WHO
The YakShop

WHAT
Assessment

HOW
Develop

Xebia Studio

BUILD THE YAKSHOP

ASSESSMENT

Xebia Full Stack Developer Assessment

This is the Xebia Full Stack Developer assessment. The assessment is intended to give us insight into your technical abilities, development approach and general technical working habits. We view your performance on this assessment as indicative of the work you will deliver as a Xebia Full Stack Developer.

The assessment consists of an assignment to prepare beforehand and a presentation about your implementation of the assignment at the Xebia office.

The assignment is the “Yak Webshop” case, described below. Read the case carefully and approach it as you would a regular project. The assignment should be implemented in a language of your choice. The case contains several test cases for each user story so you can see if your implementation works. In addition to the test cases mentioned in this document, we will subject your solution to more rigorous automated tests during the assessment.

Expect to spend at least 8 hours to prepare for the assessment. We ask you treat the assessment confidential so we can use it again in the future. Please do not publish it publically on GitHub.

At the assessment presentation, please bring a laptop with a working development environment so you can show us your solution. There will be a projector present at the office. You are free to deliver your presentation in any form, but we expect you to cover:

- the overall approach you took to the assignment
- the architecture of the solution and the technologies used
- your solution for each of the user stories

Good luck with the assignment!

The YakShop

Your new customer is a Yak shepherd living on the tundra herding a group of Yaks. Every once in a while he gets customers who come in to buy Yak wool or milk from him. However he decides to open up a shop on the internet so that he can expand his horizon and actually sell his products outside of his regular clientele. He has decided to hire you as a developer for his new webshop. You’ve had a few meetings with him and together you’ve thought up a number of user stories so that you can both have a clear focus and don’t spend time on.... well, essentially shaving yaks ;-)

From research on the internet you of course know that Yaks like humans also age, and with age they give less milk until they finally die of old age. Contrary to humans, a standard Yak year consists of 100 days.

The shepherd currently has Yaks who all stem from the “LabYaks” tribe. This tribe is known for its consistency in wool quality, milk taste and production rate of said goods. The shepherd gave you the following facts about LabYaks:

- Each day a LabYak produces 50-D*0.03 liters of milk (D =age in days).
- At most every 8+D*0.01 days you can again shave a LabYak (D =age in days).
- A yak can be first shaven when he is 1 year.
- A LabYak dies the day he turns 10.

Assumptions

- The moment you open up the YakShop webshop will be day 0, and all yaks are eligible to be shaven, as the two of you spent quite a lot of time setting up this shop and the shepherd wasn’t able to attend much to his herd.
- Each morning the shepherd milks and shaves his yaks. Yaks which aren’t eligible to be shaven on the exact day, cannot be shaved today. Example: a yak who started out on day 0 as 4 years, can be shaven again during day 13.
- When querying your app and placing orders, the elapsed time in days (T) is used. A value of for instance 13 means that day 12 has elapsed and we’re at midnight day 13.

Clari ications

The Yak shepherd is available at shepherd@xebia.studio if during the course of developing the webshop you have any questions or want clarification on the requirements.

User stories

YAK-1: As a Yak Shepherd, I want to be able to read in a XML file that contains data about my herd so that I can query it.

Input herd.xml:

```
<herd>
  <labyak name="Betty-1" age="4" sex="f"/>
  <labyak name="Betty-2" age="8" sex="f"/>
  <labyak name="Betty-3" age="9.5" sex="f"/>
</herd>
```

N.B. The age is given in standard Yak years

Your program should take 2 parameters:
1. The XML file to read
2. an integer T, representing the elapsed time in days.

Output for T = 13:

```
In Stock:
  1104.480 liters of milk
  3 skins of wool
Herd:
  Betty-1 4.13 years old
  Betty-2 8.13 years old
  Betty-3 9.63 years old
```

Output for T = 14:

```
In Stock:
  1188.810 liters of milk
  4 skins of wool
Herd:
  Betty-1 4.14 years old
  Betty-2 8.14 years old
  Betty-3 9.64 years old
```

YAK-2: As a Yak Shepherd I want to be able to load a new Herd into my webshop using an Http ReST service so that I can open my webshop.

I want to be able to run my webshop and load my herd into it. Once I load a herd, any previous state of my webshop should be reset to the initial state.

The following are the HTTP requests I want to make.

- POST /yak-shop/load: Which returns a Http Status Code 205 (Reset Content)

Samples:

```
Request: POST /yak-shop/load
<herd>
  <labyak name="Betty-1" age="4" sex="f"/>
  <labyak name="Betty-2" age="8" sex="f"/>
  <labyak name="Betty-3" age="9.5" sex="f"/>
</herd>
Response: status = 205
```

YAK-3: As a Yak Shepherd I want to be able to query my herd and my current stock using Http ReST services which output JSON data.

The following are the requests you wish to make.

- GET /yak-shop/stock/T: Which returns a view of your stock at T days
- GET /yak-shop/herd/T: Which returns a view of your herd at T days

Samples:

```
Request: GET /yak-shop/stock/13
Response: {
  "milk" : 1104.48,
  "skins" : 3
}

Request: GET /yak-shop/herd/13
Response: {
  "herd" : [
    {
      "name" : "Betty-1",
      "age" : 4.13,
      "age-last-shaved" : 4.0
    },
    {
      "name" : "Betty-2",
      "age" : 8.13,
      "age-last-shaved" : 8.0
    },
    {
      "name" : "Betty-3",
      "age" : 9.63,
      "age-last-shaved" : 9.5
    }
  ]
    }
```

YAK-4: As a Yak Shepherd I want my customers to be able to buy from my stock using my Http ReST services.

You can assume that requests come in ascending order of time. If you cannot fulfill one of the ordered goods of the order because you're out of stock, you deliver the other goods that are fully in stock.

So for instance if your stock contains 4000 liters of milk and 10 yak hides, and your customer orders 4500 liters of milk and 4 hides, you only deliver the 4 hides (and omit the milk from the result) and give a Http status code 206 (partial content). If the full order is not in stock, you only return a Http 404 status code. If the order was placed successfully you return Http status code 201 (created) with the resulting order

Request:

- POST /yak-shop/order/T where T is the day the customer orders, this means that day T has not elapsed.

Samples:

Request: POST /yak-shop/order/14

```
{
    “customer” : “Medvedev”,
    “order” : {
        “milk” : 1100,
        “skins” : 3
    }
}
```

Response: status = 201

```
{
    “milk” : 1100.0,
    “skins” : 3
}
```

Request: POST /yak-shop/order/14

```
{
    “customer” : “Medvedev”,
    “order” : {
    “milk” : 1200,
    “skins” : 3
    }
}
```

Response: status = 206

```
{
    “skins” : 3
}
```

Bonus assignments

Congratulations! You’ve made it this far! The Yak shepherd has come up with a few more user stories. These are all of equal business value to him, so you’re allowed to pick and choose which one(s) you implement.

YAK-5: As a Yak Shepherd I want to have a user interface in my browser which I can use to order goods.

The user interface should be able to place an order using the exposed ReST services, and provide feedback whether the order was placed successfully, partial successfully, or failed. In case of a partial success it should output what you will get delivered.

YAK-6: As a Yak Shepherd I want to be surprised by your ingenuity so that I can show off to my fellow shepherds.

You can do anything that we haven’t thought of, so that I can show off some cool stuff on our monthly shepherd meeting!