Задача №5

Сформировать и описать аналитический обзор проделанной работы по следующим критериям: Функциональность, Производительность, Удобство использования, Безопасность, Масштабируемость, Сопровождаемость, Переносимость, Качество кода, Тестирование

За основу для анализа взято серверное приложение Delfi + IIS + MSSQL из задания номер 4.

### Аналитический обзор проделанной работы по созданию веб-приложения на Delphi с использованием MS SQL Server и IIS по следующим критериям:

#### 1. ****Функциональность****

* Приложение выполняет базовую функциональность: подключается к базе данных MS SQL Server, выполняет SQL-запросы и отображает данные в браузере через HTML-ответы.
* Реализован механизм логирования операций с базой данных и возможных ошибок в текстовый файл, что помогает отслеживать выполнение программы и диагностику.
* Основные функции включают отображение списка клиентов с их ID, именами и фамилиями.

#### 2. ****Производительность****

* Производительность системы зависит от нескольких факторов: скорости соединения с базой данных, времени выполнения SQL-запросов и генерации HTML-ответов.
* В текущей реализации приложение использует синхронное подключение к базе данных и выполнение запросов, что может вызывать задержки при больших объемах данных или при больших нагрузках на сервер базы данных.
* Для повышения производительности можно было бы рассмотреть оптимизацию SQL-запросов, использование кеширования или внедрение асинхронных операций.

#### 3. ****Удобство использования****

* Веб-приложение имеет базовый пользовательский интерфейс, который отображает данные в виде HTML-таблицы. Это удобно для небольших объемов данных, однако при увеличении количества записей может потребоваться пагинация или фильтрация данных для улучшения восприятия.
* Интерфейс кода и структуры приложений довольно простой, что делает его доступным для модификаций и интеграций.

#### 4. ****Безопасность****

* В приложении используется Windows-аутентификация для подключения к MS SQL Server, что обеспечивает безопасное управление правами доступа через встроенные механизмы операционной системы.
* Однако текущее решение не защищено от SQL-инъекций, так как SQL-запросы формируются напрямую в коде. Рекомендуется использовать параметры запросов для предотвращения SQL-инъекций.
* Протоколирование ошибок в текстовый файл без дополнительных мер по защите также может быть уязвимостью. Необходимо контролировать доступ к файлам логов.

#### 5. ****Масштабируемость****

* Приложение в текущем виде не масштабируемо для высоких нагрузок. Для увеличения масштабируемости потребуется внедрение асинхронной обработки данных, балансировки нагрузки, а также оптимизация взаимодействия с базой данных.
* IIS позволяет разворачивать приложения на более крупные серверные среды, что в будущем может быть полезным при увеличении нагрузки.

#### 6. ****Сопровождаемость****

* Приложение написано на Delphi, что является плюсом для разработчиков, знакомых с этой средой. Однако этот стек технологий не является самым популярным, что может вызвать трудности при найме новых разработчиков.
* Логирование помогает в диагностике ошибок и упрощает процесс отладки, что положительно влияет на сопровождаемость проекта.

#### 7. ****Переносимость****

* Приложение в текущем виде достаточно тесно связано с Windows, так как использует IIS и Windows-аутентификацию для работы с базой данных.
* Переносимость ограничена платформой, на которой работает IIS и MS SQL Server. Для повышения переносимости можно было бы рассмотреть использование кросс-платформенных решений (например, Apache или Nginx).

#### 8. ****Качество кода****

* Код в целом структурирован, но требует улучшений в плане обработки исключений и повышения читаемости. Некоторые участки кода можно оптимизировать (например, генерацию HTML через шаблоны или использование ORM).
* Использование жестко закодированных строк (например, пути к файлам логов) снижает гибкость системы. В будущем лучше вынести такие параметры в конфигурационные файлы.

#### 9. ****Тестирование****

* В текущей реализации отсутствуют механизмы автоматического тестирования. Рекомендуется добавить модульные тесты для проверки корректности работы с базой данных и генерации HTML-ответов.
* Также необходимо протестировать приложение под разными нагрузками и учесть различные сценарии (например, недоступность базы данных, неправильные запросы и т.д.).

### То что можно улучшить в проекте:

* Улучшить безопасность, добавив защиту от SQL-инъекций.
* Рассмотреть возможности для масштабируемости, включая асинхронные операции.
* Добавить тестирование и улучшить качество кода за счет параметризации и рефакторинга.
* Развивать удобство использования через модернизацию UI для работы с большими данными.

На основе проведенного анализа выше предложите способы решения выявленных проблем в ходе выполнения выбранной Вами кейс-задачи

### 1. ****Функциональность****

**Проблема**: Базовая функциональность может быть расширена для удобства работы с данными. **Решение**:

* Внедрить новые функции, такие как фильтрация, пагинация и сортировка данных на уровне интерфейса.
* Добавить возможности для более сложных запросов с использованием динамических параметров, чтобы пользователь мог получать нужные данные.
* Рассмотреть возможность добавления REST API для удобной интеграции с другими системами.

### 2. ****Производительность****

**Проблема**: Приложение использует синхронные запросы, что может вызывать задержки при увеличении нагрузки. **Решение**:

* Реализовать асинхронное выполнение SQL-запросов, чтобы не блокировать интерфейс при больших запросах.
* Внедрить кеширование данных на уровне приложения или базы данных для уменьшения нагрузки.
* Оптимизировать SQL-запросы (например, с помощью индексов в базе данных) и рассмотреть возможность разделения данных на страницы для увеличения скорости обработки запросов.
* Использовать пул соединений для оптимизации работы с базой данных и предотвращения слишком частого открытия и закрытия соединений.

### 3. ****Удобство использования****

**Проблема**: Приложение обладает базовым интерфейсом для вывода данных, что может быть неудобно при увеличении объемов. **Решение**:

* Улучшить интерфейс для вывода данных, добавив поддержку пагинации, поиска и фильтрации данных по ключевым полям.
* Интегрировать сторонние библиотеки для визуализации данных, например, через диаграммы или графики.
* Добавить адаптивный дизайн для работы на мобильных устройствах.

### 4. ****Безопасность****

**Проблема**: Отсутствует защита от SQL-инъекций, что может привести к уязвимостям. **Решение**:

* Использовать параметризованные запросы для работы с базой данных, чтобы избежать SQL-инъекций.
* Ограничить доступ к лог-файлам и использовать шифрование для хранения конфиденциальной информации.
* Рассмотреть использование HTTPS для обеспечения шифрования данных при передаче.

### 5. ****Масштабируемость****

**Проблема**: Приложение не масштабируемо для работы под высокой нагрузкой. **Решение**:

* Внедрить горизонтальное масштабирование, добавив несколько серверов IIS для балансировки нагрузки.
* Использовать асинхронные запросы и разгрузить сервер базы данных с помощью репликации данных или применения распределенных баз данных.
* Добавить обработку фоновых задач для длительных операций.

### 6. ****Сопровождаемость****

**Проблема**: Использование Delphi ограничивает гибкость в отношении новых разработчиков. **Решение**:

* Обеспечить документирование кода для упрощения сопровождения.
* Рассмотреть возможность миграции на более современные технологии или добавление слоев абстракции для упрощения сопровождения в будущем.
* Обеспечить обучение разработчиков для работы с текущей технологией или рассмотреть миграцию на кросс-платформенные решения.

### 7. ****Переносимость****

**Проблема**: Приложение сильно связано с Windows и IIS. **Решение**:

* Рассмотреть переход на платформы, которые поддерживают кросс-платформенные веб-сервера (например, Apache, Nginx).
* Использовать контейнеризацию (например, Docker), чтобы облегчить переносимость и развертывание приложения на других серверах.

### 8. ****Качество кода****

**Проблема**: Код нуждается в оптимизации и улучшении читаемости. **Решение**:

* Реализовать обработку исключений на всех уровнях приложения для предотвращения аварийных остановок.
* Вынести конфигурационные параметры (например, строку подключения к базе данных) в отдельные файлы конфигурации, чтобы облегчить сопровождение и модификацию.
* Улучшить структуру кода, разделив логику бизнес-правил и доступа к данным на отдельные модули.

### 9. ****Тестирование****

**Проблема**: Отсутствуют автоматизированные тесты. **Решение**:

* Внедрить автоматизированное тестирование, начиная с модульных тестов для проверки работы с базой данных и бизнес-логики.
* Добавить нагрузочное тестирование для проверки производительности приложения при высоких нагрузках.
* Использовать инструменты для мониторинга производительности и ошибок в режиме реального времени.

Эти решения помогут улучшить функциональность, производительность, безопасность и другие аспекты веб-приложения, а также сделают его более надежным и гибким для будущего использования.