПоКоду()). Затем создаётся запись регистра сведений через РегистрыСведений.КурсыВалют.СоздатьЗапись(), задаётся период (дата из запроса), измерение Валюта и ресурс Курс, после чего вызывается Записать(). Вся работа с внешним API должна быть обёрнута в конструкцию Попытка...Исключение для обработки ошибок подключения, таймаутов, некорректного JSON или кода состояния, отличного от 200. Пользователю выводится информативное сообщение. В результате регистр

заполняется актуальными курсами на указанную дату, что демонстрирует полный цикл интеграции 1С с внешним REST-сервисом.

Методика выполнения работы

1. Создание эмулятора внешнего API на Python (FastAPI)

Создайте файл currency_api.py:

```python
from fastapi import FastAPI
from datetime import date
from typing import Dict

app = FastAPI()

@app.get("/rates")
def get_rates(date_str: str = None):
    """
    Пример эндпоинта /rates?date=2025-01-15
    Возвращает курсы валют к рублю.
    """
    # Имитация данных: для простоты всегда возвращаем фиксированные курсы
    rates = {
        "USD": 91.50,
        "EUR": 99.20,
        "CNY": 12.75,
        "GBP": 116.80
    }
    # В реальном проекте здесь был бы запрос к внешнему источнику или база
    return {
        "date": date_str or str(date.today()),
        "base": "RUB",
        "rates": rates
    }

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="127.0.0.1", port=8000)
```

Запуск: pip install fastapi uvicorn → python currency_api.py

Проверка: curl http://127.0.0.1:8000/rates?date=2025-01-15

2. Создание в 1С регистра сведений «КурсыВалют»

- Имя: КурсыВалют.
- Периодичность: В пределах дня.
- Измерения: Валюта (тип СправочникСсылка.Валюты – предварительно создать справочник «Валюты» с предопределёнными USD, EUR, CNY, GBP).

- Ресурсы: Курс (тип Число, длина 10, точность 4).

3. Разработка обработки (или формы) с командой «Загрузить курсы»

Создайте внешнюю обработку или общую форму. Поместите кнопку и реквизит ДатаЗапроса (тип Дата).

Модуль формы (основной код):

```
&НаКлиенте
Процедура ЗагрузитьКурсы(Команда)
   ЗагрузитьКурсыНаСервере(Объект.ДатаЗапроса);
КонецПроцедуры

&НаСервере
Процедура ЗагрузитьКурсыНаСервере(ДатаЗапроса)
   // 1. Настройка HTTP-соединения
   Попытка
      HTTPСоединение = Новый HTTPСоединение("127.0.0.1", 8000, , , , , 30); // таймаут 30 сек
   Исключение
      Сообщить("Ошибка подключения: " + ОписаниеОшибки());
      Возврат;
   КонецПопытки;

   // 2. Формирование запроса
   URL = "/rates?date=" + Формат(ДатаЗапроса, "ДФ=гггг-ММ-дд");
   HTTPЗапрос = Новый HTTPЗапрос(URL);
   HTTPЗапрос.Метод = "GET";
   HTTPЗапрос.Заголовки.Вставить("Accept", "application/json");

   // 3. Выполнение запроса
   Попытка
      Ответ = HTTPСоединение.ВызватьHTTPМетод(HTTPЗапрос);
   Исключение
      Сообщить("Ошибка при вызове API: " + ОписаниеОшибки());
      Возврат;
   КонецПопытки;
```

```
Если Ответ.КодСостояния <> 200 Тогда
    Сообщить("Сервер вернул ошибку: " + Ответ.КодСостояния);
    Возврат;
КонецЕсли;

// 4. Парсинг JSON
Попытка
    ЧтениеJSON = Новый ЧтениеJSON(Ответ.ПолучитьТелоКакСтроку());
    Данные = ПрочитатьJSON(ЧтениеJSON, Ложь);
Исключение
    Сообщить("Ошибка разбора JSON: " + ОписаниеОшибки());
    Возврат;
КонецПопытки;

// Ожидаем, что Данные - структура с ключами: date, base, rates
Если ТипЗнч(Данные) <> Тип("Структура") Тогда
    Сообщить("Неверный формат ответа");
    Возврат;
КонецЕсли;

Курсы = Данные.rates; // Структура вида { "USD": 91.5, "EUR": 99.2, ... }

// 5. Запись в регистр сведений
ЗаписьРегистра = РегистрыСведений.КурсыВалют.СоздатьЗапись();
ЗаписьРегистра.Период = ДатаЗапроса;

Для каждого КлючЗначение Из Курсы Цикл
    КодВалюты = КлючЗначение.Ключ;  // строка "USD"
    Курс = КлючЗначение.Значение;

    // Найти валюту в справочнике по коду (предположим, есть реквизит Код)
    ВалютаСсылка = Справочники.Валюты.НайтиПоКоду(КодВалюты);
    Если ВалютаСсылка = Справочники.Валюты.ПустаяСсылка() Тогда
        Сообщить("Валюта " + КодВалюты + " не найдена в справочнике");
```

```
            Продолжить;
        КонецЕсли;

        ЗаписьРегистра.Валюта = ВалютаСсылка;
        ЗаписьРегистра.Курс = Курс;
        ЗаписьРегистра.Записать();
    КонецЦикла;

    Сообщить("Загружено курсов: " + Курсы.Количество());
КонецПроцедуры
```

4. Проверка результата

- Запустить эмулятор API.
- В 1С открыть обработку, указать дату (например, текущую), нажать «Загрузить курсы».
- Открыть регистр сведений «КурсыВалют» – должны появиться записи.

Индивидуальные задания

1. Задания для индивидуального выполнения (15 вариантов). Каждый вариант отличается **эндпоинтом API**, **типом получаемых данных** и **целевым регистром**.

Общее для всех вариантов:

- Создать эмулятор API на Python (FastAPI) с одним эндпоинтом.
- В 1С создать регистр сведений или другой объект для хранения данных.
- Реализовать вызов API, парсинг JSON, запись в регистр.
- Обработать ошибки: таймаут, некорректный ответ, отсутствие данных.

Вариативная часть (для оценки «Отлично»):

- Добавить в обработку **фоновое задание** для асинхронной загрузки (не блокирует интерфейс).
- Реализовать **журнал загрузок** (когда, сколько записей, статус).

Вариант	Эмулятор API (эндпоинт, возвращаемые данные)	Целевой регистр 1С	Измерения / Ресурсы	Особенность обработки
1	/weather?city=Moscow – температура, влажность	Погода (периодический)	Город (справочник); Температура, Влажность	Обработать отсутствие города
2	/exchange?from=RUB&to=USD – курс обмена	КурсыВалют (добавить пару валют)	ВалютаИз, ВалютаВ; Курс	Два измерения – валюта
3	/cbrf/daily – официальные курсы ЦБ РФ (XML, но преобразуйте в JSON)	КурсыЦБ	Валюта; Курс, Номинал	Парсинг JSON с вложенным массивом
4	/metals?date= – цены на драгметаллы (золото, серебро)	ЦеныМеталлов	Металл (справочник); ЦенаЗаГрамм	Дробные числа с высокой точностью
5	/stocks?ticker=SBER – цена акции	КотировкиАкций	Тикер (справочник); ЦенаОткрытия, ЦенаЗакрытия	Два ресурса, обработка отсутствия данных
6	/crypto?pair=BTC/USDT – цена криптовалюты	КурсыКриптовалют	Криптопара (справочник); Цена, Объем24ч	Асинхронный запрос (имитация)
7	/delivery/cost?weight=10&city=1 – стоимость доставки	СтоимостьДоставки	Город, ВесИнтервал; Стоимость	Преобразование весовых интервалов
8	/holidays?year=2025 – список праздников на год	Праздники (регистр, непериодический)	Дата, Регион; НазваниеПраздника	Множественная запись (массив объектов)
9	/convert?from=USD&to=EUR&amount=100 – результат конвертации	Конвертация (регистр-журнал)	ВалютаИз, ВалютаВ, СуммаВход; СуммаВыход, Комиссия	Вычисление комиссии на основе полученного курса
10	/ipgeo?ip=8.8.8.8 – геолокация по IP	ГеолокацияIP (непериодический)	IP-адрес (строка); Страна, Город, Широта, Долгота	Парсинг вложенных объектов (country.code)
11	/sms/send?phone=...&text=... – имитация отправки SMS (POST)	Журнал регистрации отправленных SMS (регистр или документ)	НомерТелефона; Текст, Статус, ДатаОтправки	Использовать POST-запрос с телом JSON

12	/qr/generate?data=... – генерация QR-кода (возвращает base64-изображение)	Регистр сведений с двоичными данными	КлючДанных; QRИзображение (ХранилищеЗначения)	Преобразование base64 в картинку
13	/translate?text=Hello&lang=en-ru – перевод текста	Переводы (регистр)	ИсходныйТекст, Направление; ПереведенныйТекст	Обработка длинных строк
14	/validate/inn?inn=1234567890 – проверка ИНН (валидность)	РезультатыПроверкиИНН	ИНН; Валидный, Комментарий	GET-параметр с ИНН, ответ – структура
15	/rates (как в примере) + date – курсы валют	КурсыВалют	Валюта; Курс, Дата (период)	Дополнительно: если курс уже есть на дату – обновить или пропустить

Требования к отчёту

Отчёт должен быть оформлен в электронном виде (PDF) и содержать:

- **Титульный лист**.
- **Цель и задачи работы**.
- **Краткое теоретическое обоснование** (REST, JSON, HTTP-объекты 1С, регистр сведений).
- **Практическая часть**:
  - Листинг Python-скрипта эмулятора API.
  - Скриншот запущенного эмулятора (или вывод curl).
  - Скриншот структуры регистра сведений.
  - Листинг кода 1С (обработка/форма) с комментариями.
  - Скриншот выполненной загрузки (заполненный регистр).
- **Результаты выполнения** (таблица с загруженными данными).
- **Анализ результатов** (какие ошибки возникли, как решались, время выполнения запроса).
- **Ответы на контрольные вопросы**.

– **Вывод** (освоенные навыки интеграции, работы с JSON, регистрами сведений).

Контрольные вопросы

1. Какие объекты 1С используются для выполнения HTTP-запросов?
2. В чём разница между HTTPСоединение и HTTPЗапрос?
3. Как задать таймаут соединения и зачем это нужно?
4. Как преобразовать строку JSON в структуры 1С?
5. Что произойдёт, если сервер вернёт ответ не в кодировке UTF-8?
6. Как проверить код состояния HTTP-ответа?
7. Как передать параметры в URL GET-запроса (например, дату)?
8. Для чего нужен регистр сведений периодического типа?
9. Как записать несколько записей в регистр сведений в одном цикле?
10. Какие исключения могут возникнуть при работе с HTTP в 1С и как их обработать?

Критерии оценивания

Оценка	Критерии
Отлично	Работа выполнена полностью без ошибок. Эмулятор API реализован корректно, 1С-код выполняет GET-запрос, парсит JSON и записывает данные в регистр. Обработка ошибок присутствует (попытка, проверка кода состояния). Реализована вариативная часть (фоновое задание или журнал загрузок). Отчёт полный, скриншоты, листинги, анализ. На защите — уверенные ответы на все вопросы.
Хорошо	Работа выполнена полностью. Данные загружаются, но есть замечания: например, отсутствует обработка таймаута, или парсинг JSON без проверки структуры, или нет обработки ошибки при недоступности сервера. Вариативная часть не выполнена. Отчёт неполный (нет одного скриншота). На защите — ответы на большинство вопросов.

Удовлетворительно	Работа выполнена частично (например, эмулятор есть, но 1С не записывает данные или запрос не работает). Код содержит критические ошибки. Отсутствует обработка ошибок. Отчёт урезан. Студент затрудняется ответить на вопросы.
Неудовлетворительно	Работа не представлена. Эмулятор не запускается или возвращает неверный формат. 1С не может выполнить запрос. Данные в регистр не записываются. Отчёт не соответствует требованиям.

Лабораторная работа №5.
Публикация HTTP-сервиса в 1С и интеграция с очередью сообщений

Цель работы: Освоить публикацию HTTP-сервиса на платформе 1С, научиться принимать HTTP-запросы, обрабатывать их, взаимодействовать с очередью сообщений RabbitMQ (отправка/получение) с использованием Python в роли producer/consumer.

Задачи:

- Развернуть RabbitMQ в Docker-контейнере (с включённым management-плагином).
- Создать и опубликовать HTTP-сервис в 1С с методом POST /GetRemains, принимающим код номенклатуры и возвращающим остаток в формате JSON.
- Разработать Python-скрипт producer, который:
  - Читает список номенклатур (из файла или встроенного списка).
  - Для каждой отправляет POST-запрос к HTTP-сервису 1С.
  - Полученный результат публикует в очередь RabbitMQ (remains_queue).
- Разработать Python-скрипт consumer, который читает сообщения из очереди и логирует их в консоль или файл.
- Оформить спецификацию API в формате OpenAPI (YAML/JSON или Markdown).

Теоретическая часть

Платформа 1С может выступать в роли HTTP-сервера: создаётся объект конфигурации «HTTP-сервис», определяются шаблоны URL и методы (GET, POST), а в модуле-обработчике доступны входящие данные через ПолучитьТелоКакСтроку() и параметры запроса. Для интеграции с брокером сообщений RabbitMQ, который не имеет нативного клиента в 1С,

используется Python в качестве адаптера. Сначала разворачивается RabbitMQ в Docker-контейнере с management-плагином (образ rabbitmq:3-management, порты 5672 для AMQP и 15672 для веб-интерфейса, учётные данные user/pass123). В конфигураторе 1С создаётся HTTP-сервис, например WarehouseService, с методом GetRemains (шаблон URL /WarehouseService/GetRemains, метод POST, тип ответа application/json). В модуле HTTP-сервиса обработчик получает тело запроса, парсит JSON, извлекает код номенклатуры. Через запрос к регистру остатков вычисляется доступное количество, формируется структура ответа (КодНоменклатуры, Наименование, Остаток) и возвращается в виде JSON с кодом состояния 200 или 404 при отсутствии номенклатуры. Далее разрабатывается Python-скрипт producer, который подключается к RabbitMQ через библиотеку pika, объявляет очередь remains_queue (durable=True для сохранения сообщений при перезапуске). Producer формирует список запросов (например, список кодов номенклатуры) и для каждого выполняет POST-запрос к HTTP-сервису 1С с помощью библиотеки requests. Полученный ответ (статус, тело) упаковывается в JSON-сообщение и публикуется в очередь через channel.basic_publish с routing key, равным имени очереди. Producer обрабатывает возможные ошибки подключения к 1С (таймаут, отказ сервера) и записывает их в сообщение. Python-скрипт consumer подключается к той же очереди, устанавливает basic_qos(prefetch_count=1) для справедливого распределения, определяет функцию callback, которая декодирует сообщение, выводит его в консоль (или сохраняет в файл) и отправляет basic_ack для подтверждения обработки. Consumer запускается в бесконечном цикле start_consuming(). Такая архитектура демонстрирует паттерн асинхронного обмена, где 1С остаётся синхронным поставщиком данных, а очередь сообщений обеспечивает надёжную доставку и развязывает отправителя и получателя. Спецификация OpenAPI описывает эндпоинт /GetRemains с указанием формата запроса (например, {"Код": "ТОВ-001"}) и ответа ({"КодНоменклатуры": "...", "Наименование": "...", "Остаток":

150.5}). Весь сценарий проверяется запуском всех трёх компонентов: RabbitMQ, HTTP-сервис 1С (опубликованный на веб-сервере или через встроенный веб-сервер с портом 8080), producer и consumer — при этом сообщения о остатках успешно пересылаются из 1С в очередь и логируются..

Методика выполнения работы

1. Развёртывание RabbitMQ в Docker

docker-compose.yml:

```
version: "3.8"
services:
  rabbitmq:
    image: rabbitmq:3-management
    container_name: rabbitmq_1c
    ports:
      - "5672:5672"   # AMQP
      - "15672:15672" # Management UI
    environment:
      RABBITMQ_DEFAULT_USER: user
      RABBITMQ_DEFAULT_PASS: pass123
```

Запуск: docker-compose up -d

Web-интерфейс: http://localhost:15672 (user/pass123)

2. Создание HTTP-сервиса в 1С

Шаг 1: В конфигураторе добавить HTTP-сервис:

- Имя: WarehouseService
- Шаблон URL: /WarehouseService/{method}

Шаг 2: Добавить метод GetRemains:

- Шаблон URL метода: GetRemains
- Метод HTTP: POST
- Тип ответа: application/json

Шаг 3: В модуле HTTP-сервиса написать обработчик:

```
Функция ОбработкаЗапроса(Запрос)
```

```
// Получение тела запроса (JSON)
Тело = Запрос.ПолучитьТелоКакСтроку();
Если ПустаяСтрока(Тело) Тогда
    Возврат СоздатьОтвет("Ошибка: пустой запрос", 400);
КонецЕсли;

// Парсинг JSON
Попытка
    Чтение = Новый ЧтениеJSON(Тело);
    Данные = ПрочитатьJSON(Чтение);
Исключение
    Возврат СоздатьОтвет("Ошибка парсинга JSON: " + ОписаниеОшибки(), 400);
КонецПопытки;

КодНоменклатуры = Данные.Код; // ожидаем {"Код": "ТОВ-001"}

// Поиск номенклатуры по коду
Номенклатура = Справочники.Номенклатура.НайтиПоКоду(КодНоменклатуры);
Если Номенклатура = Справочники.Номенклатура.ПустаяСсылка() Тогда
    Возврат СоздатьОтвет("Номенклатура не найдена", 404);
КонецЕсли;

// Запрос остатка из регистра
ЗапросОстатка = Новый Запрос;
ЗапросОстатка.Текст =
    "ВЫБРАТЬ
    |    ОстаткиТоваров.КоличествоОстаток КАК Остаток
    |ИЗ
    |                РегистрНакопления.ОстаткиТоваров.Остатки(,    Номенклатура    =
&Номенклатура) КАК ОстаткиТоваров";
ЗапросОстатка.УстановитьПараметр("Номенклатура", Номенклатура);
Результат = ЗапросОстатка.Выполнить();

Остаток = 0;
Если Не Результат.Пустой() Тогда
```

```
        Выборка = Результат.Выбрать();
        Выборка.Следующий();
        Остаток = Выборка.Остаток;
    КонецЕсли;

    // Формирование ответа
    СтруктураОтвета = Новый Структура;
    СтруктураОтвета.Вставить("КодНоменклатуры", КодНоменклатуры);
    СтруктураОтвета.Вставить("Наименование", Номенклатура.Наименование);
    СтруктураОтвета.Вставить("Остаток", Остаток);

    JSON = ЗаписьJSON(СтруктураОтвета);
    Возврат СоздатьОтвет(JSON, 200, "application/json");

КонецФункции

Функция СоздатьОтвет(ТелоСтрока, КодСостояния, ТипСодержимого = "text/plain")
    Ответ = Новый HTTPОтвет(КодСостояния);
    Ответ.Заголовки.Вставить("Content-Type", ТипСодержимого);
    Ответ.УстановитьТелоИзСтроки(ТелоСтрока);
    Возврат Ответ;
КонецФункции
```

Шаг 4: Опубликовать HTTP-сервис (через веб-клиент 1С или через публикацию на веб-сервере). Для отладки можно использовать встроенный веб-сервер 1С (запуск сервера с параметром --http-port=8080).

3. Python Producer (отправляет запросы в 1С и публикует в очередь)

```
import pika
import requests
import json

# Настройки RabbitMQ
RABBIT_HOST = 'localhost'
RABBIT_QUEUE = 'remains_queue'
RABBIT_USER = 'user'
```

```
RABBIT_PASS = 'pass123'

# Настройки 1С HTTP-сервиса
ONEC_URL = 'http://localhost:8080/WarehouseService/GetRemains'

# Список номенклатур для проверки
items = [
    {"Код": "ТОВ-001"},
    {"Код": "ТОВ-002"},
    {"Код": "ТОВ-003"},
    {"Код": "НЕ-СУЩ-001"}  # несуществующий
]

# Подключение к RabbitMQ
credentials = pika.PlainCredentials(RABBIT_USER, RABBIT_PASS)
connection = pika.BlockingConnection(pika.ConnectionParameters(host=RABBIT_HOST, credentials=credentials))
channel = connection.channel()
channel.queue_declare(queue=RABBIT_QUEUE, durable=True)

for item in items:
    # Вызов 1С
    try:
        response = requests.post(ONEC_URL, json=item, timeout=10)
        result = {
            "request": item,
            "status_code": response.status_code,
            "response": response.json() if response.status_code == 200 else response.text
        }
    except Exception as e:
        result = {
            "request": item,
            "error": str(e)
        }
```

```
        # Публикация в очередь
        message = json.dumps(result, ensure_ascii=False)
        channel.basic_publish(
            exchange='',
            routing_key=RABBIT_QUEUE,
            body=message.encode('utf-8'),
            properties=pika.BasicProperties(delivery_mode=2)  # persistent
        )
        print(f"Sent: {message}")

    connection.close()
```

4. Python Consumer (читает из очереди и логирует)

```
import pika
import json

RABBIT_HOST = 'localhost'
RABBIT_QUEUE = 'remains_queue'
RABBIT_USER = 'user'
RABBIT_PASS = 'pass123'

def callback(ch, method, properties, body):
    message = body.decode('utf-8')
    data = json.loads(message)
    print(f"[Consumer] Received: {json.dumps(data, indent=2, ensure_ascii=False)}")
    ch.basic_ack(delivery_tag=method.delivery_tag)

credentials = pika.PlainCredentials(RABBIT_USER, RABBIT_PASS)
connection = pika.BlockingConnection(pika.ConnectionParameters(host=RABBIT_HOST, credentials=credentials))
channel = connection.channel()
channel.queue_declare(queue=RABBIT_QUEUE, durable=True)
channel.basic_qos(prefetch_count=1)
channel.basic_consume(queue=RABBIT_QUEUE, on_message_callback=callback)
```

```
print("Waiting for messages...")
channel.start_consuming()
```

5. Спецификация OpenAPI (YAML)

```
openapi: 3.0.0
info:
  title: Warehouse API
  version: 1.0.0
  description: HTTP-сервис 1С для получения остатков товаров
servers:
  - url: http://localhost:8080/WarehouseService
paths:
  /GetRemains:
    post:
      summary: Получить остаток товара
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                Код:
                  type: string
                  example: "ТОВ-001"
              required: [Код]
      responses:
        '200':
          description: Успешный ответ
          content:
            application/json:
              schema:
                type: object
                properties:
                  КодНоменклатуры:
```

```
        type: string
      Наименование:
        type: string
      Остаток:
        type: number
        example: 150.5
  '400':
    description: Ошибка запроса (неверный JSON)
  '404':
    description: Номенклатура не найдена
```

Индивидуальные задания

1. Добавить третий сервис – «Сервис инвентаризации» (резервирование товара). Реализовать Saga из трёх шагов с компенсацией: заказ → оплата → резерв.

Вариант	**HTTP-сервис 1С (метод, действие)**	**Очередь RabbitMQ (назначение)**	**Python producer (источник данных)**	**Особенность**
1	POST /GetPrice – цена товара по коду	price_queue	список товаров из CSV	Цена из регистра сведений
2	POST /CheckStock – доступный остаток на складе (с учётом резерва)	stock_queue	JSON-файл с запросами	Использовать два склада
3	POST /CreateOrder – создание заказа клиента (возвращает номер)	order_queue	список заказов из внешней системы (имитация)	Запись документа в 1С
4	POST /CalculateCost – расчёт себестоимости (партия товара)	cost_queue	данные о партиях из Pandas DataFrame	Вызов сложного алгоритма
5	POST /GetContractorBalance – сальдо по контрагенту	balance_queue	список ИНН из текстового файла	Взаимодействие с регистром бухгалтерии

6	POST /ValidateProduct – проверка сертификатов (дата годности)	validate_queue	XML-файл с перечнем	Обработка дат, сравнение с текущей
7	POST /ConvertUnits – пересчёт в базовую единицу (кг, шт, л)	convert_queue	список товаров с разными единицами	Требуется справочник коэффициентов
8	POST /GetExchangeRate – курс валюты на дату	currency_queue	список валют (USD, EUR, CNY)	Получение из регистра сведений
9	POST /CheckDiscount – проверка права на скидку (по карте клиента)	discount_queue	номера дисконтных карт	Сложная логика (условия по сумме покупок)
10	POST /ReserveGoods – резервирование товаров (временное)	reserve_queue	список товаров и количеств	Проведение документа резерва
11	POST /GetDocumentsByPeriod – список документов за период	docs_queue	параметры периодов (начало, конец)	Возвращать массив JSON-объектов
12	POST /GetWarehouseCapacity – загрузка склада (м³/тонны)	capacity_queue	артикулы товаров	Суммировать объём/вес из справочника
13	POST /CheckLogin – аутентификация пользователя 1С	auth_queue	логин/пароль (хеш)	Проверка через ПользователиИнформационнойБазы
14	POST /GenerateReport – формирование отчёта (PDF в base64)	report_queue	параметры отчёта	Возвращать строку base64, consumer сохраняет в файл
15	POST /GetNomenclatureTree – иерархия групп товаров	nomenclature_queue	корневая группа	Рекурсивный обход справочника

Требования к отчёту

Отчёт должен быть оформлен в формате PDF и содержать:

1. **Титульный лист**.
2. **Цель и задачи работы**.
3. **Краткое теоретическое обоснование** (HTTP-сервисы 1С, RabbitMQ, AMQP, OpenAPI).
4. **Практическая часть**:
    - Docker-команды / docker-compose.yml для RabbitMQ.
    - Листинг модуля HTTP-сервиса 1С.
    - Листинг Python producer и consumer.
    - Скриншоты: работающий HTTP-сервис (тест в Postman), producer в действии, consumer вывод, очередь в Management UI.
5. **Результаты выполнения** (логи consumer, примеры сообщений).
6. **OpenAPI-спецификация** (YAML-файл, вставленный в отчёт).
7. **Анализ результатов** (сравнение прямого вызова 1С и асинхронной очереди, замеры времени).
8. **Ответы на контрольные вопросы**.
9. **Вывод** (освоены навыки публикации HTTP-сервиса, асинхронного обмена через очередь, документирования API).

Контрольные вопросы

1. Каким образом в 1С создать HTTP-сервис и опубликовать его?
2. Как в обработчике HTTP-сервиса получить тело POST-запроса?
3. Как задать параметр URL в шаблоне, например, `/GetRemains/{id}`?
4. Какие коды состояния HTTP являются успешными, а какие – ошибочными?
5. Что такое RabbitMQ, exchange, queue, routing key?
6. Зачем нужен флаг `durable=True` при объявлении очереди?

7. Как гарантировать, что сообщение в RabbitMQ не будет потеряно при падении consumer?
8. В чём отличие между `basic_ack` и `basic_nack`?
9. Как протестировать HTTP-сервис 1С без Python (например, через Postman)?
10. Для чего нужна спецификация OpenAPI и как её сгенерировать?

Критерии оценивания

Оценка	Критерии
Отлично	Работа выполнена полностью без ошибок. HTTP-сервис 1С реализован, корректно обрабатывает POST-запросы, возвращает JSON. Producer отправляет запросы, публикует в очередь. Consumer читает и логирует. OpenAPI-спецификация корректна. Реализована вариативная часть (запись в PostgreSQL или retry-механизм). Отчёт полный, все скриншоты, листинги. На защите — уверенные ответы на все вопросы.
Хорошо	Работа выполнена полностью. Основной функционал работает. Есть замечания: например, producer не обрабатывает ошибки HTTP, consumer не подтверждает сообщения (ack), OpenAPI неполный. Вариативная часть не выполнена. Отчёт неполный (нет скриншота очереди). На защите — ответы на большинство вопросов.
Удовлетворительно	Работа выполнена частично (например, HTTP-сервис есть, но producer не публикует в очередь, или consumer не запускается). Код содержит логические ошибки. OpenAPI отсутствует. Отчёт урезан. Студент затрудняется с ответами.
Неудовлетворительно	Работа не представлена. HTTP-сервис 1С не отвечает или возвращает неверный формат. Очередь не создаётся. Producer/consumer не работают. Отчёт не соответствует требованиям.

Лабораторная работа №6.
Транзакции, блокировки и отказоустойчивость в 1С

Цель работы: Исследовать механизмы транзакционности, блокировок и отказоустойчивости в платформе 1С, научиться выявлять и предотвращать взаимоблокировки, реализовывать управляемые блокировки и компенсирующие обработки (Saga).

Задачи:

- Создать два сеанса 1С, одновременно пытающихся списать один и тот же товар (документ «Расход товаров»).
- Проанализировать блокировки через журнал регистрации 1С (установить уровень детализации).
- Реализовать управляемую блокировку через объект БлокировкаДанных.Заблокировать() для предотвращения конфликта.
- Сравнить поведение транзакций 1С (по умолчанию – автоматический повтор с эскалацией) с уровнем SERIALIZABLE в PostgreSQL (через внешнее подключение или анализ).
- Написать компенсирующую обработку – аналог Saga-компенсации для случая сбоя при списании (например, если вторая транзакция не удалась, откатить первую).

Теоретическая часть

Платформа 1С автоматически выполняет операции записи данных (проведение документов, запись справочников) в неявных транзакциях, а для явного управления предоставляются методы НачатьТранзакцию(), ЗафиксироватьТранзакцию() и ОтменитьТранзакцию(). По умолчанию используется уровень изоляции READ COMMITTED, что при параллельных изменениях одной строки регистра может привести к некорректным остаткам, если не применить явные блокировки. Для исследования поведения транзакций создаются два сеанса 1С, одновременно

пытающихся списать один и тот же товар (например, документ «Расход товаров» с количеством 5 при начальном остатке 10). В первом сеансе документ проводится, во втором — либо ждёт, либо получает ошибку отрицательного остатка в зависимости от наличия проверки в коде проведения. Чтобы проанализировать конфликты, настраивается журнал регистрации: в свойствах конфигурации добавляются события Блокировки и ТранзакцияБлокировки с уровнем детализации «Информация». После запуска параллельных сеансов в журнале видны записи о блокировках, включая идентификаторы сеансов и ожидаемые ресурсы. Для предотвращения конфликтов реализуется управляемая блокировка через объект БлокировкаДанных. В обработке проведения документа перед формированием движений создаётся блокировка, для каждой номенклатуры из табличной части добавляется элемент блокировки на регистр «Остатки товаров» с исключительным режимом, затем вызывается Блокировка.Заблокировать(). Если блокировка успешна, выполняется проверка остатка и формирование движений, после чего блокировка снимается. В случае ошибки блокировки устанавливается Отказ = Истина. Для сравнения с PostgreSQL создаётся таблица остатки_товаров с единственной строкой, и два параллельных сеанса выполняют UPDATE ... SET количество = количество - 5 в транзакциях с уровнем изоляции SERIALIZABLE. В PostgreSQL второй коммит завершится ошибкой сериализации, тогда как 1С при READ COMMITTED допустит оба изменения, что может привести к отрицательному остатку — это демонстрирует необходимость ручного управления блокировками в 1С для бизнес-критичных операций. Для отказоустойчивости применяется паттерн Saga: если после успешного списания товара последующая операция (например, резервирование под заказ) завершается ошибкой, запускается компенсирующее действие — создаётся документ сторно или корректирующая запись, которая возвращает остаток в исходное состояние. Компенсация реализуется в блоке Исключение после отмены транзакции: создаётся новый

документ (например, «Сторно расхода»), заполняется ссылка на исходный документ, и он проводится, восстанавливая остаток. Такой подход имитирует распределённую транзакцию на уровне прикладной логики.

Методика выполнения работы

4.1. Подготовка тестовых данных

- Справочник «Номенклатура»: один товар «Стол» с начальным остатком = 10.
- Регистр накопления «Остатки товаров» (остаточный, измерение – Номенклатура, ресурс – Количество).
- Документ «Расход товаров» (табличная часть: Номенклатура, Количество) с проведением, уменьшающим остаток.

2. Сценарий одновременного списания (без управляемой блокировки)

В **первом сеансе** (запущенном из под одного пользователя) и **втором сеансе** (второй экземпляр 1С) выполнить одинаковый код:

```
// В обоих сеансах одновременно (вручную нажать кнопку)
Док = Документы.РасходТоваров.СоздатьДокумент();
Док.Дата = ТекущаяДата();
Док.Товары.Добавить();
Док.Товары[0].Номенклатура =
Справочники.Номенклатура.НайтиПоНаименованию("Стол");
Док.Товары[0].Количество = 5;   // каждый пытается списать 5, итого 10
Попытка
   Док.Записать(РежимЗаписиДокумента.Проведение);
   Сообщить("Успешно проведён");
Исключение
   Сообщить("Ошибка: " + ОписаниеОшибки());
КонецПопытки;
```

Ожидаемый результат: первый сеанс проводит документ (остаток = 5), второй сеанс либо ждёт, либо получает ошибку, что остаток не может

стать отрицательным (если в движении есть проверка Если Остаток < КоличествоТогда Отказ = Истина). Журнал регистрации покажет блокировку на строке регистра.

3. Реализация управляемой блокировки

В модуль документа «Расход товаров» добавим явную блокировку перед списанием:

```
Процедура ОбработкаПроведения(Отказ, Режим)
   // Блокируем строку регистра по конкретной номенклатуре
   Блокировка = Новый БлокировкаДанных;
   Для Каждого СтрокаТЧ Из Товары Цикл
      ЭлементБлокировки = Блокировка.Добавить("РегистрНакопления.ОстаткиТоваров");
      ЭлементБлокировки.УстановитьЗначение("Номенклатура", СтрокаТЧ.Номенклатура);
      ЭлементБлокировки.Режим = РежимБлокировкиДанных.Исключительный;
   КонецЦикла;

   Попытка
      Блокировка.Заблокировать();
   Исключение
      Сообщить("Не удалось заблокировать данные: " + ОписаниеОшибки());
      Отказ = Истина;
      Возврат;
   КонецПопытки;

   // Теперь безопасно формируем движения
   Движения.ОстаткиТоваров.Записать = Истина;
   Для Каждого СтрокаТЧ Из Товары Цикл
      ТекущийОстаток = ПолучитьОстаток(СтрокаТЧ.Номенклатура);
      Если ТекущийОстаток < СтрокаТЧ.Количество Тогда
         Сообщить("Недостаточно остатка");
         Отказ = Истина;
         Блокировка.Разблокировать();
         Возврат;
```

```
        КонецЕсли;
        Движение = Движения.ОстаткиТоваров.Добавить();
        Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
        Движение.Номенклатура = СтрокаТЧ.Номенклатура;
        Движение.Количество = СтрокаТЧ.Количество;
    КонецЦикла;

    Блокировка.Разблокировать();
КонецПроцедуры
```

4. Анализ через журнал регистрации

1. В конфигурации открыть свойства корня → «Журнал регистрации» → добавить события ТранзакцияБлокировки, Блокировки.
2. Запустить два сеанса.
3. После конфликта открыть журнал регистрации (раздел «Администрирование» → «Журнал регистрации»).
4. Найти записи с видом события «Блокировка» (код, данные, ожидание). Проанализировать: какой сеанс кого блокировал.

5. Сравнение с SERIALIZABLE в PostgreSQL

Создать в PostgreSQL таблицу-аналог регистра 1С:

```
CREATE TABLE остатки_товаров (
    номенклатура VARCHAR(50),
    количество NUMERIC(15,3),
    PRIMARY KEY (номенклатура)
);
INSERT INTO остатки_товаров VALUES ('Стол', 10);
```

Запустить два параллельных сеанса psql с уровнем изоляции SERIALIZABLE:

```
sql
-- Сеанс 1
BEGIN ISOLATION LEVEL SERIALIZABLE;
UPDATE остатки_товаров SET количество = количество - 5 WHERE номенклатура =
'Стол';
COMMIT;
```

```
-- Сеанс 2 (запустить почти одновременно)
BEGIN ISOLATION LEVEL SERIALIZABLE;
UPDATE остатки_товаров SET количество = количество - 5 WHERE номенклатура =
'Стол';
COMMIT;
```

Второй COMMIT вызовет ошибку ERROR: could not serialize access due to read/write dependencies. В 1С при аналогичном сценарии (без управляемой блокировки) второй сеанс будет ждать, пока первый не зафиксируется, и затем выполнится, что может привести к отрицательному остатку. Вывод: 1С использует более слабый уровень изоляции (READ COMMITTED), что требует ручной блокировки для бизнес-корректности.

6. Компенсирующая обработка (Saga)

Если у нас есть две операции: сначала списание, затем резервирование под заказ. При сбое резервирования нужно отменить списание.

Пример:

```
Процедура ВыполнитьКомплектОпераций()
   НачатьТранзакцию();
   Попытка
      // Шаг 1: списание
      ДокументРасхода.Провести();
      // Шаг 2: резервирование
      ДокументРезерва.Провести();
      ЗафиксироватьТранзакцию();
   Исключение
      ОтменитьТранзакцию();
      // Компенсация: если списание уже провелось, создаём сторно
      Попытка
         ДокументСторно = Документы.СторноРасхода.СоздатьДокумент();
         ДокументСторно.ИсходныйДокумент = ДокументРасхода.Ссылка;
         ДокументСторно.Провести();
      Исключение
```

```
            Сообщить("Критическая ошибка компенсации: " + ОписаниеОшибки());
            // Здесь можно отправить уведомление администратору
        КонецПопытки;
    КонецПопытки;
КонецПроцедуры
```

Индивидуальные задания

1. Задания для индивидуального выполнения (15 вариантов). Каждый вариант исследует **определённый тип конфликта** и реализует **соответствующий механизм управления**.

Общее для всех вариантов:

- Провести эксперимент с двумя параллельными сеансами.
- Включить и проанализировать журнал регистрации.
- Реализовать управляемую блокировку через БлокировкаДанных.
- Написать компенсирующую обработку (Saga) для своего варианта.
- Сравнить поведение 1С с уровнем SERIALIZABLE в PostgreSQL (на тестовой таблице).

Вариативная часть (оценка «Отлично»):

- Реализовать динамическое определение нужного уровня изоляции через УстановитьУровеньИзоляцииТранзакций() (если доступно в версии платформы) или эмуляцию через явные блокировки.
- Добавить мониторинг блокировок в Prometheus/Grafana (через экспорт журнала регистрации).

Вариант	Тип транзакционного конфликта	Объект блокировки	Компенсационное действие	Дополнительное исследование
1	Два документа списания одного товара	Регистр остатков по номенклатуре	Создание документа сторно	Сравнить с `REPEATABLE READ` в PostgreSQL
2	Проведение документа	Регистр остатков +	Корректировка партии	Эскалация блокировок (как происходит)

	прихода и расхода одновременно	регистр партий	(добавить корректирующую запись)	
3	Изменение справочника номенклатуры во время проведения документов	Справочник (запись)	Отказ от изменения справочника (логирование)	Замер времени блокировки
4	Документ перемещения между складами (две записи регистра)	Два измерения (склад-отправитель, склад-получатель)	Обратное перемещение	Взаимоблокировка (deadlock) – смоделировать
5	Пакетная обработка документов (цикл)	Все затронутые товары	Частичный откат – отмена только неудачных документов	Уровень изоляции в 1С и SET TRANSACTION
6	Регистр сведений (цены) – одновременное обновление курсов	Измерение «Валюта»	Восстановление предыдущего значения из истории	Ручная блокировка через БлокировкаДанных.Заблокировать() с таймаутом
7	Регистр бухгалтерии (проводки) – списание со счёта	Корреспонденция счетов	Сторно проводки	Сравнение с двухфазной фиксацией (2PC)
8	Регистр расчётов (зарплата) – начисление и удержание	Измерение «Сотрудник»	Пересчёт удержания	Использование УстановитьБлокировкуДанных() в запросе
9	Документ заказа и резерва – одновременное резервирование	Регистр резерва (два измерения: Номенклатура, Заказ)	Снятие резерва	Анализ deadlock в журнале регистрации

10	Обмен данными через план обмена – одновременная регистрация изменений	Узел плана обмена	Отказ от регистрации (логирование)	Сравнение с оптимистической блокировкой
11	Документ-основание и корректировка (проведение под одним номером)	Блокировка по номеру документа (реестр номеров)	Анулирование номера	Самодельная очередь задач (таблица ожидания)
12	Регистр пересчётов (агрегаты) – обновление итогов	Измерения агрегата	Пересчёт агрегата из движений	Сравнение с механизмом Materialized View в PostgreSQL
13	Периодический регистр сведений «Курсы валют» – вставка за один день	Измерение «Валюта» + период	Удаление ошибочной записи	Использование ПЕРИОД в блокировке
14	Документ «Переоценка товаров» – массовое изменение цен	Группа номенклатуры (иерархия)	Восстановление старых цен из регистра «Цены»	Блокировка всей иерархии
15	Выполнение отчёта с сохранением итогов во временной таблице (конфликт с записью)	Таблица значений на сервере (не СУБД)	Повтор выполнения отчёта	Сравнение с транзакциями в памяти (ИнформацияОбОшибке)

Требования к отчёту

Отчёт должен содержать:

- Титульный лист.

- Цель и задачи работы.
- Краткое теоретическое обоснование (транзакции, уровни изоляции, блокировки, Saga).
- Практическая часть:
- Схема объектов (регистры, документы) для эксперимента.
- Листинг кода двух сеансов без управляемой блокировки.
- Скриншоты ошибок/ожиданий.
- Листинг кода с управляемой блокировкой.
- Скриншоты журнала регистрации с анализом.
- Код сравнения с PostgreSQL (SQL-скрипт и результаты).
- Реализация компенсирующей обработки.
- Результаты выполнения:
- Таблица сравнения поведения (1C vs PostgreSQL).
- Выводы по необходимости ручных блокировок.
- Анализ результатов (какие типы конфликтов выявлены, какой метод блокировки эффективнее).
- Ответы на контрольные вопросы.
- Вывод (освоены навыки управления транзакциями, анализа блокировок, реализации компенсаций).

Контрольные вопросы

1. Что такое транзакция в 1С? Какие операции автоматически попадают в транзакцию?
2. Как явно управлять транзакциями с помощью НачатьТранзакцию?
3. В чём разница между автоматической блокировкой и управляемой через БлокировкаДанных?
4. Как настроить журнал регистрации для отслеживания блокировок?
5. Какой уровень изоляции используется в 1С по умолчанию? Почему он опасен для бизнес-логики остатков?

6. Что произойдёт, если попытаться выполнить две транзакции одновременно без блокировок в 1С?
7. Как в PostgreSQL ведёт себя SERIALIZABLE при двух параллельных обновлениях одной строки? Чем отличается поведение 1С?
8. Что такое взаимоблокировка (deadlock)? Как её обнаружить в 1С?
9. Для чего нужен паттерн Saga и как его применить в 1С?
10. Как реализовать компенсацию, если исходная транзакция уже зафиксирована и откатить её невозможно?

Критерии оценивания

Оценка	Критерии
Отлично	Работа выполнена полностью без ошибок. Проведён воспроизводимый эксперимент с параллельными сеансами. Журнал регистрации проанализирован, сделаны выводы. Реализована управляемая блокировка, предотвращающая конфликт. Компенсирующая обработка работоспособна. Выполнено сравнение с PostgreSQL SERIALIZABLE (скрипты, скриншоты). Реализована вариативная часть (динамический уровень изоляции или интеграция с Prometheus). Отчёт полный, содержит анализ. На защите — уверенные ответы.
Хорошо	Работа выполнена полностью. Эксперимент проведён, блокировка реализована. Компенсация работает. Сравнение с PostgreSQL есть, но неполное (только вывод, без скриптов). Журнал регистрации проанализирован поверхностно. Вариативная часть не выполнена. Отчёт имеет мелкие недочёты. На защите — ответы на большинство вопросов.
Удовлетворительно	Работа выполнена частично: например, эксперимент показал конфликт, но управляемая блокировка не работает или компенсация отсутствует. Сравнение с PostgreSQL не проводилось. Журнал регистрации не анализировался. Отчёт урезан. Студент затрудняется с ответами.

Неудовлетворительно	Работа не представлена. Эксперимент не проведён или код не выполняет списание. Блокировки не реализованы. Компенсация отсутствует. Отчёт не соответствует требованиям.

Лабораторная работа №7.
Комплексный проект: ERP-расширение с интеграцией и мониторингом

Цель работы: Разработать комплексное ERP-расширение, объединяющее все изученные механизмы: объектную модель, язык запросов, управляемые формы, интеграцию через HTTP-сервисы и потребление REST API, транзакционную корректность и базовый мониторинг. Студент должен продемонстрировать способность проектировать и реализовывать промышленное расширение на платформе 1С.

Задачи:

- Спроектировать метаданные мини-ERP:
  - Справочники: «Контрагенты», «Номенклатура» (иерархический).
  - Документы: «Заказ покупателя», «Отгрузка товаров».
  - Регистры накопления: «Остатки товаров» (остатки), «Заказы к отгрузке» (обороты резерва).
- Реализовать проведение документов с соблюдением транзакционной целостности (явные блокировки при резервировании).
- Создать управляемые формы списка и формы документов с командным интерфейсом (провести, отменить, скопировать).
- Публиковать HTTP-сервис GetOrderStatus (по номеру заказа возвращает статус и состав).
- Реализовать интеграцию с внешним эмулятором учётной системы (Python FastAPI) для синхронизации контрагентов (выгрузка новых контрагентов из 1С во внешнюю систему).
- Настроить журнал регистрации: логировать проведение документов, ошибки интеграции, подозрительные блокировки.
- (Опционально) Настроить экспорт метрик в Prometheus и визуализацию в Grafana (частота проведения документов, количество ошибок).

- Подготовить итоговый отчёт в Markdown, включающий ER-диаграмму регистров, инструкцию по развёртыванию, скриншоты.

Теоретическая часть

Комплексный ERP-проект объединяет все ранее изученные механизмы: объектную модель, язык запросов, управляемые формы, HTTP-сервисы, потребление REST API, транзакции и мониторинг. Спроектируем мини-ERP, включающую справочники «Контрагенты» и «Номенклатура» (иерархический), документы «Заказ покупателя» и «Отгрузка товаров», а также два регистра накопления — «Остатки товаров» (вид Остатки, измерение Номенклатура, ресурс Количество) и «Заказы к отгрузке» (вид Обороты, измерения Номенклатура и Заказ, ресурс КоличествоРезерв). При проведении заказа необходимо соблюсти транзакционную корректность: сначала исключительно блокируются строки регистра остатков по всем номенклатурам заказа через БлокировкаДанных. Затем с помощью запроса вычисляется свободный остаток (фактический остаток минус уже зарезервированный другими заказами). Если его достаточно, в регистр «Заказы к отгрузке» добавляются движения с видом Расход, фиксируя резерв. При отгрузке аналогично блокируются оба регистра, списывается остаток и снимается резерв (движения в «Заказы к отгрузке» с обратным знаком). Для внешнего взаимодействия публикуется HTTP-сервис с методом GET /erp/order/{OrderNumber}, который по номеру документа возвращает JSON с полями статус (Проведён/Не проведён/Отгружен) и список позиций заказа. В обработчике сервиса выполняется запрос к документу и его табличной части, ответ формируется с помощью ЗаписьJSON. Интеграционный сценарий включает синхронизацию контрагентов с внешней учётной системой. На Python разрабатывается эмулятор на FastAPI с эндпоинтом POST /sync/contractors, принимающим массив контрагентов в JSON. В 1С создаётся обработка или регламентное задание, которое выбирает контрагентов, изменённых после последней синхронизации (по

реквизиту ДатаИзменения или флагу), преобразует их в JSON через ЗаписьJSON и отправляет методом POST с использованием HTTPСоединение. Ответ эмулятора логируется. Для мониторинга настраивается журнал регистрации: в свойствах конфигурации активируются события Документ.Проведение (уровень Информация), HTTPСервис.Вызов (Примечание), а также пользовательское событие Интеграция.Ошибка, которое вызывается через ЗаписьЖурналаРегистрации при сбоях синхронизации или HTTP-вызовов. В результате получается законченное ERP-расширение с резервированием, внешним API и наблюдаемостью. Для вариативной части (оценка «Отлично») дополнительно настраивается экспорт метрик в Prometheus: HTTP-сервис 1С отдаёт метрики по эндпоинту /metrics в формате, понятном Prometheus (например, счётчик проведённых документов), а в Grafana создаётся дашборд с графиком их количества по часам.

Методика выполнения работы

1. Состав метаданных (предлагаемая структура)

Справочники:

- Контрагенты: реквизиты – ИНН, КПП, Телефон.
- Номенклатура: иерархический, реквизиты – Артикул, ЕдиницаИзмерения.

Документы:

- ЗаказПокупателя: реквизиты – Контрагент, ДатаОтгрузки; табличная часть – Номенклатура, Количество, Цена.
- ОтгрузкаТоваров: реквизиты – ЗаказОснование, Контрагент; табличная часть – Номенклатура, Количество.

Регистры накопления:

- ОстаткиТоваров: вид – Остатки; измерения – Номенклатура; ресурс – Количество.

- ЗаказыКОтгрузке: вид – Обороты; измерения – Номенклатура, Заказ (тип ДокументСсылка.ЗаказПокупателя); ресурс – КоличествоРезерв.

2. Проведение заказа (фрагмент)

```
Процедура ОбработкаПроведения(Отказ, Режим)
   // Блокировка по всем номенклатурам заказа
   Блокировка = Новый БлокировкаДанных;
   Для каждого Строка Из Товары Цикл
      Эл = Блокировка.Добавить("РегистрНакопления.ОстаткиТоваров");
      Эл.УстановитьЗначение("Номенклатура", Строка.Номенклатура);
      Эл.Режим = РежимБлокировкиДанных.Исключительный;
   КонецЦикла;
   Блокировка.Заблокировать();

   // Снятие старого резерва (если перепроводим)
   Движения.ЗаказыКОтгрузке.Записать = Истина;
   Движения.ЗаказыКОтгрузке.Очистить();

   Для каждого Строка Из Товары Цикл
      // Проверка доступности
      Запрос = Новый Запрос;
      Запрос.Текст = "ВЫБРАТЬ Остатки.КоличествоОстаток КАК Остаток ИЗ
РегистрНакопления.ОстаткиТоваров.Остатки(, Номенклатура = &Номенклатура)
КАК Остатки";
      Запрос.УстановитьПараметр("Номенклатура", Строка.Номенклатура);
      Остаток = Запрос.Выполнить()[0].Остаток;

      ЗапросРезерв = Новый Запрос;
      ЗапросРезерв.Текст = "ВЫБРАТЬ СУММА(КоличествоРезерв) КАК Занято ИЗ
РегистрНакопления.ЗаказыКОтгрузке.Обороты(, Номенклатура = &Номенклатура)";
      ЗапросРезерв.УстановитьПараметр("Номенклатура", Строка.Номенклатура);
      Занято = ЗапросРезерв.Выполнить()[0].Занято;

      Если Остаток - Занято < Строка.Количество Тогда
```

```
            Сообщить("Недостаточно товара " + Строка.Номенклатура);
            Отказ = Истина;
            Блокировка.Разблокировать();
            Возврат;
        КонецЕсли;

        Движение = Движения.ЗаказыКОтгрузке.Добавить();
        Движение.Период = Дата;
        Движение.Номенклатура = Строка.Номенклатура;
        Движение.Заказ = Ссылка;
        Движение.КоличествоРезерв = Строка.Количество;
        Движение.ВидДвижения = ВидДвиженияНакопления.Расход; // резервирование
    КонецЦикла;
    Блокировка.Разблокировать();
КонецПроцедуры
```

3. HTTP-сервис GetOrderStatus

Шаблон: /erp/order/{OrderNumber} (GET).

В ответе JSON: статус (Проведён/Не проведён/Отгружен), список позиций.

4. Внешний эмулятор (FastAPI) для синхронизации контрагентов

Эндпоинт POST /sync/contractors принимает массив контрагентов. В ответе – количество принятых.

В 1С – обработка (или регламентное задание), которая выбирает контрагентов, изменённых после последней синхронизации (использовать дату изменения или флаг), формирует JSON и отправляет.

5. Журнал регистрации (настройка)

В свойствах конфигурации:

- Событие Документ.Проведение – уровень Информация, записывать пользователя, номер документа.
- Событие HTTPСервис.Вызов – уровень Примечание.
- Событие Интеграция.Ошибка (пользовательское) – вызывать через ЗаписьЖурналаРегистрации().

Индивидуальные задания

1. Индивидуальные задания (15 вариантов). Каждый вариант – это **тематическое расширение** базовой ERP-модели. Студент должен реализовать все обязательные компоненты, но с учётом специфики варианта.

Обязательная база для всех вариантов:

- Реализовать все метаданные (справочники, документы, регистры) согласно описанной в примере базовой структуре + расширения по варианту.
- Проведение документов с блокировками.
- HTTP-сервис для получения статуса (или аналогичной информации).
- Интеграция (потребление или публикация) с внешним Python-сервисом.
- Настройка журнала регистрации (минимум 3 вида событий).
- Полный отчёт в Markdown (ER-диаграмма, инструкция, скриншоты).

Вариативная часть (оценка «Отлично»):

- Реализовать CI/CD для расширения: сборка .cf или .epf через скрипт (например, через 1c-dt), автотесты на проведение документов (в режиме предприятия).
- Настроить экспорт метрик в Prometheus + дашборд в Grafana (хотя бы один график – количество проведённых документов по часам).

Вариант	Дополнительная предметная область	Особые требования к интеграции	Особые метрики для мониторинга
1	Торговля с партиями и сроками годности	Внешнее API для проверки сертификатов	Остатки с истекающим сроком
2	Интернет-магазин (заказы с сайта)	Приём заказов через REST API (POST /order)	Количество заказов в час
3	Складская логистика (ячейки, адреса)	Внешнее API для расчёта оптимальной ячейки	Заполненность склада (%)
4	Производство (спецификации, списание материалов)	Внешний расчёт потребности в материалах (MRP)	Оборачиваемость материалов
5	Услуги и подписки (абонементы)	Интеграция с биллинговой системой (начисление платежей)	Количество активных подписок

6	Розничная торговля (чеки, фискализация)	Отправка чеков во внешнюю систему ОФД	Количество чеков с ошибками
7	Управление проектами (бюджет, этапы)	Интеграция с системой учёта рабочего времени (Jira-подобная)	Превышение бюджета по проектам
8	Учёт ГСМ (топливные карты, путевые листы)	API для получения остатка литров от поставщика топлива	Расход на 100 км
9	Аренда оборудования (почасовой учёт)	Интеграция с IoT-счётчиками (эмулятор – случайные показания)	Процент использования оборудования
10	Учёт животных (животноводство)	Внешнее API ветеринарного надзора (прививки)	Поголовье по видам
11	Логистика (заказы на перевозку)	Интеграция с картографическим сервисом (расчёт расстояния)	Стоимость доставки на км
12	Бюджетирование (План-факт)	Интеграция с банковским API (выгрузка выписок)	Отклонение плана от факта
13	Управление персоналом (графики, табели)	Интеграция с системой контроля доступа (турникеты)	Количество опозданий
14	Управление качеством (рекламации, брак)	API для создания рекламации во внешней CRM	Процент брака по видам продукции
15	Свободная тема (согласование с преподавателем)	Должна включать 1С → внешнее API и наоборот	Любые метрики, экспортируемые в Prometheus

Требования к отчёту

Отчёт должен быть оформлен в виде PDF и содержать:

- Титульный лист (ФИО, группа, вариант, № работы – 7).
- Аннотация (краткое описание разработанной ERP-расширения).
- Архитектура решения:
  - ER-диаграмма (текстовое описание или ASCII-схема) связей справочников и регистров.
  - Схема потоков данных: как движутся заказы, резервы, отгрузки.
- Описание метаданных:

    - Перечень справочников, документов, регистров с реквизитами (таблицы).
- Листинги ключевых модулей:
    - Обработка проведения заказа.
    - Код HTTP-сервиса.
    - Код интеграции с внешним API.
- Инструкция по развёртыванию (файловый режим):
    - Создание ИБ.
    - Загрузка конфигурации.
    - Запуск эмулятора Python (требуемые библиотеки).
    - Настройка публикации HTTP-сервиса (если нужно).
- Скриншоты:
    - Форма документа «Заказ покупателя».
    - Результат вызова HTTP-сервиса (из Postman или браузера).
    - Запись в регистрах после проведения.
    - Журнал регистрации с заданными событиями.
- Результаты интеграционных тестов (логи эмулятора, сообщения 1С об успехе/ошибке).
- Анализ результатов (сложности, нереализованные идеи, сравнение с промышленными ERP).
- Ответы на контрольные вопросы (по выбору преподавателя, но лучше подготовить все).
- Вывод (какие компетенции сформированы, какие технологии интегрированы).

Контрольные вопросы

1. Почему в ERP-системе необходимо разделять резерв и остатки?
2. Какие объекты метаданных вы использовали для обеспечения транзакционной целостности резервирования?

3. Каким образом вы проверяли отсутствие deadlock-ов при одновременном проведении нескольких заказов?

4. Как вы тестировали HTTP-сервис 1С (какие инструменты использовали)?

5. В какой кодировке передаются данные в вашем интеграционном сценарии? Как боретесь с проблемой кириллицы?

6. Какие события журнала регистрации вы настроили и почему именно их?

7. Какую архитектуру интеграции вы выбрали (синхронную или асинхронную)? Почему?

8. Как реализована обработка ошибок внешнего API (повторы, fallback)?

9. Какие метрики вы бы вывели в Prometheus для мониторинга производительности 1С?

10. Какие шаги CI/CD вы реализовали (если выполняли вариативную часть)?

Критерии оценивания

Оценка	Критерии
Отлично	Разработана полноценная ERP-расширение, удовлетворяющее всем обязательным требованиям (метаданные, проведение с блокировками, HTTP-сервис, интеграция, журнал регистрации). Дополнительная специфика варианта полностью реализована. Код соответствует стандартам (структурирован, обработка ошибок). Отчёт в Markdown содержит все разделы, ER-диаграмму, скриншоты, инструкцию. Выполнена вариативная часть (CI/CD или Prometheus+Grafana). На защите студент даёт развёрнутые ответы, демонстрирует понимание архитектурных решений.
Хорошо	Все обязательные требования выполнены. Дополнительная специфика варианта реализована, но с небольшими недочётами (например, блокировка не на все измерения, HTTP-сервис не обрабатывает ошибки). Вариативная часть не выполнена. Отчёт есть, но неполный (отсутствует ER-диаграмма или скриншоты журнала регистрации). На защите отвечает на большинство вопросов.

Удовлетворительно	Выполнено не менее 70% обязательных требований. Например, HTTP-сервис есть, но интеграция с внешним API не работает; или транзакционная корректность нарушена (отрицательные остатки). Отчёт содержит грубые пропуски. Студент путается в объяснении логики.
Неудовлетворительно	Проект не представлен или неработоспособен (ключевые документы не проводятся, регистры не заполняются). Отсутствует интеграция или HTTP-сервис. Отчёт не соответствует требованиям. На защите не может ответить ни на один вопрос.

СПСИОК ЛИТЕРАТУРЫ

1. Радченко, М. Г. 1С:Предприятие 8.3. Практическое пособие разработчика. — М.: 1С-Паблишинг, 2022. — 964 с. — ISBN 978-5-9677-2956-7.

2. Филиппов, Е. В. Язык запросов 1С:Предприятие 8. Подробное руководство. — СПб.: БХВ-Петербург, 2021. — 448 с. — ISBN 978-5-9775-6678-2.

3. Бойко, В. В., Савинков, В. М. Администрирование 1С:Предприятие 8. Кластеры, серверы, мониторинг. — М.: 1С-Паблишинг, 2023. — 512 с. — ISBN 978-5-9677-3124-9.

4. Краковский, Д. Ю. Управляемые формы в 1С:Предприятие 8. Разработка интерфейсов. — М.: 1С-Паблишинг, 2022. — 352 с. — ISBN 978-5-9677-3001-6.

5. Петлин, А. В. Интеграция 1С с внешними системами: HTTP-сервисы, REST, SOAP. — М.: 1С-Паблишинг, 2021. — 288 с. — ISBN 978-5-9677-2912-5.

6. Моргунов, Е. П. СУБД PostgreSQL для администраторов 1С. — М.: ДМК Пресс, 2022. — 304 с. — ISBN 978-5-9706-0987-2.

7. Бармин, Д. С. Паттерны разработки в 1С: Предприятие 8. От процедур к моделям данных. — М.: 1С-Паблишинг, 2023. — 420 с. — ISBN 978-5-9677-3156-0.

8. Кардашевский, А. Н. Транзакции, блокировки и оптимизация в 1С. — СПб.: БХВ-Петербург, 2022. — 272 с. — ISBN 978-5-9775-6912-3.

9. Виноградов, С. В. Docker и контейнеризация для разработчиков 1С. — М.: 1С-Паблишинг, 2022. — 192 с. — ISBN 978-5-9677-3088-4.

10. Рыжков, П. С. RabbitMQ в микросервисных интеграциях. — М.: ДМК Пресс, 2021. — 336 с. — ISBN 978-5-9706-0876-9.

11. Гришин, А. Ю. Prometheus + Grafana: мониторинг и наблюдение систем. — М.: ДМК Пресс, 2022. — 288 с. — ISBN 978-5-9706-1034-2.

12. Ньюмен, С. Создание микросервисов. Проектирование систем реального времени. — СПб.: Питер, 2022. — 528 с. — ISBN 978-5-4461-1895-3. (Главы по CQRS, Event Sourcing, Saga применимы к архитектуре ERP).