

Programming Applications with Databases

Exercise Set 12

1. Introduce 3 business validation rules using built-in validators and 2 business custom validation rules using a self-validation mechanism. Implement them in the domain model using the library `FluentValidation` (or analogous in the other technology).
[2p]
2. Ensure that at least 3 business validation rules (select ones which make sense) are also implemented in the UI using the vendor's approach for validation. Ensure the logic is consistent between these implementations.
[2p]
3. Plan and design an HTTP API to support *product* entity and one more from your domain model. Create a mock-up of this API at <https://www.mockapi.io/> (or equivalent) and populate it with sample data. Prepare in Postman a set of 6 example requests to the API, covering different possible usages, including all CRUD operations.
[2p]
4. Using <https://services.odata.org/AdventureWorksV3/AdventureWorks.svc/> service, create and execute in Postman the following OData query: from a product's catalog, get black bikes with a weight between 15 and 25, ordered by list price, and select only a minimum list of columns such as product name, color, weight, and list price.
[1p]
5. Using <https://hygraph.com/graphql-playground> service, create a GraphQL query that leverages the following features: where clause with and and or operators, order, selecting only 5 attributes, and fetching 3 middle items (i.e. a specific "page").
[1p]
6. Following a tutorial from <https://docs.microsoft.com/pl-pl/azure/service-bus-messaging/service-bus-dotnet-get-started-with-queues>, create and present an Azure Service Bus queue, a sender application, and a receiver application.
[2p]

Pawel Rajba