**Кейс-задача № 5 Сформировать и описать аналитический обзор проделанной работы.**

**Аналитический обзор кейс-задачи №4**

Аналитический обзор оценивает разработку веб-сайта “Tourism”, созданного на Delphi 10.3 с использованием MS SQL Server в качестве базы данных и IIS в качестве веб-сервера. В качестве отправной точки для обзора использовались предыдущие ответы, описывающие создание и развертывание этого веб-сайта.

**1. Функциональность:**

* **Достижения:**
  + Реализована базовая функциональность: отображение списка туров из базы данных в формате JSON.
  + Спроектирована база данных с необходимыми таблицами (типы туров, страны, отели, туры, заказы) и связями между ними.
  + Созданы базовые ISAPI Actions для доступа к данным (например, /tours).
  + Обеспечено взаимодействие между веб-сайтом и базой данных через ADOConnection и ADOQuery.
* **Ограничения:**
  + Реализована только базовая функциональность чтения данных. Отсутствует функциональность добавления, редактирования и удаления данных (CRUD операции).
  + Отсутствует функциональность аутентификации и авторизации пользователей.
  + Отсутствует обработка ошибок и валидация входных данных.
  + Отсутствует пользовательский интерфейс (фронтенд). Данные отображаются только в формате JSON.
  + Не реализована функциональность заказов (создание, просмотр, управление).

**2. Производительность:**

* **Достижения:**
  + Использование индексов в базе данных (например, индекс на CountryID в таблице Hotels) для ускорения запросов.
* **Ограничения:**
  + Не проводилось профилирование производительности.
  + Не реализовано кэширование данных.
  + Не оптимизированы SQL-запросы для сложных сценариев.
  + Отсутствует анализ производительности при большой нагрузке.
  + Использование ADO может быть не самым производительным способом доступа к данным по сравнению с другими решениями (например, DataSnap с соответствующей оптимизацией).

**3. Удобство использования (юзабилити):**

* **Ограничения:**
  + Отсутствует пользовательский интерфейс.
  + Отсутствует документация для API.
  + Недружелюбные сообщения об ошибках (технические детали вместо понятных сообщений для пользователя).

**4. Безопасность:**

* **Ограничения:**
  + Хранение паролей в коде (недопустимо).
  + Уязвимость к SQL-инъекциям (использование конкатенации строк для построения SQL-запросов).
  + Отсутствует валидация входных данных.
  + Отсутствие аутентификации и авторизации делает сайт уязвимым для несанкционированного доступа.
  + Недостаточная защита от XSS-атак (неправильное экранирование данных, отображаемых в браузере).
  + Недостаточно настроены права доступа к файлам и базе данных.

**5. Масштабируемость:**

* **Ограничения:**
  + Архитектура WebBroker может быть недостаточно масштабируемой для высоконагруженных веб-сайтов.
  + Не продумана возможность горизонтального масштабирования (например, использование нескольких серверов IIS).
  + Использование ADO может создать проблемы с масштабируемостью при большом количестве одновременных подключений к базе данных.

**6. Сопровождаемость:**

* **Ограничения:**
  + Отсутствует документация.
  + Недостаточно комментариев в коде.
  + Код не структурирован с использованием паттернов проектирования.
  + Отсутствует автоматизированное тестирование.

**7. Переносимость:**

* **Ограничения:**
  + Зависимость от Windows и IIS (сложность переноса на другие платформы).
  + Зависимость от MS SQL Server (сложность переноса на другие СУБД).

**8. Качество кода:**

* **Ограничения:**
  + Отсутствие единого стиля кодирования.
  + Использование устаревших подходов (например, ADO вместо более современных ORM).
  + Отсутствие обработки исключений.
  + “Магические числа” и строковые литералы в коде (например, SQL-запросы, жестко закодированные имена полей).
  + Дублирование кода.

**9. Тестирование:**

* **Ограничения:**
  + Отсутствует автоматизированное тестирование (unit-тесты, интеграционные тесты, UI-тесты).
  + Не проводилось нагрузочное тестирование.
  + Тестирование проводилось только вручную.

**Предложения по решению выявленных проблем:**

Основываясь на анализе, вот предложения по улучшению веб-сайта “Tourism”:

**A. Улучшение безопасности:**

* **Переход на параметризованные запросы:** Использовать параметризованные SQL-запросы *всегда*, чтобы предотвратить SQL-инъекции. ADOQuery и ADOCommand это поддерживают.
* **Хранение паролей:** Никогда не хранить пароли в коде или конфигурационных файлах в открытом виде. Использовать:
  + **Хранение хешей паролей:** Хранить в базе данных только хеши паролей с использованием надежного алгоритма (например, bcrypt, Argon2).
  + **Конфигурационные файлы:** Хранить строку подключения к базе данных в зашифрованном конфигурационном файле.
  + **Key Vault:** Использовать Key Vault (например, Azure Key Vault) для хранения секретов.
* **Валидация входных данных:** Проверять все входные данные на корректность (тип, формат, диапазон значений) перед использованием. Использовать регулярные выражения, чтобы валидировать строковые поля (например, email, телефон).
* **Аутентификация и авторизация:** Реализовать систему аутентификации и авторизации пользователей. Использовать:
  + **Стандартные протоколы:** Использовать стандартные протоколы аутентификации, такие как OAuth 2.0 или OpenID Connect.
  + **Библиотеки:** Использовать библиотеки для работы с аутентификацией и авторизацией (например, Spring Security (если переходить на Java)).
* **Защита от XSS:** Экранировать данные, отображаемые в браузере, чтобы предотвратить XSS-атаки. Использовать:
  + **HTML-кодирование:** Преобразовывать специальные символы HTML (например, <, >, &, ", ') в их HTML-сущности.
* **Настройка прав доступа:** Предоставить учетной записи, под которой работает Application Pool, только минимально необходимые права доступа к базе данных и файлам.

**B. Улучшение функциональности:**

* **Реализация CRUD операций:** Добавить функциональность добавления, редактирования и удаления данных (CRUD - Create, Read, Update, Delete) для всех сущностей (туры, отели, страны и т.д.).
* **Реализация API для заказов:** Реализовать API для создания, просмотра и управления заказами.
* **Добавление пользовательского интерфейса (фронтенда):** Создать пользовательский интерфейс с использованием HTML, CSS и JavaScript (например, с использованием фреймворка React, Angular или Vue.js) для отображения данных и взаимодействия с пользователем.
* **Добавление поиска и фильтрации:** Реализовать функциональность поиска и фильтрации туров, отелей и т.д.

**C. Улучшение производительности:**

* **Профилирование производительности:** Использовать инструменты профилирования (например, AQTime) для выявления узких мест в коде и SQL-запросах.
* **Кэширование:** Реализовать кэширование данных (например, с использованием Redis или Memcached) для уменьшения нагрузки на базу данных.
* **Оптимизация SQL-запросов:** Оптимизировать SQL-запросы, используя индексы, правильные типы данных и избегая ненужных операций.
* **Использование Connection Pooling:** Убедитесь, что ADO Connection Pooling включен, чтобы избежать затрат на установку и разрыв соединения для каждого запроса.

**D. Улучшение сопровождаемости и качества кода:**

* **Документирование кода:** Добавить комментарии к коду, объясняющие назначение классов, методов и переменных.
* **Использование паттернов проектирования:** Реструктурировать код с использованием паттернов проектирования (например, MVC, Repository, Factory).
* **Автоматизированное тестирование:** Реализовать автоматизированное тестирование (unit-тесты, интеграционные тесты, UI-тесты) для проверки корректности кода и предотвращения регрессий. Использовать фреймворки для юнит-тестирования Delphi, такие как DUnitX.
* **Code Reviews:** Проводить code reviews для выявления ошибок и улучшения качества кода.
* **Code Style Guidelines:** Разработать и применять code style guidelines для поддержания единообразия и читаемости кода.

**E. Улучшение масштабируемости:**

* **Переход на микросервисную архитектуру:** Разделить приложение на микросервисы, каждый из которых отвечает за определенную функциональность.
* **Использование Message Queue:** Использовать Message Queue (например, RabbitMQ или Kafka) для асинхронного взаимодействия между микросервисами.
* **Горизонтальное масштабирование:** Настроить IIS для работы с несколькими серверами (Web Farms).

**F. Улучшение переносимости:**

* **Использование ORM:** Использовать ORM (Object-Relational Mapping) для абстрагирования от конкретной СУБД (например, <https://www.devart.com/linqconnect/>).
* **Контейнеризация:** Использовать Docker для контейнеризации приложения, что упростит его развертывание на различных платформах.

**Вывод:**

Веб-сайт “Tourism” обладает базовой функциональностью, но требует значительных улучшений в области безопасности, производительности, удобства использования, сопровождаемости, масштабируемости и переносимости. Предложенные решения помогут создать более надежный, быстрый, удобный и масштабируемый веб-сайт, отвечающий современным требованиям. Важно помнить, что улучшение безопасности должно быть приоритетом.