Dokumentation
PONGO TECH

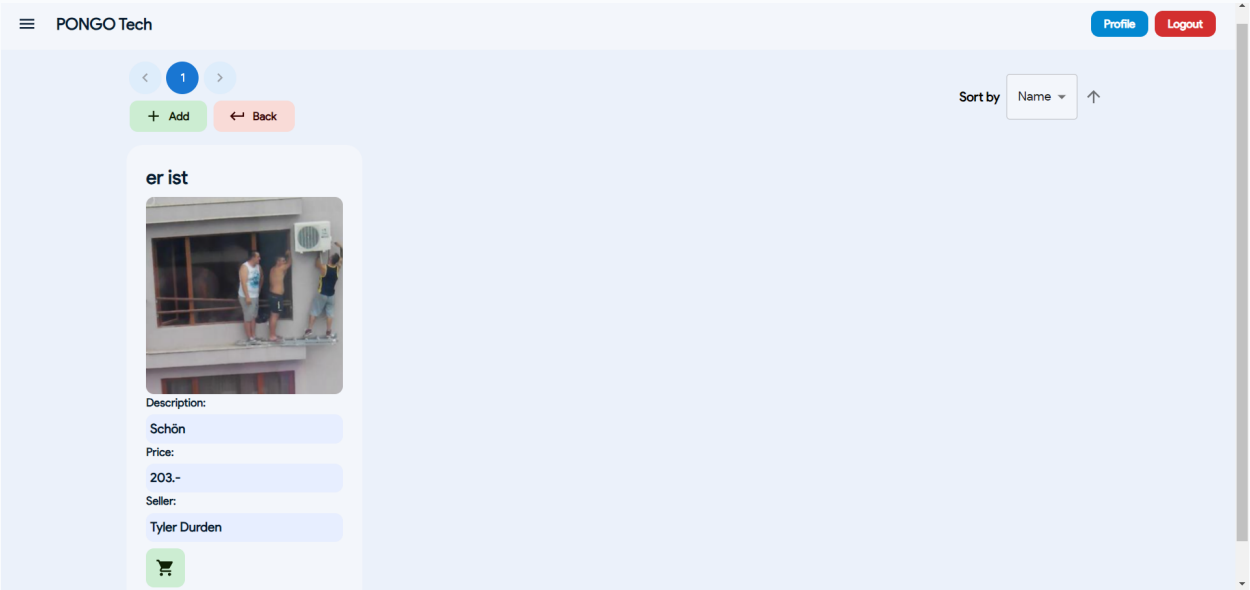
Orçun Bat
João Baixinho
Danny Nguyen
31.10.2023

Technische Berufsschule Zürich
Ausstellungsstrasse 70
8005 Zürich

Inhaltsverzeichnis

Einführung

Die folgende Doku enthält eine ausführliche Beschreibung unseres Projektes. PONGO Tech ist ein Online-Shop, um die neuesten Technologien an unsere Kunden leicht und schnell zu liefern. Mit Inspiration von Digitec haben wir uns vorgenommen, eine bessere Website zu entwickeln.



Planung

Alles wurde geplant, sodass die Arbeit sorgfältig und gerecht aufgeteilt werden konnte. Als die Aufgaben durchgearbeitet wurden, sind immer die Unteraufgaben rechtmäßig aufgeteilt worden. Zum Beispiel: Orcun macht a); Joao macht b); Danny macht c). Am Ende wurde alles zusammengefügt und zusammen die kleinen Einzelheiten poliert und bereit zur Abgabe gemacht.

Am Anfang des Projekts wurde eine klare Idee gemacht, die die Ziele und den Nutzen des Projekts festlegte. Das hat dem Team geholfen, ein gemeinsames Ziel zu setzen. Kommunikation war uns sehr wichtig, damit wir immer auf dem aktuellsten Stand sind und im Falle eines Problems immer bereit sind uns gegenseitig zu helfen, um großen Zeitverlust zu verhindern. Regelmäßige Updates waren ein wichtiger Bestandteil unserer Kommunikation. So konnten wir stets den Fortschritt im Auge behalten und die Planung im Laufe des Projekts anpassen.

Architektur

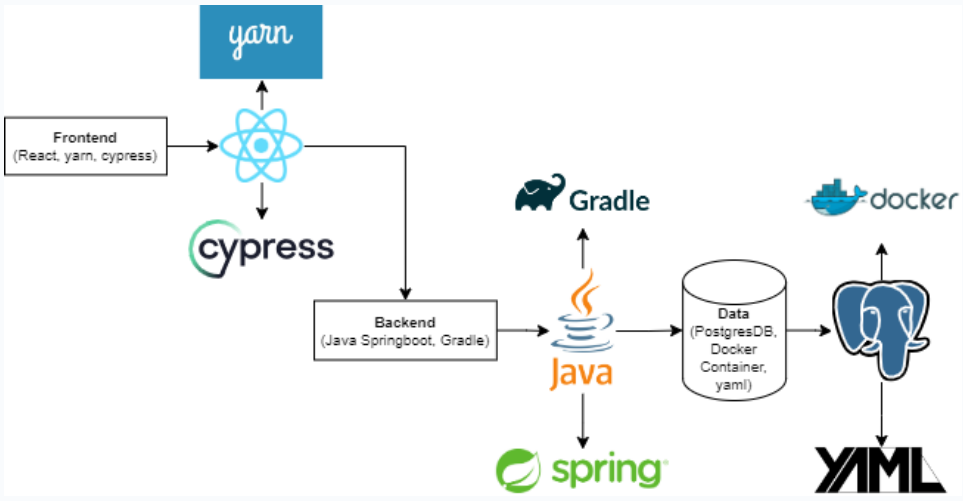
Unsere Software-Architektur ist die Grundlage unseres Entwicklungsprojekts. Die Architektur besteht aus drei Komponenten: dem Frontend, dem Backend und der Datenbank.

Das Frontend ist der Teil der Anwendung, den die Benutzer sehen und mit dem sie interagieren. Hier verwenden wir React für die Entwicklung. Für Tests und die Paketverwaltung verwenden wir die Tools Cypress und Yarn. Cypress ermöglicht uns, die Benutzeroberfläche gründlich zu testen, um sicherzustellen, dass sie einwandfrei funktioniert.

Im Backend erfolgt die Verarbeitung von Benutzeranfragen, die Geschäftslogik und die Kommunikation mit der Datenbank. Wir setzen auf das Java Framework Spring. Gradle wird verwendet, um den Build und die Dependencies zu steuern.

Die Datenbank ist der Speicherort für alle Informationen, die unsere Anwendung benötigt. Wir nutzen PostgreSQL, eine weit verbreitete relationale Datenbank, die sich gut für die Speicherung strukturierter Daten eignet. Um die Datenbank einfach und effizient zu betreiben, verwenden wir Docker-Container, die es uns ermöglichen, die Datenbankumgebung zu isolieren und zu verwalten. Die Konfiguration und Orchestrierung der Container erfolgt mithilfe von YAML-Dateien.

Diese Architektur bietet eine gute Grundlage für unsere Software und ermöglicht es uns, eine zuverlässige Anwendung zu entwickeln.



Testkonzept

Einleitung

Die vorliegende Applikation ist eine E-Commerce-Plattform, die es Benutzern ermöglicht, Artikel in einem Warenkorb zu sammeln und zu kaufen.

Test Items

Zu den zu testenden Elementen gehören:

- **User Service**: Verwaltet Benutzerinformationen und -authentifizierung.
- **Shopping Cart Service**: Verwaltet den Warenkorb eines Benutzers.
- **Item Service**: Verwaltet Artikelinformationen.
- **Cart Item Service**: Verwaltet Artikel innerhalb eines Warenkorbs.

Eine Skizze der Architektur, mit Komponenten wie Frontend, Backend und Datenbank, ist am Ende des Abschnitts “Architektur” zu finden.

Features to be tested

Die zu testenden Funktionalitäten umfassen:

- Benutzerregistrierung und -anmeldung
- Hinzufügen, Aktualisieren und Entfernen von Artikeln im Warenkorb
- Anzeigen von Artikelinformationen
- Verwaltung von Artikeln im Warenkorb

Features not to be tested

Nicht getestet werden:

- Performance und Skalierbarkeit der Applikation
- Sicherheit und Datenschutz

Approach

In unserem Projekt werden die zwei Testarten “Unit Testing” und “E2E Testing” verwendet, welche durch TDD umgesetzt werden.

Item pass / fail criteria

Geringfügige Fehler: Applikation hat kleine Mängel, funktioniert aber grundlegend. Beispiele:

- Schreibfehler
- Design/Visuelle Fehler (unpassende Farben, schlechte Anordnung der Items, etc.)
- Leistungsprobleme (Ladezeiten, Systembelastung, etc.)

Mittelschwere Fehler: Offensichtliche Fehler, die die Funktionalität beeinträchtigen. Beispiele:

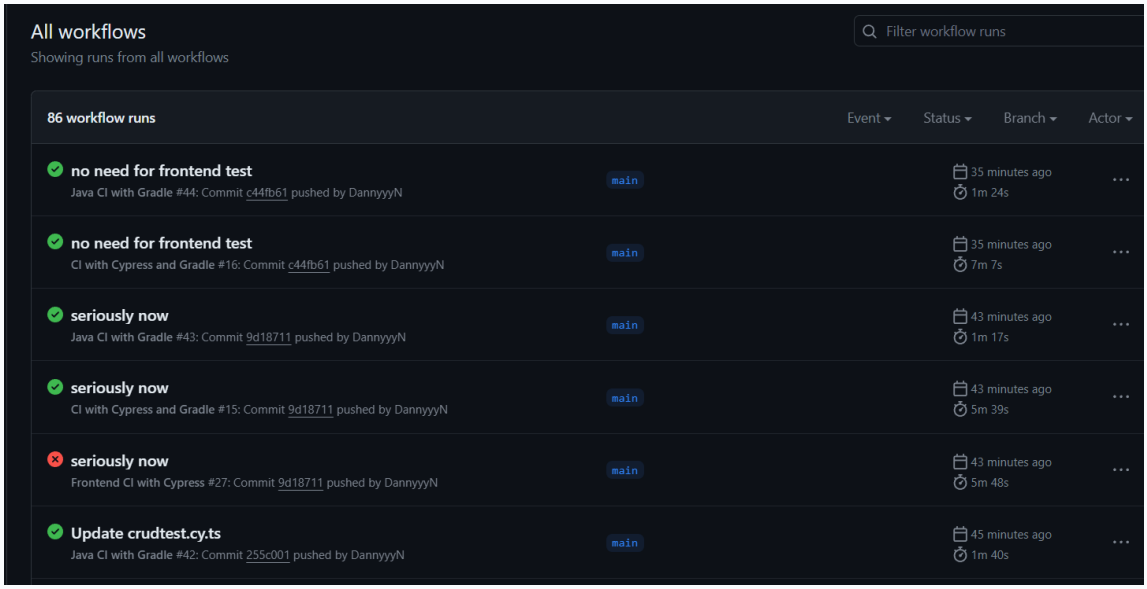
- Falsche Angaben (Preis, Name, Anzahl, etc.)
- Logout/Registrierung funktioniert nicht
- Keine Responsiveness für Mobile Ansichten

Schwerwiegende Fehler: Applikation stürzt ab oder ist nicht funktionsfähig.

Test Deliverables

Zu den Test-Artefakten gehören:

- **Testkonzept**: Dieses Dokument.
- **Testfälle**: Dokumentation der Testfälle und -szenarien.
- **Testwerkzeuge**: JUnit für Unit-Tests, Cypress für E2E-Tests und Postman für REST-API-Tests.



Testing Tasks

Die Teststufen umfassen:

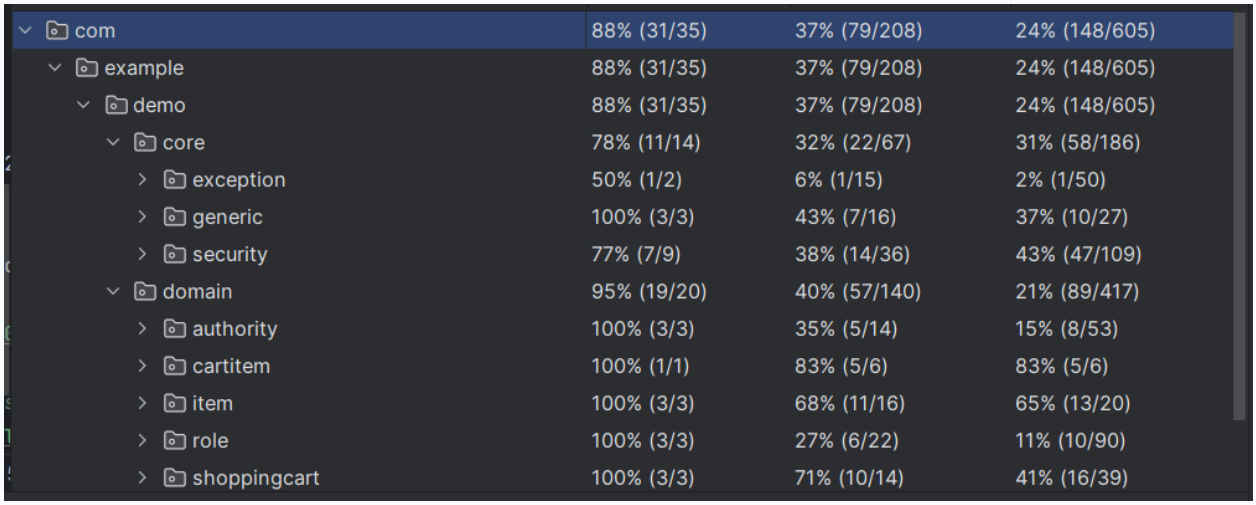
- **Unit-Tests**: Testen einzelner Komponenten.
- **E2E-Tests**: Testen des Workflow und die Interaktionen, die ein Benutzer in einer Applikation durchführen würde.

Environmental Needs

Die Testumgebung besteht aus:

- **Hardware**: Server mit ausreichend Speicher und Rechenleistung.
- **Software**: Java, Spring Boot, Datenbank-Management-System (PostgreSQL), React, Cypress, JUnit, Docker

Bei den JUnit Tests haben wir uns bewusst dazu einige Tests zu überspringen, da es schlicht recht unnötig und reine Zeitverschwendung wäre und den Fokus auf die Domains zu setzen. Daher beträgt unser Code Coverage im Backend auch “nur” 88%, was aber kein Problem darstellt.



com	88% (31/35)	37% (79/208)	24% (148/605)
example	88% (31/35)	37% (79/208)	24% (148/605)
demo	88% (31/35)	37% (79/208)	24% (148/605)
core	78% (11/14)	32% (22/67)	31% (58/186)
exception	50% (1/2)	6% (1/15)	2% (1/50)
generic	100% (3/3)	43% (7/16)	37% (10/27)
security	77% (7/9)	38% (14/36)	43% (47/109)
domain	95% (19/20)	40% (57/140)	21% (89/417)
authority	100% (3/3)	35% (5/14)	15% (8/53)
cartitem	100% (1/1)	83% (5/6)	83% (5/6)
item	100% (3/3)	68% (11/16)	65% (13/20)
role	100% (3/3)	27% (6/22)	11% (10/90)
shoppingcart	100% (3/3)	71% (10/14)	41% (16/39)

Reflexion

Joao

Am Anfang dieses Moduls wusste ich ehrlich gesagt nicht ganz, was wir für 10 Wochen machen würden. "Testing" war für mich ein vages Thema, mit dem ich nicht viel anfangen konnte. Erst mit diesem Modul wurde mir wirklich klar, wie wichtig Testing in einem Projekt sein kann und wie vielfältig es eigentlich ist. Bis jetzt habe ich das Testing vernachlässigt, aber jetzt nicht mehr.

Leider sind wir die erste Klasse, die dieses Modul vorgenommen hat, und ich muss zugeben, es hat vieles, was meiner Meinung nach verbessert werden kann. Die Aufgaben waren ab und zu schwer zu verstehen, was genau von einem erwartet wird, und die vorgeschriebene Funktionen für uns lokal zu testen sind nicht immer gelaufen wie erwartet. Die Gewichtung des Moduls war nicht ganz selbstverständlich und wurde immer wieder geändert. Und auch die Dokumentation, die uns gegeben wurde, war oft nicht erklärend genug, hatte viele redundante Teile und man konnte leicht falsch interpretieren, was darauf stand.

Die schriftliche Prüfung war aber ziemlich gut. Auch wenn die Dokumente, die zur Verfügung gestellt wurden, manchmal nicht das Gelbste vom Ei waren, hatte ich das Gefühl, ich konnte mich gut darauf vorbereiten.

Das Thema hat mir allgemein gefallen und es war spannend, zusammen mit meiner Gruppe tiefer in dieses Thema einzutauchen.

Danny

Das Modul war im ganzen sehr spannend und ich konnte unironisch viel Neues über Testing lernen. Das Konzept mit dem Miro-Board ist nichts Neues, aber ich war immer Fan davon, weil es einfach und übersichtlich ist.

Die Prüfung war eine gute Einbindung in das Modul und war grundlegend nicht schwierig oder zu einfach, sondern perfekt. Die Aufgaben waren auch machbar und die Theorie, die uns vom GitLab zugestellt wurde, ist informativ und hilfreich, aber einige Male musste auch von anderen Quellen Hilfe geholt werden.

Die Projektarbeit war ein ziemliches Chaos, da ich für die Pipelines insgesamt 86 Versuche gebraucht habe, weil jedes Projekt individuell ist und es keine Dokumentation gibt. Zum Beispiel war das Einbinden von Cypress ziemlich nervig, sowie auch einige dumme Fehler von Git, dass einige Files einfach ignoriert wurden, obwohl sie nicht im .gitignore enthalten waren.

Dafür, dass es ein erstes Modul war, gab es natürlich einige Schwierigkeiten, ich würde mir mehr wünschen, dass die Übungen etwas weniger werden oder eine andere Art für die LB2 gewählt wird, da parallel Übungen und eine Projektarbeit zu machen sehr viel Zeit und Stress kostet.

Orcun

Ich fand die Zeit in diesem Projekt und im Modul allgemein wirklich gut. Es war toll, dass wir die im Miro-Board erlernten Konzepte in den Aufgaben vertiefen und im eigentlichen Projekt anwenden konnten.

Beim Test Driven Development lief es nicht so reibungslos, da wir ein bestehendes Projekt verwendet haben, und es war schwer, neue Funktionen hinzuzufügen. Dennoch konnten wir sehen, wie der Prozess funktioniert, indem wir Tests für bestehende Funktionen erstellt haben.

Es war auch schwierig, die Pipelines und den CI/CD-Workflow einzurichten, da wir Schwierigkeiten hatten, die passenden Ressourcen zu finden. Aber am Ende haben wir ein funktionierendes Ergebnis erzielt.

Die Kommunikation im Team war wie immer sehr gut, da wir uns ständig über Teams oder andere Tools ausgetauscht haben. Jeder im Team war hilfsbereit, und die Verantwortlichkeiten wurden fair aufgeteilt.

Ich bin froh, dass ich das Gelernte aus diesem Modul in meiner zukünftigen Arbeit nutzen und meine Testing Kenntnisse erweitern konnte.