Universidad ORT Uruguay

Facultad de Ingeniería

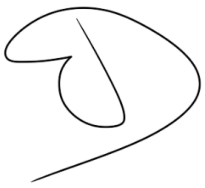
OBLIGATORIO 1

DISEÑO DE APLICACIONES 2

-

Evidencia del diseño y especificación de la API

Nicolás Edelman
251363



Juan Diego Etcheverry
252443

Mateo Goldwasser
239420

Tutores: Francisco Bouza, Juan Irabedra, Santiago Tonarelli

2023

Indice:

Link al repositorio:
https://github.com/IngSoft-DA2-2023-2/239420-252443-251363

¿Por qué nuestra API de ecommerce es RESTful?

Las APIs RESTful se guían por 6 principales restricciones que plantea REST, a continuación nos estaremos refiriendo a cada una de estas restricciones y fundamentando cómo impactan estas en nuestro sistema.

1 - Client-Server

Hemos diseñado nuestra API para interactuar con los clientes a través de una WebApp. La WebApp se comunica con la API, que a su vez se encarga del acceso a la base de datos. Este enfoque nos brinda una mayor flexibilidad para crear interfaces que funcionen en diversas plataformas (Cross Platform) y simplifica la escalabilidad del sistema, ya que los componentes en el servidor están optimizados y simplificados.

2 - Stateless

Nuestras request no mantienen un estado en el sistema. El sistema el mismo podrá trabajar solamente con la información que se pasa en esa request individualmente, sin necesidad de que se realice una request antes que genere un cambio en el sistema para luego utilizar otra. Sin embargo, mantenemos cierto estado en la base de datos para poder manejar las sesiones. Igualmente, esta práctica sigue siendo considerada stateless.

3 - Cacheable

Lo ideal sería que nuestra API sea cacheable para que cuando el cliente haga la misma request, la response pueda ser reutilizada. Esto tambien ayuda a que el servidor no tenga tanto trabajo. De igual forma, nuestro sistema al ser como una primera entrega creemos que no aplica. Sin

embargo es posible que el framework que utilizamos nos esté cacheando algunas cosas para mejorar la performance.

4 - Interfaz Uniforme

En nuestro sistema, aplicamos un conjunto de modelos para gestionar los datos de entrada y salida de las solicitudes. Estos modelos se utilizan exclusivamente en los controladores. Cuando avanzan a la segunda etapa de la Lógica de Negocios, se transforman en objetos del Dominio. Estos objetos contienen toda la información que no es necesario que el cliente conozca.

El funcionamiento de todo el sistema sigue un patrón similar. La solicitud ingresa a un controlador, la información se parsea de JSON a un Modelo, luego se llama a la Lógica correspondiente de la parte del sistema que desea ser afectada. Esta lógica se encarga de aplicar las reglas de negocio previo a interactuar con la base de datos. Después, la solicitud se traslada a la base de datos, que se encarga de la lógica de la base de datos. Finalmente, se devuelve lo que se debía recuperar o no.

Este enfoque se mantiene consistente en todo el sistema, y cada solicitud sigue el mismo curso. Gracias a esta estandarización y uniformidad, el sistema es más fácil de entender.

5 - Sistema en Capas

El sistema está dividido en las capas principales Web Api, Logic y Base de datos, teniendo también capas de apoyo como Domain.

6 - Code on demand

Este principio no se aplica al alcance del obligatorio.

Autenticación de Request:

En nuestro sistema, para saber si un cliente tiene una sesion activa, nos tiene que mandar en la request, una cookie con la clave “Authorization” y un valor. Luego validamos ese valor en la tabla de Sesiones en nuestra base de datos. Allí validamos si existe esa sesión. Para algunas acciones como el Signup y el Login no es necesario mandar una Authorization.

Algo que implementamos fue que se puede tener una sesion sin un usuario asociado. Esto es útil cuando alguien quiere realizar una accion que no sea necesario estar logueado. Sin embargo, al momento de hacer Signup o Login se asocia el Authorization con el nuevo usuario.

Códigos de Error:

Codigo	**Mensaje**
200	OK
201	Created
400	Bad Request (errores de Validation)
401	Unauthorized (no mando Auth)
403	Forbidden (no tiene permiso)
500	Server error

Recursos:

La URL base de la api es : https://localhost:7034

User:

Endpoint	Description	Headers	Body	Params	Responses
GET /api/users/{id}	Get user	Authorization			200 (User)
DELETE /api/users/{id}	Delete user	Authorization			200 (User)
PUT /api/users/{id}	Update user	Authorization	Email: string, Address:string, Role:int		200 (User)
POST /api/users	Create user	Authorization	Email:string, Address:string, Role:int, Password:string, PasswordConfirmation:string		201 (User)
POST /api/login	Login		Email: string, Password:string		200
POST /api/signup	Signup		Email:string, Address:string, Password:string, PasswordConfirmation:string		201 (User)
POST /api/logout	Logout	Authorization			200

Products:

Endpoint	Description	Headers	Body	Params	Responses
POST /api/products	Add product	Authorization	Name:string, Price:float, Description:string, Brand:Brand, Category:Category, Colors:[Color]		201 (Product)
GET /api/products	Get products			Text:string, Brand:string, Category:string	200 (Products)
GET /api/products/{id}	Get product				200 (Products)
DELETE /api/products/{id}	Delete product	Authorization			200

Purchases:

Endpoint	Description	Headers	Body	Params	Responses
GET /api/purchases/{id}	Get my purchase history	Authorization			200 (Purchases)
POST /api/purchases	Add purchase	Authorization	ProductIds:[Guid]		201 (Purchase)
GET /api/purchases	Get all purchases	Authorization			200 (Purchases)