

„Wie „Null Bock auf immer das gleiche“ ein Motivator für Prozessoptimierung sein kann.“

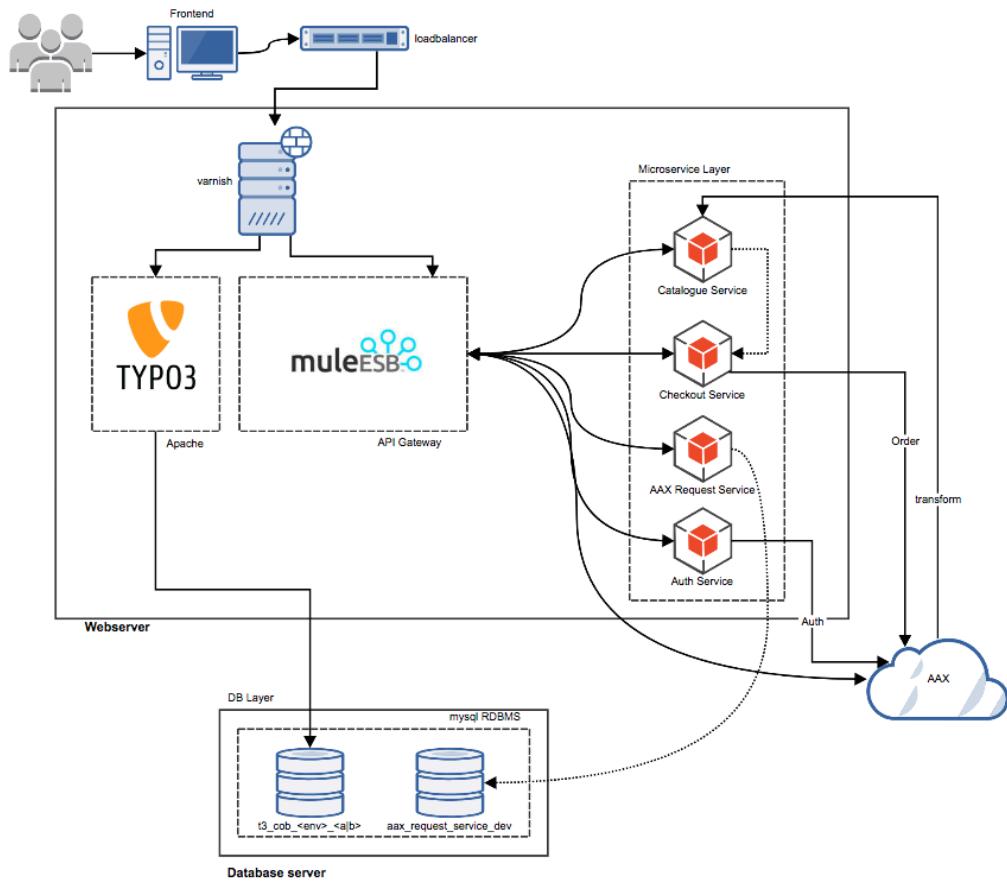
In diesem Artikel schreibe ich über meine Motivation eigene Projekte anzufangen und voranzutreiben. Ein zunächst beschwerlich erscheinender Weg kann im Rückblick oft zu Erkenntnissen führen, die im Arbeitsalltag im Verborgenen bleiben.

Team und Projekt

Wir sind ein kleineres Entwicklungsteam vier Entwicklern am Standort Wiesbaden und arbeiten für einen großen deutschen Telekommunikationsdienstleister. Unsere Aufgabe ist es Internetprodukte die an einer Adresse verfügbar sind zu ermitteln und im Weiteren diese „Bestellbar“ zu machen. Was auf den ersten Blick wenig spannend klingt — man entwickelt ja nichts das besonders „sexy“ anmutet, wird erst auf den zweiten Blick spannend. Denn setzt man sich mit den technischen Anforderungen auseinander die notwendig sind um einen solchen Service zu entwickeln und zu betreiben wird schnell sichtbar, dass die spannenden Aspekte mannigfaltig sind und sich nicht rein auf Basis der sichtbaren Oberfläche bewerten lassen.

Projekthistorie

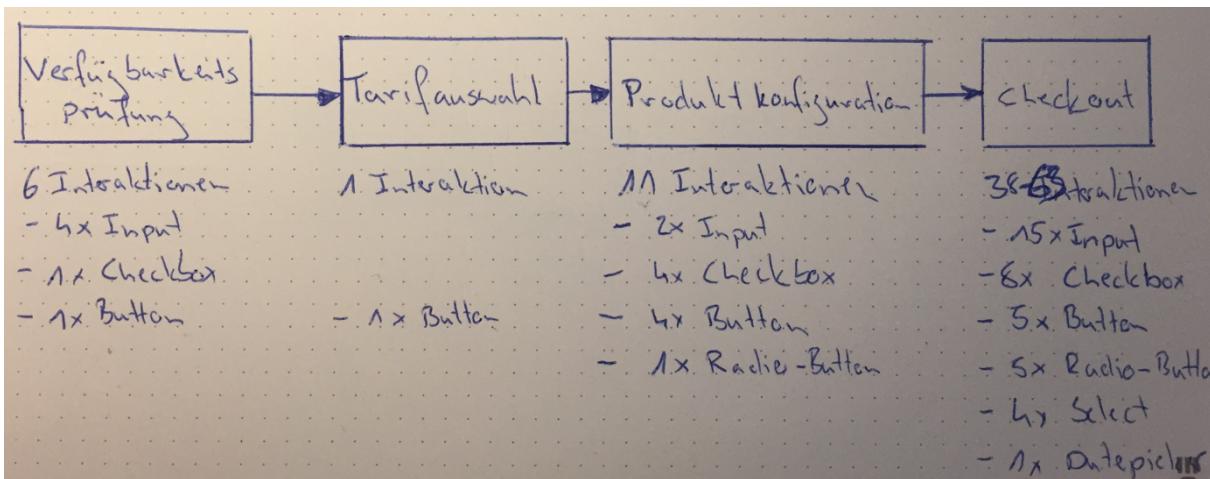
Unser Projekt begann auf Basis [des EFT Frameworks](#) um zeitnah nach Projektbeginn eine funktionsfähige Umgebung bereitstellen zu können. Neue Anforderungen setzten wir in einer Micro Service Architektur um. Dieser Schritt war notwendig um einerseits logische Trennungen zu vollziehen und andererseits fein granularer skalieren zu können. Ein weiterer durch die Granularität entstandener Mehrwert war die Reduzierung der Deploymentzeiten was zur Folge hatte, dass vor allem die Frontentwicklung und die Qualitätssicherung in kürzeren Zyklen entwickeln und testen konnten.



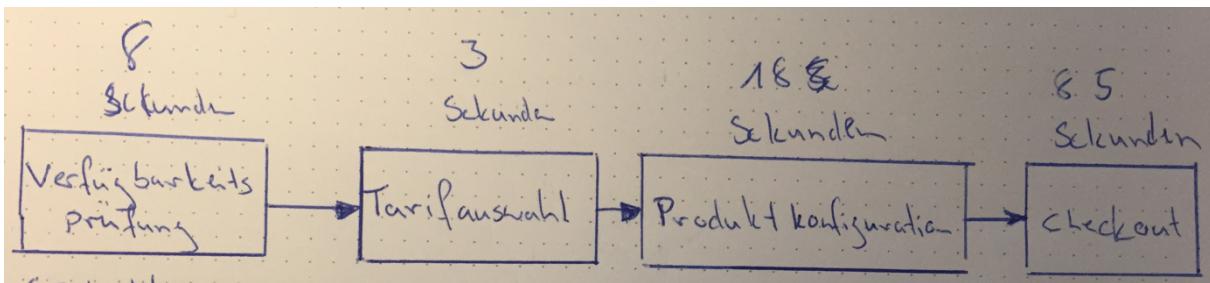
Highlevel Architektur

Faktor (anspruchsvoller) Mensch

Was im Frontendteil unseres Teams beim Entwickeln und manuellen Testen der Oberflächen und sekundär den beim Backendteil des Teams bei Tests der Services im Zusammenspiel mit weiteren Services zu hohen Aufwänden führte die aber gleichzeitig sehr linear waren mündete beim Benutzen des Frontend in ein Gefühl der Eintönigkeit was zuletzt auch negativen Einfluss auf die Aufmerksamkeit hatte, da hier ein „Routineprozess“ immer und immer wieder durchlaufen werden musste. Hierbei handelte es sich im Prinzip um eine Tätigkeit, die wenig automatisiert war und dadurch zu einem hohen Teil vom Faktor Mensch beeinflusst gewesen ist.



Buchungsprozess 56 – ca. 90 Interaktionen // Ø73 Interaktionen



Aufenthaltsdauer pro Prozessschritt bei manueller Buchung mit Routine zwei Minuten

Inspiration

In der Qualitätssicherung (QA) unseres Projekts wird mithilfe von Selenium sichergestellt, dass die Funktion des „Main Business Cases“ immer gegeben ist (Cross-Browser [IE 11, Edge, Chrome, Firefox]). Es wird automatisiert sichergestellt, dass sich nach Updates oder Hotfixes (dreimal in zwei Jahren bisher 🤘) alle Bereiche der Anwendung bei Interaktion mit dem Frontend vorhersehbar verhalten. Der zu treibende Aufwand um diese Qualitätsanforderungen sicherzustellen ist wegen der Menge des zu schreibenden Automatisierungscodes (Java/Selenium) und dem Bereitstellen der Testattribute im Markup nicht unerheblich. Dennoch ist dies notwendig um die Funktionalität der Plattformen über die verschiedenen Browser hinweg sicherzustellen.

Idee

„Die Seleniumtests automatisieren exakt die Bereiche, die in der Entwicklung rein manuell getestet werden.“

Wenn sich eine Automatisierung bereits in der Entwicklung des Frontends bewerkstelligen ließe — ohne das Schreiben von Mengen zusätzlichem Codes — kann der Anteil an manuellen Tests reduziert werden.

Die Anforderungen an die oben beschriebene Idee lässt sich, wie folgt zusammenfassen :

„Automatisiere jegliche Interaktion die notwendig ist um die Buchungsoberfläche bedienen zu können und biete ein hohes Maß an Flexibilität damit nach Änderungen am Frontendcode die Automatisierung mit geringem Zeitaufwand angepasst werden kann.“

So entstand das Side Project „FetchBot“

Als API für die Interaktion mit dem Browser kam [puppeteer](#) zum Einsatz und dessen Methoden für das Aufrufen und Steuern von Webseiten wurden in einem JSON Schema gemappt.

```
{  
  "http://google.com": {  
    "type": [ ["input", "web companies in wiesbaden\n"] ],  
    "waitFor": [ [3000] ]  
  }  
}
```

Das Schema folgt dem in der Abbildung beschriebenen Muster

Mithilfe eines Interpreters wird das Schema in ausführbaren Code übersetzt. Dieser Interpreter wurde als OpenSource Tool [veröffentlicht und FetchBot](#) genannt — zum „Fetch“ später mehr.

FetchBot kann als Library in eigene Projekte eingebunden- oder mithilfe eines Command Line Interfaces (CLI) betrieben werden.

```
npm install fetchbot
```

Installieren kann man FetchBot via npm

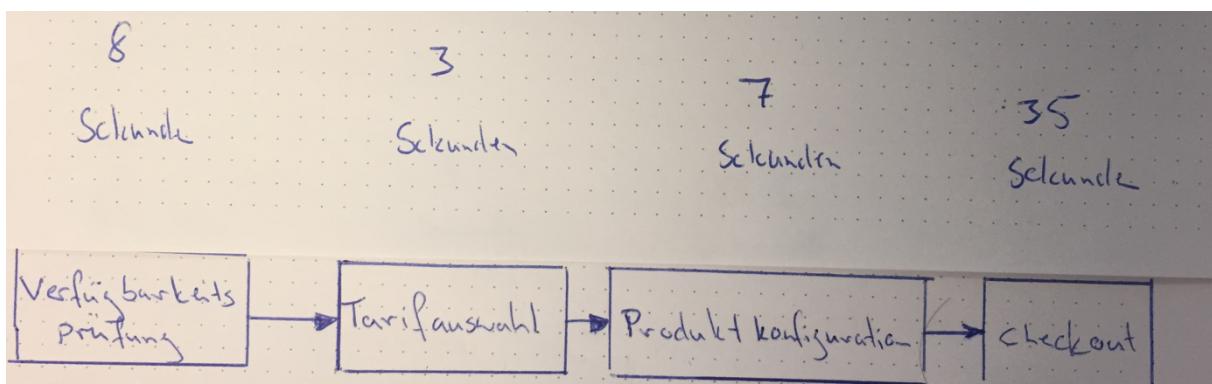
Integration in unserem Projekt einem Buchungsworkflow für Internetanschlüsse

Element Typ	Anzahl
Eingabefeld	21
Auswahllisten	4
Checkboxen	13
Radio Buttons	6
Datepicker	1
Buttons	11
Summe	min. 56 – max. ca. 90 Interaktionen

Interaktionen aufgeschlüsselt nach UI-Elementen im Buchungsprozess

Um eine Buchung abzuschließen sind im Mittel 73 Userinteraktionen notwendig und es dauert etwa zwei Minuten den Buchungsprozess manuell zu durchlaufen.

Praxis



Mit Fetchbot wurde die Buchungsdauer auf eine Minute reduziert. Nach den ersten Versuchen in der Praxis kristallisierte sich heraus, dass nicht vollständig auf manuelle Tests verzichtet werden kann, weil gerade die Usability nicht „Erfahrbar“ wird. Dennoch lassen sich sämtliche Interaktionen automatisieren, die notwendig sind, um zu einem bestimmten Buchungsschritt zu gelangen.

Soll beispielsweise im letzten Schritt des Checkoutprozesses eine Anpassung vorgenommen werden, so kann mithilfe von FetchBot bis genau zu diesem Schritt automatisiert gebucht werden um dann manuell weiter testen zu können. Diese Symbiose aus automatisierter und manueller Interaktion beschleunigte die Frontendentwicklung deutlich.

Automatisierungs Konfiguration

Mit einer dem o.g. Schema entsprechenden JSON-Datei (Konfiguration) lassen sich so auf 140 Zeilen alle Interaktionen zum Abschließen einer Buchung in unserem Workflow abbilden.

Um nicht in jedem neuen Sprint eine neue Konfiguration schreiben zu müssen, pflegen wir ein „**MasterSchema**“, welches die „**Maximalausprägung**“ abbildet. Ausgehend von dieser Maximalausprägung lässt sich durch entfernen der irrelevanten Anteile eine am Anwendungsfall ausgerichtete Automatisierung erstellen.

```
{
  "https://testserver.local/PostForms/webshopform.php": {
    "root": true,
    "click": "input[type='submit']",
    "waitFor": "[5000]
  },
  "https://testserver.local/tarifuebersicht-kabel": {
    "click": "#content > div > div.cobra-product-footer > p > button",
    "waitFor": "[1000]
  },
  "https://testserver.local/breitband/produktkonfiguration/": [
    {
      "click": "label[data-test-action='select-vlz-2a']",
      "waitFor": "[1000]
    },
    {
      "click": "button[data-test-action='continue-to-options']",
      "waitFor": "[1000]
    },
    {
      "click": [
        {"data-test-action='Flat International 1'}",
        {"data-test-action='Flat International 2'}",
        {"button[data-test-action='continue-to-hw']"
      ],
      "waitFor": "[1000]
    },
    {
      "click": [
        {"label[data-test-action='select-hw-5001']"
      ],
      "waitFor": "[1000]
    },
    {
      "click": "button[data-test-action='submit']",
      "waitFor": "[2000]
    }
  ],
  "https://testserver.local/bestellen/": [
    {
      "click": "[data-test-action='continue-to-contract-data']",
      "waitFor": "[1000]
    },
    {
      "click": [
        {"data-test-action='customer--salutation-2-via-label'"
      ],
      "type": [
        ["name\\customer--first-name"], "Bernhard",
        ["name\\customer--last-name"], "Behrendt",
        ["name\\customer--birthday_dd"], "16",
        ["name\\customer--birthday_yyyy"], "1984",
        ["name\\customer--headline-phone-number-area-code"], "+49",
        ["name\\customer--headline-phone-number-all"], "4624232424",
        ["name\\customer--email"], "bernhard.bezdek@aoe.com",
        ["name\\customer--email-repeat"], "bernhard.bezdek@aoe.com"
      ],
      "click": "[data-test-action='continue-to-line-and-internet-details']",
      "waitFor": "[1000]
    },
    {
      "click": "[data-test-action='products-dsl-order--current-provider']",
      "waitFor": "[1000]
    },
    {
      "click": "[data-test-action='products-dsl-order--current-provider-value-1']",
      "waitFor": "[1000]
    },
    {
      "type": [
        ["name\\current-line-owner--line-phone-no-area-code"], "+06661",
        ["name\\current-line-owner--line-phone-no-all"], "532454"
      ],
      "click": "[data-test-action='products-dsl-order--data-storing-option-0-via-label']"
    },
    {
      "click": "[data-test-action='order--media-wall-outlet-available-0-via-label']"
    },
    {
      "click": "[data-test-action='order--building-details--building-type-0-via-label']",
      "type": [
        {"name\\products-dsl-order--building-details--floor": "1"
      ]
    },
    {
      "click": "[data-test-action='continue-to-payment-details']",
      "waitFor": "[1000]
    },
    {
      "type": [
        ["name\\bank-account--iban"], "DE92530513960000525042"
      ],
      "click": "[data-test-action='sepa-mandate-check-via-label']",
      "waitFor": "[1000]
    },
    {
      "click": "[data-test-action='continue-to-terms-and-conditions']",
      "waitFor": "[1500]
    },
    {
      "click": "[data-test-action='terms-and-conditions-check-via-label']"
    },
    {
      "click": "[data-test-action='services-privacy-instruction-check']"
    },
    {
      "click": "[data-test-action='right-of-recession-check']"
    },
    {
      "click": "[data-test-action='credit-rating-check-via-label']"
    },
    {
      "click": "[data-test-action='continue-to-order-confirmation']",
      "waitFor": "[999999]
    }
  ]
}
```

Beispiel (ggf. als highlighted Code in Seite einbinden oder nur einen Ausschnitt zeigen)

Entwicklung des „Fetch“Anteils

Im Gespräch mit Kollegen aus anderen Teams erfuhr ich, dass diese puppeteer auch in ihren Side Projects einsetzen, um Daten aus Webseiten zu extrahieren und eigene Services damit zu betreiben. Hierfür war es notwendig JavaScript Code auf einer geöffneten Seite zu evaluieren. Extrahierte Daten werden in puppeteer als String zurückgegeben und müssen konvertiert werden um bspw. mathematische Operationen damit durchzuführen.

Das Extrahieren von Daten via Konfiguration ist eine weitere Funktion von FetchBot, die letztendlich auch namensgebend für das Tool ist.

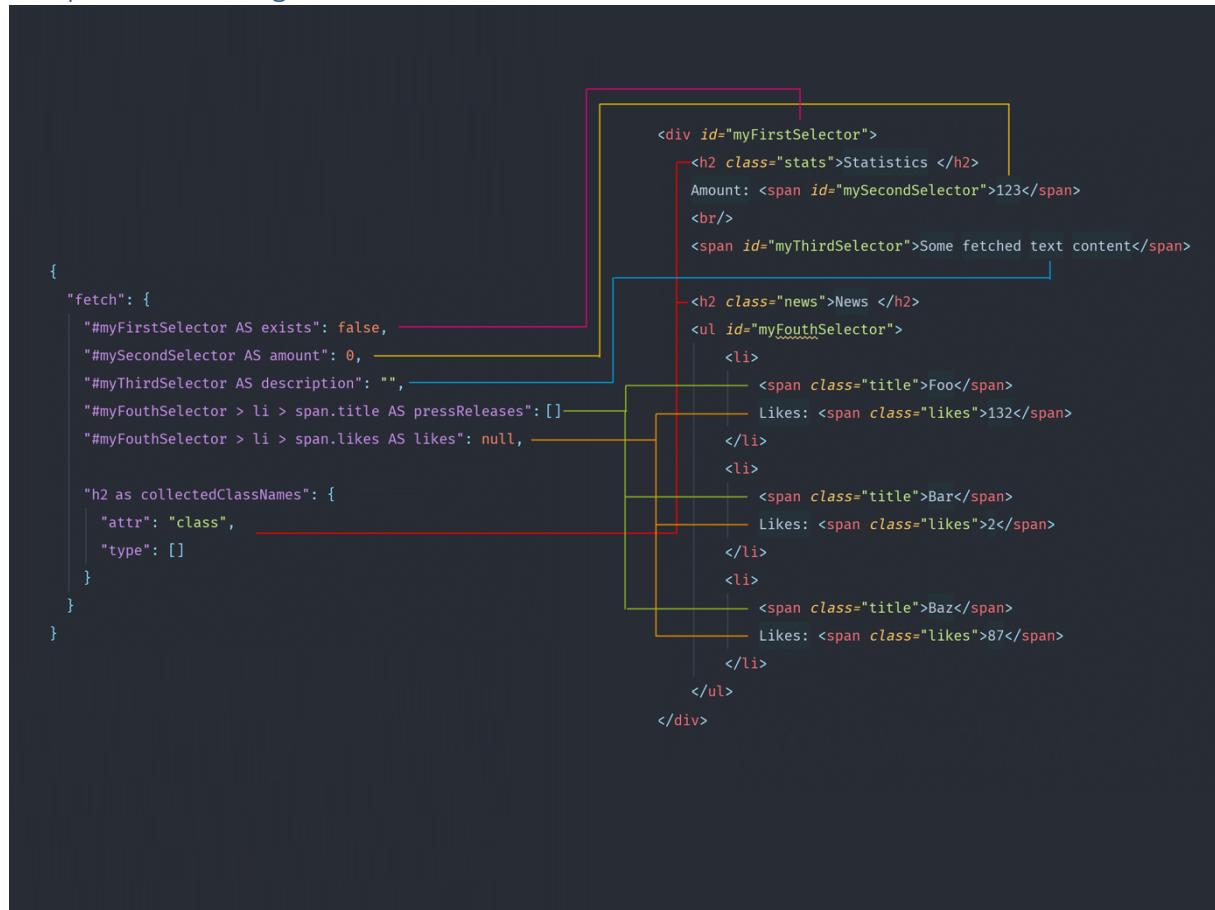
```
SELECTOR AS PROPERTYNAME : OF TYPE  
  
SELECTOR AS PROPERTYNAME : {  
    OF TYPE,  
    ATTRIBUTENAME  
}
```

Zu extrahierende Daten auf einer Seite werden wie folgt beschrieben

Boolean	Ermittelt ob Selektor existiert
Number	Casted numerischen String zu einer Zahl
String	Zeichenkette
String enthaltendes Array	Array das mit Strings befüllt ist
Number enthaltendes Array	Array das mit Zahlen befüllt ist
Attribute Objects	Wenn z.B. ein Attribut anstelle von Text ausgelesen werden soll

Typecastings werden wie folgt konfiguriert:

Beispiel Fetchkonfiguration



Das Mapping einer Konfiguration auf ein HTML Markup

Wird ein solcher Block auf einer Seite evaluiert, wird an den gefundenen Selektoren der Content oder Attributwert extrahiert.

```
{  
  "exists": true,  
  "amount": 123,  
  "description": "Some fetched text content",  
  "pressReleases": ["Foo", "Bar", "Baz"],  
  "likes": [132, 2, 87],  
  "collectedClassNames": ["stats", "news"]  
}
```

Ausgabe Fetchkonfigurations Ergebnis

Möglichkeiten

Mit dieser Technik lassen sich Webseiten - auch wenn diese Daten asynchron nachladen - wie eine API ansprechen „Apifizieren“. Damit ergeben sich viele Möglichkeiten, wie man klassische Webseiten nutzen kann bspw. als Datendienste.

In einem neuen Projekt erzeuge ich die Konfigurationen maschinell, um mit größeren Seiten Kontextbezogen interagieren zu können — es bleibt also spannend.

Schlusswort:

FetchBot zu entwickeln war eine interessante Herausforderung und hat den langweiligeren Teil meines Arbeitsalltages deutlich reduziert. Das endlos wiederholende Ausfüllen von Formularen kann schon sehr eintönig werden. Außerdem konnte ich so mit Kollegen Gedanken austauschen und ein Projekt vorantreiben, das mir bei meinen privaten Datenprojekten ebenfalls den Alltag erleichtert.

FetchBot ist zwar nicht so performant, wie ein für den Anwendungsfall optimierter Crawler aber dennoch schnell genug um viele Webseiten maschinell zu analysieren und dafür sehr einfach zu bedienen. Es reicht hier aus die Chrome Entwicklertools und JSON zu beherrschen um jede Webseite als API anzusprechen zu können.