Universidad ORT Uruguay

Facultad de Ingeniería

Bernard Wand Polak

Diseño de Aplicaciones II

Anexo 1 - 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(new string[]{nameof(RoleType.*RoleName*), nameof(RoleType.*RoleName*) })]** siendo *RoleName* 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.

Cambios a nivel del dominio

Purchase

Se saca la propiedad Pharmacy y se agrega la propiedad TrackingCode (string).

PurchaseDetail

Se agrega la propiedad Pharmacy y la propiedad Status (string)

De esta forma se soporta realizar compras de medicamentos de diferentes farmacias (en lugar de que todos los ítems de una compra fueran de la misma farmacia). Además, permite mantener un status a nivel de ítem, donde los posibles valores de este status pueden ser: 'Approved', 'Rejected' o 'Pending'.

Presentation

Se cambio para que el nombre ahora sea de tipo string y no un enumerado, esto permite que se envíe al front-end la tupla (id, name), y que en el caso de que se quiera agregar una nueva Presentación solamente sea necesario hacer un Insert por base de datos, y si se quiere dar de baja una Presentación se pueda hacer un Update por base de datos (deleted='true'). Y que esto se vea reflejado automáticamente en el front-end.

UnitMeasure

Idem con respecto a la entidad 'Presentation'.

Descripción de los cambios en los resources de la API.

LoginController

Se modifica para que ahora además de devolver el token de autenticación se devuelve el rol del usuario y el nombre de usuario (username). Con el rol se permite desde el front-end identificar el tipo de usuario y permitirle acceder solo a sus funcionalidades. El username se utiliza en el componente Alert (de estado éxito y color verde) para dar visibilidad y darle la bienvenida al usuario recién logueado.

- 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, el rol del usuario y su nombre de usuario,
- 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")

PurchasesController

Se cambia para que el método GET, que solo se permite acceder con el Rol=’Employee’ devuelva todas las compras realizadas en la farmacia del empleado que accede al endpoint, para esto se toma el token de Autorización enviado y se busca al usuario empleado para luego obtener las compras de su farmacia.

- All()
  - GET
  - [HttpGet]

- [Route("[action]")]
- 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")

Se cambia el endpoint **purchases/bymonth** por el endpoint **purchases/bydate**, de esta forma se soporte obtener las compras entre 2 fechas y no las compras de un determinado mes/año como estaba anteriormente.

- ByDate(DateTime? start, DateTime? end)
  - GET
  - [HttpGet]
  - [Route("[action]")]
  - Obtiene una lista de Compras entre las fechas 'start' y 'end'
  - Requiere autenticación de usuario.
  - Acceso solo a usuarios de Rol ***Owner***.

- /api/purchases/bydate
- Ejemplo de petición
  - GET /api/purchases/bydate?start=2022-11-01T10:34&end=2022-11-06T10:34
  - Se le deberá pasar un Header ‘Authorization’ con un token.
- Códigos de estado y salida
  - Ok 200 - Retorna una lista con todas las compras en el rango de fechas especificado.
  - Error 401 - Token de autorización no válido. (“Invalid authorization token”)
  - Error 403 - Acceso con un Rol prohibido.(“Forbidden role”)
  - Error 400 - La fecha de inicio no es enviada (“Start date is required”)
  - Error 400 - La fecha de fin no es enviada (“End date is required”)
  - Error 400 - La fecha de inicio es mayor a la de fin (“Start date can't be bigger than End date”)

Al crear una compra se persisten todos los ítems de la misma con estado=’Pending’ y se calcula el monto total de la misma. No se hace más el check de si la misma tiene o no stock en su farmacia.

- 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")

Se agrega el endpoint de tracking **purchases/tracking** que permite enviar un código de rastreo de una compra (mediante query param) y devolver el detalle de la misma, incluido el estado de cada uno de los ítems de la misma. El código es un string autogenerado alfanumérico de largo 16 (solo letras mayúsculas).

- Tracking(string? code)
    - GET

- [HttpGet]
- [Route("[action]")]
- Obtiene una compra identificada por su código de rastreo ‘code’
- No requiere autenticación de usuario.
- /api/purchases/tracking
- Ejemplo de petición
  - GET /api/purchases/tracking?code=FROR7HWPUH5JWW4C
- Códigos de estado y salida
  - Ok 200 - Retorna la compra identificada por el código de rastreo ‘code’.
  - Error 400 - El código de rastreo no puede ser vacío (“Tracking Code is can't be empty"”)

Se agregan los endpoints **purchases/approve** y **purchases/reject** (ambos PUT) que permiten aprobar y rechazar la compra de un ítem (aplica a items en estado “Pending”) respectivamente, ambos endpoints son solo accedidos por usuarios de rol=’Employee’.

Al aprobar la compra de un ítem se verifica el stock del mismo, y en caso de que la cantidad no supere al stock se actualiza el stock del mismo. Luego su estado pasa a ‘Approved’.

Al rechazar la compra de un ítem su estado pasa a ‘Rejected’ y se actualiza el monto total de la compra restando el (precio unitario*cantidad) de la misma.

- Approve(int id, PurchaseAuthorizationModel model)
  - PUT
  - [HttpPut]
  - [Route("[action]")]
  - Aprueba la compra de un ítem, se actualiza el stock del ítem.
  - Devuelve el detalle del ítem modificado de la compra.

- Requiere autenticación de usuario.
- Acceso solo a usuarios de Rol ***Employee***.
- /api/purchases/approve
- Ejemplo de petición
  - PUT /api/purchases/approve/1 (identificador único de la compra)
  - El PurchaseAuthorizationModel consta del identificador de la farmacia y del código (único dentro de la farmacia) del ítem a aprobar.
  - Body: { "pharmacyId": 1, "drugCode": "XDEA"}
  - Se le deberá pasar un Header 'Authorization' con un token.
- Códigos de estado y salida
  - Ok 200 - Retorna el detalle del ítem aprobado.
  - Error 401 - Token de autorización no válido. ("Invalid authorization token")
  - Error 403 - Acceso con un Rol prohibido.("Forbidden role")
  - Error 404 - La compra de Id = 1 no fue encontrada ("Purchase not found 1")
  - Error 404 - El detalle de la compra no fue encontrado ("Purchase Detail not found for Pharmacy 1 and Drug XDEA")
  - Error 404 - La farmacia con Id = 1 no fue encontrada ("Pharmacy with Id: 1 not found")
  - Error 404 - El medicamento de código "XDEA" en la farmacia de Id = 1 ("Drug XDEA not found in Pharmacy")
  - Error 400 - El medicamento de código "XDEA" no tiene stock en la farmacia de Id = 1 ("The Drug XDEA is out of stock in Pharmacy 1")

- Reject(int id, PurchaseAuthorizationModel model)

- PUT
- [HttpPut]
- [Route("[action]")]
- Rechaza la compra de un ítem, se resta el monto del ítem del total de la compra.
- Devuelve el detalle del ítem modificado de la compra.
- Requiere autenticación de usuario.
- Acceso solo a usuarios de Rol ***Employee***.
- /api/purchases/reject
- Ejemplo de petición
    - PUT /api/purchases/reject/1 (identificador único de la compra)
    - El PurchaseAuthorizationModel consta del identificador de la farmacia y del código (único dentro de la farmacia) del ítem a aprobar.
    - Body: { "pharmacyId": 1, "drugCode": "XDEA"}
    - Se le deberá pasar un Header 'Authorization' con un token.
- Códigos de estado y salida
    - Ok 200 - Retorna el detalle del ítem aprobado.
    - Error 401 - Token de autorización no válido. ("Invalid authorization token")
    - Error 403 - Acceso con un Rol prohibido.("Forbidden role")
    - Error 404 - La compra de Id = 1 no fue encontrada ("Purchase not found 1")
    - Error 404 - El detalle de la compra no fue encontrado ("Purchase Detail not found for Pharmacy 1 and Drug XDEA")

        - Error 404 - La farmacia con Id = 1 no fue encontrada (“Pharmacy with Id: 1 not found”)
        - Error 404 - El medicamento de código “XDEA” en la farmacia de Id = 1 (“Drug XDEA not found in Pharmacy")

InvitationController

Se agrega endpoint GetAll, GetById, UserCode (todos bajo el verbo GET) con la finalidad de obtener una lista de invitaciones la cual se puede filtrar pasando determinados parámetros mediante queryString, obtener una invitación en particular dado su identificador o obtener un nuevo código de usuario para poder actualizar el dato en la invitación.

- GetAll([FromQuery] InvitationSearchCriteriaModelRequest)
    - GET
    - [HttpGet]
    - /api/invitations
    - Ejemplo de petición
        - GET /api/invitations
        - Se puede filtrar pasando parámetros pharmacy y/o username y/o role.
    - Códigos de estado y salida
        - Ok 200 - Todas las invitaciones.
- GetById([FromRoute] id)
    - GET
    - [HttpGet]
    - /api/invitations/{id}
    - Ejemplo de petición
        - GET /api/invitations/1

- Códigos de estado y salida
    - Ok 200 - Obtiene una invitación

- UpdateInvitation([FromRoute] id, [FromBody] InvitationModelRequest)
    - PUT
    - [HttpPut]
    - /api/invitations/{id}
    - Ejemplo de petición
        - GET /api/invitations/1
    - Códigos de estado y salida
        - Ok 200 - Cuando la invitación se actualiza 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.”.

- UserCode()
    - GET

- [HttpGet]
- /api/invitations/usercode
- Ejemplo de petición
  - GET /api/invitations/usercode
- Códigos de estado y salida
  - Ok 200 - Devuelve nuevo usercode.

StockRequestController

Se modificó el endpoint CreateStockRequest para que ya no sea necesario enviarle el usuario empleado en el body del mismo, ahora mediante el token de autorización se obtiene al usuario empleado, el cual es el mismo que realiza la petición de stock.

- 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.”.

PharmacyController

Se sacó el AuthorizationFilter del endpoint GetAll (al que solo podían acceder los usuarios de rol='Administrator') para que cualquier tipo de usuario pueda acceder, de esta forma los usuarios anónimos pueden acceder a todas las farmacias del sistema desde el front-end para poder filtrar los medicamentos a comprar por farmacia.

- 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.

DrugController

Se modificó para que el endpoint para obtener medicamentos GET **/drug** que se utiliza desde el front-end para filtrar los medicamentos desde el home por nombre y/o farmacia solo devuelva medicamentos que no fueron dados de baja (Deleted='false').

- 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.

Se sacó el AuthorizationFilter del endpoint GetById que antes solo podían acceder los usuarios de rol='Administrator' para que todos los usuarios (incluidos los anónimos) puedan acceder. De esta forma los usuarios anónimos desde el front-end pueden acceder al mismo para ver el detalle de un medicamento.

- 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

Se modificó el endpoint POST **/drug** que permite crear un medicamento, ahora no es necesario enviarle la farmacia donde se creará el medicamento, sino que mediante el token de autorización del usuario empleado que solicita la creación se identifica la farmacia donde trabaja, y se usa esa farmacia para crear al mismo.

- 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

Se creó un nuevo endpoint GET **/drug/user** que devuelve todos los medicamentos (no dados de baja) de la farmacia en la que trabaja el usuario de rol empleado. De esta forma un usuario empleado tendrá acceso al listado de los medicamentos de la farmacia en la que trabaja. Permitiendo, por ejemplo, visualizar en el front-end una tabla con un listado de los mismos para permitirle darlos de baja.

- User()
  - GET
  - [HttpGet]
  - [Route("[action]")]
  - Devuelve la lista de medicamentos (no dados de baja) de la farmacia asociada al empleado logueado.
  - Requiere autenticación de usuario.
  - Acceso solo a usuarios de Rol ***Employee***.
  - /api/drug/user

- Ejemplo de petición
    - GET /api/drug/user
    - Se le deberá pasar un Header ‘Authorization’ con un token.
- Códigos de estado y salida
    - Ok 200 - Retorna una lista de medicamentos asociados al usuario empleado.

Se realizaron cambios para que el DTO de salida DrugBasicModel ahora también devuelva el Precio, Síntoma y Nombre/Código de la farmacia a la que pertenece el medicamento. Permitiendo al usuario anónimo ver esta información desde el front-end.

Se realizaron cambios en el DTO DrugModelResponse para que ahora además se devuelva el stock de un medicamento, de esta forma el usuario empleado tendrá la posibilidad desde el front-end de visualizar el stock de cada medicamento antes de hacer una solicitud.

PresentationsController

Se creó este controlador con el endpoint GET **/presentations** para devolver todas las Presentaciones no eliminadas persistidas en la base de datos, de esta forma al momento de que un empleado quiera crear un medicamento se le pueda listar todos las presentaciones existentes.

- GetAll()
    - GET
    - [HttpGet]
    - Obtiene una lista de Presentaciones, tupla [Id, Nombre]
    - No requiere autenticación de usuario.
    - /api/presentations
    - Ejemplo de petición
        - GET /api/presentations
    - Códigos de estado y salida

        - Ok 200 - Retorna la lista de Presentaciones del sistema que no están borradas (borrado lógico).

UnitMeasuresController

Se creó este controlador con el endpoint GET **/unitmeasures** para devolver todas las Unidades de Medida no eliminadas persistidas en la base de datos, de esta forma al momento de que un empleado quiera crear un medicamento se le pueda listar todos las unidades de medida existentes.

- GetAll()
    - GET
    - [HttpGet]
    - Obtiene una lista de Unit Measures, tupla [Id, Nombre]
    - No requiere autenticación de usuario.
    - /api/unitmeasures
    - Ejemplo de petición
        - GET /api/unitmeasures
    - Códigos de estado y salida
        - Ok 200 - Retorna la lista de Unidades de Medida del sistema que no están borradas (borrado lógico).

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.

Cambios de la API a Nivel Global

Cors

Se agrego una política de nombre “MyAllowedOrigins” en el Program.cs para que permita requests desde los navegadores desde cualquier origen (“*”).

AuthorizationFilter

Se realizó un cambio que ahora permite que un endpoint pueda ser accedido por diferentes roles, en lugar de enviarse un solo rol al filtro se le envía una lista de roles. De esta forma el sistema queda más flexible para que usuarios de diferentes roles puedan acceder a un mismo endpoint.