

Software Development Assignment

The work you will do as a part of this assignment is to implement the user stories specified below. It should give you a good understanding of what it is like to work with Coolblue, and it should give us a better understanding of your skills. So, good for you, and good for us!

You have one whole week (including the weekend) to hand your solution back in but please plan your days to give yourself enough time for the assignment. It's not a requirement to finish all stories at all costs but whatever you manage to complete should be done to the standards we expect. If it is not done by the standards we expect, the second round will be used to give you feedback.

Our technical values at Coolblue:

1. We value team communication so we document our design/architecture choices
2. We have a strong culture of Test Driven Development, before we write production code we prefer to write tests first
3. We like to know what our applications and services are doing in production without necessarily having to debug code or without needing the customer to tell us something is wrong so we proactively monitor them and invest in logging.
4. We value SOLID principles and are pragmatic around applying them to the systems we build
5. We ensure our applications and systems are self-healing and resilient in face of failures

What we expect in your solution:

- Behavioural tests
- Appropriate separation of concerns
- **A solution with an architecture that you can defend and would feel comfortable putting in production.**
- Readable code with meaningful names for variables/classes/interfaces etc.
- Appropriate exception handling and fault tolerance
- Should be runnable on our machines so keep that in mind while picking third party tools and apps etc.
- Documentation of your design decisions and technical choices for the tasks. A simple markdown or text file (included in your submission) showing the following will be enough:
 - **Assumption/Decision Made**
 - **Reason**

Your colleagues

We are, of course, not going to let you deal with the assignment on your own. You will have two software developers available via Slack. Please feel free to ask your questions or clarifications,

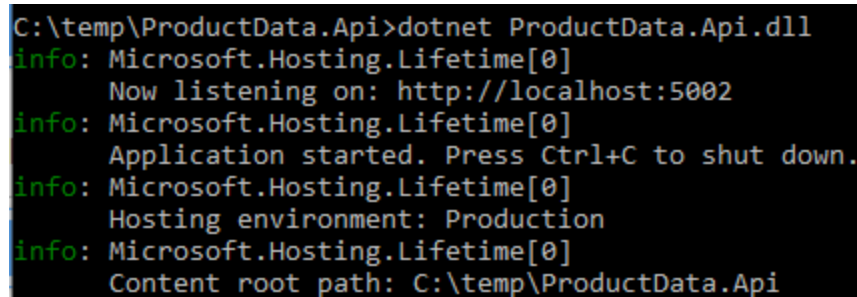
they are happy to help. Please make sure you send us your questions before/after the weekend, as we don't work on the weekends.

User Story:

At Coolblue, we want to be able to insure the products that we sell to a customer, so that we get money back in case the product gets lost or damaged before reaching the customers. For that, we need a REST API that is going to be used by Coolblue webshop. You're going to get access to our [Product Information API](#):

1. Unzip it onto your machine
2. Navigate to the unzipped folder in a terminal
3. On the terminal, type **dotnet ProductData.Api.dll** and hit *Enter*.

Your screen should look like as shown below if the API has successfully started:



```
C:\temp\ProductData.Api>dotnet ProductData.Api.dll
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5002
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\temp\ProductData.Api
```

The Products API is Swagger enabled, and you can access it by navigating to **<http://localhost:5002/>** in your browser. In the example the port is 5002.

Functionality already implemented [here](#):

There is an existing endpoint that, given the information about the product, calculates the total cost of insurance according to the rules below:

- If the product sales price is less than € 500, no insurance required
- If the product sales price=> € 500 but < € 2000, insurance cost is € 1000
- If the product sales price=> € 2000, insurance cost is €2000
- If the type of the product is a *smartphone* or a *laptop*, add € 500 more to the insurance cost.

** Download the current code, so you can continue with the further tasks.*

Task 1 [BUGFIX]:

The financial manager reported that when customers buy a laptop that costs less than € 500, insurance is not calculated, while it should be € 500.

Task 2 [REFACTORING]:

It looks like the already implemented functionality has some quality issues. Refactor that code, but be sure to maintain the same behavior.

** Please make sure to include in the documentation about the approach that you chose for the refactoring.*

Task 3 [FEATURE 1]:

Now we want to calculate the insurance cost for an *order* and for this, we are going to provide all the products that are in a shopping cart.

** While developing this feature, please document your assumptions and feel free to reach the stakeholders for doubts via Slack.*

Task 4 [FEATURE 2]:

We want to change the logic around the insurance calculation. We received a report from our business analysts that digital cameras are getting lost more than usual. Therefore, if an order has one or more digital cameras, add € 500 to the insured value of the order.

** While developing this feature, please document your assumptions and feel free to reach the stakeholders for doubts via Slack.*

Task 5 [FEATURE 3]:

As a part of this story we need to provide the administrators/back office staff with a new endpoint that will allow them to upload surcharge rates per product type. This surcharge will then need to be added to the overall insurance value for the product type.

Please be aware that this endpoint is going to be used simultaneously by multiple users.

** While developing this feature, please document your assumptions and feel free to reach out to the stakeholders for doubts via Slack.*