Universidad ORT
Facultad de Ingeniería

Evidencia de Diseño y especificación de la API

Diseño de Aplicaciones 2 - 6/10/2022

Repositorio Git
https://github.com/ORT-DA2/DA2-163471-223139-201250.git

Federico Czarnievicz - 201250 - M6C-1
Ignacio Olivera - 223139 - M6C
Cristhian Maciel - 163471 - N6A

Índice:

Criterios de diseño de la API

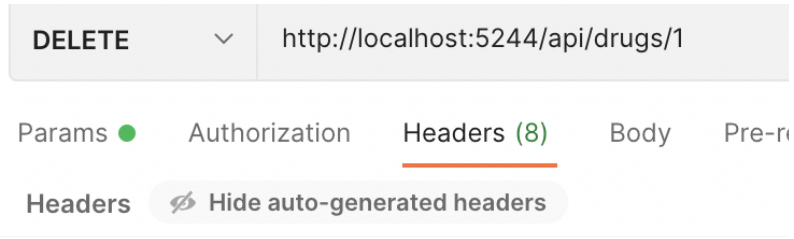
Discusión de los criterios seguidos para asegurar que la API cumple con los criterios REST.

De forma tal de asegurarnos mantenibilidad y extensibilidad frente a cualquier tipo de modificaciones, el trabajo presentado cumple con la arquitectura basada en REST, siendo esta una arquitectura cliente-servidor. Para asegurarnos que se está cumpliendo con los criterios, tuvimos en cuenta los siguientes puntos:

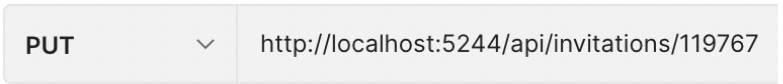
Se sigue una comunicación HTTP mediante distintas operaciones que hacen posible acceder al servidor en un punto de acceso, enviar información y recibir respuesta. Las operaciones utilizadas en esta primera instancia son GET - para obtener datos - , POST - para crear datos -, PUT - para modificar datos - y DELETE - para eliminar datos -. Se incluyen dentro del header y body HTTP todos los parámetros, contexto y datos necesarios para que el servidor genere la respuesta sin la necesidad que el servidor, mientras procesa la solicitud, tenga que almacenar ningún tipo de contexto o sesión. Se pudieron evidenciar ejemplos de lo mencionado en las capturas de las distintas requests.

Dentro de las requests se siguieron determinados lineamientos que hacen al cumplimiento de Rest. A continuación detallamos alguno de ellos:

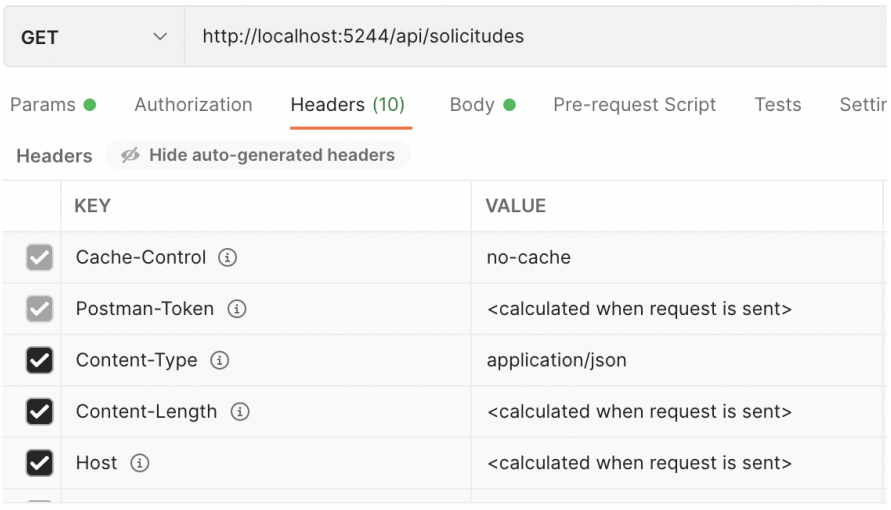
- Uso de sustantivos en plural y no verbos



- No extender URL más de 3 niveles



- Utilización de headers para identificar formato



Por otra parte, nos parece importante agregar que la WebApi de nuestra solución está dividida en controllers, models y filters. A continuación explicaremos brevemente en qué consiste cada uno de ellos:

Controllers: Aquí se definen los distintos endpoints que van a permitir la interacción con el sistema. Se configuraron distintas rutas de forma que se le pueda pegar a la API y se cumplan todos los pedidos que la misma realiza. Cabe aclarar que en varias de las rutas se utilizan DTO's que se encuentran como parte del dominio. Cada uno de los controllers, hereda de ControllerBase lo que permite exponer los servicios de la api. Todas las peticiones HTTP que se realizan se delegan a la lógica de negocio y las mismas realizan las acciones correspondientes devolviendo el resultado o en caso contrario mensajes del error encontrado. Se fueron creando según la demanda lógica del sistema en cuanto a los recursos que se deben de acceder y utilizar.

Models: Son utilizados por los controllers para manejar las entidades. A cada una de ellas se le creó un ModelRequest donde allí se encontrarán los atributos de entrada del JSON y por otro lado el ModelResponse donde estarán los atributos de salida del mismo. Cabe aclarar que no todas las entidades utilizan ambos models, incluso en algunos casos donde se edita cierta información, se crea un tercer modelo de respuesta para que devuelva solamente el dato editado.

Filters: Por un lado, son utilizados por los controllers para validar que ciertas requests estén autorizadas para realizarse según el rol con el que se esté logueado en ese momento. Por otro lado, también se utilizan para manejar excepciones antes o después de que la petición llegue al controller con mensajes retornados como un model exception.

Mecanismo de autenticación y autorización

Para manejar distintos usuarios según sus responsabilidades, se definieron los siguientes roles:

- Admin
- Owner
- Employee

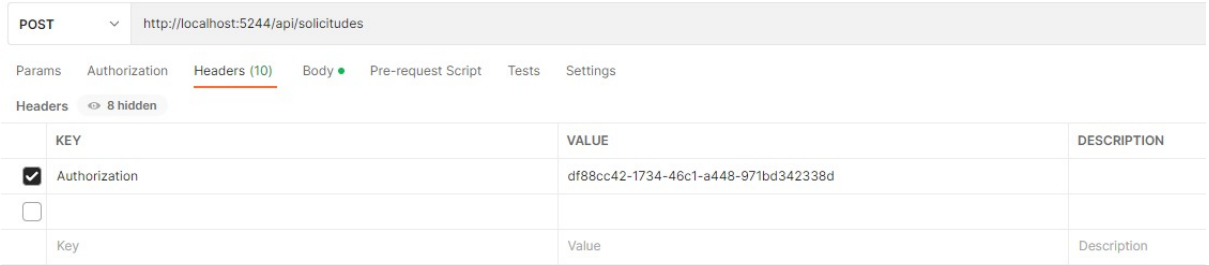
En primer lugar, cabe aclarar que para navegar dentro del sistema, el usuario debe obligatoriamente loguearse en el mismo, es decir autenticarse. El estado del cliente no se mantiene logueado y cada vez que se hace un POST de sesión, se inicia una sesión - en caso que las credenciales ingresadas sean las correctas - se crea un token de la misma, retornado en el body de la request. Dicho token se envía como header Authorization dentro de la request HTTP y el sistema validará si el usuario tiene los permisos necesarios para realizar la acción que corresponda con dicho endpoint.

A continuación se detalla la API de session para la creación de una sesión de un usuario registrado.

Cada endpoint expuesto tiene asociado una lista de roles de usuarios los cuales tienen permiso sobre dicho endpoint. Los roles de usuario se definen de la siguiente manera en la base de datos:

ID	ROL
1	Admin
2	Owner
3	Employee

En la sección Recursos se listan todos los endpoints y se describe qué rol debe tener el usuario para tener el permiso de acceder al recurso.



Para la autenticación y permisos de cada uno de los roles se utilizan los filtros para verificar que usuario puede acceder a dicho endpoint. Se expresan de la siguiente manera en las data anotations:

```
[ServiceFilter(typeof(AuthorizationAttributeFilter))]
```

En el proyecto WebApi.Filters es donde se controlan y gestionan los filtros. En este punto es donde se setean los códigos de error a retornar en la API, según cada una de las excepciones que se controlan en el sistema. A su vez se verifica si el usuario tiene los permisos correspondientes para acceder al endpoint. Mediante el método OnAuthorization, se realiza el chequeo del token y roles necesarios para acceder a cada uno de los recursos permitiéndonos desacoplar la lógica y separar la responsabilidad de autenticación para la clase AuthorizationAtrtributeFilter.

Descripción general de códigos de response

Descripción general de códigos de error

Para el desarrollo de una Web API se utilizan diversos códigos HTML que otorgan una mejor información al cliente sobre el estado de las respuestas de la API. En este caso fueron utilizados los siguientes códigos: 200 Ok, 400 Bad Request, 401 Unauthorized, 403 Forbidden y 404 Not Found. Los mismos se procesaron como excepciones propias, las cuales fueron mapeadas posteriormente a mensajes de error de HTML.

200: Ok
Se lanza cuando la solicitud fue procesada de forma correcta.

400: Bad Request
Este error se lanza cuando ocurre un error lógico, datos faltantes o inválidos, se traduce a nivel lógico como ValidationException

401: Unauthorized
Este error ocurre cuando se trata de hacer una acción y no se tienen los permisos adecuados, particularmente cuando alguien trata de hacer acciones ajenas a su rol. A nivel lógico se traduce como AuthorizationException

403: Forbidden
Este error ocurre cuando se ingresa la contraseña incorrecta o hacer PUT con un token sin permiso. Se traduce como AuthenticationException

404: Not Found
Este error se lanza cuando se solicita un recurso que no existe, a nivel de lógica se traduce como ResourceNotFoundException.

500: Internal Server Error
Este error ocurre cuando hay un fallo no manejado en el servidor, es autogenerado así que no lo hicimos.

Recursos de la API

Recurso: Session

Crear sesión

FUNCIONALIDAD: Crear sesión		
URL: POST /sessions		

BODY:
{
“userName”: <string>,
“password”: <string>
}		

RESPUESTA	**CÓDIGO**	**BODY**

Request correcta	200: OK	{
		“token”: <string>
		}

Formato inválido de la request	400: Bad Request	{
		“error”: “validationError”
		“message”: “resource does not exist”
		}

Token de sesión inválido	401: Unauthorized	{
		“error”: ”permissionError”,
		“message”: “Invalid token”
		}

Password incorrecta	403: Forbidden	{
		“error”:”Forbidden”,
		“message”: "This action is forbidden"
		}

Username incorrecto	404: Not found	{
		“error”: “notFoundError”
		“message”: ”resource not found”
		}

Error interno del servidor	500: Internal Server Error	{
		“error”: ”InternalServerError”,
		“message”: “Internal server error”
		}

Recurso: Invitation

Crear invitación

FUNCIONALIDAD: Crear invitación		
URL: POST /invitations		
HEADER: Authorization:{token}		
ROLES AUTORIZADOS: Admin		

BODY:
{
“userName”: <string>,
“roleName”: <string>,
“pharmacyName”: <string>
}		

RESPUESTA	**CÓDIGO**	**BODY**

Request correcta	200: OK	{
		“userName”: <string>,
		“roleName”: <string>,
		“invitationCode”: <string>,
		“pharmacyName”: <string>
		}

Formato inválido de la request	400: Bad Request	{
		“error”: “validationError”
		“message”: “resource does not exist”
		}

Usuario sin autorización de crear invitación.	401: Unauthorized	{
		“error”:”Permission error”,
		“message”:"Invalid token"
		}

Usuario no tiene permiso para realizar la acción	403: Forbidden	{
		“error”:”Forbidden”,
		“message”: "This action is forbidden"
		}

Error interno del servidor	500: Internal Server Error	{
		“error”:”InternalServerError”,
		“message”: "Internal server error"
		}

Actualizar invitación

FUNCIONALIDAD: (Crear Usuario)		
URL: PUT /invitations/{invitationCode}		

BODY:
{
“userName”: <string>,
“email”: <string>,
“address”: <string>,
“password”: <string>
}		

RESPUESTA	**CÓDIGO**	**BODY**

Request correcta	200: OK	{
		“userName”: <string>,
		“roleName”: <string>,
		“pharmacyName”: <string>,
		“email”: <string>,
		“address”: <string>,
		}

Formato inválido de la request	400: Bad Request	{
		“error”: “validationError”
		“message”: “invalid invitation code”
		}

Error interno del servidor	500: Internal Server Error	{
		“error”:”InternalServerError”,
		“message”: "Internal server error"
		}

Recurso: Pharmacy

Crear farmacia

FUNCIONALIDAD: Crear farmacia		
URL: POST /pharmacies		
HEADER: Authorization:{token}		
ROLES AUTORIZADOS: Admin		

BODY:
{
“name”: <string>,
“address”: <string>
}		

RESPUESTA	**CÓDIGO**	**BODY**

Request correcta	200: OK	{
		“id”:<int>,
		“name”: <string>,
		“address”: <string>
		}

Formato inválido de la request	400: Bad Request	{
		“error”:”validationError”,
		“message”: “Pharmacy name already exists”
		}

Token de sesión inválido	401: Unauthorized	{
		“error”:”permissionError”,
		“message”:"Invalid token"
		}

Usuario no tiene permiso para realizar la acción	403: Forbidden	{
		“error”:”Forbidden”,
		“message”: "This action is forbidden"
		}

Error interno del servidor	500: Internal Server Error	{
		“error”:”InternalServerError”,
		“message”: "Internal server error"
		}

Recurso: Solicitude

Crear una solicitud

FUNCIONALIDAD: Crear una solicitud de reposición de stock		
URL: POST /solicitudes		
HEADER: Authorization:{token}		
ROLES AUTORIZADOS: Employee		

BODY:
{
“solicitudeItems”: [
{
“drugCode”: <string>,
“quantity”: <int>
}
]
}		

RESPUESTA	**CÓDIGO**	**BODY**

Request correcta	200: OK	{
		“id”: <int>,
		“state”: <string>,
		“date”:<DateTime>,
		“employee”<string>,
		“solicitudeItems”: [
		{
		“drugCode”: <string>,
		“quantity”: <int>
		}
		]
		}

Formato inválido de la request	400: Bad Request	{
		“error”:”validationError”,
		“message”: “Incorrect format ”
		}

Token de sesión inválido	401: Unauthorized	{
		“error”:”Unauthroized”,
		“message”:"Invalid token"
		}

Usuario no tiene permiso para realizar la acción	403: Forbidden	{
		“error”:”Forbidden”,
		“message”: "This action is forbidden"
		}

Drug Code erróneo	404: Not Found	{
		“error”: “notFoundError”
		“message”: “resource does not exist”
		}

Error interno del	500: Internal	{

servidor	Server Error	“error”:”InternalServerError”, “message”: "Internal server error" }

Obtener todas las solicitudes

FUNCIONALIDAD: Obtener todas las solicitudes		
URL: GET /solicitudes		
QUERY STRINGS: dateFrom: fecha de inicio en formato “yyyy-MM-ddTHH:mm:ss” dateTo: fecha fin en formato “yyyy-MM-ddTHH:mm:ss” drugCode: código del medicamento state: estado de la solicitud (pending, approved, rejected)		
HEADER: Authorization:{token}		
ROLES AUTORIZADOS: Owner, Employee		
RESPUESTA	**CÓDIGO**	**BODY**

Request correcta	200: OK	[
		{
		“id”: <int>,
		“employee”: <string>
		“state”: <string>,
		“Date”: <DateTime>,
		“solicitudeItems”: [
		{
		“drugCode”: <string>,
		“quantity”: <int>
		}
		]
		}
		]

Formato inválido de la request	400: Bad Request	{
		“error”: “validationError”,
		“message”: “The date from should be before date to”
		}

Token de sesión inválido	401: Unauthorized	{
		“error”:”Unauthroized”,
		“message”:"Invalid token"
		}

Usuario no tiene permiso para realizar la acción	403: Forbidden	{
		“error”:”Forbidden”,
		“message”: "This action is forbidden"
		}

Error interno del servidor	500: Internal Server Error	{
		“error”:”InternalServerError”,
		“message”: "Internal server error"
		}

Actualizar estado de solicitud

FUNCIONALIDAD: Actualizar estado de una solicitud		
URL: PUT /solicitudes/{solicitudeId}		
HEADER: Authorization:{token}		
ROLES AUTORIZADOS: Owner		

BODY:
{
“status”: <string>
}		

VALUES		
RESPUESTA	**CÓDIGO**	**BODY**

Request correcta	200: OK	{
		“id”: <int>,
		“employee”: <string>
		“state”: <string>,
		“Date”: <DateTime>,
		“solicitudeItems”: [
		{
		“drugCode”: <string>,
		“quantity”: <int>
		}
		]
		}

Formato inválido de la request	400: Bad Request	{
		“error”:”validationError”,
		“message”: “State doesn’t exist ”
		}

No se encuentra el recurso	404: Not Found	{
		“error”:”notFoundError”,
		“message”: “Solicitude doesn’t exist ”
		}

Token de sesión inválido	401: Unauthorized	{
		“error”:”Unauthroized”,
		“message”:"Invalid token"
		}

Usuario no tiene permiso para realizar la acción	403: Forbidden	{
		“error”:”Forbidden”,
		“message”: "This action is forbidden"
		}

Error interno del servidor	500: Internal Server Error	{
		“error”:”InternalServerError”,
		“message”: "Internal server error"
		}

Recurso: Drug

Crear un medicamento

FUNCIONALIDAD: Crear un medicamento		
URL: POST /drugs		
HEADER: Authorization:{token}		
ROLES AUTORIZADOS: Employee		

BODY:
{
“drugCode”: <string>,
“name”:<string>,
“symptoms”: <string>,
“presentation”: <string>,
“quantityPerPresentation”: <float>,
“measureUnit”: <string>,
“price”: <double>,
“recipieNedeed”: <bool>
}		

RESPUESTA	**CÓDIGO**	**BODY**

Request correcta	200: OK	{
		“id”:<Int>,
		“drugCode”: <string>,
		“name”: <string>,
		“symptoms”: <string>,
		“presentation”: <string>,
		“presentationQuantity”: <int>,
		“measureUnit”: <string>,
		“price”: <double>,
		“needsPrescription”: <bool>
		}

Formato inválido de la request	400: Bad Request	{
		“error”:”validationError”,
		“message”: “The drug code already exists in this pharmacy ”
		}

Token de sesión inválido	401: Unauthorized	{
		“error”:”Unauthroized”,
		“message”:"Invalid token"
		}

Usuario no tiene permiso para realizar la acción	403: Forbidden	{
		“error”:”Forbidden”,
		“message”: "This action is forbidden"
		}

Error interno del servidor	500: Internal Server Error	{
		“error”:”InternalServerError”,
		“message”: "Internal server error"
		}

Eliminar un medicamento

FUNCIONALIDAD: Eliminar un medicamento		
URL: DELETE /drugs/{drugId}		
HEADER: Authorization:{token}		
ROLES AUTORIZADOS: Employee		
RESPUESTA	**CÓDIGO**	**BODY**

Request correcta	200: OK	{
		“message”: “Se eliminó el medicamento correctamente”
		}

No se encuentra el recurso	404: Not Found	{
		“error”:”NonExistentDrug”,
		“message”: “Drug doesn’t exist ”
		}

Token de sesión inválido	401: Unauthorized	{
		“error”:”Unauthroized”,
		“message”:"Invalid token"
		}

Usuario no tiene permiso para realizar la acción	403: Forbidden	{
		“error”:”Forbidden”,
		“message”: "This action is forbidden"
		}

Error interno del servidor	500: Internal Server Error	{
		“error”:”InternalServerError”,
		“message”: "Internal server error"
		}

Obtener todos los medicamentos

FUNCIONALIDAD: Obtener todos los medicamentos		
URL: GET /drugs		
QUERY STRINGS: drugName: nombre de medicamento withStock: indica si tiene o no stock (true, false)		
ROLES AUTORIZADOS: Anonymous		
RESPUESTA	**CÓDIGO**	**BODY**

Request correcta	200: OK	[
		{
		“id”:<Int>,
		“drugCode”: <string>,
		“stock”: <int>,
		“pharmacy”:<string>
		}

		]

Error interno del servidor	500: Internal Server Error	{
		“error”:”InternalServerError”,
		“message”: "Internal server error"
		}

Obtener un medicamento

FUNCIONALIDAD: Obtener un medicamento		
URL: GET /drugs/{drugId}		
HEADER: Authorization:{token}		
ROLES AUTORIZADOS: Employee		
RESPUESTA	**CÓDIGO**	**BODY**

Request correcta	200: OK	{
		“id”:<int>,
		“drugCode”: <string>,
		“name”: <string>,
		“symptoms”: <string>,
		“presentation”: <string>,
		“presentationQuantity”: <int>,
		“measureUnit”: <string>,
		“price”: <double>,
		“recipieNedeed”: <bool>
		}

No se encuentra el recurso	404: Not Found	{
		“error”:”NonExistentDrug”,
		“message”: “Drug doesn’t exist ”
		}

Token de sesión inválido	401: Unauthorized	{
		“error”:”Unauthroized”,
		“message”:"Invalid token"
		}

Usuario no tiene permiso para realizar la acción	403: Forbidden	{
		“error”:”Forbidden”,
		“message”: "This action is forbidden"
		}

Error interno del servidor	500: Internal Server Error	{
		“error”:”InternalServerError”,
		“message”: "Internal server error"
		}

Recurso: Purchase

Comprar un medicamento

FUNCIONALIDAD: Comprar un medicamento		
URL: POST /purchases		

BODY:
{
“userEmail”: <string>,
“pharmacyName”: <string>,
“items”: [
{
“drugCode”: <string>,
“quantity”: <int>
}
]
}		

RESPUESTA	**CÓDIGO**	**BODY**

Request correcta	200: OK	{
		“id”:<int>,
		“userEmail”: <string>,
		“createdDate”: <string>,
		“price”:<double>,
		“pharmacyName”: <string>
		“items”: [
		{
		“drugCode”: <string>,
		“quantity”: <int>
		}
		]
		}

Formato inválido de la request	400: Bad Request	{
		“error”:”NonExistentDrug”,
		“message”: “Drug doesn’t exist ”
		}

Usuario sin autorización de crear invitación.	401: Unauthorized	{
		“error”:”permissionError”,
		“message”:"Invalid token"
		}

Usuario no tiene permiso para realizar la acción	403: Forbidden	{
		“error”:”Forbidden”,
		“message”: "This action is forbidden"
		}

Error interno del servidor	500: Internal Server Error	{
		“error”:”InternalServerError”,
		“message”: "Internal server error"
		}

Obtener todas las compras de medicamentos

FUNCIONALIDAD: Obtener todas las compras de medicamentos		
URL: GET /purchases		
QUERY STRINGS: dateFrom: fecha de inicio en formato “yyyy-MM-ddTHH:mm:ss” dateTo: fecha fin en formato “yyyy-MM-ddTHH:mm:ss”		
HEADER: Authorization:{token}		
ROLES AUTORIZADOS: Owner, Employee		
RESPUESTA	**CÓDIGO**	**BODY**

Request correcta	200: OK	{
		“totalPrice”:<double>,
		“purchases”:
		[
		{
		“id”:<int>,
		“userEmail”: <string>,
		“createdDate”: <string>,
		“price”:<double>,
		“pharmacyName”: <string>
		“items”: [
		{
		“drugCode”: <string>,
		“quantity”: <int>
		}
		]
		}
		]
		}

Usuario sin autorización de crear invitación.	401: Unauthorized	{
		“error”:”Unauthroized”,
		“message”:"Invalid token"
		}

Usuario no tiene permiso para realizar la acción	403: Forbidden	{
		“error”:”Forbidden”,
		“message”: "This action is forbidden"
		}

Error interno del servidor	500: Internal Server Error	{
		“error”:”InternalServerError”,
		“message”: "Internal server error"
		}