Evidencia y especificación de la API

Diego Acuña-222675, Felipe Briosso-269851, Nicole Uhalde-270303

Con respecto a la especificación de la API, esta se incluye a continuación. Dado que no se ve muy bien, se incluye el Excel que contiene la tabla entera.

Recur	Acti	Path	Header	Parameter	Model in	Model out	server	justification

purchare	Get	/api/purchar er	Authorization (Only the user can see his purchare hirtory)	-	-	[
						{
						"id":"string",
						"cart":[
						{
						"name":"string",
						"brand":"string",
						"price":0,
						"category":
						"description":
						"string",
						"color":["string"
						]
						}
						],
						"idUser":"string",
						"date":"DateTime",
						"promotion":
						"total":"int"
						}
						]
							200-everything
							400-user error
							401-Not unauthenticated
							500-server error
								the user can see a list with their purchares. If the user is an admin he can see all the purchares

purchare	part	/api/purchar er	Authorization (The administrator can't buy, only the buyer)	-	{
					"cart":[
					{
					"name":"string",
					"brand":"string",
					"price":0,
					"category":
					"description":
					"string",
					"color":[
					"string"
					]
					}
					]
					}
						{
						"id":"string",
						"cart":[
						{
						"name":"string",
						"brand":"string",
						"price":0,
						"category":
						"description":
						"string",
						"color":[
						"string"
						]
						}
						],
						"buyerId":"string",
						"date":
						"selectedPromotion":"string",
						"total":"number"
						}
							201-Purchare done correctly
							400-User error
							401-Not unauthenticated
							500- Server error
								the user can create a purchare having the cart

product	part	/api/product s	Authorization (Only the administratior can create products)	-	{"name":"String" "precio":"int" "description":"String" "marca":"String" "category": "color":"String"}	{"id":"String" "name":"String" "precio":"int" "description":"String" "marca":"String" "category": "color":"String"}	201-Product created correctly 400-User error 401-Not unauthenticated 403- Not authorized 500-Server error	only admins can create a product
product	put	/api/product s	Authorization (Only the administrator can update products).	-	{"id":"String" "name":"String" "precio":"int" "description":"String" "marca":"String" "category": "color":"String"}	{"id":"String" "name":"String" "precio":"int" "description":"String" "marca":"String" "category": "color":"String"}	200-Updated correctly 400-User error 401-Not unauthenticated 403- Not authorized 500-Server error	only admins can update a product

product	get	/api/product	Authorization (anyone who is authorize can enter)	/product?text-String&brand-String&category-String&operation-String	-	[{"id":"String" "name":"String" "price":"int" "description": "Brand":"String" "category": "colour":["string"] }]	200-Listshown correctly 400-User error 401-Not unauthenticated 500-Server error	the user can see a list with all the products filters.
product	get	/api/product/{id}	Authorization (Only the administrator can get)	-	-	{"id":"String" "name":"String" "precio":"int" "description":"String" "marca":"String" "category": "color":["String"]}	200-Listshown correctly 400-User error 401-Not unauthenticated 500-Server error	the user can see a list with all the products filters
user	delete	/api/users/{id}	Authorization (Only the administrator can delete)		-	{"id":"String" "name":"String" "email":"String" "address":"String" "Rolo:["String"]}	200-Deleted 400-User error 401-Not unauthenticated 500-Server error	only admins can delete a user
user	put	/api/users/{id}/admin	Authorization (Only the administrator can update it)		{"rolo":["string"] "name":"String" "email":"String" "address":"String" }	{"id":"String" "name":"String" "email":"String" "address":"String" "Rolo:["String"] }	200-Updated correctly 400-User error 401-Not unauthenticated 500-Server error	only the user and the admins can update an user
user	get	/api/users	Authorization (Only the administrators can see the list)		-	{"id":"String" "name":"String" "email":"String" "address":"String" "Rolo:["String"]}	200-Shown correctly 400-User error 401-Not unauthenticated 500-Server error	only admins can see all the users
user	post	/api/users/admin	Authorization(Admin Can Create users with specific rolos)	-	{"name":"String" "email":"String" "rol":["String"] "Password":"String "address":"String"}	{"id":"String" "name":"String" "email":"String" "rolo":["String"] "address":"String"}	201-Created correctly 400-User error 500-Server error	the admin can create an user

user	post	/api/users	-	-	{
					"name":"String",
					"email":"String"
					"password":"String
					"address":"String"}
					}
						{"id":"String"
						"name":"String"
						"email":"String"
						"rolo":["string"]
						"address":"String"
						}
							201-Created corre
							400-User error
							500-Server error
								user can create their account

user	put	/api/users/{id}	Authorization (Only the user can modify their own profile)	-	{"name":"String" "Password":"String "email":"string" "Rolo":["String"] "address":"String"}	{"id":"String" "name":"String" "email":"String" "rolo":["String"] "address":"String"}	200-Updated corre 400-User error 401-Not unauthent 500-Server error	only the user and the admins can update an user
session	post	/api/sessions			{"email":"string" "password": }	{"Token":"String"}	200-Logged in Correctly 400-User error 500-Server error	the user can log in
session	delete	/api/sessions	Authorization (Only the users can log out)	-	-	-	200-Logged out correctly 400-User error 500 Server error	the user can log out

Cumplimiento de REST:

Restricciones:

- Uniform Interface: Se define una interfaz para la interacción entre el cliente y el servidor. En este caso son los endpoints detallados arriba
- Stateless: Como REST no maneja estado a nivel de servidor, cuando se hace una request, esta incluye toda la información necesaria para que se ejecute. Esto se logra utilizando los model in, query strings,etc. Por ejemplo, al querer traer un producto, se utiliza GET /api/products/{id}, es decir que al hacer la solicitud ya se indica la id del recurso que queremos.
- Cacheable: En nuestro caso no se utilizó.
- Client-Server: La arquitectura es cliente-servidor. Los clientes envían sus requests utilizando los endpoints de la tabla de arriba. El servidor responde con lo que en la tabla de arriba se llama “Model out”.
- Layered system: Se utilizó una estructura por capas, controlador, lógica y repositorio, aunque se explica con mayor detalle en el documento *Especificación del diseño*.

Buenas prácticas de REST utilizadas

- Se utilizaron sustantivos en lugar de verbos
- Los sustantivos se encontraban todos en plural
- Las query strings son muy utilizadas en API Rest para búsquedas que contienen filtros. En nuestro caso, se utilizó para el filtrado de productos según el criterio(OR o AND), texto, marca y categoría.
- Bajo ningún concepto se devolvieron respuestas sin recursos, ya que esto es una mala práctica
- El GET no modifica estado en ningún momento (solo recupera datos)

Justificación del cumplimiento de API Rest[1]

- Recursos Identificables: Cada uno de los endpoints de la API se refiere a un recurso específico, como "purchase", "product", "user", y "session". Esto es fundamental en REST, ya que los recursos son la base de la arquitectura REST.

- Verbos HTTP significativos: Por ejemplo, en el endpoint "purchase" (GET /api/purchases), se utiliza el verbo GET para recuperar una lista de compras. Esto es coherente con el principio de usar verbos HTTP significativos para operaciones en recursos.

- Uso de Rutas Descriptivas: Las rutas de los endpoints son descriptivas y se relacionan directamente con los recursos y las acciones que se están realizando. Por ejemplo, "/api/purchases" para compras, "/api/products" para productos, "/api/users/{id}" para usuarios, etc.

- Uso de Códigos de Estado HTTP: Los códigos de estado devueltos son los más comunes. Por ejemplo se devuelve estado 201 (cuando se crea una compra exitosamente), 401 (cuando el usuario no se encuentra autorizado), etc. Esto es coherente con el uso de códigos de estado HTTP apropiados en REST para indicar el resultado de la operación.

- Autenticación y Autorización: En todos los endpoints donde se requiere autenticación o autorización, como en "purchase" y "product", se utilizan cabeceras de autorización para garantizar que solo los usuarios autorizados puedan realizar acciones específicas. Por ejemplo, solo los administradores pueden crear productos.

- Uso de Parámetros de Consulta: En el endpoint "product" (GET /api/products), se utilizan parámetros de consulta, como "text", "brand", "category" y "operation", para filtrar la lista de

[1] Mejorado con los comentarios de ChatGPT

productos. Esto es consistente con la práctica común de usar parámetros de consulta en REST para personalizar las respuestas.

Códigos de error utilizados

Informational response (1xx): El equipo no vio necesario hacer uso de estos códigos

Success (2xx): El equipo utilizó únicamente el estado 200-OK y 201-Created, dependiendo de la operación. En caso de tratarse de un POST, el código a retornar era 201, mientras que se retornaba 200 en otro caso.

Redirection(3xx): No fue necesario

Client Errors(4xx):

- 400 User error: para errores generales
- 401 Unathenticated: cuando el usuarion no se encontraba logeado
- 403 Not authorized: cuando no tenía los permisos de realizar la acción.

Server Errors (5xx): 500-Internal server error: Errores generales del lado del servidor.