Universidad ORT Uruguay

Facultad de Ingeniería

Bernard Wand Polak

Diseño de Aplicaciones II

Evidencia del diseño y especificación de la API

https://github.com/ORT-DA2/164660-185720-234059

Santiago Alvarez – N° Estudiante: 185720

Juan Castro – N° Estudiante: 164660

Marcelo Tilve – N° Estudiante: 234059

Contenido

Criterios para asegurar que la API cumple con los criterios REST.

Se puede asegurar que la API cumple con criterios REST dado que, por ejemplo, todas las requests están pensadas para que el servidor de aplicaciones sea "stateless" y toda la información necesaria para llevar a cabo una acción esté del lado del cliente y no del lado del servidor. Esto permite tener una independencia al estado del servidor.

No importa la cantidad de servidores (escalabilidad del sistema), si estos caen o vuelven a levantarse, las request serán recibidas y procesada de la misma manera.

Además, siempre se mantiene la url base lo más simple e intuitiva posible, se utilizan sustantivos en plural y no verbos.

No hay estado, esto quiere decir que cada petición que recibe el servidor es independiente.

Se utilizan los verbos Http GET, POST, PUT y DELETE para el acceso, creación, actualización y borrado de recursos.

Por otro lado, los mensajes que se envían al cliente son auto descriptivos, por lo que contienen toda la información necesaria y describen lo que ocurrió y cómo eventualmente continuar.

Descripción del mecanismo de autenticación de requests.

Se definió un filtro de autenticación por el que pasan todas las requests que así lo requieran, el filtro toma el header “Authorization” del request y valida dicho token con los existentes en el sistema. Para esto se definió una tabla de sesiones (Session) en donde se almacenan las sesiones de los usuarios y el identificador del usuario en cuestión. Se definió que los token de autenticación sean con formato *GUID* - Identificador único global- el cual es una implementación de Microsoft del *UUID*.

De esta forma al recibir un token primero se valida que este no sea vacío para luego, primero validar que su formato corresponda al de un *GUID* y luego validar su existencia en la tabla de Sesiones.

En caso de no existir dicha sesión en la tabla de Sesiones se devuelve al cliente 401 Unauthorized y en caso de existir dicha sesión se valida que la misma esté asociada a un usuario con el rol correspondiente, se envía el rol a la capa de negocio y en caso de no corresponderse se devuelve al usuario un 403 Forbidden.

Para el manejo de los Roles del lado de la Web Api se utiliza una decoración a nivel de endpoint del formato ***[AuthorizationFilter(nameof(RoleType.Value))]*** siendo *RoleType* un enumerado con los siguientes valores posibles: Administrator (Administrador), Owner (Dueño) o Employee (Empleado).

Descripción general de códigos de error.

Para manejar los códigos de error se lanzan excepciones de diferentes tipos que luego son capturadas por un ExceptionFilter a nivel global el cual se encarga de devolver los códigos de error y los mensajes asociados a los mismos.

Se utilizaron dos excepciones custom que extienden de "*Exception*" siendo estas: ***InvalidResourceException*** y ***ResourceNotFoundException***. La primera se utiliza en los casos donde los datos enviados no son correctos o válidos siendo luego capturada por el filtro y devolviendo 400, la segunda siendo utilizada en los casos donde no se encuentra una instancia específica del dominio devolviendo 404.

En caso de que la validación del el token de Autenticación no sea correcta se lanza una excepción de tipo ***FormatException*** que luego es capturada por el filtro devolviendo al cliente 400 con el mensaje "Invalid token format" (Formato Invalido del token).

En caso de que un error no se corresponda con el de los antes mencionados se captura por parte del filtro devolviendo al cliente status 500 y el mensaje específico de ese error.

El filtro de excepciones siempre retorna el siguiente formato en forma de JSON:

```
{
  "message": "Mensaje de error"
}
```

A su vez, se definió el status 401 (Unauthorized) y mensaje "***Invalid authorization token***" para cuando el token de sesión no existe en la base de datos y un status 403 (Forbidden) con mensaje "***Forbidden role***" cuando el usuario existe en la base de datos pero su rol no permite acceder al recurso.

Descripción de los resources de la API.

LoginController

- Login([FromBody] LoginModelRequest)
    - POST
    - [HttpPost]
    - Si la sesión no existe se crea una y se genera un nuevo GUID, si existe se devuelve el GUID.
    - Se devuelve un token de formato GUID.
    - No requiere autenticación de usuario.
    - Acceso para cualquier usuario.
    - /api/login
    - Ejemplo de petición
        - POST /api/login
        - Se le deberán pasar los datos de usuario y contraseña en el body
    - Códigos de estado y salida
        - Ok 200 - Token de sesión devuelto de forma correcta.
        - Error 400 - Cuando el nombre de usuario está vacío. ("Invalid Username")
        - Error 400 - Cuando la contraseña está vacía o la misma no coincide con la del usuario existente. ("Invalid Password")
        - Error 404 - Cuando no se encuentra en el sistema un usuario con el nombre de usuario enviado. ("The user does not exist")

UsersController

- CreateUser([FromBody] UserModelRequest)
    - POST
    - [HttpPost]
    - Se crea un usuario a partir de una invitación con un rol asociado y en caso de que corresponda una farmacia asociada.
    - Retorna al usuario creado.
    - No requiere autenticación de usuario.
    - Acceso para cualquier usuario.
    - /api/users
    - Ejemplo de petición
        - POST /api/users
        - Se le deberán pasar los datos del usuario en el body
    - Códigos de estado y salida
        - Ok 200 - Usuario creado de forma correcta.
        - Error 400 - Cuando el código de invitación no es válido. ("Invalid UserCode")
        - Error 400 - Cuando el nombre de usuario no es válido. (“Invalid Username”)
        - Error 400 - Cuando el formato del Email no es válido. (“Invalid Email”)
        - Error 400 - Cuando el formato del Password es inválido. ("Invalid Password")
        - Error 400 - Cuando el formato de la dirección es invalido ("Invalid Address")
        - Error 400 - Cuando el nombre de usuario ya existe en el sistema. ("Invalid Username, Username already exists")

- Error 400 - Cuando el email del usuario ya existe en el sistema ("Invalid Email, Email already exists")
- Error 404 - Cuando no se encuentra la invitación en el sistema o ya está activa ("Invitation not found or is not currently active").

PurchasesController

- All()
  - GET
  - [HttpGet]
  - [Route("[action]")]
  - Crea una compra junto a sus detalles y el email del comprador.
  - Retorna una lista de todas las compras sin el detalle de las mismas.
  - Requiere autenticación de usuario.
  - Acceso solo a usuarios de Rol ***Employee***.
  - /api/purchases
  - Ejemplo de petición
    - GET /api/purchases
    - Se le deberá pasar un Header ‘Authorization’ con un token.
  - Códigos de estado y salida
    - Ok 200 - Retorna una lista con todas las compras registradas sin el detalle de las mismas.
    - Error 401 - Token de autorización no válido. (“Invalid authorization token”)
    - Error 403 - Acceso con un Rol prohibido.(“Forbidden role”)

- ByMonth([FromQuery] int? year, [FromQuery] int? month)
    - GET
    - [HttpGet]
    - [Route("[action]")]
    - Retorna una lista de las compras del mes con el detalle de las mismas.
    - Requiere autenticación de usuario.
    - Acceso solo a usuarios de Rol ***Owner***.
    - /api/purchases/bymonth
    - Ejemplo de petición
        - GET /api/purchases/bymonth?year=2022&month=09
        - Se le deberá pasar un Header ‘Authorization’ con un token.
    - Códigos de estado y salida
        - Ok 200 - Retorna una lista de las compras del mes con el detalle de las mismas.
        - Error 400 - cuando no se envía un mes y/o año por query param. (“Year and Month are required”)
        - Error 400 - Cuando el año no es válido. ("Invalid year {year}: should be 'yyyy' (1900-2099) format")
        - Error 400 - Cuando el mes no es válido. ("Invalid month {month}: should be 'MM' (01-12) format")
        - Error 401 - Token de autorización no válido. (“Invalid authorization token”)
        - Error 403 - Acceso con un Rol prohibido.(“Forbidden role”)

- CreatePurchase([FromBody] PurchaseModelRequest)
    - POST
    - [HttpPost]
    - Se crea una compra con el detalle de sus ítems.
    - Retorna la compra realizada por el usuario.
    - No requiere autenticación de usuario.
    - Acceso para cualquier usuario.
    - /api/purchases
    - Ejemplo de petición
        - POST /api/purchases
    - Códigos de estado y salida
        - Ok 200 - Cuando la compra se registra correctamente.
        - Error 400 - Cuando el formato del Email enviado por el comprador tiene formato invalido. (“Invalid Email”)
        - Error 400 - Cuando la lista de ítems está vacía. (“The list of items can't be empty”)
        - Error 400 - Cuando la cantidad de un medicamento a comprar es mayor que la del stock disponible ("The Drug {drug.Code} is out of stock")
        - Error 400 - Cuando la cantidad a comprar de algún medicamento es <= 0. ("The Quantity is a mandatory field")
        - Error 400 - Cuando se envía una fecha de compra vacía. ("The purchase date is a mandatory field")

- Error 400 - Cuando se envía una farmacia vacía. ("Pharmacy Id is a mandatory field")
- Error 404 - Cuando no se encuentra la farmacia asociada a la compra ("Pharmacy {Pharmacy.Id} not found")
- Error 404 - Cuando no se encuentra alguno de los medicamentos por los que se realiza la compra ("Drug {drugCode} not found")

InvitationController

- CreateInvitation([FromBody] InvitationModelRequest)
  - Método: POST
  - Crea una invitación con los datos proporcionados, solo los usuarios administradores pueden realizarla.
  - Ejemplo de petición:
    - POST api/invitation
  - Códigos de estado y salida
    - 200 Ok - Cuando la invitación se crea correctamente
    - 400 Bad Request
      - Si la invitación es para un Administrador, no se debe indicar una farmacia “A pharmacy is not required.”.
      - Si la invitación es para un Dueño o Empleado, se debe indicar una farmacia “A pharmacy is required.”.
      - Los nombre de usuario son únicos, por lo que si ya existe una invitación creada para un usuario, el sistema no deja continuar mostrando mensaje “Invitation already exist.”
      - Si no se pasa un usuario “Invalid UserName.”

- Si no se pasa un rol o el rol no es válido (los especificados en el sistema) “Invalid Rol.”.

StockRequestController

- CreateStockRequest([FromBody] StockRequestModelRequest)
  - Método: POST
  - Crea una solicitud de stock con los datos proporcionados, solo los usuarios con rol empleado pueden utilizarla.
  - Ejemplo de petición:
    - POST api/stockRequest
  - Códigos de estado y salida
    - 200 Ok - Cuando la solicitud de stock se crea correctamente.
    - 400 Bad Request
      - Si la solicitud tiene un medicamento que no es válido “Stock request has invalid drug.”.
      - Si la solicitud no tiene detalle de medicamentos “Invalid stock details.”.
- ApproveStockRequest(int id)
  - Método: PUT
  - Aprueba una solicitud de stock, solo los usuarios con rol dueño pueden utilizarla.
  - Ejemplo de petición:
    - PUT api/stockRequest/approveStockRequest/{id}
    - Códigos de estado y salida:
      - 200 Ok - Si la solicitud fue aprobada correctamente

- 400 Bad Request
    - Si la solicitud ya fue aprobada
    - Si la solicitud ya fue rechazada

- RejectStockRequest(int id)
    - Método: PUT
    - Rechaza una solicitud de stock, solo los usuarios con rol dueño pueden utilizarla.
    - Ejemplo de petición:
        - PUT api/stockRequest/rejectStockRequest/{id}
    - Códigos de estado y salida:
    - 200 Ok - Si la solicitud fue aprobada correctamente
    - 400 Bad Request
        - Si la solicitud ya fue aprobada
        - Si la solicitud ya fue rechazada

- GetAll()
    - Método: GET
    - Obtiene las solicitudes de reposición de stock de medicamentos realizadas por los empleados, sólo los usuarios con rol dueño pueden utilizarla.
    - Ejemplo de petición:
        - GET api/stockRequest
    - Códigos de estado y salida:
    - 200 Ok
    - 403 Forbidden

        - Si el usuario no tiene permisos.

- ByEmployee([FromQuery] StockRequestSearchCriteriaModelRequest)
    - Método: GET
    - Obtiene las solicitudes de reposición de stock de medicamentos realizadas por el empleado logueado, solo los usuarios con rol empleado.
    - Este endpoint utiliza filtros que vienen en el request dentro de query params
    - Ejemplo de petición:
        - GET /api/stockRequest/byemployee?Code=XXX
    - Códigos de estado y salida:
        - 200 Ok
        - 403 Forbidden
            - Si el usuario no tiene permisos.

PharmacyController

Todos los endpoints de este controlador utilizan el ExceptionFilter y el AuthorizationFilter, teniendo en cuenta para este último que solamente los usuarios Administradores pueden acceder a los recursos.

- GetAll([FromQuery] PharmacySearchCriteria pharmacySearchCriteria)
    - Método GET
    - Obtiene todas las farmacias que tengan el mismo nombre y/o dirección del objeto pharmacySearchCriteria pasado por query params. En caso de que la query sea vacía, se devuelven todas las farmacias.
    - Ejemplos de petición:
        - GET /api/Pharmacy

        - GET /api/Pharmacy?Name=pharmacy&Address=address
    - Códigos de estado y salida:
        - 200 OK
        - 403 Forbidden
            - Si el usuario no tiene permisos.

- GetById([FromRoute] int id)
    - Método GET
    - Obtiene si existe la farmacia con el id de la uri de la request.
    - Ejemplo de petición:
        - GET /api/Pharmacy/{id}
    - Códigos de estado y salida:
        - 200 OK
        - 403 Forbidden
            - Si el usuario no tiene permisos.
        - 404 ResourceNotFoundException
            - Si la farmacia a obtener no existe

- Create([FromBody] PharmacyModel pharmacyModel)
    - Método POST
    - Crea una farmacia nueva y la almacena en la base de datos
    - Ejemplo de petición:
        - POST /api/Pharmacy

        - Se le deberán pasar los datos de la farmacia en el body

    - Códigos de estado y salida:

        - 200 OK

        - 403 Forbidden

            - Si el usuario no tiene permisos.

        - 400 Bad Request

            - Si la farmacia a crear ya existe

            - Si la farmacia de la request es null

            - Si el nombre de la farmacia es null o vacío

            - Si el largo del nombre de la farmacia es mayor a 50 caracteres

            - Si la dirección de la farmacia es null o vacío

            - Si la lista de usuarios de la farmacia es null

            - Si la lista de medicamentos de la farmacia es null

- Update([FromRoute] int id, [FromBody] PharmacyModel updatedPharmacy)

    - Método PUT

    - Modifica si existe una farmacia existente en la base de datos a partir del objeto de la request updatedPharmacy

    - Ejemplo de petición:

        - PUT /api/Pharmacy/{id}

        - Se le deberán pasar los datos actualizados de la farmacia en el body

    - Códigos de estado y salida:

        - 200 OK

- 403 Forbidden
  - Si el usuario no tiene permisos.
- 400 Bad Request
  - Si la farmacia de la request es null
  - Si el nombre de la farmacia es null o vacío
  - Si el largo del nombre de la farmacia es mayor a 50 caracteres
  - Si la dirección de la farmacia es null o vacío
  - Si la lista de usuarios de la farmacia es null
  - Si la lista de medicamentos de la farmacia es null
- 404 ResourceNotFound
  - Si la farmacia a modificar no existe

- Delete([FromRoute] int id)
  - Método DELETE
  - Elimina la farmacia con el id de la request si existe y no tiene medicamentos asociados.
  - Ejemplo de petición:
    - DELETE /api/Pharmacy/{id}
  - Códigos de estado y salida:
    - 200 OK
    - 403 Forbidden
      - Si el usuario no tiene permisos.
    - 404 ResourceNotFound

- Si la farmacia a eliminar no existe

DrugController

- GetAll([FromQuery] DrugSearchCriteria drugSearchCriteria)
  - Método GET
  - Obtiene todos los medicamentos que tengan el mismo nombre y/o código del objeto drugSearchCriteria pasado por query params. En caso de que la query sea vacía, se devuelven todos los medicamentos.
  - Ejemplos de petición:
    - GET /api/Drug
    - GET /api/Drug?Code=drugCode&Name=drugName
  - Códigos de estado y salida:
    - 200 OK
    - 403 Forbidden
      - Si el usuario no tiene permisos.
- GetById([FromRoute] int id)
  - Método GET
  - Obtiene si existe el medicamento con el id de la uri de la request.
  - Este endpoint solo puede ser accedido por usuarios administradores, controlando esto con el AuthorizationFilter.
  - Ejemplo de petición:
    - GET /api/Drug/{id}
  - Códigos de estado y salida:

        - 200 OK
        - 403 Forbidden
            - Si el usuario no tiene permisos.
        - 404 ResourceNotFoundException
            - Si la farmacia a obtener no existe

- Create([FromBody] DrugModel drugModel)
    - Método POST
    - Crea un medicamento nuevo y lo almacena en la base de datos
    - Ejemplo de petición:
        - POST /api/Drug
        - Se le deberán pasar los datos del medicamento en el body
    - Códigos de estado y salida:
        - 200 OK
        - 403 Forbidden
            - Si el usuario no tiene permisos.
        - 400 Bad Request
            - Si el medicamento de la request es null
            - Si el medicamento a crear ya existe
            - Si el nombre del medicamento es null o vacío
            - Si el código del medicamento es null o vacío
            - Si el síntoma del medicamento es null o vacío

- Si la cantidad del medicamento es menor o igual a 0
- Si el precio del medicamento es menor a 0
- Si el stock del medicamento es menor a 0
- Si la unidad de medida del medicamento es null
- Si la presentación del medicamento es null
- Si la farmacia asociada al medicamento es null

- 404 ResourceNotFoundException
  - Si la farmacia del medicamento no existe
  - Si la unidad de medida del medicamento no existe
  - Si la presentación del medicamento no existe

- Update([FromRoute] int id, [FromBody] UpdateDrugModel updatedDrug)
  - Método PUT
  - Modifica si existe un medicamento existente en la base de datos a partir del objeto de la request updatedDrug
  - Ejemplo de petición:
    - PUT /api/Drug/{id}
    - Se le deberán pasar los datos actualizados del medicamento en el body
  - Códigos de estado y salida:
    - 200 OK
    - 403 Forbidden
      - Si el usuario no tiene permisos.
    - 400 Bad Request

- Si el nombre del medicamento es null o vacío
- Si el código del medicamento es null o vacío
- Si el síntoma del medicamento es null o vacío
- Si la cantidad del medicamento es menor o igual a 0
- Si el precio del medicamento es menor a 0
- Si el stock del medicamento es menor a 0
- Si la unidad de medida del medicamento es null
- Si la presentación del medicamento es null
- Si la farmacia asociada al medicamento es null

■ 404 ResourceNotFound

- Si el medicamento de la request es null
- Si el medicamento no existe

- Delete([FromRoute] int id)
  - Método DELETE
  - Elimina el medicamento con el id de la request si existe
  - Ejemplo de petición:
    - DELETE /api/Drug/{id}
  - Códigos de estado y salida:
    - 200 OK
    - 403 Forbidden
      - Si el usuario no tiene permisos.

- 404 ResourceNotFound
    - Si el medicamento a eliminar no existe

Notas

- Un nombre de usuario se considera no válido cuando está vacío o ya existe en el sistema.
- Un código de invitación se considera no válido cuando está vacío o no es numérico de largo 6.
- Un email se considera no válido cuando está vacío o no cumple con el formato de correos electrónicos.
- Un password se considera válido cuando cumple que debe tener largo mínimo 8, y contar con al menos una mayúscula, un número y un carácter especial (#?!@$%^&.*-).
- Una dirección se considera no válida cuando está vacía.