

*Vinted Go*

# How to QA a sorting robot from your laptop

Jelizaveta Blaze

QA Engineer



*Vinted* Go

# What you will leave with

## 1. The Setup

How I test a robot I never stood next to

## 2. The Blindspots

Why mocks can look fine but still lie to us

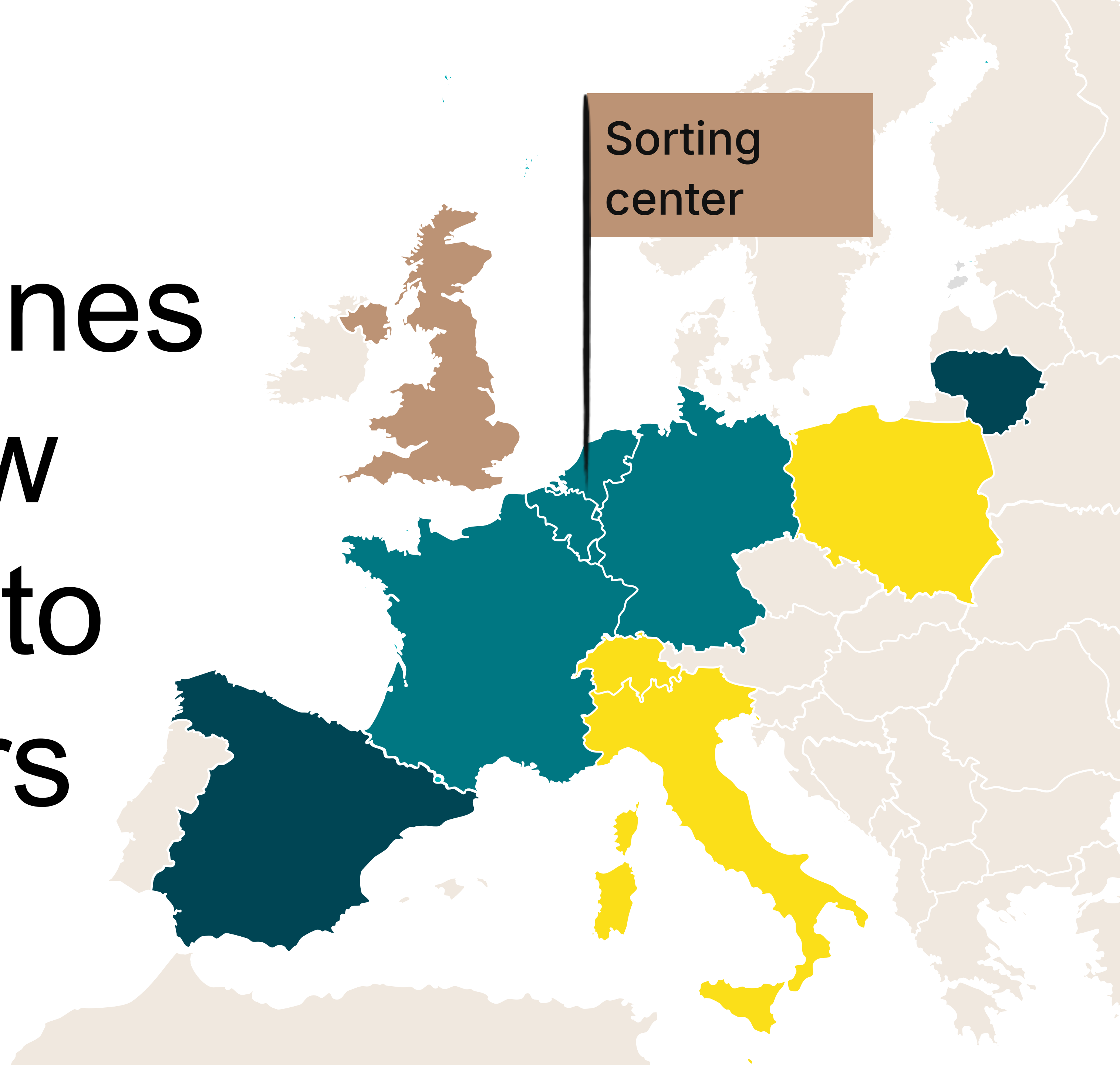
## 3. Three bugs

Full bag, two machines, and a green chute

## 4. What to copy

Concrete testing patterns you can use

I test machines  
that throw  
parcels into  
containers





# Fast context



Chute

The exit the parcel falls through

Container

The bag or pallet under the exit

Reject

The “I don’t know where it goes” container

Mock

The vendor’s simulator in the Test env

# Parcel path



| Step   | Physical reality                    | Where QA verifies it                               |
|--------|-------------------------------------|----------------------------------------------------|
| Create | Parcel exists with a target address | API or Admin                                       |
| Scan   | Robot scans parcels barcode         | Vendor mock or database                            |
| Decide | Group and chute are chosen          | Chute is open and has room                         |
| Fall   | Parcel lands in a container         | Event confirmation                                 |
| Leave  | Full container is taken away        | Chute state locks until new container is connected |

# Four layers



Layer

The happy illusion

The silent trap

Admin UI

The chute is connected

API /  
events

Scan returns 200 OK

Database

Row says the bag is attached

Mock

Status says “sorted”

All layers are secretly disagreeing  
with each other

# Bug 1 - Full container

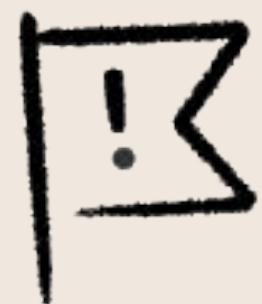


## Sandbox

Chute stays open and parcels keep falling in. The happy path never ends, so the test “passes”

## Reality

A full container must immediately close that chute so parcels don't pile up and spill onto the floor



**Lesson:** Silence is not a confirmation of safety. No message does not mean “not full”.

# Bug 2 - Split-Brain warehouse

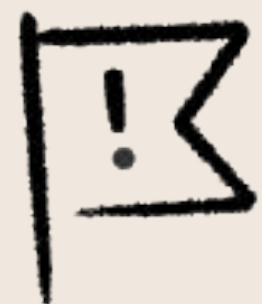


## You think you test

The warehouse  
The sorting machines  
The light on the chute

## You actually test

Only one machine's config:  
Machine A calls it "bin status"  
Machine B calls it "sort state"



**Note:** Same thing as two payment providers, or "the same" feature on the web and in the mobile app. "It works" on one of them.

# Bug 3 - Half-connected pallet

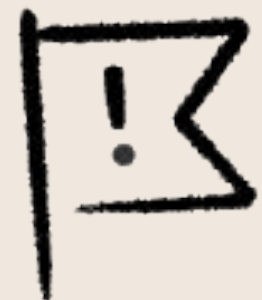


## 1 chute

Looks connected, but is  
not visible in UI

## 2 chutes

The pallet is so wide, so 2  
exits must be locked to the  
same pallet barcode



**Lesson:** One green chute is not a connected pallet. The screen can look ready while the parcel has nowhere to land.

# What to do



Talks path

Who sends the event, who saves it, who shows it?  
If you can understand that, you can test remotely

Don't stop at first green

A 200 OK only means that the request arrived. It does not mean the machine did the thing

Test the “no”

Full container, route disabled, container missing.  
That's where these bugs were

Check the same fact  
twice

Cross-check across two layers. If UI says “Ready”, make sure the DB and the mock says that too

# Try this on your system



## The “Missing signal” test

Don't just test error codes. Test what happens when an expected error never arrives (silent timeouts, dropped webhooks, hanging jobs)

## The “Refusal” test

Specifically test the moment a system must reject an action: cart limit reached, storage full, account suspended mid-flow

## The Split-Brain test

If your system uses two providers (iOS + Android), test them with the exact same edge cases

## The Cross-Layer check

If the screen says “Saved”, check the database or the event, not only the screen.

# Three sentences



You do not need the robot in the room

You need to know who talks to whom. If you can name that, you can test remotely

A green UI is a hypothesis

Confirm it in a second layer, or you tested the brochure.

The bug is often a state that never happened

If the mock never says “full”, the bag never fills and parcels never stop.

# What's your version of this?



If you test something you cannot stand next to - a payment provider, a printer, a partner API, a device in another city - I'd like to hear that story.

*Vinted* Go

