

# Programming Applications with Databases

## Exercise Set 8

The high-level main requirements of the e-Shop application are as follows:

- customer is able to browse the product catalog and review the products' cards,
- customer is able to put products into the cart,
- customer is able to see and manage the cart content,
- manager is able to create, update, and delete products.

Please keep in mind that the scope of the application should be minimal and basic.

1. Prepare prototypes of all different types of views, i.e., prepare a visual representation of how the application is going to look. It can be created in any application, including famous MS Paint or even MS PowerPoint, but usage of a dedicated *mockup* tool is highly recommended. Please note that the solution must later align with that visualization.  
[1p]

2. Following the attached example *DDDSample* create *application*, *domain* and *infrastructure* layers expressed by projects (in .NET) or in another way aligned with the selected technology. On top of that, create a *cross-cutting component* (e.g. *CommonComponent*, *GenericComponent*, or *UtilityComponent*) and an empty placeholder for the user interface.  
[1p]

3. Referring to the e-Shop requirements and prototypes created in Exercise 1, propose a draft domain model, i.e.:
  - define objects and mark them as either entities or value objects,
  - for entities define their lifecycle, i.e., enumerate stages of "life" (e.g., order created, order completed, order paid, etc.),
  - propose aggregates keeping in mind transactions that are required to perform all operations in the context of all views.

As a result, prepare an appropriate class structure in the *domain* layer. At this stage, each object needs to include all properties and methods.

[3p]

4. For each aggregate, create a respective repository interface in the *domain* layer and then the respective *in-memory* implementation in the *infrastructure* layer. Consider creating a generic base class implementation.  
[2p]

5. Create appropriate application service(s), i.e. at the current stage, for each domain model aggregate create an application service class with methods to support respective use cases specified at the beginning of this document (or, in others words, methods required by UI views), including methods which supposed to return data (ensure the implementation that returns some fake data for now).

Ensure that all application services cover all the requirements (or views) as specified at the beginning of this document.

[3p]

*Paweł Rajba*