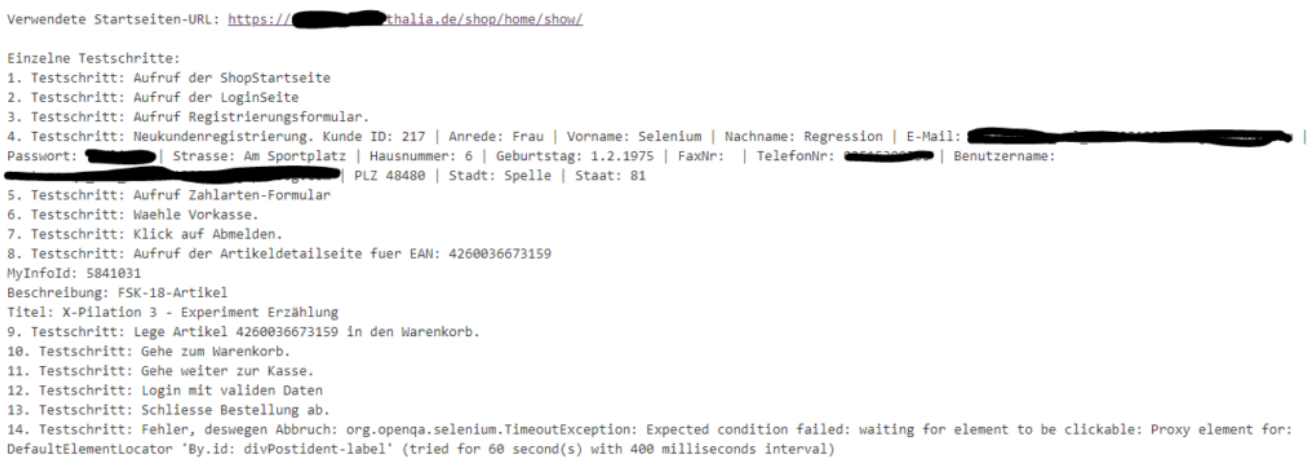
Wie UI-Component Tests unsere E2E-Tests schneller und robuster machen

E2E-Tests reduzieren ohne Testabdeckung einzubüßen? UI-Componenten Tests können hier helfen.

True Grid oder wie wir unsere Testausführung durch Parallelisierung 6x schneller machten

Bei Thalia gibt es nicht nur crossfunktionale Teams mit QAlern, es gibt auch die zentrale QA. Wir testen prozesskettenübergreifend – d.h. während die Kollegin aus Team Kaufen alle Eingabemöglichkeiten der Zahlarten durchtestet oder der Kollege aus Team Kunde alle möglichen (und unmöglichen) Lieferadressen hinterlegt, legen wir in einem Test einen Neukunden an, ändern die Zahlart, hinterlegen eine Lieferadresse, suchen einen Artikel und kaufen ihn dann. Alles natürlich automatisiert, und für 4 verschiedene Mandanten.



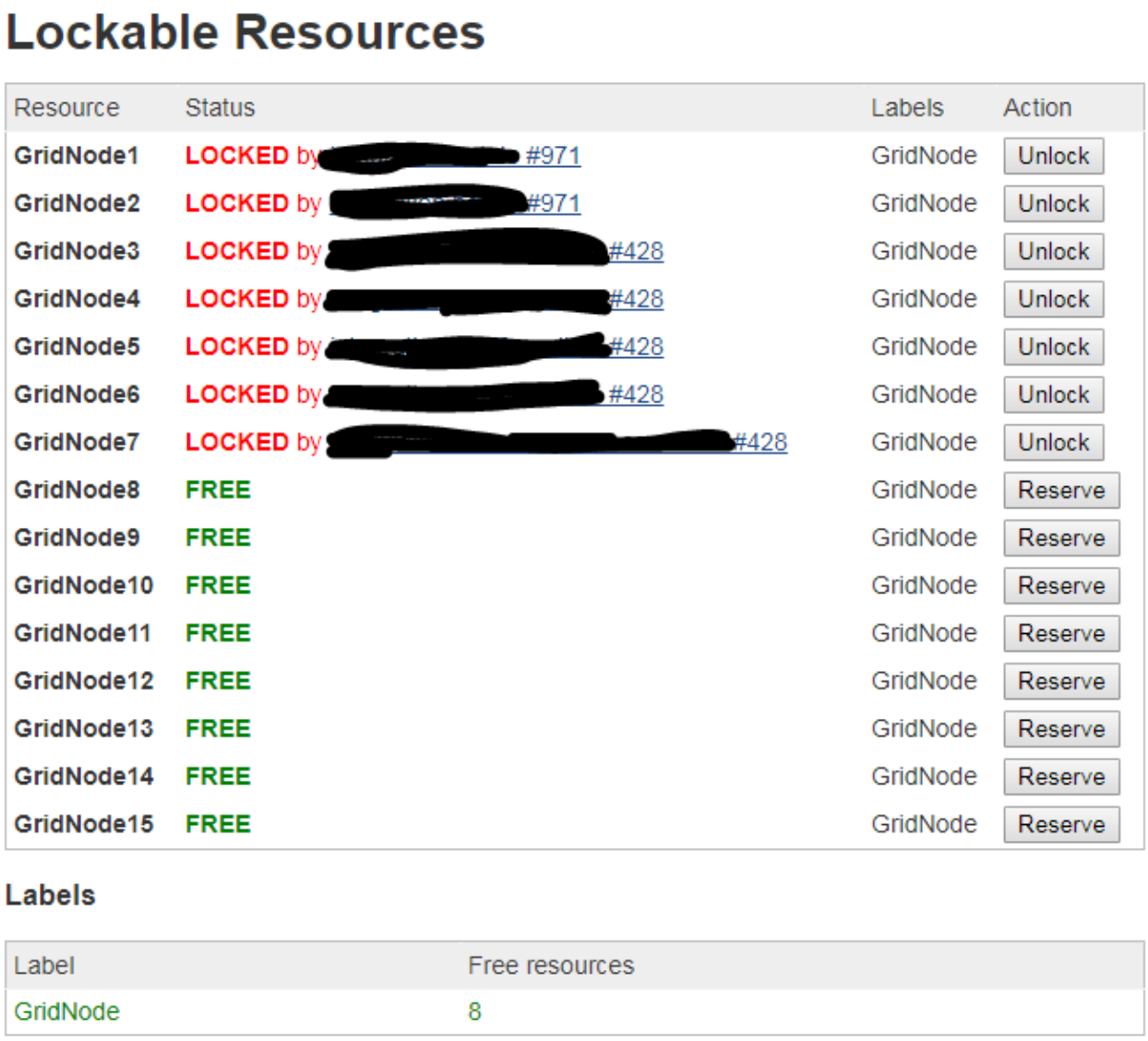
Insgesamt haben wir rund 100 Tests, die wir auch noch in 2 Browsern mit unterschiedlichen Auflösungen ausführen. Bis vor wenigen Wochen noch dauerte der größte Integrationstest ca. 6 Stunden… Wenn dann aber ein Hotfix aufgespielt werden soll und jemand schnell die Antwort haben möchte, ob denn die automatischen GUI-Tests fehlerfrei durchgelaufen sind, ist „Frag morgen nochmal nach“ keine angenehme Antwort – unsere Tests mussten schneller werden.

Wir überlegten kurz, ob weniger Tests in weniger Browsern oder ohne Screenshots eine Alternative wären, stellten aber schnell fest, dass unser Problem leichter mit mehr Rechnern und paralleler Testausführung zu lösen war. Selenium Grid mit allen Möglichkeiten zur einfachen Parallelisierung von ferngesteuerten Browsern war die Antwort.

Alle machen Docker, warum nicht wir? Bisher hatten wir unsere Tests auf Windows durchgeführt (die Mehrzahl der Kunden nutzt Windows), aber wenn man z.B. Zalenium (einen Docker-basierten, dynamischen Selenium Grid) nutzen will, muss man Linux als Plattform wählen. Leider scheint es mit Firefox unter Linux bei automatisierten Tests manchmal Probleme mit unserem Hauptmenü zu geben, was sich an unerwarteten Stellen über die wichtigen Elemente legt und damit den Testweitergang behindert. Außerdem kann man mit Zalenium Chrome nicht in der kleinsten Auflösung unserer Webseite benutzen – das ist uns aber sehr wichtig, daher nahmen wir wieder Abstand von einer Docker- und Linux-basierten Lösung.

Die Anzahl unserer Tests ändert sich nicht so häufig, und wie viele Nodes wir parallel brauchen, können wir einfach pro Testinstanz im Jenkins festlegen – daher entschlossen wir uns, mit einer festen Anzahl Windows-Rechner zu

arbeiten, die sich jeweils mit 4 Node-Prozessen an einem Selenium-Grid (der ausnahmsweise unter Linux läuft) anmelden. Um die genaue Anzahl von verfügbaren Browser-Handles und damit die Anzahl von Threads, die pro Test verwendet werden können, zu verwalten, sei hier noch löblich auf das Lockable Resource-Plugin im Jenkins verwiesen, mit dem wir recht einfach Flaschenhälse bei der gleichzeitigen Ausführung des Tests verhindern konnten.



Lockable Resources

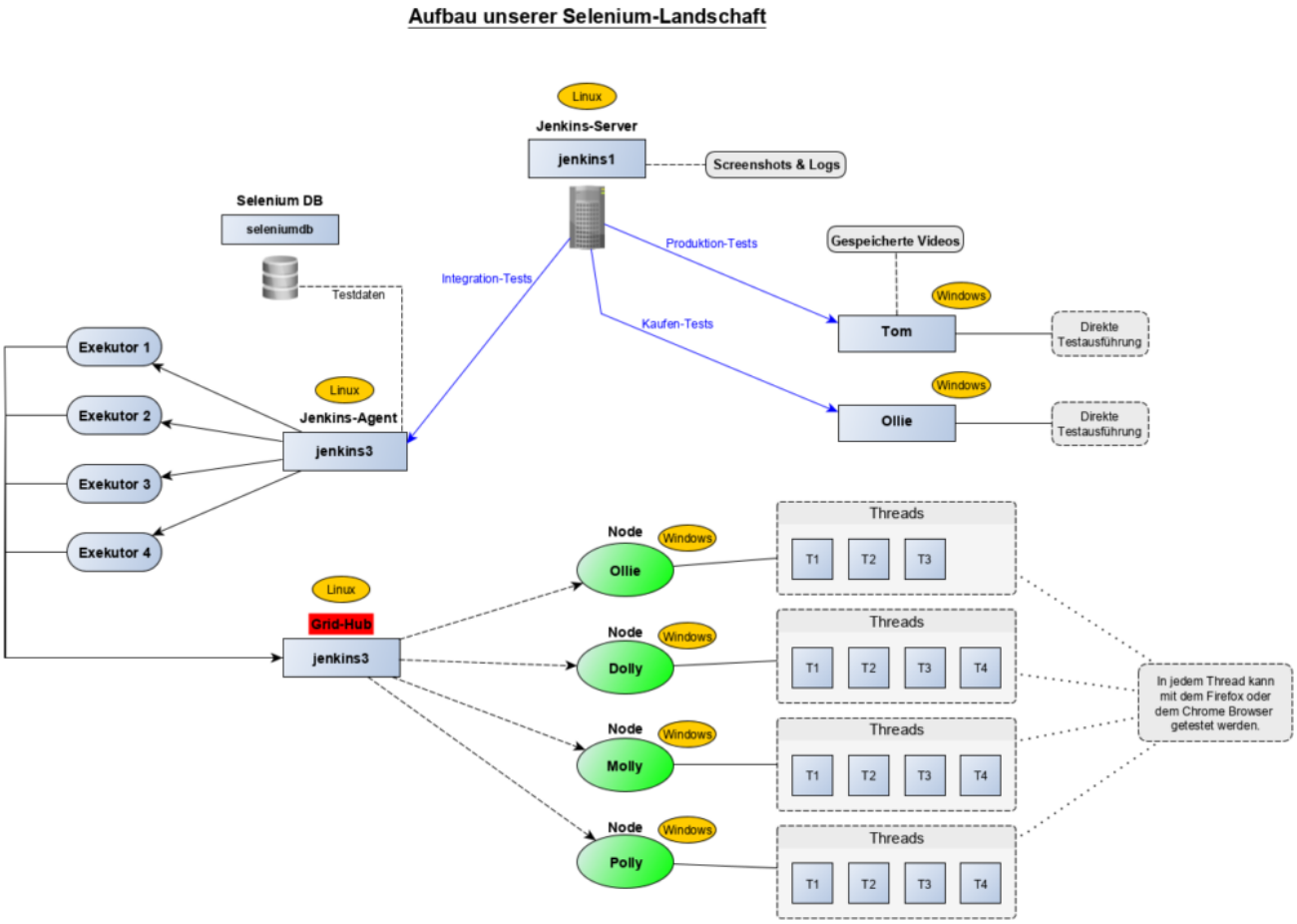
Resource	Status	Labels	Action
GridNode1	LOCKED by [redacted] #971	GridNode	Unlock
GridNode2	LOCKED by [redacted] #971	GridNode	Unlock
GridNode3	LOCKED by [redacted] #428	GridNode	Unlock
GridNode4	LOCKED by [redacted] #428	GridNode	Unlock
GridNode5	LOCKED by [redacted] #428	GridNode	Unlock
GridNode6	LOCKED by [redacted] #428	GridNode	Unlock
GridNode7	LOCKED by [redacted] #428	GridNode	Unlock
GridNode8	FREE	GridNode	Reserve
GridNode9	FREE	GridNode	Reserve
GridNode10	FREE	GridNode	Reserve
GridNode11	FREE	GridNode	Reserve
GridNode12	FREE	GridNode	Reserve
GridNode13	FREE	GridNode	Reserve
GridNode14	FREE	GridNode	Reserve
GridNode15	FREE	GridNode	Reserve

Labels

Label	Free resources
GridNode	8

Jetzt haben wir also 4 Windows-Rechner als Selenium Nodes. Ein weiterer Windows-Rechner wird für stündliche Tests gegen die Produktion verwendet. Und auf denen soll man jetzt Firefox- bzw. Chrome-Versionen im Blick behalten und die passenden Treiber für Selenium (geckodriver oder chromedriver) updaten? Sowas macht man heutzutage nicht mehr per Hand, es musste eine automatisierte Update-Lösung her. Dank Git und einem Batch-Update-Skript, was nächtlich läuft, kann ich auf meinem lokalen Rechner das Zusammenspiel von

Browser-Treiber und Browser testen, einchecken und am nächsten Tag davon ausgehen, dass die Updates auf alle Rechner verteilt sind.

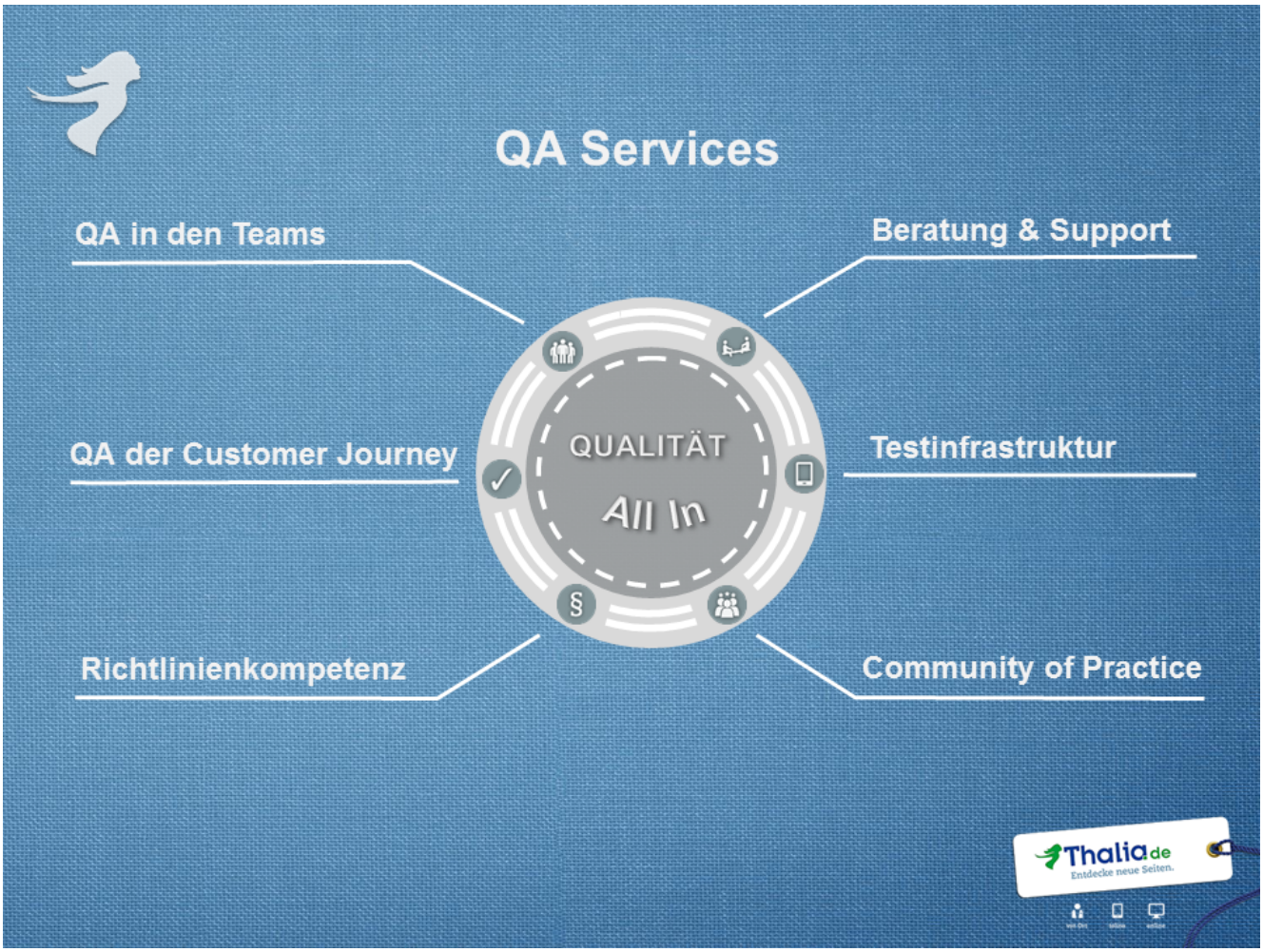


Und wie lange brauchen unsere Tests jetzt? Wir sind tatsächlich mit Tests für alle 4 Mandanten und noch einigen wenigen komplexen Testszenarien jetzt in rund 1 Stunde mit allen Tests durch – wenn alles gut läuft und die Testumgebung nicht zu sehr verändert wurde.

Build ↑	Duration	Agent
#1186	1 hr 1 min	um-qajenkins3
#1185	1 hr 1 min	um-qajenkins3
#1184	1 hr 20 min	um-qajenkins3
#1183	1 hr 3 min	um-qajenkins3

QA Services – Mittendrin statt nur am Ende

Der Bereich Thalia eCommerce Quality Assurance Services unterstützt die anderen Bereiche bei dem Ziel, gegenüber den Kunden und Kundinnen ein hochwertiges Produkt zur Verfügung zu stellen. Wir sind dabei entweder direkt einem Team zugeordnet oder in der zentralen QA tätig. Durch die permanente Weiterentwicklung von Produkten und Prozessen werden wir stets gefordert, neue passende Lösungen zu entwickeln. Derzeit stehen bei uns sechs Themen im Fokus:



QA in den Teams

Wir sind als QA-Lead direkter Bestandteil eines cross-functional Teams und nehmen vielfältige Aufgaben wahr:

- Abstimmung der teaminternen Qualitätskriterien

- Erster Ansprechpartner zum Thema Qualitätssicherung und Test
- Themen zur Testautomation im Team bündeln und vorantreiben
- und natürlich das normale Testgeschäft

Beratung und Support

Wir beraten die einzelnen Teams und Fachbereiche bei Themen wie Testfallermittlung, Testkonzeption, Tooleinsatz etc.

QA der Customer Journey

Wir prüfen zyklisch, dass die Customer Journey für unsere Kunden keine Stolpersteine enthält. Die Prüfung durch automatisierte Tests bauen wir stetig aus.

Testinfrastruktur

Wo und wann kann ich testen? Woher bekomme ich Testdaten? Womit kann ich Lasttests ausführen? Das sind einige Fragestellungen, die wir beantworten. Und falls wir die Antwort nicht kennen, sorgen wir für Lösungen.

Richtlinienkompetenz

Wir definieren Spielregeln, damit die Flexibilität und hohe Innovationsgeschwindigkeit der agilen Produktentwicklung nicht zu Lasten von Stabilität und Qualität geht.

Community of Practice

Wir tauschen uns zu spannenden Themen im Bereich der QA aus, um voneinander zu lernen und besser zu werden.