logyca

investigación
consultoría
analítica

Especificaciones Técnicas

Objeto genérico de respuesta a peticiones HTTP.

Bogotá D.C., junio 2021

Conten

Introducción

Este documento presenta las especificaciones técnicas para la recepción y envío de respuesta a peticiones HTTP (Request / Response) a implementar en los desarrollos en curso y futuros en LOGYCA. Estas se convierten en un marco de referencia para el trabajo del desarrollador de software para la construcción de APIs y Backends indiferentemente la tecnología en la que estos estén construidos. Los ejemplos presentados durante este documento están en .Net, se deberá hacer el código homologo cuando se utilicen otros frameworks.

1 Objetivo

Estandarizar los objetos de recepción y envío a peticiones HTTP en los sistemas de LOGYCA.

2 Solicitud (Request)

- Para los frameworks que lo permitan, se debe indicar la forma en la que se recibe la información, ya se por el cuerpo (FromBody) o por la cadena de consulta (FromQuery)
- Para las peticiones que la información sea recibida en el cuerpo, el objeto deberá llamarse **RequestPayload**, el cual debe ser genérico y soportar siempre una lista de registros.
- Para las peticiones que la información sea recibida FromQuery se debe crear un objeto que contenga los parámetros a recibir
- El objeto de las peticiones Get debe incluir las propiedades:
    - **LastId**: Último Id reportado por el recurso solicitado como respuesta en una petición.
    - **PageSize**: Cantidad de puntos de registros que se devolverán en la respuesta. **Nota:** Si la cantidad de información a devolver excede la capacidad de procesamiento del recurso se retornará el máximo que el recurso permita.
- El idioma a utilizar será siempre en Ingles.
- Se deben nombrar los métodos según su verbo HTTP, si se tiene mas de uno con el mismo verbo se debe dar otro nombre del segundo en adelante.

Ejemplo:

```
Post([FromBody] RequestPayload<BrandDTO> brand)

Get([FromQuery] BrandConsultDTO brandConsult)
```

```
public class RequestPayload<T>
{
    /// <summary>
    /// Initializes a new instance of the <see cref="RequestPayload{T}"/> class.
    /// </summary>
    20 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public RequestPayload()
    {
        Payload = new List<T>();
    }

    /// <summary>
    /// Gets or sets payload
    /// </summary>
    99+ references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public List<T> Payload { get; set; }
}
```

3 Respuesta (Response)

- La estructura del objeto de respuesta a todas las peticiones se debe llamar **APIResult,** este se detallará su contenido en el siguiente apartado**.**
- Para peticiones POST, el objeto de respuesta debe ser el mismo que fue recibido, más la asignación del ID generado por la base de datos en la propiedad que corresponda.
- Para peticiones PUT, PATCH, el objeto de respuesta debe ser el mismo que fue recibido más el resto de propiedades del objeto **APIResult**.
- Uso de procesamiento asíncrono para los escenarios que la arquitectura lo requiera.
- Para las peticiones GET, el objeto de respuesta será el creado para su fin.
- El idioma a utilizar será siempre en Ingles.

Ejemplo:

```
public async Task<APIResult<BrandDTO>>
```


```
public async Task<APIResult<BrandDTO>>
```

4 APIResult.

Campo	Descripción	Tipo
ResultToken	Objeto que contiene la información del token de acceso.	Token
ResultObject	Array de objetos que contiene el resultado exitoso de una petición síncrona o asíncrona.	Array Cada elemento tiene el mismo tipo del recurso solicitado.
ValidationErrors	Devuelve los errores de validación que se encontraron al procesar la solicitud. Por cada propiedad del recurso que presentó error se envía una propiedad con el mensaje de error de la validación que falló. Por cada elemento en el payload de entrada, se coloca un elemento en el validationErrors. Si un elemento en particular no tiene errores, se envía el objeto de validationErrors vacío.	Array Cada elemento tiene el mismo tipo del recurso solicitado.
TransactionId	Id de la transacción asíncrona para las peticiones que tenga un sistema de procesamiento posterior, con este Id se podría saber el estado del proceso por medio de recursos que lo permita.	Number
TraceabilityId	Identificador interno usado para trazabilidad sobre el motivo específico del error.	String
ResultMessage	Contiene mensaje del estado de la petición	String
LogycaStatus	Contiene el resultado de la operación, como un código LOGYCA. Los códigos LOGYCA funcionan como códigos HTTP estándar y están inspirados en los códigos de respuesta de GRPC. Sirven para identificar una razón más detallada del error que la que nos permite entregar el código estándar HTTP.	Number
LastId	Contiene el último Id devuelto para el recurso solicitado como respuesta en una petición GET paginada. Por ejemplo, la petición GET para obtener negociaciones tiene como parámetro “lastId” el Id de la última negociación retornada en la respuesta.	Number
TotalRecordsByQuery	Indica la cantidad de registros entregados en la respuesta, este dato junto con LastId son manejados para paginar la información.	Number
DataError	Indica si la respuesta contiene algún error, de ser verdadero, el objeto **ValidationErrors** deberá contener el detalle.	Boolean

```
/// <summary>
/// Object result API integration
/// </summary>
/// <typeparam name="T">Genereric Object</typeparam>
99+ references | 0 changes | 0 authors, 0 changes
public class APIResult<T>
{
    3 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public Tokens ResultToken { get; set; }
    76 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public List<T> ResultObject { get; set; }
    99+ references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public List<ValidationError> ValidationErrors { get; set; }
    2 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public long? TransactionId { get; set; }
    4 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public Guid? TraceabilityId { get; set; }
    99+ references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string ResultMessage { get; set; }
    99+ references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public LogycaStatusEnum LogycaStatus { get; set; }
    12 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public long? LastId { get; set; }
    12 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public long TotalRecordsByQuery { get; set; }
    57 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public bool? DataError { get; set; }
}
```

4.1 Token

Atributo	Descripción	Tipo de dato
Token	Contiene el auth token JWT codificado en base64 el cual tiene una vigencia determinada.	String
RefreshToken	Contiene el auth token JWT codificado en base64 el cual tiene la información para renovar el token	String
Result	Mensaje que indica si fue autorizado “Authorization”	String
EmailActiveDirectory	Indica el email del usuario logueado, el email es único. Devuelve vacío – Uso interno	String
Message	Contiene mensaje personalizado. Devuelve vacío – Uso interno	String

```
public class Tokens
{
    /// <summary>
    /// Gets or sets result Token
    /// </summary>
    25 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string Token { get; set; }

    /// <summary>
    /// Gets or sets Refresh token
    /// </summary>
    8 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string RefreshToken { get; set; }

    /// <summary>
    /// Gets or sets result request
    /// </summary>
    5 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string Result { get; set; }

    /// <summary>
    /// Gets or sets email Active Directory
    /// </summary>
    5 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string EmailActiveDirectory { get; set; }

    /// <summary>
    /// Gets or sets user message
    /// </summary>
    6 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string Message { get; set; }
}
```

4.2 ValidationError

Atributo	Descripción	Tipo de dato
TransactionId	Indica el Id de transacción generado o asociado al error en el escenario que este pueda ser capturado.	Number
Description	Elemento característico o descriptivo el cual presento el error. Ejemplo GTIN, GLN.	String
DetailError	Detalle del error presentado.	String

```
/// <summary>
/// ValidationError
/// </summary>
99+ references | ggomez@logyca.com, 252 days ago | 1 author, 1 change
public class ValidationError
{
    /// <summary>
    /// Gets or sets Status
    /// </summary>
    23 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public long? TransactionId { get; set; }

    /// <summary>
    /// Gets or sets Description
    /// </summary>
    64 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string Description { get; set; }

    /// <summary>
    /// Gets or sets DetailError
    /// </summary>
    37 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string DetailError { get; set; }
}
```

4.3 LogycaStatus.

Lo siguientes códigos fueron tomados como referencia del estándar de Google.

Estados Personalizados LOGYCA			
Status codes and their use in gRPC			
Code	**Number**	**Description**	**Closest HTTP Mapping**
OK	0	Not an error; returned on success.	200 OK
CANCELLED	1	The operation was cancelled, typically by the caller.	499 Client Closed Request
UNKNOWN	2	Unknown error. For example, this error may be returned when a `Status` value received from another address space belongs to an error space that is not known in this address space. Also errors raised by APIs that do not return enough error information may be converted to this error.	500 Internal Server Error
INVALID_ARGUMENT	3	The client specified an invalid argument. Note that this differs from `FAILED_PRECONDITION`. `INVALID_ARGUMENT` indicates arguments that are problematic regardless of the state of the system (e.g., a malformed file name).	400 Bad Request
DEADLINE_EXCEEDED	4	The deadline expired before the operation could complete. For operations that change the state of the system, this error may be returned even if the operation has completed successfully. For example, a successful response from a server could have been delayed long	504 Gateway Timeout
NOT_FOUND	5	Some requested entity (e.g., file or directory) was not found. Note to server developers: if a request is denied for an entire class of users, such as gradual feature rollout or undocumented whitelist, `NOT_FOUND` may be used. If a request is denied for some users within a class of users, such as user-based access control, `PERMISSION_DENIED` must be used.	404 Not Found
ALREADY_EXISTS	6	The entity that a client attempted to create (e.g., file or directory) already exists.	409 Conflict

PERMISSION_DENIED	7	The caller does not have permission to execute the specified operation. `PERMISSION_DENIED` must not be used for rejections caused by exhausting some resource (use `RESOURCE_EXHAUSTED`instead for those errors). `PERMISSION_DENIED` must not be used if the caller can not be identified (use `UNAUTHENTICATED` instead for those errors). This error code does not imply the request is valid or the requested entity exists or satisfies other pre-conditions.	403 Forbidden
UNAUTHENTICATED	16	The request does not have valid authentication credentials for the operation.	401 Unauthorized
RESOURCE_EXHAUSTED	8	Some resource has been exhausted, perhaps a per-user quota, or perhaps the entire file system is out of space.	429 Too Many Requests
FAILED_PRECONDITION	9	The operation was rejected because the system is not in a state required for the operation's execution. For example, the directory to be deleted is non-empty, an rmdir operation is applied to a non-directory, etc. Service implementors can use the following guidelines to decide between `FAILED_PRECONDITION`, `ABORTED`, and `UNAVAILABLE`: (a) Use `UNAVAILABLE` if the client can retry just the failing call. (b) Use `ABORTED` if the client should retry at a higher level (e.g., when a client-specified test-and-set fails, indicating the client should restart a read-modify-write sequence). (c) Use `FAILED_PRECONDITION` if the client should not retry until the system state has been explicitly fixed. E.g., if an "rmdir" fails because the directory is non-empty, `FAILED_PRECONDITION`should be returned since the client should not retry unless the files are deleted from the directory.	400 Bad Request
ABORTED	10	The operation was aborted, typically due to a concurrency issue such as a sequencer check failure or transaction abort. See the guidelines above for deciding between `FAILED_PRECONDITION`, `ABORTED`, and `UNAVAILABLE`.	409 Conflict

OUT_OF_RANGE	11	The operation was attempted past the valid range. E.g., seeking or reading past end-of-file. Unlike `INVALID_ARGUMENT`, this error indicates a problem that may be fixed if the system state changes. For example, a 32-bit file system will generate `INVALID_ARGUMENT` if asked to read at an offset that is not in the range [0,2^32-1], but it will generate `OUT_OF_RANGE` if asked to read from an offset past the current file size. There is a fair bit of overlap between `FAILED_PRECONDITION` and `OUT_OF_RANGE`. We recommend using `OUT_OF_RANGE` (the more specific error) when it applies so that callers who are iterating through a space can easily look for an `OUT_OF_RANGE` error to detect when they are done.	400 Bad Request
UNIMPLEMENTED	12	The operation is not implemented or is not supported/enabled in this service.	501 Not Implemented
INTERNAL	13	Internal errors. This means that some invariants expected by the underlying system have been broken. This error code is reserved for serious errors.	500 Internal Server Error
UNAVAILABLE	14	The service is currently unavailable. This is most likely a transient condition, which can be corrected by retrying with a backoff.	503 Service Unavailable
DATA_LOSS	15	Unrecoverable data loss or corruption.	500 Internal Server Error
TRANSACTION_PENDING	1001	Pending	200 OK
TRANSACTION_IN PROCESS	1002	Transaction in process.	200 OK