SEM 13a - DelftBlue assignment 1

Michiel Bakker - Bink Boëtius - Ionut Ciobanu - Alican Ekşi - Ivans Kravcevs - Nils Mikk

Task 2 – Design Patterns

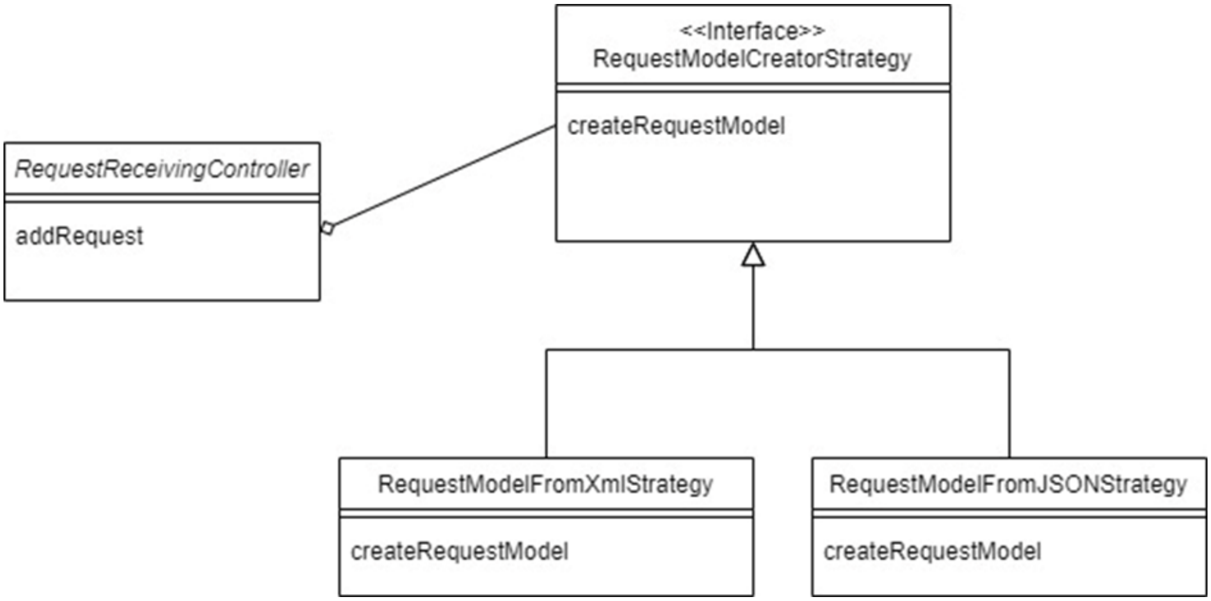
Strategy pattern

Why and how is it implemented?

Our software application should be compatible with different client applications. Because of this, it was decided to use a strategy design pattern to accept requests through the User microservice API in form of different semi-structured data formats. Therefore, we are using 2 different strategies to receive requests. This API supports now both JSON and XML formatted requests. We had the plain text strategy before as well, but it was decided to remove this strategy during the final discussion because plain text is practically not used for object transfer nowadays.

We are using a Strategy design pattern with one common interface and 2 different strategies to receive requests. A client can choose what strategy will be used by specifying the headers of HTTP requests. The controller will create a specific strategy class based on it.
All created classes are in the requestmodelget folder in the user-microservice module. The API that is using them is defined in the addRequest() method in the RequestReceivingController in the user-microservice module.

Class diagram

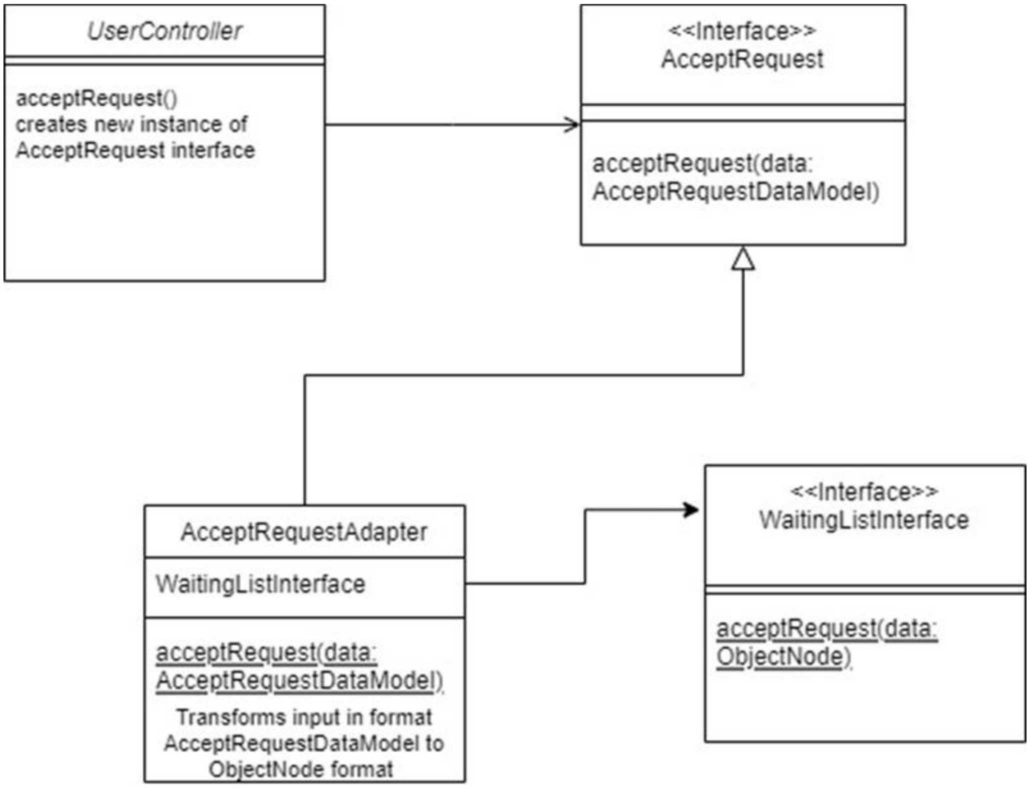

<<Interface>>
RequestModelCreatorStrategy
createRequestModel
RequestReceivingController
addRequest
RequestModelFromXmlStrategy
createRequestModel
RequestModelFromJSONStrategy
createRequestModel

Adapter pattern

Why and how is it implemented?

There was a problem in the application that 2 different microservices were using different data transfer objects to communicate with one another. Therefore, we decided to use an adapter design pattern to make one microservice use the output of another microservice. We have used an adapter pattern for accepting requests in the FacultyAdminController class, which should communicate with the WaitingListController class. We are using the AcceptRequest interface and Adapter class that inherits from it and takes the WaitingListInterface as the parameter.
All created classes are in the acceptrequestadapter folder in the user-microservice module and the API that is using them is defined in the acceptRequest() method in the FacultyAdminController class in the user-microservice module.

Class diagram


UserController
acceptRequest()
creates new instance of
AcceptRequest interface
<<Interface>>
AcceptRequest
acceptRequest(data:
AcceptRequestDataModel)
AcceptRequestAdapter
WaitingListInterface
acceptRequest(data:
AcceptRequestDataModel)
Transforms input in format
AcceptRequestDataModel to
ObjectNode format
<<Interface>>
WaitingListInterface
acceptRequest(data:
ObjectNode)