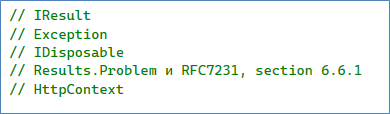
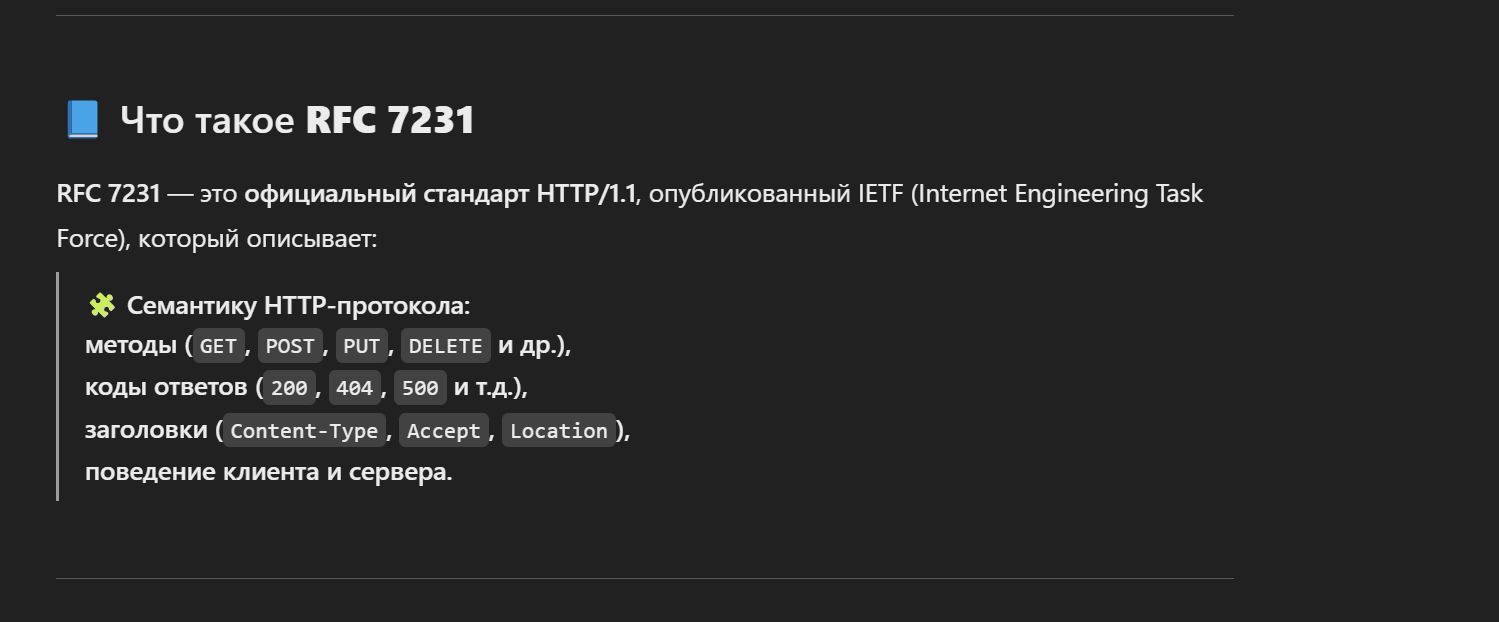
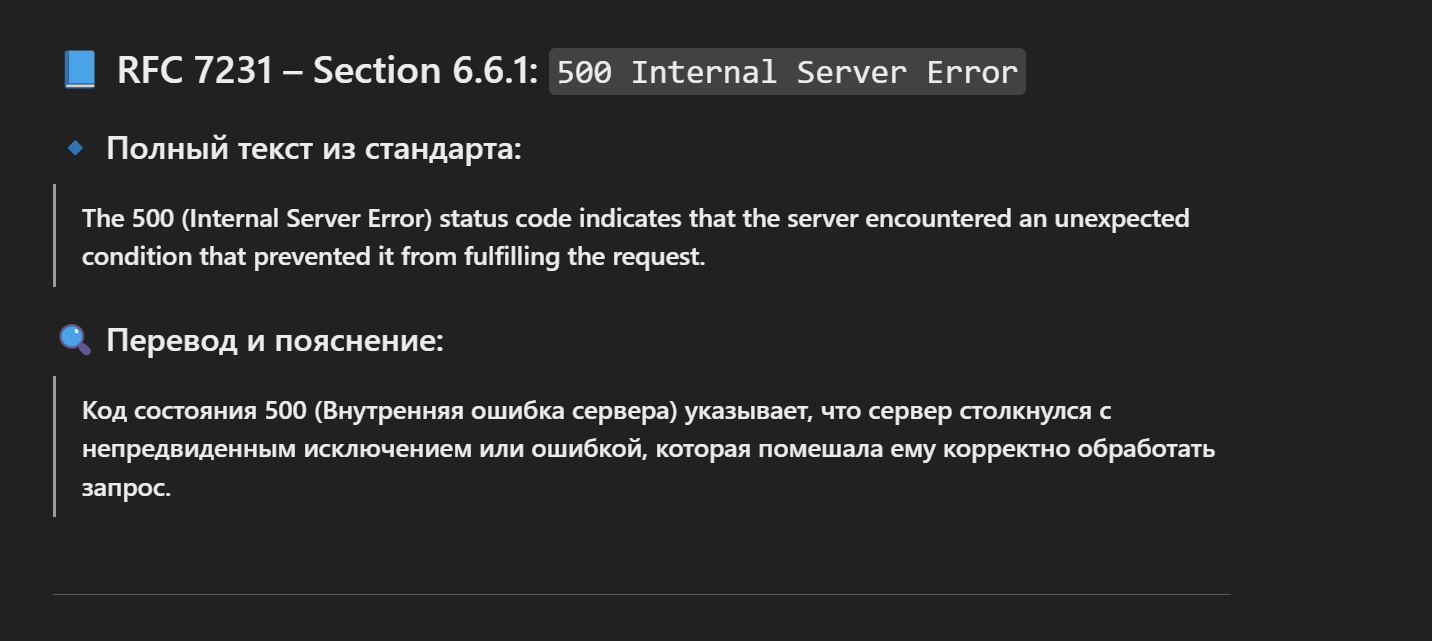


1. **MapGet()**
   * Регистрирует маршрут для обработки HTTP‑запросов с методом **GET**.
   * Обычно используется, когда клиент запрашивает данные (например, чтение списка ресурсов).
2. **MapPost()**
   * Регистрирует маршрут для обработки HTTP‑запросов с методом **POST**.
   * Чаще всего применяется, когда нужно создать новый ресурс или передать данные на сервер (например, отправить форму).
3. **MapPut()**
   * Регистрирует маршрут для обработки HTTP‑запросов с методом **PUT**.
   * Как правило, используется для обновления существующего ресурса целиком.
4. **MapDelete()**
   * Регистрирует маршрут для обработки HTTP‑запросов с методом **DELETE**.
   * Предназначен для удаления ресурса.
5. **MapFallback()**
   * Определяет «запасной» маршрут, который срабатывает, если ни один из других маршрутов не подошёл к запросу.
   * Часто применяется, чтобы вернуть страницу «404 Not Found» или перенаправить в SPA (Single Page Application).
6. **UseExceptionHandler()**
   * Настраивает глобальную обработку исключений и перенаправляет в специальный маршрут (например, "/Error"), если в приложении возникает необработанное исключение.
   * Позволяет централизованно обрабатывать ошибки и возвращать пользователю понятные сообщения.
7. **UseStaticFiles()**
   * Подключает middleware для обслуживания статических файлов (HTML, CSS, JavaScript, изображения и т.д.) из заданной директории (по умолчанию wwwroot).
   * Используется, когда нужно раздавать контент, не требующий динамической обработки.



1. **IResult**
   * В минимальных API (ASP.NET Core) интерфейс **IResult** представляет результат выполнения HTTP-запроса.
   * Объекты, реализующие IResult, содержат логику формирования HTTP-ответа (статус-код, заголовки, тело ответа и т.д.).
   * Примеры готовых реализаций: Results.Ok(), Results.NotFound(), Results.Problem() и т.д.
2. **Exception**
   * Это базовый класс для всех исключений (ошибок) в .NET.
   * Позволяет обрабатывать ошибочные ситуации, используя механизм try/catch или пробрасывать ошибку выше по стеку вызовов.
   * Содержит сообщение об ошибке, стек вызовов и другие детали, необходимые для диагностики.
3. **IDisposable**
   * Интерфейс, который определяет метод Dispose().
   * Используется для освобождения неуправляемых ресурсов (например, файловые дескрипторы, сетевые подключения, ресурсы ОС).
   * Объекты, реализующие IDisposable, обычно используют конструкцию using, чтобы гарантировать вызов Dispose() даже в случае ошибок.
4. **Results.Problem** и **RFC7231, section 6.6.1**
   * Results.Problem — это метод из минимальных API в ASP.NET Core, который возвращает стандартизированный ответ об ошибке (Problem Details).
   * Под капотом он формирует HTTP-ответ с телом в формате RFC 7807 (Problem Details for HTTP APIs).
   * 
   * Упоминание **RFC7231, section 6.6.1** часто связано с кодом ответа **500 Internal Server Error**, описанным в спецификации HTTP/1.1 (RFC 7231).
   * Таким образом, при возникновении ошибок сервер может вернуть структуру «Problem Details» с описанием проблемы, соответствуя стандартам HTTP.
   * 
5. **HttpContext**
   * Класс, представляющий контекст текущего HTTP-запроса в ASP.NET Core.
   * Содержит информацию о запросе (URL, заголовки, тело), ответе (статус-код, заголовки), а также о связанных сервисах и функциональности (Cookies, Session, Features и пр.).
   * Позволяет middleware и конечным точкам взаимодействовать с деталями запроса и формировать ответ.

Чем известны Ada Lovelace и Charles Babbage?☺

**Ада Лавлейс** известна как первый программист в истории – она написала алгоритм для аналитической машины Чарльза Бэббиджа.  
**Чарльз Бэббидж** считается «отцом вычислительной техники», так как разработал идеи и проекты первых механических вычислительных машин, таких как Разностная машина и Аналитическая машина.