* We've been getting resources throughout this module, and all of this looks good, right?
* But let's set this off against one of our constraints, the manipulation of resources through representations constraint, that's the one that stated that when a client holds a representation of a resource, including any possible metadata, it must have enough information to modify or delete that resource on the server, provided it has permission to do so.
* In the next few modules, we'll allow creating and deleting resources. So, if the consumer of the API gets the response we see now, does it have enough information to modify or delete it? Well, more or less, but not really. What should be in the response to allow for that is the resource URI, the identifier of the resource. We include an id, and often that's considered enough. From the id, a consumer can create a URI. But if we think about this a bit further, it isn't completely correct. An id isn't what identifies the resource, it's the URI that identifies the resource, and the resource URI is part of the HTTP request, but it isn't part of the HTTP response. So to adhere to this constraint, we could include the URI in each representation if update or delete is allowed. We could do that now already, it's just a matter of adding an extra field and filling it up with the URI. But there's a much better way of handling this, and that's through HATEOAS.

That's one of the more advanced REST constraints, it's covered in my Implementing Advanced RESTful Concerns with ASP.NET Core 3 course. For now, we're going to leave this just as is, but I do want to already mention this in case you do not want to implement HATEOAS. Make sure to at least include the resource URI in your response if updating or deleting the resource is allowed.