Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Кузбасский государственный технический университет
имени Т. Ф. Горбачева»

Институт профессионального образования

Кафедра информатики и информационных систем

Составитель О. С. Семенова

Сопровождение информационных систем

Методические материалы

Рекомендовано цикловой методической комиссией
Информационных систем и программирования
в качестве электронного издания
для использования в образовательном процессе

Кемерово 2026

Рецензент:
С. А. Асанов, старший преподаватель кафедры ИиАПС

Семенова Ольга Сергеевна
Сопровождение информационных систем : методические материалы для обучающихся специальности СПО 09.02.11 Разработка и управление программным обеспечением / Кузбасский государственный технический университет имени Т. Ф. Горбачева ; кафедра информатики и информационных систем ; составитель О. С. Семенова. – Кемерово : КузГТУ, 2026. – 1 файл (1440 КБ). – Текст : электронный.

Методические материалы по дисциплине «Сопровождение информационных систем» содержат указания для проведения лабораторных работ и для организации самостоятельной работы.

Введение

Целью освоения дисциплины «Сопровождение информационных систем» является формирование компетенций, необходимых для эффективного технического сопровождения информационных систем на всех этапах их жизненного цикла.

Основными задачами изучения дисциплины «Сопровождение информационных систем», являются:

- освоение этапов сопровождения информационных систем (ИС);
- изучение регламентов и норм обновления и технического сопровождения ИС;
- изучение взаимосвязи процессов сопровождения с другими стадиями жизненного цикла (проектирование, внедрение, эксплуатация).
- овладение навыками настройки ИС для пользователей согласно технической документации;
- овладение методами исправления ошибок в программном коде;
- освоение процедуры резервного копирования, восстановления и обновления данных;
- изучение политики безопасности в современных ИС;
- овладение навыками оценки качества и надёжности функционирования ИС;
- освоение методов мониторинга и предотвращения сбоев ИС.
- получение навыков составления технических заданий на сопровождение;
- изучение принципов организации комплексной системы сопровождения;
- освоение методов сбора и анализа данных об ошибках в ИС.
- получение навыков разработки обучающих материалов для пользователей;
- изучение основных документов системы сертификации РФ.

В соответствии с учебным планом изучение дисциплины «Сопровождение информационных систем» предусматривает проведение лекционных и лабораторных занятий.

При подготовке к лабораторным занятиям обучающиеся самостоятельно изучают основную и дополнительную литературу, готовят конспекты по темам, предложенным преподавателем.

На лабораторных занятиях преподаватель осуществляет контроль качества знаний обучающегося, используя опрос, обсуждение вопросов по темам изучаемой дисциплины, письменный опрос при текущем контроле и проверку отчетов по лабораторным работам.

Лабораторная работа № 1.
Настройка логирования и журналирования событий. Анализ и интерпретация логов системы

Цель работы: изучение порядка настройки логирования и журналирования событий и проведения анализа логов информационной системы.

Теоретические положения

Отчёт об ошибке (англ. error report или crash report) – это файл, содержащий техническую информацию об исключительной ситуации (исключении), произошедшей в программе на компьютере пользователя.

Отчеты об ошибках создаются, когда в системе возникают неполадки с аппаратным или программным обеспечением. Отчеты об ошибках содержат следующие разделы: сведения о состоянии компьютера при возникновении ошибки, версия используемой операционной системы и аппаратного обеспечения, а также цифровой код продукта, используемый для определения лицензии. Также передается IP-адрес компьютера, поскольку для отправки отчета необходимо подключиться к сетевой службе. Отчеты об ошибках могут содержать данные файлов журналов, например, имена пользователей, IP-адреса, URL-адреса, имена файлов и пути к ним, а также адреса электронной почты. Отчеты об ошибках отправляются с использованием шифрования в базу данных с ограниченным доступом и не используются в каких-либо коммерческих целях.

Настройка параметров отбора событий

Можно настроить параметры сбора данных диагностики для ведения журнала событий. События можно регистрировать как в журнале событий Windows (рис. 1), так и журнале трассировки (рис. 2).

Можно настроить параметры регулирования событий, чтобы задавать число событий, регистрируемых в каждом журнале в соответствии с критичностью события. Для расширения возможностей управления при регулировании событий можно задать регулирование всех событий или любой отдельной категории событий. Доступно несколько категорий событий в зависимости от служб и возможностей.

Категории событий могут определяться по отдельным службам или по группам связанных событий. К категориям выбранных событий относятся:

- для всех;
- категории, определенные в соответствии с используемым продуктом, например, Office SharePoint Server 2007 или Microsoft Office Project Server 2007;
- административные функции, такие как "Администрирование", "Резервное копирование и восстановление", и др.

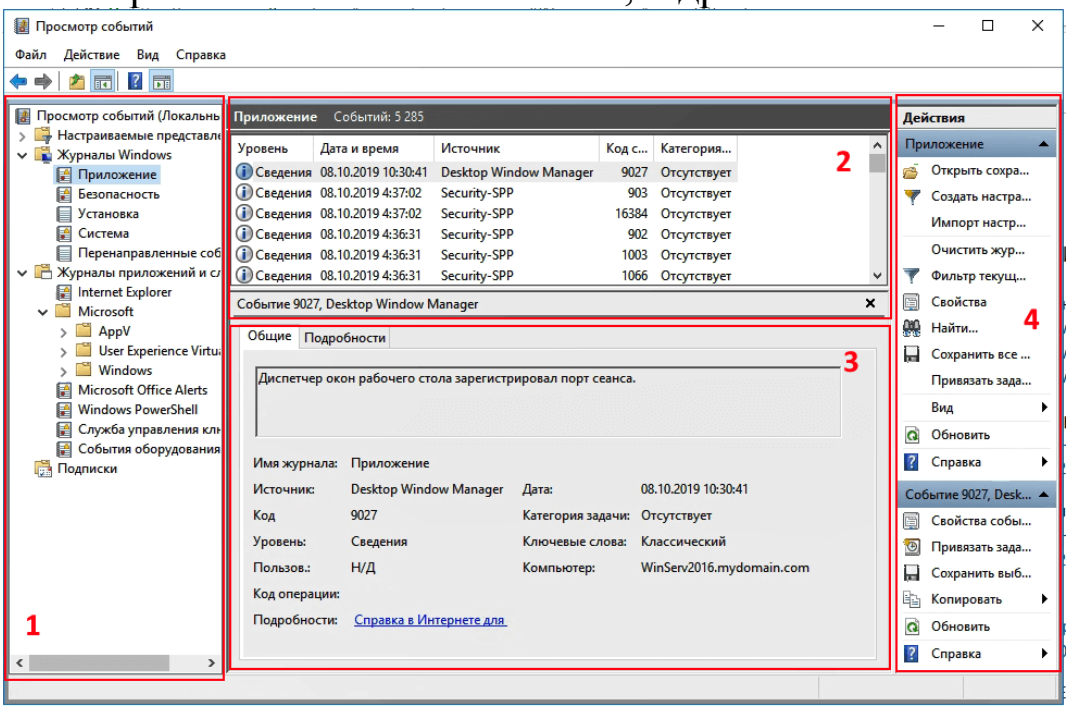

Рисунок 1 – Журнал событий Windows

Для выбранной категории устанавливается минимальный уровень критичности событий, которые следует регистрировать в журнале событий Windows и в журнале трассировки. В каждом журнале будут регистрироваться события этого или более высокого уровня критичности. Записи в этом списке сортируются в порядке от максимального уровня критичности к минимальному.

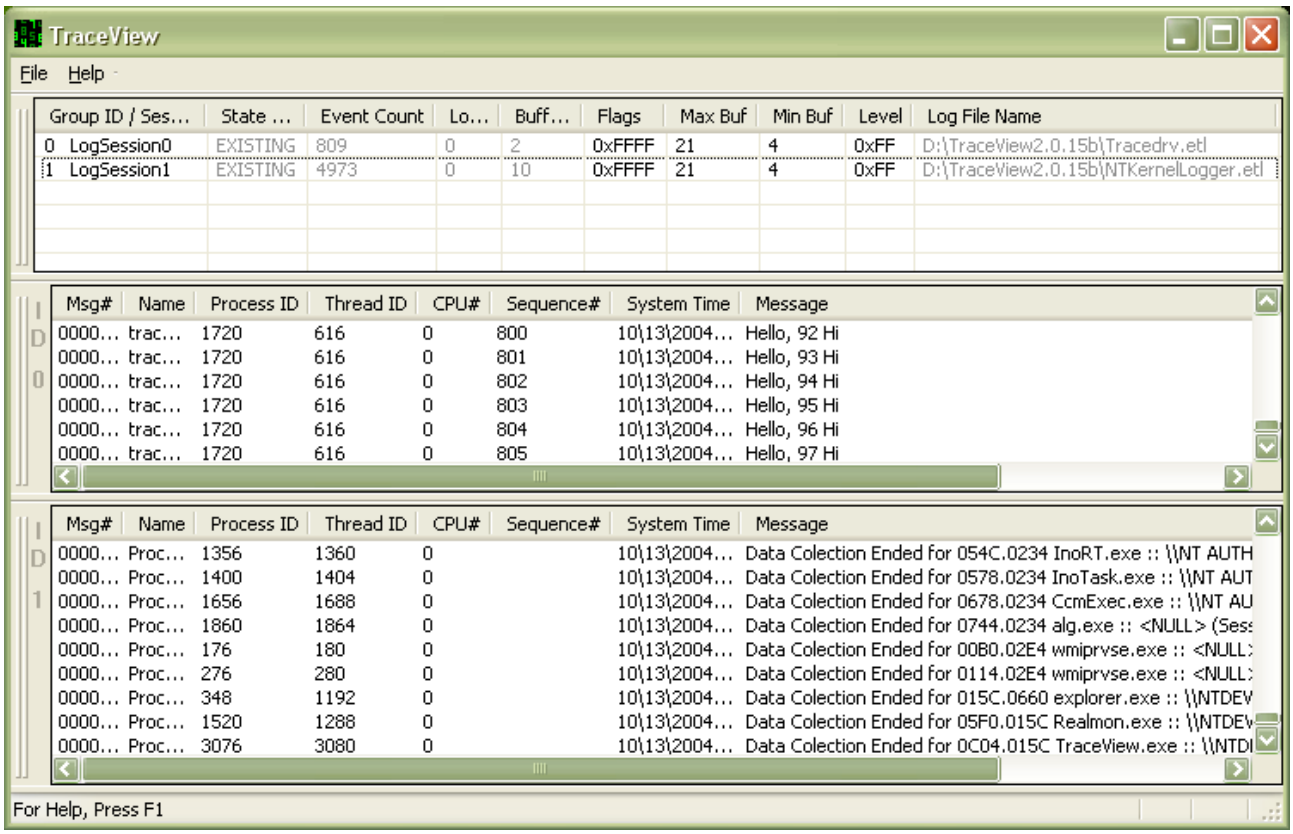

Рисунок 2 – Журнал трассировки Windows

События журнала Windows могут иметь следующие уровни:

- не используется;
- error (ошибка);
- warning (предупреждение);
- не удалось выполнить аудит;
- успешное выполнение аудита;
- information (информация).

События журнала трассировки могут иметь следующие уровни:

- не используется;
- unexpected (непредвиденный);
- monitorable (контролируемый);
- high (высокая);
- medium (средняя);
- verbose (подробный).

Структура отчета об ошибке

ID (Идентификатор, Номер). Каждой записи в системе учета ошибок присваивается уникальный идентификатор или номер. Как правило, он задается самой системой по определенному шаблону. Это может быть просто числовой номер (1,2,3,…3487), а может быть

идентификатор вида ПРОЕКТ-НОМЕР, например, TPO-2367, REG-335 и так далее.

Тип (Трекер). Системы управления задачами, как правило, содержат в себе записи различных типов - задача (Task), улучшение (Feature), баг (Bug), пользовательская история (User Story) и так далее. Для каждого типа может быть свой набор полей и свой жизненный цикл.

Заголовок (Тема, Title). Краткое описание проблемы. Оно отображается в списках, результатах поиска, фильтрах и позволяет быстро понять, в чем суть проблемы. Оно не должно быть слишком коротким и общим, но одновременно и не должно быть слишком длинным. Существует мнемоническое правило для грамотного составления описания «Что? Где? Когда?»: описание должно описывать, что именно сломалось, где сломалось и при каких условиях. Например, «Не работает поиск» – плохое описание ошибки, а «В форме поиска после отправки запроса выдается ошибка „Internal Error“ вместо результатов» – уже лучше.

Проект. Как правило, один большой продукт подразделяется на несколько проектов для более удобного управления, и в системе учета задач необходимо указать, к какому именно проекту относится данная ошибка.

Версия. Версия продукта, в которой ошибка была обнаружена.

Компонент. Компонент продукта, где была обнаружена ошибка. Как правило, список доступных компонентов ограничен и создается администратором системы.

Приоритет (Priority). Приоритет показывает, с какой срочностью должен быть исправлен дефект.

Серьезность (Важность, Severity). Серьезность показывает степень влияния дефекта на систему.

Окружение. Описание системы - программного и аппаратного обеспечения, на котором воспроизводится данный дефект.

На кого назначен. Кто будет ответственен за решение данного дефекта. В зависимости от принятых правил и процессов в компании, это может быть конкретный разработчик, руководитель группы разработчиков, или же поле по умолчанию остается пустым, а разработчики сами решают, кто будет править этот дефект.

Описание. Подробное описание проблемы. Иногда бывает одно поле для описания проблемы, и тогда в нем нужно указать шаги для воспроизведения, фактический результат, ожидаемый результат.

Иногда вместо этого поля могут быть 3 разных - для шагов, фактического и ожидаемого результатов.

Шаги для воспроизведения. Не всегда этот пункт выделяется как отдельное поле. Если его нет, то нужно шаги для воспроизведения написать в поле «Описание».

Фактический результат. Должен быть обязательно - нужно указать, что мы получили в результате выполнения указанных шагов.

Ожидаемый результат. Необходимо указать, чтобы было понятно, как система должна работать. Если есть возможность, то необходимо давать ссылку на документацию, требования или иные документы, в которых описывается ожидаемое поведение системы. В некоторых случаях этот пункт можно опустить, если ожидаемый результат тривиален (сервер отдает ошибку 500, программа аварийно завершается и т.п.)

Вложения (скриншоты, файлы журнала и пр.). Файлы, которые могут помочь для понимания, воспроизведения, локализации и исправления дефекта.

Отчет об ошибках системы создается для того, чтобы отслеживать количество ошибок и предупреждений, регистрируемых за указанный отчетный период для указанного компьютера, сервера, для указанной группы серверов.

Рассмотрим создание отчета об ошибках на примере Application Virtualization Server.

Данный отчет создает линейчатую диаграмму, которая отображает неустранимые ошибки, ошибки и предупреждения журнала в порядке возрастания в зависимости от времени записи сообщений в журнал.

При создании отчета указываются параметры, используемые для сбора данных при выполнении отчета. Отчеты не выполняются автоматически; их необходимо запускать явным образом для создания выходных данных. Продолжительность времени, необходимого для выполнения этого отчета, определяется объемом данных, собранных в хранилище данных.

Процесс создания отчета из консоли управления сервером Application Virtualization Server одинаковый независимо от типа отчета. При выборе типа отчета в окне отображается краткое описание выбранного отчета.

При создании отчета указываются параметры, используемые для сбора данных при выполнении отчета. Пока вы не запустите отчет, данные не будут собираться.

Создание отчета

1. Запустите мастер создания отчетов, щелкнув правой кнопкой мыши узел "Отчеты" и выбрав пункт "**Новый отчет"** во всплывающем меню.
2. На первой странице мастера создания отчетов введите имя в поле "Имя отчета" и выберите тип отчета из раскрывающегося списка отчетов. В зависимости от выбранного отчета остальные страницы в мастере изменяются в соответствии с требованиями этого типа отчета. Просмотрите следующий список страниц, чтобы найти страницы, которые ссылаются на отчет:
1. **Период отчета** – выберите переключатель, чтобы указать частоту выполнения отчета.
2. **Сервер** – выберите переключатель **"Сервер", "Группа серверов**" или "**Корпоративный**", а затем выберите группу серверов и сервер в соответствующем раскрывающемся списке и поле, если включено.
3. **Приложение** – выберите приложение из раскрывающегося списка доступных приложений.
3. Нажмите кнопку **Готово.**

Процесс выполнения отчета одинаковый независимо от типа отчета. При выборе типа отчета в консоли управления сервером Application Virtualization Server в окне отображается краткое описание выбранного отчета.

Следует заметить, что отчеты не выполняются автоматически; Их необходимо запускать явным образом для создания выходных данных. Продолжительность выполнения отчета определяется объемом данных, собранных в хранилище данных.

Запуск отчета

1. Щелкните узел **"Отчеты**" в области навигации.
2. Щелкните нужный отчет правой кнопкой мыши и выберите команду **"Выполнить отчет**" во всплывающем меню.
3. Страницы, которые необходимо выполнить для запуска отчета, зависят от типа отчета. Чтобы запустить отчет, заполните соответствующие страницы из следующего списка:
1. Выберите переключатель "**Период отчета**", чтобы указать частоту выполнения отчета.

2. Укажите дату начала и окончания в соответствующих полях, чтобы определить диапазон дат, включенных в отчет. Эти даты можно ввести вручную или использовать функцию календаря и выбрать даты.
3. Выберите переключатель **"Сервер", "Группа серверов"** или "**Корпоративный**", а затем выберите группу серверов и сервер в соответствующем раскрывающемся списке и поле, если включено.
4. Выберите нужное приложение из раскрывающегося списка приложений.
4. Нажмите кнопку **Готово**.

Задание к лабораторной работе №1

Изучить порядок сбора информации об ошибках и управление настройками программного обеспечения сбора информации. Собрать информацию об ошибках указанной информационной системы.

Контрольные вопросы

1. Какие типы журналов можно просматривать средствами утилиты просмотра событий? Для чего предназначен каждый из них?
2. Какие уровни событий предусмотрены в журнале?
3. Какова структура отчета об ошибках?
4. Что такое отчет об ошибках?
5. Каковы источники информации для создания отчета об ошибках?

Лабораторная работа № 2.
Разработка схемы резервного копирования

Цель работы: изучение порядка составления плана резервного копирования и создания резервных копий информационной системы.

Теоретические положения

Схема резервного копирования – это комплекс мер и последовательность действий для создания актуальной копии защищаемых данных на резервном носителе. План может состоять из одного или нескольких заданий, включающих в себя следующие параметры:

- Объекты копирования: сервера, виртуальные машины, диски, тома, папки, файлы, приложения, базы данных.
- Способ копирования: полное, инкрементальное, дифференциальное.
- Расписание: дата, время, период или событие для запуска копирования данных.
- Хранилище: место для хранения резервной копии (облако, локальные или сетевые папки и диски, устройства хранения).
- Параметры хранения: жизненный цикл резервной копии в хранилище и действия, выполняемые после окончания срока хранения.
- Дополнительные параметры: уровень компрессии (сжатия), шифрование потока, проверка целостности резервной копии и др.

При создании плана резервного копирования ответьте на следующие вопросы:

- **Насколько важны данные?** Этот критерий поможет решить, как, когда и какую информацию архивировать. Для критичной информации, например, баз данных, следует создавать избыточные архивные наборы, охватывающие несколько периодов архивации. Для менее важной информации, например, для текущих пользовательских файлов, сложный план архивации не нужен, достаточно регулярно сохранять их и уметь легко восстанавливать.
- **К какому типу относится архивируемая информация?** Тип информации поможет определить необходимость архивации данных: как и когда данные должны быть сохранены.
- **Как часто изменяются данные?** Частота изменения влияет на выбор частоты архивирования. Например, ежедневно меняющиеся данные необходимо сохранять каждый день.

- **Нужно ли дополнить архивацию созданием теневых копий?** При этом следует помнить, что теневая копия – это дополнение к архивации, но ни в коем случае не ее замена.
- **Как быстро нужно восстанавливать данные?** Время – важный фактор при создании плана архивации. В критичных к скорости системах нужно проводить восстановление очень быстро.
- **Какое оборудование оптимально для архивации и есть ли оно у вас?** Для своевременной архивации вам понадобится несколько архивирующих устройств и несколько наборов носителей. Аппаратные средства архивации включают ленточные накопители (это наименее дорогой, но и самый медленный тип носителя), оптические диски и съемные дисковые накопители.
- **Кто отвечает за выполнение плана архивации и восстановления данных?** В идеале и за разработку плана, и собственно за архивацию и восстановление должен отвечать один человек.
- **Какое время оптимально для архивации?** Архивация в период наименьшей загрузки системы пройдет быстрее, но не всегда возможно провести ее в удобные часы. Поэтому с особой тщательностью архивируйте ключевые данные.
- **Нужно ли сохранять архивы вне офиса?** Хранение архивов вне офиса – важный фактор на случай стихийного бедствия. Вместе с архивами сохраните и копии ПО для установки или переустановки ОС.

Для построения правильной и эффективной системы резервного копирования необходимо детально изучить и задокументировать все файловые ресурсы, используемые в компании, а затем тщательно спланировать стратегию резервного копирования и реализовать ее на практике

Утилиты архивации и восстановления создают копию файлов и папок на указанном пользователем носителе информации. В случае потери или повреждения пользовательских данных их можно восстановить из файла резервной копии. Работу информационной системы без выполнения регулярного резервного копирования нельзя признать удовлетворительной. Частота архивации (резервного копирования) определяется планом резервного копирования (см. Лабораторная работа №1). Администратор может выбирать различные типы архивации в зависимости от его требований:

- Для типа Обычная (Normal) происходит архивация всех выбранных файлов и системных настроек для определенной папки или диска, и

каждый файл маркируется как прошедший архивацию (имеющий резервную копию).
- Для типа Копирование (Copy) происходит архивация всех выбранных файлов и системных настроек для определенной папки или диска, но файлы не маркируются как прошедшие архивацию.
- Для типа Добавочная (Incremental) происходит архивация только тех файлов, которые были созданы или изменены вслед за последней обычной или добавочной архивацией, и каждый файл маркируется как прошедший архивацию.
- Для типа Разностная (Differential) происходит архивация только тех файлов, которые были созданы или изменены вслед за последней обычной или добавочной архивацией, но файлы не маркируются. Для типа
- Ежедневная (Daily) происходит архивация только тех файлов, которые были созданы или изменены в данный день, но файлы не маркируются.

Тип архивации, который применяется, определяет, насколько сложным будет процесс восстановления. Для восстановления после нескольких добавочных или разностных архиваций необходимо выполнить восстановление из последней обычной резервной копии и из всех добавочных или разностных копий, полученных после обычной архивации и вплоть до настоящего момента. Выполняя архивацию данных, администратор указывает имя и место для файла резервной копии. Файлы архивации можно сохранять на жестком диске или на любом другом типе съемного носителя. При выборе места для резервной копии нужно учитывать размер файла архивации, типы имеющихся носителей, а также возможное требование того, что файлы резервных копий нужно хранить отдельно от компьютера на случай катастрофы.

Функция восстановления информационной системы – позволяет выполнить откат состояния информационной системы к одной из точек восстановления, фиксирующих состояние на момент, когда система стабильно работала. Преимуществом данной функции заключается в том, что она предоставляет возможность быстрого восстановления ("отката" состояния системы к состоянию, в котором она находилась в один из предыдущих моментов во времени) без переустановки системы, а также не подвергает риску случайного перезаписывания рабочих файлов пользователей.

Возможно выполнение отката к любому из следующих типов контрольных точек и точек восстановления:

- Начальная контрольная точка (initial system checkpoint) системы создается при первом запуске компьютера с вновь установленной ОС.
- Точки восстановления для автоматических обновлений (Automatic update restore points) создаются, когда инсталлируются обновления, которые загружаются с помощью штатных средств обновления информационной системы.
- Точки восстановления при восстановлении с резервной копии (Backup recovery restore points) создаются, когда пользователь использует утилиты восстановления данных из резервной копии.
- Пользователь может создавать свои собственные точки восстановления вручную ("ручные" контрольные точки - manual checkpoints) в любой момент с помощью утилит управления информационной системы.
- Точки восстановления при инсталляции программ (Program name installation restore points) создаются, при установке программного обеспечения.
- Системные контрольные точки (System checkpoints) - это запланированные точки восстановления, которые создаются компьютером регулярно, даже если пользователь не вносил никаких изменений в систему.
- Точки восстановления для неопознанного устройства (Unsigned device driver restore points) создаются, когда устанавливается драйвер устройства, который не был опознан или сертифицирован.

Средства восстановления системы обычно сохраняют набор контрольных точек восстановления за период от одной до трех недель. Количество контрольных точек восстановления, доступных в любой заданный момент времени, ограничено объемом пространства, которое выделено пользователем для работы системы восстановления.

При работе с базами данных до настройки резервного копирования следует выбрать ***модель восстановления*** (рис. 3). Для оптимального выбора следует оценить требования к восстановлению и критичность потери данных, сопоставив их с накладными расходами на реализацию той или иной модели.

Как известно, база данных MS SQL состоит из двух частей: собственно базы данных и лога транзакций к ней. База данных содержит пользовательские и служебные данные на текущий момент времени,

лог транзакций включает в себя историю всех изменений базы данных за определенный период, располагая логом транзакций мы можем откатить состояние базы на любой произвольный момент времени.

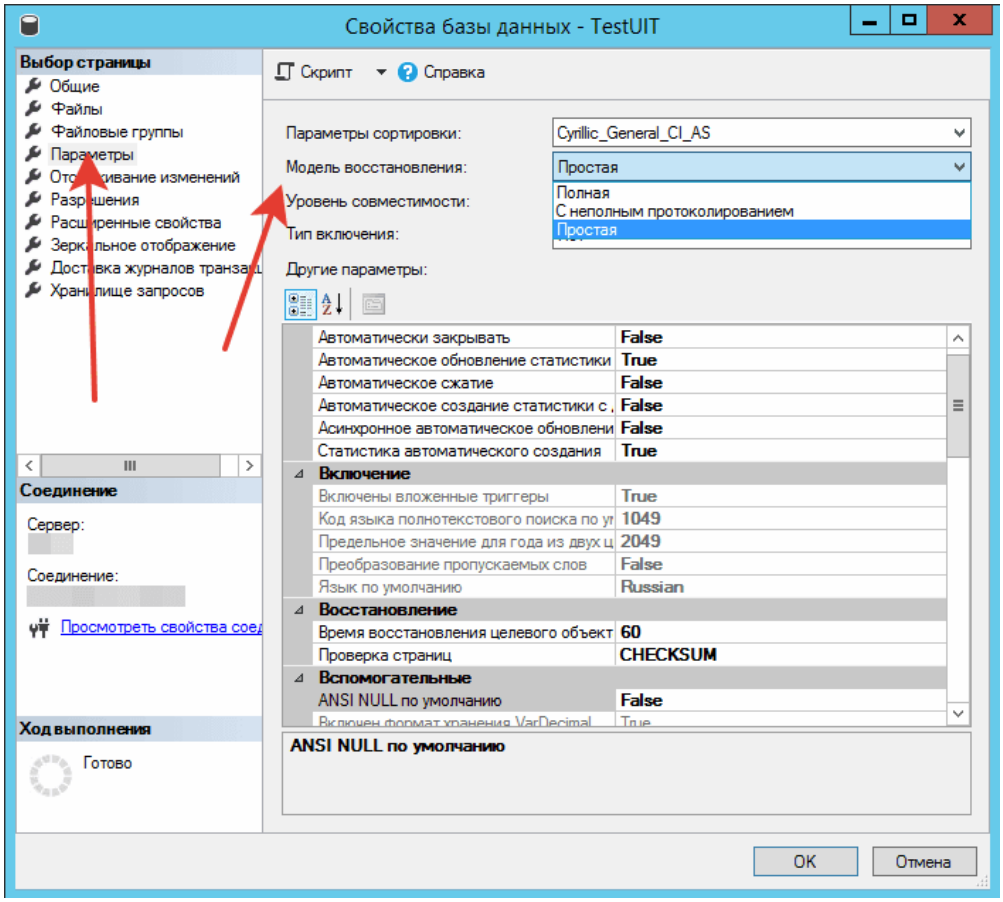

Рисунок 3 – Выбор модели восстановления

Для использования в производственных средах предлагается две модели восстановления: простая и полная. Существует также модель с неполным протоколированием, но она рекомендуется только как дополнение к полной модели на период крупномасштабных массовых операций, когда нет необходимости восстановления базы на определенный момент времени.

- Простая модель предусматривает резервное копирование только базы данных, соответственно восстановить состояние БД мы можем только на момент создания резервной копии, все изменения в промежуток времени между созданием последней резервной копии и сбоем будут потеряны. В тоже время простая схема имеет небольшие накладные расходы: вам необходимо хранить только копии базы данных, лог транзакций при этом автоматически усекается и не растет в

размерах. Также процесс восстановления наиболее прост и не занимает много времени.

- Полная модель позволяет восстановить базу на любой произвольный момент времени, но требует, кроме резервных копий базы, хранить копии лога транзакций за весь период, для которого может потребоваться восстановление. При активной работе с базой размер лога транзакций, а следовательно, и размер архивов, могут достигать больших размеров. Процесс восстановления также гораздо более сложен и продолжителен по времени.

Для баз с небольшим объемом добавления информации может быть выгоднее использовать простую модель с большой частотой копий, которая позволит быстро восстановиться и продолжить работу, введя потерянные данные вручную. Полная модель в первую очередь должна использоваться там, где потеря данных недопустима, а их возможное восстановление сопряжено со значительными затратами.

Виды резервных копий базы данных

Полная копия базы данных – как следует из ее названия, представляет собой содержимое базы данных и часть активного лога транзакций за то время, которое формировалась резервная копия (т.е. сведения обо всех текущих и незавершенных транзакциях). Позволяет полностью восстановить базу данных на момент создания резервной копии.

Разностная копия базы данных – полная копия имеет один существенный недостаток, она содержит всю информацию базы данных. Если резервные копии нужно делать довольно часто, то сразу возникает вопрос неэкономного использования дискового пространства, так как большую часть хранилища будут занимать одинаковые данные. Для устранения этого недостатка можно использовать разностные копии базы данных, которые содержат только изменившуюся со времени последнего полного *копирования информацию.*

Резервная копия журнала транзакций - применяется только при полной модели восстановления и содержит копию журнала транзакций начиная с момента создания предыдущей копии. Важно помнить следующий момент – копии журнала транзакций никак не связаны с копиями базы данных и не содержат информацию предыдущих копий, поэтому для восстановления базы вам необходимо иметь непрерывную цепочку копий того периода, в течении которого вы хотите иметь возможность откатывать состояние базы. При этом мо-

мент последнего успешного копирования должен быть внутри этого периода.

При выполнении задачи резервного копирования с помощью SQL Server Management Studio можно создать соответствующий скрипт Transact-SQL BACKUP, нажав **кнопку "Скрипт"** и выбрав назначение скрипта.

1. После подключения к соответствующему экземпляру ядро СУБД Microsoft SQL Server разверните дерево сервера в **обозреватель объектов**.
2. Разверните узел **Базы данных** и выберите пользовательскую базу данных или разверните узел **Системные базы данных** и выберите системную базу данных.
3. Щелкните правой кнопкой мыши базу данных, которую вы хотите создать резервную копию, наведите указатель на **задачи** и выберите команду **"Создать резервную копию..."**.
4. В диалоговом окне **Резервное копирование базы данных** выбранная база данных приводится в раскрывающемся списке (ее можно изменить на любую другую базу данных на сервере) (рис. 4).

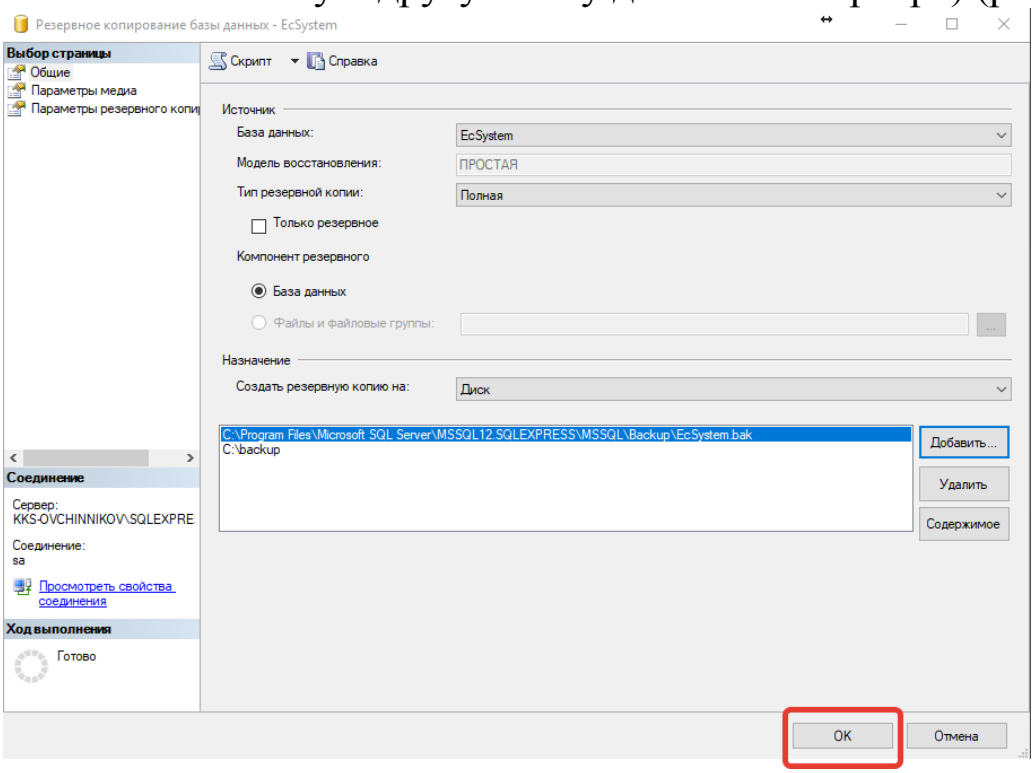

Рисунок 4 – Создание резервных копий

5. В раскрывающемся списке **Тип резервной копии** выберите нужный вариант (по умолчанию выбран тип **Полная**). Перед тем как выполнять разностное резервное копирование или резервное копиро-

вание журналов транзакций, необходимо произвести по крайней мере одно полное резервное копирование базы данных.

6. В разделе **Компонент резервного копирования** выберите **База данных**.

7. В разделе **Назначение** проверьте расположение по умолчанию для файла резервной копии (в папке ../mssql/data). Чтобы выбрать другое устройство, можно использовать раскрывающийся список **Создать резервную копию на**. Щелкните **Добавить**, чтобы добавить объекты резервного копирования и (или) целевые объекты. Резервный набор данных можно перераспределить между несколькими файлами, чтобы повысить скорость резервного копирования.

Чтобы удалить целевой объект резервного копирования, выберите его и щелкните **Удалить**. Чтобы просмотреть содержимое существующего целевого объекта резервного копирования, выберите его и щелкните **Содержимое**.

8. Чтобы начать резервное копирование, нажмите кнопку **ОК**. После успешного завершения резервного копирования щелкните **ОК**, чтобы закрыть диалоговое окно SQL Server Management Studio.

Задание к лабораторной работе №2

Изучить порядок составления плана резервного копирования. Составить план резервного копирования для заданной информационной системы. Создать резервную копию для заданной информационной системы.

Контрольные вопросы

1. Укажите особенности различных типов резервного копирования?
2. Что необходимо учитывать при назначении инкрементального или дифференциального резервного копирования?
3. Какие атрибуты файловой системы учитываются системами резервного копирования?
4. Для чего нужна функция восстановления информационной системы?
5. Каковы причины резервирования данных?
6. Какие типы резервного копирования вы знаете? В чем их особенности?
7. Кто планирует, какие данные нужно резервировать?
8. Какие данные необходимо резервировать?

9. Из каких разделов состоит план резервного копирования?
10. Какие виды контрольных точек существуют?

Лабораторная работа № 3.
Подготовка регламента ввода ИС в эксплуатацию

Цель работы: Освоить процедуру разработки регламента ввода информационной системы (ИС) в эксплуатацию, сформировать навыки работы с нормативной документацией и составления организационно‑распорядительных документов в сфере ИТ.

Теоретические положения

Подготовка регламента ввода информационной системы (ИС) в эксплуатацию требует учёта нормативных требований, этапов разработки и внедрения, а также обеспечения безопасности и соответствия техническим заданиям. Основой для разработки такого регламента служат постановления Правительства РФ, в частности, №676 от 06.07.2015, которое определяет порядок создания, развития, ввода в эксплуатацию, эксплуатации и вывода из эксплуатации государственных информационных систем.

Основные этапы подготовки регламента:

1. Анализ нормативных требований

Необходимо изучить действующие нормативные правовые акты, методические документы и национальные стандарты, которым должна соответствовать система. Это включает требования к защите информации, классификацию системы по уровню безопасности, а также разработку модели угроз безопасности информации.

2. Разработка документации

На этапе создания системы разрабатывается документация, включающая концепцию, техническое задание, проектные решения и рабочую документацию. В ней должны быть описаны автоматизируемые процессы и архитектура системы, требования к защите информации и подсистеме защиты, технико-экономическое обоснование, включая оценку ресурсов для создания, ввода в эксплуатацию и дальнейшей эксплуатации системы.

3. Разработка или адаптация ПО

Включает создание программного обеспечения системы, выбор и адаптацию приобретаемого ПО, а также сертификацию разработанного ПО и средств защиты информации в установленных случаях.

4. Пусконаладочные работы

Проводится автономная наладка технических средств и ПО, загрузка информации в базу данных, комплексная наладка системы, включая средства защиты информации.

5. Предварительные испытания

Разрабатывается программа и методика предварительных испытаний, проводится проверка системы на работоспособность и соответствие техническому заданию. Устраняются выявленные неисправности, оформляются протокол испытаний и акт о приёмке системы в опытную эксплуатацию.

6. Опытная эксплуатация

Система эксплуатируется в соответствии с разработанной программой и методикой. При обнаружении недостатков проводится доработка ПО и дополнительная наладка технических средств. Оформляется акт о завершении опытной эксплуатации с перечнем устранимых недостатков.

7. Приёмочные испытания

Проводятся испытания системы на соответствие техническому заданию. Анализируются результаты устранения недостатков, указанных в акте о завершении опытной эксплуатации, оформляется акт о приёмке системы в эксплуатацию.

Содержание регламента ввода в эксплуатацию

- Перечень действий сотрудников при выполнении задач по эксплуатации системы, включая виды, объёмы и периодичность работ по обеспечению функционирования системы.
- Контроль работоспособности системы и компонентов, обеспечивающих защиту информации.
- Перечень неисправностей, которые могут возникнуть в процессе эксплуатации, и рекомендации по действиям при их возникновении.
- Перечень режимов работы системы и их характеристики, а также порядок и правила перевода системы с одного режима работы на другой.

Основанием для ввода системы в эксплуатацию является правовой акт органа исполнительной власти, который определяет перечень мероприятий по обеспечению ввода системы в эксплуатацию и устанавливает срок начала эксплуатации. Этот акт включает:

- Мероприятия по разработке и утверждению организационно-распорядительных документов, определяющих мероприятия по защите информации в ходе эксплуатации системы.
- Мероприятия по аттестации системы по требованиям защиты информации.

Дополнительные рекомендации:

- Аттестация системы по требованиям защиты информации проводится в установленных законодательством случаях. В некоторых ситуациях её можно совместить с приемочными испытаниями.
- Документирование всех этапов разработки и внедрения системы, включая изменения в инфраструктуре и настройки. Это важно для поддержания актуальности документации и упрощения последующего сопровождения.
- Обучение пользователей и специалистов по обслуживанию системы перед началом промышленной эксплуатации.

При разработке регламента также стоит учитывать специфику организации, отраслевые особенности и дополнительные требования, которые могут быть установлены внутренними нормативными актами.

Для более детальной проработки регламента рекомендуется обратиться к актуальным версиям постановлений Правительства РФ и другим нормативным документам, а также проконсультироваться с экспертами в области информационных технологий и защиты информации.

Задание к лабораторной работе №3

1. Выпишите основные нормативные акты, регулирующие ввод ИС в эксплуатацию. Для каждого акта укажите: название и номер; дату принятия; ключевые требования к процессу ввода в эксплуатацию.

2. Составьте таблицу этапов подготовки к вводу ИС в эксплуатацию (не менее 6 этапов). Для каждого этапа укажите: название этапа; цель этапа; перечень выполняемых работ; ответственные лица/подразделения; входные и выходные документы.

3. Предложите структуру регламента (не менее 5 разделов). Для каждого раздела сформулируйте: заголовок; краткое содержание (2–3 предложения); список документов/приложений, которые должны быть включены.

4. Выберите 2–3 раздела из разработанной структуры и напишите их черновые версии (объём каждого раздела 0,5-1 страница). Обя-

зательно включите: перечень действий сотрудников при эксплуатации системы (с указанием ролей и периодичности), порядок контроля работоспособности системы и средств защиты информации (методы, инструменты, периодичность), алгоритм действий при нештатных ситуациях (перечень типовых неисправностей и способы их устранения).

5. Оформите регламент в виде текстового документа (формат .docx или .pdf): титульный лист (название документа, организация, дата, ответственные); оглавление; основной текст с заголовками разделов; приложения (шаблоны актов, чек-листов, журналов).

Контрольные вопросы

1. Какие нормативные акты регламентируют ввод ИС в эксплуатацию в РФ?
2. Перечислите основные этапы подготовки к вводу ИС в эксплуатацию.
3. Какие разделы обязательно должны присутствовать в регламенте?
4. Кто несёт ответственность за ввод ИС в эксплуатацию?
5. Как оформляется акт о приёмке системы в эксплуатацию?

Лабораторная работа № 4.
Настройка мониторинга ресурсов приложения

Цель работы: освоить практические навыки настройки системы мониторинга ресурсов приложения, научиться выявлять и анализировать ключевые метрики производительности.

Теоретические положения

Мониторинг ресурсов приложения – это непрерывный процесс сбора, анализа и визуализации данных о работе программного обеспечения и инфраструктуры, позволяющий оперативно выявлять проблемы, оценивать производительность и принимать обоснованные решения по оптимизации системы.

Суть мониторинга заключается в преобразовании сырых технических данных (метрик) в понятную информацию о состоянии приложения. Без грамотного мониторинга невозможно обеспечить стабильную работу сервиса, вовремя заметить нарастающие проблемы или спрогнозировать потребность в дополнительных ресурсах.

В основе любой системы мониторинга лежат несколько ключевых компонентов. Агенты сбора данных устанавливаются на серверах и собирают показатели работы операционной системы и приложений. Эти данные отправляются в специальное хранилище – базу данных временных рядов, где они сохраняются как в актуальном виде, так и в исторической перспективе. Для наглядного представления информации используются системы визуализации – дашборды, на которых отображаются графики и индикаторы состояния. Важнейший элемент – система оповещений, которая автоматически реагирует на критические события, уведомляя ответственных лиц или даже запуская сценарии восстановления.

При настройке мониторинга важно определить набор ключевых метрик, отражающих состояние системы. На уровне операционной системы отслеживают загрузку процессора, использование оперативной памяти, свободное место на диске и сетевую активность. На уровне приложения критически важны время отклика на запросы, количество обрабатываемых запросов в секунду, коды ответов сервера (особенно ошибки 5xx и 4xx), а также длина очереди задач для систем с асинхронной обработкой. В зависимости от специфики приложения могут потребоваться дополнительные метрики: для баз данных –

время выполнения запросов, для систем кэширования – соотношение удачных и неудачных обращений, для микросервисных архитектур – задержки между сервисами.

Особую роль играет настройка системы оповещений (алертинга). Её задача – информировать команду только о действительно значимых событиях, требующих вмешательства. Эффективные оповещения (алерты) строятся на трёх основных принципах: превышение критического порога в течение определённого времени (например, загрузка CPU выше 85 % на протяжении трёх минут), превышение допустимого количества событий за период (более 100 ошибок 500 за минуту) или существенное отклонение от нормального тренда (резкий рост времени отклика на 50 % относительно среднего значения за последний час). При этом важно избегать ложных срабатываний из‑за кратковременных всплесков и обязательно включать в уведомления контекстную информацию – какой сервер или сервис затронут и какие могут быть причины проблемы.

Существует два основных подхода к мониторингу. Мониторинг «белого ящика» предполагает доступ к внутренним компонентам системы и позволяет точно диагностировать причины сбоев – например, анализ логов приложения или метрик виртуальной машины. Мониторинг «чёрного ящика» оценивает систему извне, имитируя пользовательский опыт, – это может быть проверка доступности эндпоинта или проведение синтетических тестов. Оба подхода дополняют друг друга и часто используются совместно.

При выборе метрик для мониторинга необходимо учитывать несколько факторов: целевое назначение приложения (для интернет‑магазина критично время отклика и количество ошибок), среду развёртывания (облако, физические серверы, контейнеры), доступные ресурсы (бюджет, мощность агентов, объём хранилища) и требования к уровню обслуживания (SLA), включая допустимое время простоя и целевые показатели производительности.

Чтобы мониторинг был эффективным, следует придерживаться ряда лучших практик. Во‑первых, выстраивать иерархию метрик – начинать с базовых системных показателей и постепенно добавлять прикладные метрики по мере необходимости. Во‑вторых, обеспечивать контекстную визуализацию – объединять связанные показатели на одном дашборде (например, загрузку CPU, использование памяти и количество запросов) и использовать аннотации для отметки значимых событий. В‑третьих, сохранять исторические данные для ана-

лиза трендов и сравнения текущих показателей с прошлыми. В-четвёртых, по возможности автоматизировать реагирование – настраивать автоматическое масштабирование на основе метрик и интегрировать систему оповещений с инструментами управления инцидентами. И наконец, регулярно проводить аудит мониторинга – удалять неактуальные метрики и алерты, корректировать пороги срабатывания на основе накопленного опыта эксплуатации.

Среди типичных ошибок при настройке мониторинга можно выделить отслеживание только поверхностных показателей без углублённого анализа, отсутствие исторических данных для выявления трендов, избыточное количество алертов, приводящее к «оповестительному шуму», несинхронизированное время на разных узлах системы (что искажает корреляцию событий) и игнорирование метрик зависимых компонентов (баз данных, систем кэширования, сетевого взаимодействия). Избегая этих ошибок и следуя описанным принципам, можно построить эффективную систему мониторинга, обеспечивающую стабильность и производительность приложения.

Задание к лабораторной работе №4

1. Выберите и установите инструменты мониторинга: определите набор инструментов для сбора метрик (например: Prometheus для сбора, Grafana для визуализации), установите и настройте агенты мониторинга на сервере с приложением, проверьте подключение агента к серверу мониторинга.

2. Определите ключевые метрики: составьте таблицу метрик, которые необходимо отслеживать. Для каждой метрики укажите: название (например, `cpu_usage`); единицу измерения (%, мс, шт./сек и т. п.); пороговое значение для алертов; источник данных (системные счетчики, логи приложения и т. п.). Обязательные метрики: загрузка CPU и памяти; количество открытых соединений; время отклика HTTP-запросов; количество ошибок 5xx; размер очереди задач (если есть); использование дискового пространства.

3. Настройте сбор данных: настройте экспортеры для сбора метрик из приложения и ОС, создайте конфигурационные файлы для сбора логов (например, Filebeat для ELK), проверьте поступление данных в систему мониторинга (убедитесь, что метрики отображаются в интерфейсе).

4. Создайте дашборды: разработайте 2-3 дашборда в системе визуализации (например, Grafana), содержащих информацию следующего характера: общий обзор приложения (графики загрузки CPU/памяти; карта запросов по эндпоинтам; счётчик ошибок 5xx), детальный анализ производительности (гистограмма времени отклика; топ медленных запросов; график количества запросов в секунду), состояние инфраструктуры (использование диска; количество активных соединений; статус сервисов).

5. Настройте алерты. Для каждой критической метрики задайте: условие срабатывания (например, `cpu_usage > 90%` в течение 5 мин), канал уведомления (email, Slack, Telegram), текст сообщения с указанием метрики и возможных причин.

Пример алерта:

```
IF: http_requests_5xx_rate > 10/min FOR 2m
 THEN: Send alert to ops-team@company.com
 MESSAGE: High error rate detected! Check
          application logs.
```

6. Проведите тестирование системы. Сгенерируйте нагрузку на приложение (например, через `ab` или `curl`), убедитесь, что метрики обновляются в реальном времени, проверьте срабатывание алертов при превышении порогов, проанализируйте логи на предмет аномалий.

Контрольные вопросы

1. Какие метрики обязательны для мониторинга веб-приложения?
2. В чём разница между *pull* и *push* моделями сбора метрик?
3. Как определить пороговое значение для алерта?
4. Какие инструменты можно использовать для мониторинга микросервисов?
5. Как связать метрики приложения с логами для диагностики проблем?

Лабораторная работа № 5. Проведение процедуры восстановления после сбоя

Цель работы: приобретение навыков восстановления данных из ранее созданных резервных копий.

Теоретические положения

Восстановлением называется процедура, которая выполняется для перемещения на жесткие диски компьютера вместо потерянного или испорченного файла или набора файлов их работающей копии из архивных (резервных) данных.

Управление восстановлением – это важная часть процесса аварийно-восстановительных работ. От того, как поддерживается готовность к аварийно-восстановительным работам, будут зависеть время простоя системы и эффективность восстановления потерянного массива информации, полученного между последним архивированием и аварией.

При восстановлении используются следующие основные модели:

- простое восстановление,
- полное восстановление,
- массовое восстановление.

В простой модели восстановления данные могут быть восстановлены только на момент последнего резервного копирования. Эта модель обеспечивает высокую эффективность выполнения массовых операций загрузки данных. Как следует из названия, простая модель копирования и восстановления наиболее легкая и удобная по сравнению с другими моделями. Максимально возможный объем данных, которые могут быть потеряны, определяется периодом времени между созданиями резервных копий.

В модели полного восстановления данные могут быть восстановлены в том виде, в котором она находилась вплоть до аварии. Модель поддерживает восстановление до контрольной точки, помеченной именованной транзакцией. Транзакция – это некоторое законченное, с точки зрения пользователя, действие в информационной системе. В модели полного восстановления массовые операции импорта протоколируются в журнал транзакций и, следовательно, могут быть полностью или частично восстановлены.

В модели массового восстановления операции импорта протоколируются в минимальном объеме. Это обеспечивает высокую производительность массовых операций загрузки, однако делает невозможным восстановление на любой заданный момент времени.

В зависимости от интервала времени, затрачиваемого на воссоздание информации, восстановление подразделяется на следующие виды:

- Восстановление в реальном времени (то есть сразу) или достаточно близко к нему. Данные, созданные не более чем несколько секунд назад, должны быть немедленно доступны пользователям и системам, даже если источник этих данных отключен. Это касается промышленных и медицинских систем, в которых задержка, определяемая восстановлением, допускается в течение долей и единиц секунд. Уровень времени в несколько секунд называется уровнем критического восстановления.
- Если восстановление требуется в течение десяти минут, пока отключен первоначальный источник, то такое восстановление называется экстренным восстановлением.
- Когда на восстановление можно затратить один час, оно называется срочным восстановлением.
- Восстановление, требующее от одного до четырех часов, называется важным восстановлением.
- Все другие восстановления, которые выполняются за интервал времени, больший предыдущих, называются рядовыми.

Стратегия архивации и восстановления

Архивирование не производится ежеминутно, поэтому полностью восстановить все данные на тот момент, когда произошла авария (если только авария не произошла сразу же после завершения архивирования), с помощью резервных копий невозможно. При восстановлении возможна потеря данных из-за того, что будут устанавливаться устаревшие данные из резервных копий. Необходимо решить, за какое время работы информационной системы потеря информации допустима. Затем, когда установлен этот допустимый уровень, необходимо отработать способы, которые позволят его поддерживать. Также необходимо определить, сколько такая поддержка будет стоить.

Так как при восстановлении будут использоваться устаревшие данные, необходимо определить срок их годности для восстановле-

ния. Существует следующий список сроков годности. Для восстановления можно использовать данные, созданные:

- месяц назад или ранее;
- от одной до четырех недель назад;
- от четырех до семи дней назад;
- один-три дня назад;
- шесть-двенадцать часов назад;
- от двух до пяти часов назад;
- от одной до 60 минут назад.

Исходя из сроков годности данных, необходимо определить, какую из технологий архивирования можно использовать. Например, при сохранении на ленте последний уровень срока годности из-за низкой скорости копирования реализован быть не может.

При выработке стратегии восстановления необходимо оценить, насколько требуется немедленное восстановление данных (в реальном времени), и уточнить еще один фактор – возможны ли изъяны в резервных копиях. Это нужно принимать во внимание, если данные непрерывно меняются. Кроме того, файлы могут быть повреждены вирусами. Причем не только в работающей системе, но и в их резервной копии. Архивирование испорченных или зараженных вирусами файлов не обеспечивает достаточную сохранность информации. Гарантировать безопасность копий можно только с помощью предварительного тестирования их антивирусными программами высокой сложности или алгоритмов контроля качества данных.

Безопасность восстановления существенным образом зависит от требуемого времени восстановления. Чем быстрее реакция на запрос по восстановлению, тем выше шанс получить испорченные данные. Однако это не означает, что критические восстановления всегда рискованные, а восстановленные с их помощью данные – бракованные. Это означает другое. У данных, резервные копии которых получены ближе всего к моменту аварии, больше вероятность оказаться бракованными, чем у тех, что получены за несколько часов или дней до нештатной ситуации. Если авария произошла из-за порчи данных или вирусной инфекции, то вероятнее всего, что недавно резервированные данные также заражены.

Другим фактором, который следует учитывать, является то, что самые “чистые” резервные копии больше всего отличаются от данных, которые нужно восстанавливать, то есть они самые устаревшие.

При восстановлении данных необходимо руководствоваться следующим:

- Проверять, не вызовет ли восстановление данных их потерю, если файлы восстанавливаются на то же место, откуда их копировали. Например, если файл испорчен, но еще не устарел, а при восстановлении его заменяет файл, который не испорчен, но устарел, то потери могут быть большими. Лучше всего, прежде чем заменить испорченный файл, проверить возможность его исправления.
- Учитывать последствия восстановления. При восстановлении файла восстанавливается не только его содержимое, но также все атрибуты и вся информация, известная об этом файле на момент архивирования. Все новое, касающееся файла и его отношений с остальным миром, уже не будет отражено в восстановленном варианте. Примером является восстановление папки, к которой со времени последнего архивирования получило доступ несколько новых групп и пользователей. Восстановление заблокирует этих пользователей.
- Во время восстановления в то же место, откуда делалась резервная копия, доступ пользователей в это место следует заблокировать. Попытка пользователей открыть еще не полностью восстановленные файлы может привести к аварийному завершению восстановления.

Восстановление резервной копии базы данных с помощью среды SSMS

1. В **обозревателе объектов** подключитесь к экземпляру компонента SQL Server Database Engine и разверните его.
2. Щелкните правой кнопкой мыши узел **Базы данных** и выберите команду **Восстановить базу данных...** (рис. 5).

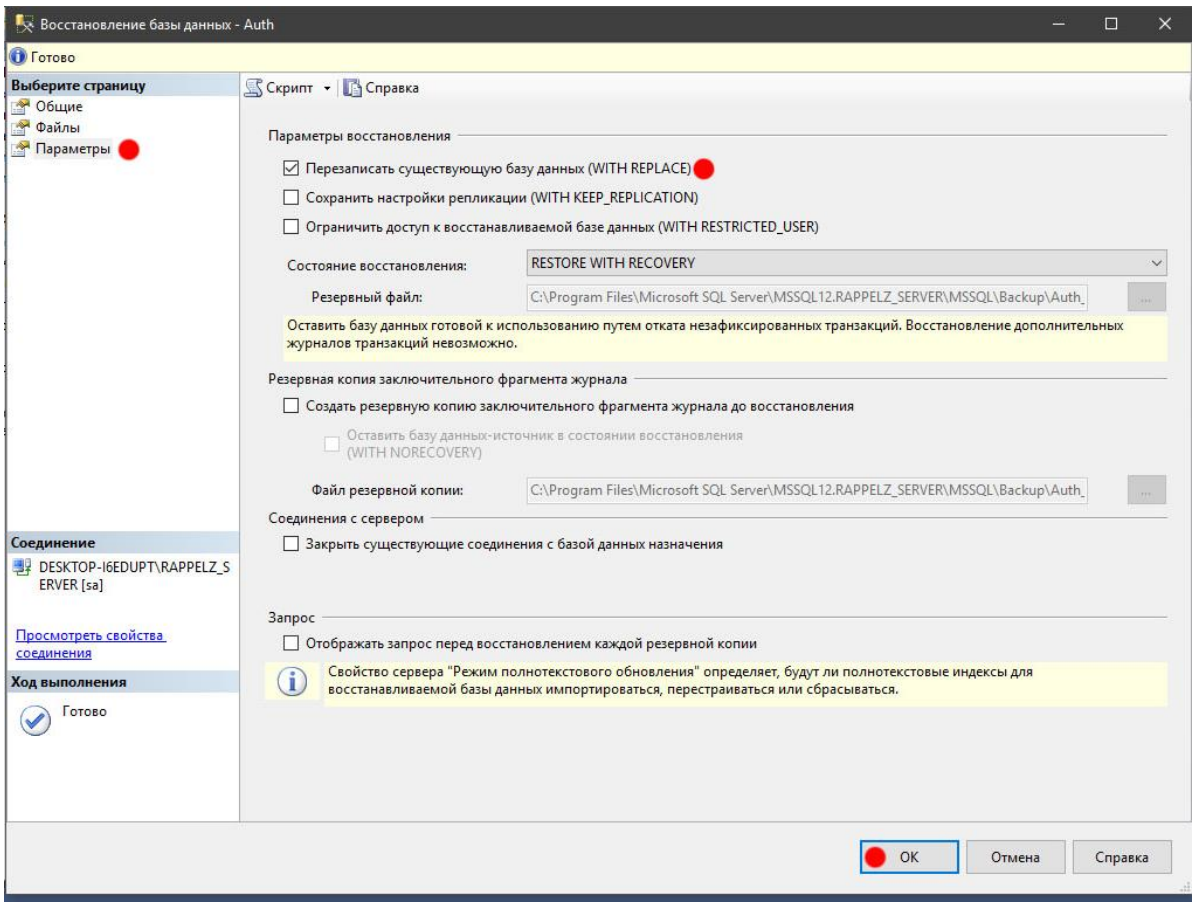

Рисунок 5 – Восстановление БД

- Чтобы указать источник и расположение восстанавливаемых резервных наборов данных, используйте страницу **Общие**, раздел **Источник**. Выберите один из следующих вариантов: **База данных.** Выберите из раскрывающегося списка базу данных для восстановления. Данный список содержит только базы данных, резервное копирование которых было выполнено в соответствии с журналом резервного копирования **msdb**.
- Если резервная копия была получена с другого сервера, на целевом сервере не будет журнала резервного копирования для указанной базы данных. В этом случае щелкните пункт **Устройство**, чтобы вручную указать файл или устройство для восстановления.
- В списке **Источник > Устройство > База данных** выберите имя базы данных, которую нужно восстановить. Данный список доступен, только если выбран параметр **Устройство**. Будут выбраны только те базы данных, резервные копии которых доступны на выбранном устройстве.

- В разделе **Назначение**, в поле **База данных** автоматически появится имя базы данных для восстановления. Для изменения имени базы данных введите новое имя в окно **База данных.**
- **В поле "Восстановление**" оставьте значение по умолчанию в качестве последней резервной копии или выберите временную шкалу для доступа к диалоговому окну временной шкалы резервного копирования, чтобы вручную выбрать точку во времени, чтобы остановить действие восстановления.
- В сетке **Резервные наборы данных для восстановления** выберите нужные резервные наборы. В этой сетке отображаются резервные копии, доступные в указанном месте. По умолчанию предлагается план восстановления. Чтобы переопределить предложенный план восстановления, можно изменить выбранные элементы в сетке. Выбор всех резервных копий, которые зависят от восстановления более ранних копий, отменяется автоматически, как только отменяется выбор более ранних копий.
- При необходимости выберите "Файлы" в области "**Выбор страницы", чтобы открыть диалоговое окно "Файлы".** Отсюда можно восстановить базу данных в новое расположение, определив новое место восстановления для каждого файла в сетке **Восстановить файлы базы данных как.**
- Чтобы просмотреть или выбрать дополнительные параметры, на странице **Параметры** на панели **Параметры восстановления** выберите любой из следующих параметров, если он подходит к данной ситуации:

ПараметрыWITH (необязательно):

- **Перезаписать существующую базу данных (WITH REPLACE)**
- **Сохранить параметры репликации (WITH KEEP_REPLICATION)**
- **Ограничить доступ к восстановленной базе данных (WITH RESTRICTED_USER)**

3. Выберите параметр в поле **Состояние восстановления**. В данном окне определяется состояние базы данных после операции восстановления.

- По умолчанию используется схема **RESTORE WITH RECOVERY**, которая выполняет откат незафиксированных транзакций и после завершения оставляет базу данных работоспособной. Дополнительные журналы транзакций не могут быть восстановлены.

Выберите этот вариант, если выполняется восстановление сразу всех необходимых резервных копий.

- Схема **RESTORE WITH NORECOVERY** оставляет базу данных в нерабочем состоянии и не выполняет откат незафиксированных транзакций. Можно восстановить дополнительные журналы транзакций. Базу данных нельзя использовать, пока она не будет восстановлена.
- Схема**RESTORE WITH STANDBY** оставляет базу данных в режиме только для чтения. С помощью данного параметра можно отменить незафиксированные транзакции и сохранить отмененные действия в резервном файле, чтобы результаты восстановления можно было отменить.

4. Создайте резервную копию заключительного фрагмента журнала до восстановления**.** Не для всех сценариев восстановления требуется резервная копия заключительного фрагмента журнала. Если имеются активные соединения с базой данных, то операция восстановления может завершиться ошибкой. Проверьте параметр "Закрыть существующие подключения", чтобы убедиться, что все активные подключения между Management Studio и базой данных закрыты. Эта настройка переводит базу данных в однопользовательский режим перед началом процедуры восстановления, а затем возвращает в многопользовательский режим после ее завершения.

5. Установите флажок **Выдавать запрос перед восстановлением каждой резервной копии**, если хотите отследить каждую операцию восстановления. Это не требуется, если не нужно наблюдать за состоянием операции восстановления для базы данных большого объема. Нажмите **ОК.**

Восстановление работоспособности системы

Современные операционные системы довольно устойчивы к сбоям и, как правило, стабильность системы тем выше, чем меньше изменений вносится в систему в процессе работы. Однако вносить изменения в конфигурацию операционной системы (установка нового ПО, обновление системы или драйверов, изменение системных параметров и компонент) все же необходимо, в результате чего работоспособность системы может быть нарушена. Обычно, процесс загрузки в операционной системе разделен на несколько частей:

- инициализация;
- работа загрузчика;
- загрузка ядра;

- регистрация.

Соответственно, если проблемы возникают на какой-либо из этих фаз, то операционная система не может выполнить успешную загрузку.

В Windows присутствуют различные средства восстановления, которые вы можете использовать для восстановления работоспособности Windows. Это *Безопасный Режим* (Safe Mode), *Консоль Восстановления* (Recovery Console) и *Диск Аварийного Восстановления* (Automatic System Recovery). Для выбора этих режимов необходимо войти в меню дополнительных вариантов загрузки, для этого во время загрузки системы нажать клавишу F8.

Использование *Последней Удачной Конфигурации* (Last Known Good Configuration) Если проблема возникла сразу после изменения настроек системы (как правило, после установки нового драйвера), следует воспользоваться загрузкой Windows в режиме *Последней Удачной Конфигурации* (Last Known Good Configuration). Этот режим восстанавливает информацию реестра и настройки драйвера, которые были использованы, когда система последний раз успешно загружалась. При этом восстанавливается только ветвь реестра HKLM\System\CurrentControlSet и поэтому не решаются проблемы, вызванные повреждением или потерей системных разделов или файлов. Если удалось загрузить Windows в режиме *Последней Удачной Конфигурации*, то последние изменения, которые были сделаны в системе, скорее всего и были причиной, препятствующей корректному запуску. Удалите или выполните обновление сбойной программы или драйвера, затем загрузитесь в обычном режиме.

Загрузка системы в *Безопасном Режиме* (Safe Mode) При загрузке в *Безопасном Режиме* (Safe Mode) Windows загружает только драйвера и службы, которые необходимы для работы. В том случае, если загрузка в *Безопасном Режиме* была выполнена успешно, то необходимо определить причину возможного сбоя в процессе загрузки. В операционной системе имеется несколько инструментов, которые могут в этом помочь. Выполните вход под учетной записью с правами администратора системы и просмотрите журналы событий (eventvwr. msc). Необходимо провести анализ журнала системы и журнала приложений на наличие предупреждений и сообщений об ошибках, обращайте внимания на источники событий.

Программа просмотра *Сведений о Системе* (msinfo32. exe) выводит различную информацию об оборудовании, системных компо-

нентах и программном окружении. Если проблемное устройство обнаружено, отключите, перенастройте или попробуйте обновить используемый им драйвер. Для отключения устройства и драйверов используйте *Диспетчер Устройств* из оснастки *Администрирование - Управление Компьютером*. Если конфликтов оборудования не обнаружено, просмотрите раздел *Программная среда – Автоматически загружаемые программы*. Попробуйте запретить программы, загружаемые автоматически, и перезагрузите компьютер. Для настройки запрета воспользуйтесь программой *Настройки Системы* (Msconfig.exe), если после запрета загрузка проходит нормально, разрешайте по одной автозагрузку программ. Если и это не помогло, воспользуйтесь режимом Диагностического Запуска, который можно установить в программе *Настройки Системы*.

Консоль восстановления

Консоль Восстановления это набор средств командной строки, способных помочь восстановить Windows в том случае если компьютер не может выполнить загрузку. Доступ к Консоли можно запустить двумя способами: с загрузочного диска Windows Server соответствующей версии или если *Консоль Восстановления* была уже установлена на компьютере. Консоль следует запускать в том случае, если ни *Режим Последней Удачной Конфигурации*, ни запуск в *Режиме Восстановления* положительного эффекта не дали.

Для того чтобы вывести на экран все доступные команды *Консоли Восстановления* наберите в командной строке help (или help < command> для получения справки по конкретной команде). Прежде чем начать работу с командами проверьте состояние вашего жесткого диска. Для этого воспользуйтесь командой chkdsk /F /R. Если chkdsk не может исправить проблемы с жестким диском, то ваши файловая система или основная загрузочная запись возможно повреждены или недоступны. Попытайтесь использовать команды Fixmbr и Fixboot для восстановления, в противном случае придется создать разделы заново и переформатировать жесткий диск или обратиться в специализированные компании, которые занимаются восстановлением жестких дисков.

Кроме того, невозможность использования *Безопасного Режима* для загрузки системы может быть обусловлена повреждением системного реестра Windows или загрузочных файлов. Загрузочные файлы (Ntldr, Ntdetect.com, Boot.ini, Ntbootdd.sys – для контроллеров SCSI, bootfont.bin – для локализованных версий Windows), располо-

женные в корне системного раздела, могут быть восстановлены из каталога i386 на установочном дистрибутиве Windows Server.

Файлы системного реестра, каждый раз после создания копии *Состояния Системы*, копируются на системный раздел в каталог % Systemroot%\Repair. Используя *Консоль Восстановления* можно восстановить поврежденные файлы реестра из этого каталога в исходный – % Systemroot%\system32\config. Не забудьте предварительно сохранить текущие файлы в другой каталог перед выполнением этой процедуры восстановления. После этого реестр Windows будет содержать информацию, которая была на момент выполнения последнего копирования *Состояния Системы*. Изменения в системе, начиная с этого момента, будут после восстановления потеряны. Если резервное копирование ни разу не производилось, то в каталоге Repair будет содержаться копия данных, сделанная после непосредственно после установки Windows.

Однако не во всех проблемах виновна операционная система и иногда сбой в загрузке возникает еще до начала самой загрузки. Например, если какой-либо другой раздел Вы пометите по ошибке как «активный», не будет содержать файлы загрузки операционной системы, компьютер не запустится. В этом случае при помощи *Консоли Восстановления* необходимо вернуть метку активного раздела системному разделу. Для этого следует воспользоваться командой diskpart. Предварительно необходимо выбрать ваш системный раздел с файлами запуска (параметры select disk < n> и select partition < m> – где n, m – номера, удовлетворяющие соглашениям об именовании ARC), после чего воспользуйтесь параметром active, чтобы пометить его как активный.

Восстановление из резервной копии

Самым последним вариантом является восстановление из резервной копии, которую вы обязательно должны были делать регулярно на работающей системе. Для ее использования необходимо установить новую копию Windows. Если локальный диск является работоспособным, то удаляем существующий системный раздел и создаем новый (при этом размер нового раздела должен быть не менее чем у прежнего). Устанавливаем новую копию Windows Server на тот же самый раздел, где размещалась Windows ранее. После этого можно приступить к восстановлению из резервной копии.

Восстановление операционной системы Windows 10

Рассмотрим восстановление операционной системы Windows 10.

• Нажмите кнопку «Пуск» в левом нижнем углу экрана и выберите «Настройки».

• В открывшемся окне «Настройки» выберите раздел «Обновление и безопасность».

• На левой панели выберите «Восстановление» (рис. 6).

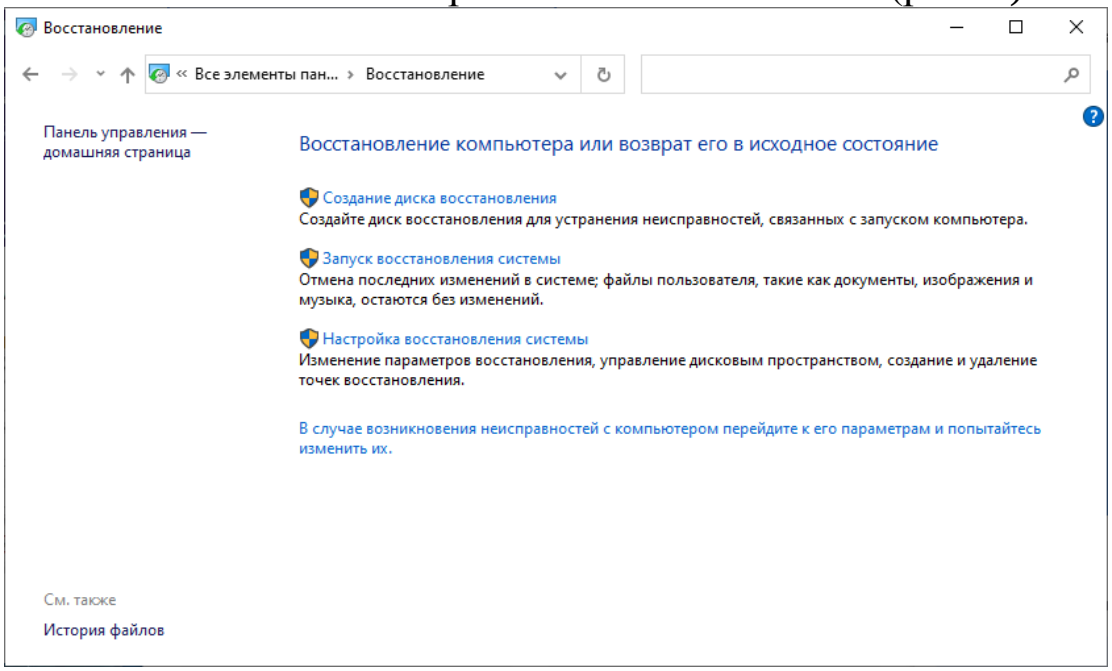

Рисунок 6 – Восстановление ОС Windows

• В разделе «Восстановление» найдите опцию «Восстановление данных фабричных настроек» и нажмите кнопку «Начать» под этой опцией.

• Следуйте инструкциям на экране и выберите соответствующий вариант восстановления системы. При этом будет предложено сохранить или удалить личные файлы. Выберите «Сохранить мои файлы», чтобы сохранить важную информацию.

• Ждите, пока система выполнит процесс восстановления. Это может занять некоторое время, поэтому будьте терпеливы.

• После завершения процесса компьютер автоматически перезагрузится и вернется в рабочее состояние.

Восстановление операционной системы Linux

Во всех современных дистрибутивах, для загрузки в режим восстановления Linux, необходимо перезагрузить компьютер и выбрать соответствующий раздел меню Grub. Для дистрибутивов Linux, таких как Debian, Linux Mint, Ubuntu и многих других, алгоритм загрузки в режим восстановления следующий:

1. Перезагружаем компьютер, и ждем загрузки меню Grub. Если меню Grub не отображается, то следует снова перезагрузить компьютер, и многократно нажать на клавишу ESC после прохождения загрузки BIOS.
2. В меню загрузчика Grub, выбираем пункт "*Дополнительные параметры для ...*" (рис. 7).

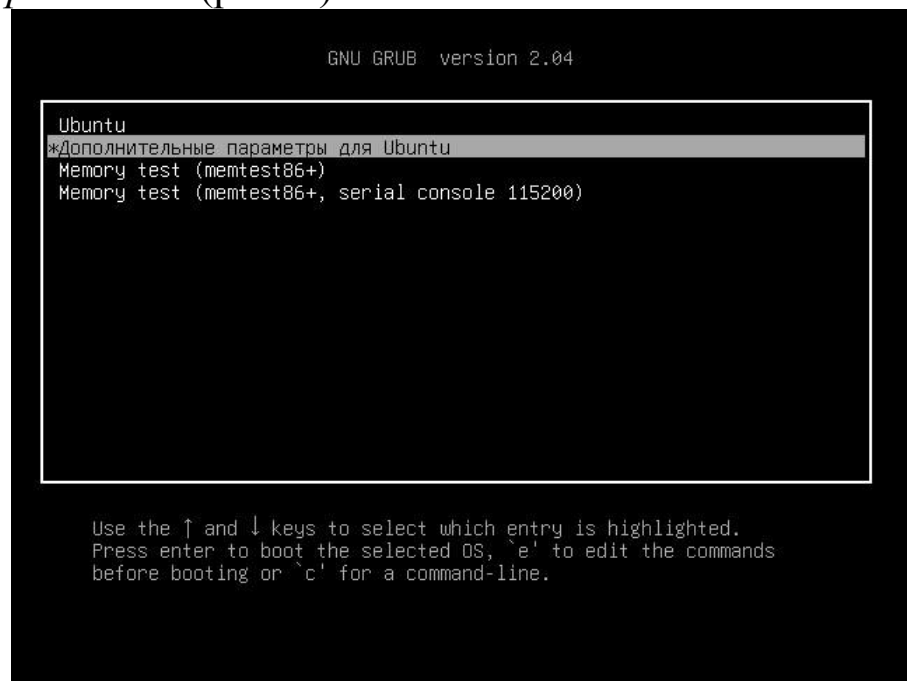

Рисунок 7 – Меню загрузчика Grub

Здесь выбираем строку, которая содержит надпись "(recovery mode)". Если таких строк несколько - скорее всего там отображаются варианты загрузки с разными версиями ядра. В таком случае, лучше выбрать строку с наиболее свежей версией ядра (рис. 8).

Рисунок 8 – Восстановление ОС Linux

После этого начнется загрузка Linux в режиме восстановления.

Задание к лабораторной работе №5

1. Изучите порядок восстановления базы данных из созданной ранее резервной копии. Восстановите базу данных из резервной копии.

2. Изучите порядок восстановления работоспособности информационной системы. Восстановите работоспособность заданной информационной системы.

Контрольные вопросы

1. Какую задачу решает процедура восстановления информации?
2. Перечислите виды восстановления информации.
3. Какие требования учитываются при разработке стратегии архивирования?
4. Опишите процедуру восстановления информации до момента сбоя в системе.
5. Какие особенности следует учитывать при выборе стратегии восстановления информации?
6. Для чего предназначена цифровая подпись системных файлов?

7. С помощью какой утилиты осуществляется проверка системных файлов? Какие функции она выполняет?

8. Какие функции выполняет утилита Msconfig?

9. Что такое безопасный режим загрузки Windows? Какие задачи с помощью его решаются?

10.Что такое точки восстановления системы? Как с помощью их решается проблема устранения проблем, вызванных установкой нового приложения?

11. Для чего служит консоль восстановления? Какие способы запуска её вы знаете?

Лабораторная работа № 6. Организация непрерывной доставки обновлений для информационных систем

Цель работы: Освоить принципы и практические навыки организации процесса непрерывной доставки (Continuous Delivery) обновлений для информационных систем.

Теоретические положения

Непрерывная доставка (Continuous Delivery, CD) представляет собой методологию разработки программного обеспечения, направленную на автоматизацию процесса выпуска обновлений с целью их максимально оперативного и безопасного внедрения в продуктивную среду. Суть подхода заключается в том, что код всегда находится в состоянии, пригодном для развёртывания – любая успешная сборка после прохождения полного цикла автоматизированных тестов может быть выпущена в эксплуатацию по решению ответственных лиц.

В основе непрерывной доставки лежит концепция конвейера (pipeline) – последовательности автоматизированных этапов, через которые проходит код от момента внесения изменений до попадания в продуктивную среду. Конвейер включает сборку проекта, выполнение тестов различных уровней, упаковку артефактов, развёртывание в тестовые среды и, при положительном результате, в продуктивную среду. Каждый этап служит фильтром, предотвращающим попадание некачественного кода на следующие ступени.

Для эффективной реализации непрерывной доставки необходимо соблюдение ряда основополагающих принципов:

- частые коммиты – разработчики регулярно (несколько раз в день) вносят изменения в основную ветку репозитория;
- автоматизированное тестирование – все этапы проверки кода выполняются без ручного вмешательства;
- единая кодовая база – использование общего репозитория для всех участников команды;
- параметризация окружений – конфигурация приложения зависит от среды развёртывания;
- версионирование артефактов – каждый собранный пакет имеет уникальный идентификатор;

• отказоустойчивость – возможность быстрого отката к предыдущей версии при возникновении проблем.

Технологический стек для организации непрерывной доставки включает несколько категорий инструментов: системы контроля версий (Git, Mercurial, SVN) для управления изменениями кода, CI/CD‑платформы (Jenkins, GitLab CI/CD, GitHub Actions, CircleCI, Travis CI) для автоматизации конвейера сборки и развёртывания, инструменты контейнеризации (Docker, Kubernetes) для обеспечения консистентности окружений, системы управления конфигурациями (Ansible, Puppet, Chef) для автоматизации настройки инфраструктуры.

Процесс непрерывной доставки реализуется через последовательность этапов:

• отправка кода в систему контроля версий;

• сборка проекта – компиляция исходного кода и создание исполняемых артефактов;

• автоматическое тестирование, включающее юнит‑тесты, интеграционные тесты, тесты производительности и безопасности;

• публикация собранных артефактов в репозиторий (Docker Registry, Nexus и др.);

• развёртывание в тестовую среду (UAT) для ручного тестирования и валидации;

• развёртывание в продуктивную среду после получения одобрения от ответственных лиц.

При внедрении непрерывной доставки критически важны как технические, так и организационные аспекты. С технической стороны требуется наличие выделенных серверов или контейнеров для разных окружений, репозиторий артефактов для хранения сборок, система мониторинга и логирования для отслеживания состояния процесса.

С организационной точки зрения необходимо чёткое разделение ролей между участниками процесса (разработчики, тестировщики, DevOps‑инженеры), документирование всех процессов и процедур, регулярные ретроспективы для анализа и улучшения процесса.

Эффективность процесса непрерывной доставки оценивается через такие ключевые метрики, как время сборки, время развёртывания, процент успешных сборок, среднее время восстановления после сбоя.

Среди основных рисков при внедрении непрерывной доставки выделяют сбои при развёртывании, потерю данных, уязвимости безопасности, несовместимость версий.

Для минимизации этих рисков применяется поэтапное развёртывание с постепенным увеличением доли пользователей, автоматический откат при обнаружении ошибок, регулярное резервное копирование данных, сканирование на уязвимости перед развёртыванием, тестирование в изолированных средах.

Успешная организация непрерывной доставки требует комплексного подхода, включающего техническую готовность инфраструктуры и инструментов, организационные изменения в процессах и ролях, формирование культуры постоянного улучшения и быстрой обратной связи.

Результатом внедрения непрерывной доставки становится существенное сокращение времени вывода обновлений на рынок, повышение качества программного обеспечения, снижение операционных затрат и улучшение взаимодействия между командами разработки и эксплуатации.

Задание к лабораторной работе №6

1. Разверните локальную или облачную среду для тестирования (например, используя Docker или виртуальные машины). Настройте систему контроля версий (Git) и создайте репозиторий для тестового проекта.
2. Установите и сконфигурируйте выбранный инструмент CI/CD (Jenkins, GitLab CI/CD или GitHub Actions).
3. Разработайте простое веб‑ приложение (например, на Node.js, Python Flask или аналогичном легковесном фреймворке). Добавьте базовые эндпоинты (например, /health для проверки работоспособности и /version для отображения версии). Убедитесь, что приложение собирается и запускается локально.
4. Создайте конфигурационный файл CI/CD (.gitlab-ci.yml, .github/workflows/main.yml или Jenkinsfile). Настройте этап сборки: автоматическую компиляцию/сборку приложения, установку зависимостей. Обеспечьте генерацию артефакта (например, Docker‑ образ или архив с приложением).
5. Напишите набор юнит‑ тестов для приложения (не менее 3–5 тестов). Интегрируйте запуск тестов в конвейер: тесты должны вы-

полняться после сборки. Настройте условия продолжения конвейера только при успешном прохождении тестов.

6. Определите тестовую среду (отдельный контейнер, виртуальная машина или облачный инстанс). Автоматизируйте развёртывание артефакта в тестовую среду через CI/CD. Добавьте проверку работоспособности приложения после развёртывания (например, запрос к /health).

7. Настройте сценарий отката к предыдущей версии приложения в случае ошибки развёртывания или сбоя тестов. Протестируйте механизм отката, симулировав сбой (например, намеренно сломав тест или конфигурацию).

8. Подключите базовое логирование процесса сборки и развёртывания. Настройте вывод ключевых метрик (время сборки, статус тестов, результат развёртывания).Сохраняйте логи для анализа в случае сбоев.

9. Составьте краткую инструкцию по настройке и запуску конвейера (включая команды, конфигурационные файлы и ключевые шаги). Опишите возможные ошибки и способы их устранения. Укажите ограничения и предположения, сделанные в ходе работы.

10. Измерьте время выполнения каждого этапа конвейера (сборка, тесты, развёртывание). Предложите 2-3 способа оптимизации процесса (например, параллелизация тестов, кэширование зависимостей). Оцените риски текущего решения и предложите меры по их снижению.

Контрольные вопросы

1. Что понимается под непрерывной доставкой (Continuous Delivery, CD) в разработке ПО? В чём её ключевое отличие от непрерывной интеграции (CI) и непрерывного развёртывания (Continuous Deployment)?

2. Перечислите основные этапы конвейера непрерывной доставки (pipeline) и кратко опишите назначение каждого этапа (например, сборка, тестирование, развёртывание).

3. Какие инструменты автоматизации обычно используются для организации CD? Приведите 3–4 примера популярных решений (например, Jenkins, GitLab CI/CD, GitHub Actions) и укажите их основные функции.

4. Что такое среда развёртывания в контексте CD? Опишите различия между средами: разработка (development), тестирование (testing), предпродуктивная (staging) и продуктивная (production).

5. Какие типы тестирования должны выполняться на этапах CD для обеспечения качества обновлений? Перечислите 3–4 вида тестов (например, юнит‑тесты, интеграционные тесты) и поясните их назначение.

6. Что такое стратегии развёртывания в CD? Опишите 2–3 подхода (например, blue‑green deployment, rolling updates, feature toggles) и укажите сценарии их применения.

7. Как обеспечивается откат обновлений при возникновении ошибок в CD? Перечислите 2–3 метода отката и объясните, в каких случаях они предпочтительны.

8. Какие метрики и мониторинг необходимы для контроля процесса CD? Укажите 3–4 ключевых показателя (например, время развёртывания, процент успешных деплоев) и объясните их значимость.

9. Назовите основные риски и проблемы, возникающие при внедрении CD в информационных системах. Предложите 2–3 способа их минимизации (например, поэтапное внедрение, автоматизация тестирования).

Лабораторная работа № 7. Основы настройки CI/CD

Цель работы: сформировать практические навыки организации процесса непрерывной интеграции и доставки (Continuous Integration / Continuous Delivery, CI/CD) для программного продукта.

Теоретические положения

CI/CD (Continuous Integration / Continuous Delivery / Deployment) – это методология автоматизации процессов разработки, призванная ускорить выпуск ПО, повысить его качество и снизить риски при развёртывании. В основе подхода лежит идея непрерывной интеграции изменений в основной код и их автоматической доставки до конечных сред.

Настройка CI/CD начинается с выбора инструментальной платформы. Наиболее распространены: Jenkins (гибкий, но требующий ручной настройки), GitLab CI/CD (встроен в GitLab, использует YAML‑конфигурацию), GitHub Actions (интегрирован с GitHub), CircleCI (удобен для Docker‑проектов). Каждая система имеет свой формат конфигурационных файлов: Jenkins работает с Jenkinsfile, GitLab – с .gitlab-ci.yml, GitHub – с YAML‑файлами в директории .github/workflows/.

Ключевой элемент настройки – описание конвейера (pipeline, пайплайна) в конфигурационном файле. Пайплайн представляет собой последовательность этапов (stages), на каждом из которых выполняются определённые задачи (jobs). Типовые этапы: checkout (получение кода из репозитория), build (сборка проекта), test (автоматическое тестирование), analyze (статический анализ кода), package (упаковка артефактов), deploy (развёртывание в целевую среду). Для каждого этапа прописываются скрипты выполнения, условия запуска и зависимости от предыдущих шагов.

Рассмотрим пример конфигурационного файла для CI/CD в GitLab. Все настройки задаются в одном файле .gitlab-ci.yml в корне репозитория.

```
# Указываем образ Docker, в котором будут выполняться команды
image: ubuntu:latest

# Определяем этапы пайплайна
stages:
  - build
```

```
  - test
  - deploy

# Задача на этапе сборки
build-job:
  stage: build
  script:
    - echo "Начинаем сборку..."
    - mkdir build
    - echo "Сборка завершена" > build/status.txt

# Задача на этапе тестирования
test-job:
  stage: test
  script:
    - echo "Запускаем тесты..."
    - test -f build/status.txt && echo "Тест пройден" || exit 1

# Задача на этапе деплоя
deploy-job:
  stage: deploy
  script:
    - echo "Деплоим в прод..."
    - echo "Готово!"
  # Выполнять только для ветки main
  only:
        - main
```

Приведенный пример пайплайна создаёт папку build и файл со статусом, проверяет наличие файла из этапа сборки, имитирует деплой (выполняется только для ветки main).

Важный механизм – триггеры запуска пайплайна. Пайплайн может стартовать при коммите в определённую ветку, по расписанию (cron), вручную или при событиях типа pull request. Для гибкости используются переменные окружения: глобальные (задаются в секции variables), локальные для задач и секретные (для токенов и паролей, хранятся в настройках CI/CD‑системы).

При настройке необходимо предусмотреть передачу артефактов между этапами. Например, результаты сборки на этапе build должны быть доступны для тестирования на этапе test. Это реализуется через секцию artifacts с указанием путей к файлам. Для ускорения пайплай применяется кэширование зависимостей (например, node_modules для Node.js‑проектов), что сокращает время повторных сборок.

Особое внимание уделяется условиям выполнения задач. С помощью директив only/except можно ограничить запуск этапов определёнными ветками, путями к файлам или тегами. Для упорядочивания зависимостей используется ключ needs, указывающий, какие за-

дачи должны завершиться перед запуском текущей. Это позволяет выстраивать параллельное выполнение независимых задач и последовательное – зависимых.

При развёртывании критически важна безопасность. Секреты (пароли, API‑ключи) не хранятся в конфигурационных файлах – они задаются как переменные CI/CD в интерфейсе платформы (например, в Settings → CI/CD → Variables в GitLab). Для изоляции окружения рекомендуется использовать контейнеры (Docker), что обеспечивает воспроизводимость сборок на разных агентах.

Настройка агентов (runners/agents) – ещё один ключевой этап. Агенты выполняют задачи пайплайна и могут быть локальными или облачными. В GitLab, например, различают Shared Runner (общий, может быть перегружен) и Specific Runner (выделенный для проекта). При регистрации агента указываются URL CI/CD‑системы, токен авторизации и тип исполнителя (shell, docker).

Оптимизация пайплайна включает: разбиение длинных задач на подзадачи, минимизацию скриптов в конфигурации (перенос сложной логики в отдельные скрипты), использование артефактов и кэша, настройку уведомлений о сбоях. Для мониторинга результатов пайплайна анализируются логи выполнения, статусы этапов и метрики времени сборки.

Типичные ошибки при настройке: отсутствие обработки ошибок в скриптах, некорректные пути к артефактам, недостаточные права доступа для агентов, хранение секретов в открытом виде, слишком длительные пайплайны (оптимально – до 10 минут). Чтобы избежать этого, рекомендуется тестировать пайплайн поэтапно, начиная с простых задач (например, вывода версии ПО), и постепенно добавлять сложные шаги.

Успешная настройка CI/CD достигается, когда пайплайн запускается автоматически при изменениях, все этапы выполняются без ручного вмешательства, результаты тестов и сборки доступны команде, а процесс развёртывания занимает не более 5–10 минут. Дополнительно важно предусмотреть механизм отката при ошибках, чтобы минимизировать риски для рабочей среды.

Задание к лабораторной работе №7

1. Создайте новый публичный репозиторий на GitHub/GitLab с шаблоном простого приложения (например, статический сайт на HTML/CSS или микросервис на Node.js/Python). Добавьте в репози-

торий файл .gitignore для вашего языка/фреймворка. Настройте ветвление: создайте ветки develop и main. Добавьте файл README.md с описанием проекта и инструкциями по запуску.

2. Выберите систему CI/CD (GitHub Actions / GitLab CI). Создайте конфигурационный файл пайплайна: для GitHub Actions – .github/workflows/ci.yml, для GitLab CI – .gitlab-ci.yml. Настройте шаги пайплайна: checkout кода; установка зависимостей (например, npm install или pip install -r requirements.txt); запуск линтера (например, ESLint, Pylint). Запустите пайплайн вручную и убедитесь, что все шаги выполняются успешно.

3. Добавьте в проект простые юнит‑тесты. Модифицируйте CI‑пайплайн, добавив шаг запуска тестов. Настройте условия: пайплайн должен завершаться с ошибкой, если тестируемое программное обеспечение тесты не проходит; добавьте отчёт о покрытии тестами.

4. Выберите среду развёртывания (например, GitHub Pages для статического сайта или Docker‑контейнер). Добавьте в пайплайн шаг развёртывания: для GitHub Pages – сборка и публикация статических файлов, для Docker – сборка образа и отправка в реестр (например, Docker Hub). Настройте триггеры развёртывания: автоматическое развёртывание в тестовую среду при коммите в develop, ручное развёртывание при слиянии веток в main.

5. Добавьте в пайплайн отправку уведомлений на почту при успешном/ошибочном выполнении или в мессенджер (например, Slack или Telegram) при развёртывании. Настройте генерацию отчёта о сборке (например, время выполнения, покрытие тестами).

Контрольные вопросы

1. Какие ключевые секции обязательно должны присутствовать в файле конфигурации CI/CD? Приведите пример минимальной рабочей конфигурации.
2. Как настроить триггеры пайплайна так, чтобы сборка запускалась только при коммитах в ветку main и при создании pull‑request’ов? Покажите синтаксис для выбранного CI‑сервиса.
3. Как правильно настроить переменные окружения в CI/CD для разных стадий (разработка/тестирование)? Приведите пример безопасного способа передачи секретных ключей.

4. Как сконфигурировать параллельное выполнение тестов в пайплайне?

5. Опишите процесс настройки развертывания на сервер через CI/CD: какие шаги включить, как обеспечить безопасное подключение (SSH/API‑ ключи), как обработать откат при ошибке.

6. Как настроить условные этапы в пайплайне (например, развертывание только при успешном прохождении всех тестов и только для ветки main)? Приведите пример синтаксиса.

7. Как организовать кэширование зависимостей в CI/CD для ускорения сборки?

8. Как настроить уведомления о статусе пайплайна (успех/не успех) в Telegram? Перечислите необходимые шаги и покажите пример конфигурации.

9. Как добавить этап статического анализа кода (linting) в пайплайн?

Лабораторная работа № 8. Сравнительный анализ версии «до» и «после» обновления

Цель работы: провести сравнительный анализ версии информационной системы «до» и «после» обновления для оценки эффективности внесённых изменений, выявления улучшений/ухудшений ключевых характеристик системы и формирования обоснованных выводов о целесообразности проведённого обновления.

Теоретические положения

При проведении сравнительного анализа версий информационной системы «до» и «после» обновления необходимо опираться на фундаментальные понятия теории информационных систем и методологии их сопровождения.

Информационная система (ИС) представляет собой организационно‑техническую систему, предназначенную для сбора, хранения, обработки и предоставления информации в целях поддержки принятия решений и автоматизации бизнес‑процессов. Её ключевая характеристика – интеграция информационных, технических, программных и кадровых ресурсов для достижения конкретных функциональных целей организации.

Жизненный цикл ИС традиционно описывается через последовательность фаз: анализ требований, проектирование, реализация, внедрение, эксплуатация и сопровождение, вывод из эксплуатации. Обновление системы относится к фазе сопровождения и является плановой или вынужденной модификацией существующей ИС с целью повышения её эффективности, расширения функциональности или адаптации к изменившимся условиям эксплуатации. При этом обновление может затрагивать отдельные компоненты (программное обеспечение, аппаратную платформу, базы данных) либо всю систему в комплексе.

Для корректного сравнительного анализа принципиально важно зафиксировать исходное состояние системы («до обновления») по ряду ключевых параметров. К таким параметрам относятся: производительность (время отклика на запросы, пропускная способность, скорость обработки транзакций), надёжность (среднее время наработки на отказ, частота сбоев, время восстановления после сбоев), функциональность (набор реализуемых бизнес‑процессов, полнота выпол-

нения требований пользователей), удобство использования (юзабилити интерфейса, время выполнения типовых операций, количество ошибок пользователя), безопасность (уровень защиты данных, устойчивость к внешним угрозам, соответствие стандартам ИБ), масштабируемость (способность системы адаптироваться к росту нагрузки), совместимость (интеграция с другими системами и сервисами), сопровождаемость (простота внесения изменений, наличие актуальной документации).

Сбор данных для сравнения осуществляется посредством комбинации методов. Инструментальные замеры позволяют объективно зафиксировать количественные показатели производительности и надёжности: логирование времени выполнения операций, мониторинг загрузки процессора и памяти, анализ сетевого трафика. Тестирование (функциональное, нагрузочное, стресс‑тестирование) выявляет изменения в работоспособности системы и её устойчивости к пиковым нагрузкам. Анкетирование и интервьюирование пользователей дают качественную оценку удобства интерфейса и общей удовлетворённости работой системы. Анализ логов и отчётов о сбоях позволяет сопоставить частоту и характер ошибок в обеих версиях.

При обработке собранных данных применяются методы сравнительного анализа: построение таблиц соответствия параметров «до» и «после», расчёт относительных изменений (в процентах или абсолютных значениях), статистическая обработка результатов (средние значения, дисперсия, доверительные интервалы). Для визуализации динамики показателей используются графики и диаграммы, отражающие тренды изменений по ключевым метрикам.

Интерпретация результатов сравнительного анализа базируется на заранее определённых критериях успешности обновления. К положительным индикаторам относятся: сокращение времени отклика системы, увеличение пропускной способности, снижение частоты критических сбоев, расширение функционала без потери стабильности, повышение удовлетворённости пользователей. Отрицательными индикаторами считаются: ухудшение производительности, рост числа ошибок, снижение удобства интерфейса, нарушение совместимости с существующими сервисами.

В случае выявления негативных последствий обновления проводится дополнительный анализ причин (например, неоптимальная настройка параметров, конфликты компонентов, недостаточное тестирование). На основе полученных данных формируются рекомен-

дации по корректировке настроек, доработке функционала или откату к предыдущей версии. Таким образом, сравнительный анализ не только оценивает эффективность проведённого обновления, но и служит основой для принятия управленческих решений по дальнейшей эксплуатации и развитию информационной системы.

Важным аспектом является документирование результатов анализа: формируется отчёт, содержащий исходные данные, методику сравнения, полученные показатели, выводы и рекомендации. Это обеспечивает прозрачность процесса, позволяет отслеживать динамику развития системы и накапливать базу знаний для будущих обновлений.

Задание к лабораторной работе №8

1. Определите и документируйте ключевые параметры ИС до обновления: перечень реализуемых функциональных модулей и их версий; аппаратные характеристики сервера/рабочей станции (CPU, RAM, объём дискового пространства); параметры СУБД (версия, настройки, объём базы данных); текущие показатели производительности (среднее время отклика, пропускная способность за 1 ч при типовой нагрузке).
2. Составьте чек‑лист тестовых сценариев для проверки базовой функциональности (не менее 10 сценариев, охватывающих ключевые бизнес‑процессы). Для каждого сценария зафиксируйте время выполнения операции (в секундах); загрузку CPU и RAM (в %); количество обработанных записей/запросов за единицу времени.
3. Выполните обновление ИС согласно инструкции разработчика (зафиксируйте версию ПО «после»). Документируйте все этапы процесса: время начала и завершения обновления; возникшие ошибки/предупреждения и способы их устранения; изменения в конфигурации системы (новые параметры, удалённые/добавленные компоненты).
4. Запустите тестовые сценарии после обновления. Для каждого сценария зафиксируйте указанные в п.2 характеристики.
5. Проведите нагрузочное тестирование (имитация работы 10/20/50 пользователей одновременно). Запишите максимальное число одновременных сеансов без сбоев; время отклика при пиковой нагрузке; количество ошибок тайм‑аута.
6. Проверьте работоспособность всех функциональных модулей после обновления. Отметьте появившиеся новые функции (опишите

их назначение и интерфейс); удалённые или изменённые функции (укажите, как это влияет на бизнес‑процессы); случаи некорректной работы (с примерами входных данных и результатов). Оцените совместимость с внешними системами (API, файлы импорта/экспорта).

7. Проведите тестирование интерфейса с участием 3–5 пользователей. Для каждого участника предложите выполнить 5 типовых операций (например, создание заказа, поиск записи, формирование отчёта); зафиксируйте время выполнения и количество ошибок; соберите субъективные оценки (удобство навигации, понятность подсказок, эстетика). Сравните результаты до и после обновления по шкале от 1 до 10 (где 10 – максимальное удобство).

8. Проведите анализ надёжности и безопасности. Сравните частоту сбоев (количество сбоев за 24 ч работы); время восстановления после сбоев; наличие новых уязвимостей, проверьте изменения в механизмах аутентификации и авторизации (новые роли, права доступа).

9. Составьте сравнительные таблицы по каждому параметру (производительность, функциональность, юзабилити, надёжность). Рассчитайте процентные изменения ключевых показателей (например, «время отклика сократилось на 15 %»).

10. Сформулируйте заключение о целесообразности обновления на основе положительных изменений (что улучшилось?), негативных последствий (что ухудшилось?), соотношения затрат на обновление и полученного эффекта. Предложите меры по оптимизации системы (например, донастройка параметров, обучение пользователей, доработка интерфейса). Определите критерии для мониторинга системы в течение 1 месяца после обновления.

Контрольные вопросы

1. Что понимается под «версией информационной системы»?
2. Каковы основные цели сравнительного анализа версий ИС «до» и «после» обновления?
3. В чём заключается ценность отслеживания изменений между версиями ИС для заказчика и пользователей?
4. Какие ключевые этапы включает процесс сравнительного анализа версий ИС?
5. По каким критериям целесообразно сравнивать версии ИС? Приведите не менее 5 примеров.

6. Как выбрать релевантные метрики для оценки производительности ИС до и после обновления?
7. Какие качественные показатели (юзабилити, удобство интерфейса и т. п.) стоит учитывать при сравнении версий?
8. Как количественно оценить изменение надёжности системы после обновления?

Лабораторная работа № 9. Проверка совместимости компонентов при миграции

Цель работы: приобрести практические навыки анализа и обеспечения совместимости компонентов информационной системы в процессе её миграции на новую платформу, выявить потенциальные конфликты и разработать меры по их устранению для гарантированного сохранения функциональности и производительности системы.

Теоретические положения

В основе миграции лежит необходимость обеспечения непрерывности бизнес-процессов при обновлении технологической базы – замене оборудования, переходе на новые версии программного обеспечения (ПО), смене облачного провайдера или консолидации разнородных систем. При этом важнейшим условием успеха выступает совместимость всех компонентов ИС как между собой, так и с целевой инфраструктурой.

Совместимость компонентов ИС понимается как способность программных и аппаратных элементов взаимодействовать без конфликтов, сохраняя заданные функциональные и эксплуатационные характеристики. Она затрагивает несколько уровней: аппаратный (совместимость с процессорами, памятью, накопителями, сетевыми адаптерами), системный (поддержка ОС, драйверов, системных библиотек), прикладного ПО (взаимодействие приложений, СУБД, middleware), данных (сохранение структуры, кодировок, ссылок) и сетевой (протоколы, порты, политики безопасности). Нарушение совместимости на любом из уровней может привести к сбоям, потере данных, снижению производительности или полной неработоспособности системы после миграции.

Процесс миграции требует предварительного анализа текущей архитектуры ИС. Необходимо проанализировать состав компонентов, их версий, зависимости, конфигурационные параметры и режимы работы. На основе этого формируется матрица совместимости, где для каждого элемента указываются требования к целевой среде и возможные риски.

Пример матрицы совместимости приведен в таблице 1.

Таблица 1 – Матрица совместимости

Компонент	**Требования к целевой среде**	**Возможные риски**	**Меры снижения риска**
СУБД Postgre SQL 14	ОС: Linux (RHEL 8+/Ubuntu 20.04+) ОЗУ: ≥ 8 ГБ Диск: ≥ 100 ГБ (SSD) Версия PostgreSQL: 14.x Порты: 5432/TCP открыт	Несовместимость с версией библиотеки libssl Недостаточный объём ОЗУ → торможение запросов Различия в настройках postgresql.conf между средами Потеря данных при некорректном восстановлении	Проверить версии зависимостей заранее Выделить резервный запас ОЗУ (+20 %) Использовать pg_dump/pg_restore с проверкой целостности Протестировать восстановление копии
Веб-сервер Nginx 1.22	ОС: Linux Порт 80/443 свободен Сертификаты SSL/TLS в формате PEM Конфигурационные файлы в /etc/nginx/	Конфликты портов с другим ПО Ошибки в конфигурации после миграции Несовместимость версий OpenSSL Потеря настроек виртуальных хостов	Проверить занятость портов до развёртывания Версионировать конфигурацию (Git) Протестировать конфигурацию командой nginx –t Обновить OpenSSL
Приложение на Python 3.9	Python 3.9+ Виртуальное окружение (venv) Зависимости из requirements.txt Доступ к СУБД и API внешних сервисов	Различия в версиях библиотек между средами Отсутствие переменных окружения Таймауты при обращении к удалённым сервисам Проблемы с кодировкой (UTF-8 vs. locale)	Использовать pip freeze > requirements.txt Настроить .env-файл с параметрами Добавить повторные обращения к удаленным сервисам для сетевых вызовов Проверить региональные и языковые настройки на целевом сервере
Сервер приложений Tomcat 9	Java 11+ (OpenJDK/Oracle JDK) ОЗУ: ≥ 4 ГБ Порт 8080 свободен	Несовместимость версии Java Переполнение памяти (OOM) Конфликты портов Ошибки развёртывания	Проверить версию Java Настроить Xmx в JAVA_OPTS Освободить порт 8080 до старта Проверить права на ди-

Компонент	Требования к целевой среде	Возможные риски	Меры снижения риска
	WAR-файлы приложения в webapps/	из-за отсутствия прав	ректорию webapps
Клиентское ПО (Windows-приложение)	ОС: Windows 10/11 (64-bit) .NET Framework 4.8+ Минимальные разрешения пользователя Доступ к серверу по HTTPS	Отсутствие .NET Framework на целевых ПК Блокировка антивирусной программой Проблемы с сертификатами сервера Несовместимость с политиками безопасности домена	Предварительно установить .NET Framework Добавить приложение в исключения для антивирусной программы Импортировать корневой сертификат центра сертификации Протестировать на тестовом ПК из домена
Система резервного копирования (Bacula)	Клиент Bacula FC 10+ на всех узлах Сеть: доступ к серверу бэкапов (порт 9102/TCP) Достаточно места в целевом хранилище	Таймауты из-за медленно работающей сети Ошибки аутентификации между клиентом и сервером Нехватка места в хранилище Конфликты версий клиента/сервера	Проверить связность сети и задержку Сверить версии клиента и сервера Проверить наличие свободного места Провести тестирование процесса восстановления из резервной копии
Сервис аутентификации (LDAP)	Порт 389/636 открыт Сертификаты для LDAPS Схема каталога совместима с приложением Права чтения для сервисных учётных записей	Блокировка портов межсетевым экраном Ошибки верификации сертификата Изменения в DN пользователей Задержки ответа сервера	Проверить правила блокировки портов Обновить сертификаты Протестировать запросы с помощью инструмента командной строки ldapsearch Настроить таймауты на клиенте

Важную роль играет инвентаризация программного обеспечения – выявление устаревших или неподдерживаемых версий, сторонних библиотек, пользовательских скриптов, которые могут не работать в новой среде.

Аналогично анализируются аппаратные ресурсы: достаточность вычислительной мощности, объёма памяти и дискового пространства, соответствие интерфейсов и стандартов.

При планировании миграции различают два основных подхода: «чистый перенос» (lift-and-shift), при котором система копируется в новую среду с минимальным изменением конфигурации, и рефакторинг (перепроектирование), предполагающий адаптацию компонентов под особенности целевой платформы. Первый метод быстрее, но выше риск несовместимости; второй требует больше времени и ресурсов, но обеспечивает лучшую интеграцию. В обоих случаях необходим этап предмиграционного тестирования, включающий проверку:

- соответствия системных требований (ОС, версии ПО, разрядность, зависимости);
- работоспособности драйверов и аппаратных абстракций;
- целостности данных при конвертации форматов и кодировок;
- сетевых настроек (DNS, маршрутизация, файрволы);
- прав доступа и политик безопасности;
- производительности в условиях новой инфраструктуры.

Для выявления конфликтов используются инструменты статического анализа (проверка манифестов, зависимостей), динамического тестирования (запуск компонентов в тестовой среде), а также средства мониторинга ресурсов (CPU, память, диск, сеть). Особое внимание уделяется обратной совместимости: новые версии ПО должны поддерживать интерфейсы и форматы, используемые в старой системе, чтобы избежать разрыва интеграционных цепочек.

В ходе миграции возможны следующие типы несовместимости:

- версия ПО – новая ОС или СУБД не поддерживает старую версию приложения;
- архитектурные различия – например, переход с 32- на 64-битную систему, смена процессора (x86 → ARM);
- зависимости библиотек – отсутствие нужных версий DLL, .so, пакетов npm/pip;
- изменения API – нарушение контрактов между сервисами при обновлении интерфейсов;
- форматы данных – несовпадение кодировок, схем БД, структур файлов;
- сетевые ограничения – блокировка портов, изменение DNS, политики SSL/TLS;

- права доступа – различия в моделях авторизации.

Для устранения несовместимости применяются следующие методы:

- обновление компонентов до совместимых версий;
- использование прослоек совместимости (например, WoW64 для 32-битных приложений в 64-битной Windows);
- контейнеризация (Docker, Kubernetes) для изоляции зависимостей;
- виртуализация для эмуляции старой среды;
- миграция данных с преобразованием форматов;
- рефакторинг кода с заменой устаревших вызовов и библиотек;
- настройка групповых политик и прав доступа в целевой среде.

После переноса проводится постмиграционное тестирование, включающее проверку функциональности критических бизнес-процессов; нагрузочное тестирование для оценки производительности; аудит безопасности (проверка прав, шифрование, логи доступа); мониторинг стабильности (аварии, утечки памяти, таймауты); сравнение метрик до и после миграции (время отклика, пропускная способность).

Важным элементом является план отката – процедура возврата к исходной системе в случае неустранимых конфликтов. Он предусматривает сохранение резервных копий данных, конфигураций и образов систем, а также проверку работоспособности механизмов восстановления.

В плане отката необходимо прописать критерии срабатывания, ответственные лица, перечень предварительных проверок (например, доступность резервных копий), детальные инструкции для каждого этапа, процедуру верификации результата.

Пример фрагмента плана отката:

```
3. Предварительные проверки:
- Убедиться, что бэкап от 2025-11-26 23:00 доступен по пути //backup-
server/prod/db_20251126.zip.
- Проверить контрольную сумму: MD5 = a1b2c3d4e5f6.

4. Шаги отката:
4.1. Остановить приложение:
   команду: sudo systemctl stop app-service
   время: 2 мин
4.2. Восстановить базу данных:
   команду: mysql -u root -p < restore_backup.sql
   время: 15 мин
4.3. Перезапустить сервисы:
   команду: sudo systemctl start app-service
   время: 3 мин
```

```
5. Верификация:
- Открыть страницу https://example.com – должен отображаться старый интерфейс.
      - Проверить логи: tail -f /var/log/app.log – нет ошибок «Database mismatch».
```

План отката необходимо протестировать до реального изменения в тестовой среде.

Задание к лабораторной работе №9

1. Проведите анализ исходной информационной системы: составьте перечень всех компонентов ИС (ПО, аппаратное обеспечение, сетевые службы, базы данных), для каждого компонента укажите версию/модель; операционную систему, зависимости от других компонентов, критичность для функционирования системы. Постройте схему взаимосвязей компонентов (можно в виде графа или диаграммы).
2. Определите требования для целевой платформы: опишите целевую среду миграции (серверная ОС, СУБД, сетевое оборудование), выпишите официальные требования вендоров для каждого компонента ИС к целевой платформе, сравните характеристики исходной и целевой инфраструктуры (процессор, ОЗУ, дисковое пространство, пропускная способность сети).
3. Проверьте совместимость ПО. Для каждого программного компонента проверьте наличие сертифицированных версий для целевой ОС, определите совместимость с новой версией СУБД (если меняется), проанализируйте изменения в API/протоколах взаимодействия. Составьте таблицу результатов. Проверьте аппаратную совместимость: сопоставьте требования к оборудованию с характеристиками целевой инфраструктуры, оцените достаточность вычислительных ресурсов, совместимость драйверов, поддержку периферийных устройств. Выделите критические несоответствия.
4. Проведите тестирование интеграционных сценариев. Для этого разработайте 3–5 тестовых сценариев, покрывающих ключевые бизнес-процессы. Для каждого сценария опишите последовательность действий, укажите ожидаемый результат, зафиксируйте фактические результаты в тестовой среде. Оформите результаты в виде таблицы.
5. Составьте список выявленных проблем совместимости (не менее 5). Для каждой проблемы оцените степень влияния на работоспособность ИС, предложите 1–2 варианта решения (замена компонента, настройка, доработка), укажите ориентировочные сроки реализации.

Контрольные вопросы

1. Какие основные типы несовместимости компонентов могут возникнуть при миграции ИС и чем они обусловлены? Перечислите не менее трёх типов с краткими примерами.
2. Что такое «матрица совместимости» в контексте миграции ИС? Опишите её структуру и назначение, приведите упрощённый пример в виде таблицы.
3. Какие ключевые этапы включает процесс проверки совместимости компонентов при миграции? Перечислите их в правильной последовательности и кратко поясните содержание каждого.
4. Какие инструменты и методы (автоматизированные и ручные) можно использовать для проверки совместимости ПО при миграции? Приведите по 2–3 примера каждого типа.
5. В чём заключаются особенности проверки совместимости аппаратных компонентов при миграции ИС? Перечислите не менее четырёх ключевых параметров, требующих проверки.
6. Как влияет версия ОС на совместимость мигрируемого ПО? Опишите не менее трёх конкретных сценариев проблем, связанных с версией ОС, и способы их решения.
7. Что такое «среда тестирования совместимости» и какие требования к ней предъявляются? Перечислите 4–5 ключевых требований и поясните их значимость.
8. Какие документы и отчёты должны быть подготовлены по итогам проверки совместимости компонентов? Опишите структуру и основное содержание каждого документа (не менее трёх).
9. Как оценить риски, связанные с выявленной несовместимостью компонентов? Опишите методику оценки (включая параметры и шкалу) и приведите пример расчёта для одной выявленной проблемы.
10. Какие стратегии можно применить при обнаружении критической несовместимости компонента, не имеющего прямых аналогов? Перечислите и кратко опишите не менее трёх стратегий с указанием их преимуществ и недостатков.

Лабораторная работа № 10.
Проведение анализа архитектуры учебной ИС

Цель работы: приобрести практические навыки системного анализа архитектур информационных систем на примере учебной ИС, сформировать умение выявлять и оценивать ключевые архитектурные решения, а также обосновывать их соответствие функциональным и нефункциональным требованиям.

Теоретические положения

Архитектура информационной системы представляет собой фундаментальную организацию её компонентов, определяющую принципы их взаимодействия, структуру данных и способы реализации функциональных требований. При анализе архитектуры учебной ИС важно понимать, что она не сводится лишь к набору технических решений – это целостная модель, обеспечивающая баланс между функциональностью, производительностью, масштабируемостью и удобством сопровождения. В основе архитектурного анализа лежит представление системы как совокупности взаимосвязанных элементов, каждый из которых выполняет определённую роль и взаимодействует с другими через чётко заданные интерфейсы.

Ключевым аспектом является многоуровневое описание архитектуры, включающее логический, физический и информационный уровни. Логический уровень отражает функциональную структуру системы – какие задачи она решает и как организованы процессы обработки данных. Физический уровень определяет конкретное воплощение системы: аппаратные средства, программные платформы, сетевые компоненты и их размещение. Информационный уровень задаёт модель данных – структуру хранилищ, форматы обмена, правила валидации и трансформации информации. Все три уровня взаимосвязаны: изменения на одном из них неизбежно влияют на остальные, поэтому анализ должен охватывать их комплексно.

При оценке архитектуры необходимо учитывать базовые проектные характеристики, такие как модульность, информационная закрытость, связность и сцепление. Модульность позволяет разбивать систему на независимые блоки, упрощая разработку и тестирование. Информационная закрытость гарантирует, что внутренние механизмы модуля недоступны извне, снижая риски непреднамеренных из-

менений. Связность отражает степень функциональной целостности модуля – высокая связность считается признаком качественного проектирования. Сцепление, напротив, показывает уровень зависимости между модулями: низкое сцепление предпочтительно, так как оно уменьшает влияние изменений в одном компоненте на другие.

В практике проектирования ИС сложились устойчивые архитектурные стили, каждый из которых имеет свои преимущества и ограничения. Монолитная архитектура объединяет все компоненты в единое приложение, что упрощает развёртывание, но затрудняет масштабирование. Клиент-серверная модель разделяет функционал на уровни (клиентский интерфейс, бизнес-логика, хранилище данных), обеспечивая чёткое распределение ответственности, но создавая зависимость от серверных мощностей. Микросервисный подход предполагает декомпозицию системы на независимые сервисы с собственными базами данных, повышая гибкость и устойчивость, но увеличивая сложность интеграции. Файл-серверная архитектура, хотя и проста в реализации, уступает в производительности и безопасности из-за централизованного хранения данных при распределённой обработке.

Анализ архитектуры требует оценки как функциональных, так и нефункциональных требований. Функциональные аспекты касаются корректности реализации бизнес-процессов и взаимодействия компонентов. Нефункциональные включают производительность (время отклика, пропускная способность), масштабируемость (способность выдерживать рост нагрузки), надёжность (устойчивость к сбоям), безопасность (контроль доступа, защита данных), сопровождаемость (простота внесения изменений) и экономическую эффективность (затраты на инфраструктуру). Эти параметры определяют, насколько архитектура соответствует целям системы и условиям её эксплуатации.

Для проведения анализа применяются стандартизированные методы и инструменты. Диаграммы UML (компонентная, развёртывания, последовательностей) позволяют визуализировать структуру и поведение системы. Метрики сложности (количество модулей, коэффициенты связности/сцепления, глубина иерархии) дают количественную оценку архитектурных решений. Анализ рисков направлен на выявление уязвимых мест – например, единых точек отказа или избыточных зависимостей. Важную роль играет изучение документации: технических спецификаций, руководств администратора и поль-

зователя, которые раскрывают исходные требования и принятые проектные решения.

Результатом анализа должно стать целостное представление об архитектуре ИС, включая описание её компонентов, их ролей и взаимосвязей. На основе этого формируются выводы о сильных сторонах решений (например, чёткое разделение уровней или высокая отказоустойчивость) и проблемных зонах (избыточная сложность, узкие места производительности). Заключительным этапом является разработка рекомендаций по оптимизации – например, рефакторинг модулей для снижения сцепления, переход на более масштабируемый архитектурный стиль или усиление механизмов безопасности. Такой подход обеспечивает не только понимание текущего состояния системы, но и обоснованные предложения по её развитию с учётом современных требований к информационным технологиям.

Рассмотрим учебную ИС для автоматизации работы кафедры вуза: учёта студентов, расписания, успеваемости, формирования отчётности.

1. Общее описание системы

Система представляет собой веб-приложение с трёхзвенной архитектурой: клиентский уровень – браузерный интерфейс для преподавателей, сотрудников деканата и студентов; серверный уровень – веб-сервер и сервер приложений, реализующий бизнес-логику; уровень данных – реляционная СУБД для хранения информации.

Основные функциональные модули:

- управление контингентом (студенты, группы, курсы);
- расписание занятий и экзаменов;
- учёт успеваемости;
- отчётность;
- личный кабинет пользователя с ролевой моделью доступа.

2. Архитектура ИС

Выбрана клиент-серверная архитектура с тонким клиентом (браузер). Обоснование: простота развёртывания (не требует установки клиентского ПО), централизованное обновление (изменения на сервере сразу доступны всем пользователям), унифицированный интерфейс (одинаковый вид для разных устройств через адаптивный дизайн), масштабируемость (возможность наращивания серверных ресурсов при росте нагрузки).

3. Анализ ключевых компонентов

Клиентская часть реализована на HTML/CSS/JavaScript с использованием фреймворка React. Преимущества: интерактивность, быстрое обновление интерфейса без полной перезагрузки страницы. Ограничения: зависимость от качества интернет-соединения, нагрузка на клиентское устройство при сложных отчётах.

Сервер приложений реализован на платформе Node.js + Express.js. Преимущества: асинхронная модель обработки запросов, высокая производительность для I/O-нагруженных задач. Ограничения: необходимость тщательного управления памятью при росте числа одновременных сессий.

СУБД – PostgreSQL. Все таблицы, входящие в состав БД («Студенты», «Группы», «Дисциплины», «Ведомость», «Расписание») нормализованы. Созданы кластеризованные индексы по ключевым полям (КодСтудента, КодГруппы, КодДисциплины, КодВедомости, КодРасписания) и некластеризованные индексы для оптимизации запросов. Периодичность резервного копирования БД – ежедневно.

4. Оценка проектных характеристик

- Модульность: система разделена на функциональные модули с чёткими интерфейсами. Например, модуль расписания не зависит от логики расчёта стипендий.
- Информационная закрытость: бизнес-логика недоступна клиенту – все операции проходят через API сервера.
- Связность: высокая внутри модулей (например, весь функционал успеваемости в одном сервисе), низкая между модулями.
- Сцепление: минимальное – модули общаются через REST API, изменения в одном минимально влияют на другие.

5. Соответствие требованиям

Все заявленные функциональные требования выполнены: реализованы сценарии зачисления студентов в группу, составление расписания, выставление оценок; реализована поддержка экспорта отчётов в PDF/Excel; реализован мультипользовательский доступ с ролями «преподаватель», «директор», «студент».

Все заявленные нефункциональные требования выполнены: время отклика интерфейса < 2 с при нагрузке до 100 пользователей; время восстановления БД после сбоя < 30 мин; высокая безопасность за счет использования HTTPS, хеширования паролей, проверки прав доступа к данным на каждом этапе; наличие документации API, комментариев в коде, логирования ошибок.

6. Выявленные проблемы:

1. Точка отказа – один сервер приложений, при сбое на нем недоступна вся ИС.

2. Ограниченная масштабируемость БД – при росте числа студентов (> 5 000) возможны задержки в формировании сложных отчётов.

3. Отсутствие кэширования, за счет чего повторные запросы к БД снижают производительность ИС.

4. Жёсткая привязка к СУБД PostgreSQL – сложно перейти на другую СУБД без переработки программного кода.

7. Рекомендации по оптимизации

1. Внедрить балансировку нагрузки между несколькими серверами приложений.

2. Добавить кэш (например, Redis) для часто запрашиваемых данных (расписание, списки групп).

3. Перевести отчёты на асинхронную обработку – формировать их в фоновом режиме и уведомлять об окончании их формирования пользователя.

4. Использовать микросервисный подход для критически важных модулей (например, отдельный сервис для расчёта стипендий).

5. Настроить репликацию БД для чтения, чтобы разгрузить основной сервер.

6. Добавить мониторинг для отслеживания производительности.

8. Итоговые выводы

Архитектура системы в целом соответствует задачам учебного заведения: обеспечивает базовый функционал для учёта студентов и успеваемости; проста в освоении для пользователей; допускает постепенное расширение (новые модули, интеграции). Ключевые риски связаны с масштабируемостью и отказоустойчивостью. Реализация предложенных рекомендаций позволит повысить доступность системы, сократить время отклика при пиковых нагрузках, упростить сопровождение за счёт модульности.

Задание к лабораторной работе №10

Провести системный анализ архитектуры выбранной информационной системы.

Контрольные вопросы

1. Что понимается под архитектурой информационной системы (ИС)?

2. Каковы основные цели проведения анализа архитектуры ИС?
3. В чём заключается ценность анализа архитектуры для жизненного цикла ИС?
4. Перечислите основные артефакты, которые должны быть получены в результате анализа архитектуры.
5. Что означает принцип модульности и почему он важен?
6. В чём разница между слабой и сильной связностью модулей?
7. Что такое сцепление и какие его типы считаются предпочтительными?

Лабораторная работа № 11.
Сбор отзывов и предложений по улучшению ИС

Цель работы: сформировать практические навыки систематического сбора, анализа и структурирования обратной связи от пользователей информационной системы для выявления проблемных зон и выработки обоснованных предложений по её совершенствованию.

Теоретические положения

Обратная связь от пользователя ИС – это ключевой механизм совершенствования информационных систем. Обратная связь представляет собой поток информации от конечных пользователей к разработчикам и администраторам системы, содержащий оценки удобства использования, описание проблемных ситуаций, предложения по новым функциям и замечания к существующим. Систематический сбор отзывов и предложений по улучшению ИС позволяет выявлять «узкие места» ИС и формировать обоснованные предложения по оптимизации, что в конечном итоге повышает эффективность бизнес‑процессов и удовлетворённость пользователей.

Эффективность сбора обратной связи определяется соблюдением ряда основополагающих принципов: системности (регулярный, а не эпизодический сбор данных), целеполагания (чёткое определение задач исследования), репрезентативности (охват различных групп пользователей), анонимности (при необходимости обеспечения откровенности ответов) и двунаправленности (фиксация не только проблем, но и позитивных аспектов работы системы).

В практике сбора отзывов применяются различные методы, каждый из которых обладает своими особенностями.

Анкетирование обеспечивает массовость и стандартизацию данных, хотя может страдать от поверхностности ответов и низкого отклика; оптимальным считается использование коротких анкет (5–7 вопросов) со шкальными оценками.

Интервью позволяют глубоко проработать проблемы благодаря возможности уточняющих вопросов, но требуют значительных временных затрат и могут быть подвержены субъективности.

Анализ логов и метрик даёт объективные данные о реальном поведении пользователей (время выполнения операций, частота оши-

бок, пути навигации), хотя интерпретация причин действий может быть затруднена.

Юзабилити‑тестирование с наблюдением за действиями пользователя и записью экрана помогает визуализировать проблемы интерфейса, но проводится в искусственно смоделированных условиях.

Форумы и чаты поддержки предоставляют естественные высказывания пользователей, однако требуют тщательной тематической категоризации поступающей информации.

Качество собранных отзывов определяется их конкретностью (чёткое описание проблемы вместо общих фраз), конструктивностью (наличие предложений по решению), актуальностью (соответствие текущим версиям ИС) и достоверностью (подтверждение примерами или скриншотами). Обработка полученной информации осуществляется поэтапно: сначала происходит сбор и фиксация всех поступающих данных, затем – категоризация по тематическим блокам (интерфейс, производительность, функционал), после чего следует приоритезация на основе частоты упоминаний, влияния на бизнес‑процессы и сложности исправления. На этапе анализа выявляются корневые причины проблем, а на завершающем этапе формируются конкретные предложения по улучшению системы.

Для эффективной работы с обратной связью используются специализированные инструменты: сервисы для проведения опросов (Google Forms, Яндекс Формы), системы веб‑аналитики (Яндекс Метрика, Google Analytics, тепловые карты), платформы управления задачами (Jira, Trello, YouTrack) и инструменты текстового анализа на базе NLP‑библиотек для автоматической классификации отзывов. При этом важно учитывать типичные проблемы: низкий отклик (решается упрощением анкет и мотивационными механизмами), субъективность оценок (компенсируется кросс‑проверкой данных), информационный шум (фильтруется автоматическими методами) и игнорирование позитивных отзывов (что лишает возможности масштабировать успешные решения).

Итоговый результат работы с обратной связью оформляется в виде аналитического документа, включающего сводную таблицу отзывов с категоризацией, диаграмму приоритетности выявленных проблем, перечень предложений по улучшению с оценкой трудозатрат и рекомендации по внедрению изменений. Таким образом, системный подход к сбору и анализу пользовательской обратной связи становится неотъемлемым элементом жизненного цикла ИС, позво-

ляя минимизировать риски отторжения системы пользователями, оптимизировать затраты на доработку и обеспечивать устойчивое развитие информационной системы в соответствии с реальными потребностями её пользователей.

Рассмотрим пример аналитического отчета по результатам сбора отзывов о работе ИС «Учебный портал».

Аналитический отчёт по результатам сбора отзывов о работе ИС «Учебный портал»

Период сбора данных: 01.11.2025 – 30.11.2025

Количество респондентов: 147 пользователей (преподаватели – 45 %, студенты – 55 %)

Методы сбора: анкетирование (80 %), интервью (15 %), анализ логов (5 %)

1. Сводная таблица отзывов с категоризацией (таблица 2)

Таблица 2 – Сводная таблица отзывов с категоризацией

№	Отзыв/проблема	Категория	Частота упоминаний	Степень влияния (1–5)
1	Долгое ожидание при загрузке учебных материалов	Производительность	68	5
2	Сложность поиска нужных курсов в каталоге	Юзабилити	54	4
3	Нет мобильного приложения	Функциональность	49	5
4	Неудобный редактор тестов для преподавателей	Интерфейс	37	4
5	Частые сбои при отправке домашних работ	Надёжность	32	5
6	Отсутствие уведомлений о новых заданиях	Коммуникация	29	3
7	Неинтуитивный процесс регистрации новых пользователей	Юзабилити	21	3
8	Нет возможности скачать материалы в PDF	Функциональность	18	2
9	Медленная работа чата поддержки	Производительность	15	3
10	Нехватка отчётов по успеваемости	Аналитика	12	4

Условные обозначения категорий:

- Производительность – скорость работы системы;
- Юзабилити – удобство использования;
- Функциональность – недостающие возможности;
- Интерфейс – визуальное оформление и элементы управления;
- Надёжность – стабильность работы;
- Коммуникация – взаимодействие с пользователем;
- Аналитика – отчётность и статистика.

2. Диаграмма приоритетности проблем

На основе матрицы «Частота × Влияние» выделены приоритетные зоны:

Высокий приоритет (красный сектор):

- Долгое ожидание при загрузке материалов (68 × 5 = 340);
- Нет мобильного приложения (49 × 5 = 245);
- Частые сбои при отправке работ (32 × 5 = 160).

Средний приоритет (жёлтый сектор):

- Сложность поиска курсов (54 × 4 = 216);
- Неудобный редактор тестов (37 × 4 = 148);
- Нехватка отчётов по успеваемости (12 × 4 = 48).

Низкий приоритет (зелёный сектор):

- Отсутствие уведомлений (29 × 3 = 87);
- Неинтуитивная регистрация (21 × 3 = 63);
- Медленный чат поддержки (15 × 3 = 45);
- Нет PDF-скачивания (18 × 2 = 36).

3. Перечень предложений по улучшению с оценкой трудозатрат (таблица 3)

Таблица 3 – Перечень предложений по улучшению с оценкой трудозатрат

№	Предложение	Ожидаемый эффект	Трудозатраты (чел./дни)	Приоритет
1	Оптимизировать загрузку материалов (кэширование, сжатие)	Сокращение времени загрузки на 60 %	10	Высокий
2	Разработать мобильное приложение	Повышение доступности на 40 %	45	Высокий
3	Внедрить систему уведомлений о новых заданиях	Снижение числа пропущенных сроков на 25 %	5	Средний
4	Переработать алгоритм поиска курсов (филь-	Ускорение поиска в 2 раза	8	Средний

	тры, сортировка)			
5	Обновить редактор тестов (drag-and-drop, шаблоны)	Сокращение времени создания тестов на 30 %	12	Средний
6	Усилить стабильность отправки работ (резервное копирование)	Снижение сбоев на 80 %	7	Высокий
7	Добавить экспорт материалов в PDF	Удобство офлайн-работы	3	Низкий
8	Автоматизировать отчёты по успеваемости	Экономия 5 чел./часов в неделю	6	Средний
9	Ускорить работу чата поддержки (шаблоны ответов)	Сокращение времени ответа на 50 %	4	Низкий
10	Упростить регистрацию (соцсети, единый вход)	Снижение отказов на 20 %	5	Низкий

4. Рекомендации по внедрению изменений

Этап 1 (срочность: 1–2 месяца)

- Устранить критические сбои при отправке работ (предложение № 6).
- Оптимизировать загрузку материалов (предложение № 1).
- Запустить разработку мобильного приложения (предложение № 2) – пилотная версия для iOS.

Этап 2 (срочность: 3–4 месяца)

- Переработать поиск курсов (предложение № 4).
- Обновить редактор тестов (предложение № 5).
- Внедрить уведомления (предложение № 3).
- Автоматизировать отчёты (предложение № 8).

Этап 3 (срочность: 5–6 месяцев)

- Добавить экспорт в PDF (предложение № 7).
- Ускорить чат поддержки (предложение № 9).
- Упростить регистрацию (предложение № 10).

Общие рекомендации:

1. Создать рабочую группу из 3 разработчиков и 1 аналитика для контроля реализации.
2. Проводить ежемесячный мониторинг удовлетворённости после внедрения изменений.
3. Запланировать A/B-тестирование новых функций перед полным развёртыванием.

4. Организовать обратную связь с пользователями по итогам каждого этапа (короткие опросы).

5. Выделить резервный бюджет в размере 15 % от общей сметы на непредвиденные доработки.

Ответственные:

- Руководитель проекта – контроль сроков;
- Ведущий разработчик – техническая реализация;
- UX-аналитик – тестирование удобств;
- Менеджер поддержки – коммуникация с пользователями.

Задание к лабораторной работе №11

Осуществить сбор, анализ и структурирование обратной связи от пользователей заданной информационной системы для выявления проблемных зон и выработки обоснованных предложений по её совершенствованию.

Контрольные вопросы

1. Что понимается под «обратной связью» в контексте информационных систем?
2. Перечислите 3–4 ключевые цели сбора отзывов о работе ИС.
3. В чём разница между общим опросом и точечным опросом? Приведите пример каждого.
4. Почему важно не просто собирать отзывы, но и предпринимать действия на их основе?
5. Назовите 5 каналов сбора отзывов о ИС. Для каждого укажите 1 преимущество и 1 ограничение.
6. В чём преимущество использования QR‑кодов для сбора отзывов по сравнению с текстовыми ссылками?
7. Чем глубинные интервью отличаются от фокус‑групп при сборе предложений по улучшению ИС?
8. Как часто рекомендуется проводить общие опросы по клиентской базе? Обоснуйте ответ.

Содержание самостоятельной работы

Цель самостоятельной работы обучающихся – получить новые знания по дисциплине «Сопровождение информационных систем».

Самостоятельная работа необходима для формирования у обучающихся способности самостоятельно решать задачи профессиональной деятельности, формирования умения и навыков планирования времени, формирования стремления развиваться и совершенствоваться.

Виды самостоятельной работы обучающихся указаны в табл. 1.

Таблица 4 – Виды самостоятельной работы

№ п/п	Вид СРС
1	Изучение базовых понятий и терминологии.
2	Изучение нормативной базы: ключевые ГОСТы (например, ГОСТ 34.601‑90), стандарты ISO/IEC 20000, ITIL.
3	Изучение правовых аспектов: лицензионные соглашения, SLA, ответственность за сбои.
4	Изучение процессов сопровождения: управление инцидентами, управление изменениями, конфигурационное управление, релиз-менеджмент, мониторинг
5	Изучение инструментов и технологий: ITSM‑системы (принципы работы ServiceNow, Jira Service Management, ITSM 365), системы мониторинга (Zabbix, Nagios, Prometheus, настройка триггеров, дашбордов). Написание скриптов для рутинных операций (Bash, PowerShell, Python). Изучение виртуализации и контейнеризации (основы Docker, Kubernetes для тестирования обновлений). Изучение правил ведения технической документации (Confluence, Wiki, Markdown).
6	Самостоятельная работа с кейсами: разбор реальных инцидентов (анализ отчётов о сбоях, взятых из публичных источников), составление договора для гипотетической ИС с учётом KPI, разработка инструкции по сопровождению для типового приложения. Проведение сравнительного анализа ITSM‑систем под заданные требования.

Обучающиеся должны изучить интернет-ресурсы, литературу по вопросам, представленным в табл. 1.

Учебно-методические материалы по дисциплине

1. Гвоздева, В. А. Основы построения автоматизированных информационных систем : учебник / В. А. Гвоздева, И. Ю. Лаврентьева. – Москва : НИЦ ИНФРА-М, 2025. – 318 с.
2. Проектирование информационных систем: учебник и практикум для СПО / Д. В. Чистов, П. П. Мельников, А. В. Золотарюк, Н. Б. Ничепорук. – 2-е изд., пер. и доп. – Москва : Юрайт, 2025. – 273 с.
3. Фролов, А. Б. Web-сайт. Разработка, создание, сопровождение: учебное пособие / А. Б. Фролов,И. А. Нагаева, И. А. Кузнецов ; под редакцией И. А. Нагаевой. – 2-е изд. – Саратов : Вузовское образование, 2024. – 355 с.
4. Фролов, А. Б. Основы web-дизайна. Разработка, создание и сопровождение web-сайтов : учебное пособие для СПО / А. Б. Фролов, И. А. Нагаева, И. А. Кузнецов. – 2-е изд. – Саратов : Профобразование, 2024. – 244 с.
5. Нестеров, С. А. Анализ и управление рисками в информационных системах на базе операционных систем Microsoft : учебное пособие / С. А. Нестеров. – 4-е изд. – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2024. – 250 с.
6. Зыков, С. В. Архитектура информационных систем. Основы проектирования : учебник для СПО / Зыков С. В. – Москва : Юрайт, 2025. – 260 с.
7. Мартишин, С. А. Базы данных. Практическое применение СУБД SQL- и NoSQL-типа для проектирования информационных систем : учебное пособие / С. А. Мартишин, В. Л. Симонов, М. В. Храпченко. – Москва : ФОРУМ : ИНФРА-М, 2021. – 368 с.
8. Небаев, И. А. Администрирование информационных систем. Введение в стек протоколов. Информационные службы и утилиты прикладного уровня : учебное пособие / И. А. Небаев. – Санкт-Петербург : Санкт-Петербургский государственный университет промышленных технологий и дизайна, 2023. – 76 с.
9. Конфигурирование и поддержка сетевой инфраструктуры. Основы конфигурирования информационных систем : учебное пособие для СПО / Н. В. Тутова, Е. О. Шишканова, А. В. Тутов, И. А. Андреев. – Саратов, Москва : Профобразование, Ай Пи Ар Медиа, 2024. – 96 с.

СОДЕРЖАНИЕ