Webapp Pattern: Die Status-Seite

Meine allererste Homepage bestand aus HTML-Seiten, die ich von Hand geschrieben habe. Diese kamen auf einen Apache Webserver - und die Homepage stand.

Seither haben sich meine privaten Webseiten - und später die für die Arbeit stetig weiter entwickelt. Eine professionelle Webapp in Ruby on Rails besteht oft aus vielen einzelnen Komponenten, die alle laufen müssen, damit die Anwendung korrekt läuft:

- Webserver (Apache / Nginx)
- Rails Server (Passenger, Unicorn, …)
- Datenbank / Redis
- Memcached
- Hintergrund Jobs
- E-Mail Versand
- intene und externe Webservices

Wie finde ich nun beim einrichten der Webapp auf einen neuen Server heraus, ob „alles passt“? Und wie richte ich für alles ein Monitoring ein?

Mit einer Status-Seite. Diese ist z.B. unter `/status` erreichbar und prüft alle benötigten Services durch, ob sie laufen. Auf der Seite sind dann alle laufenden Services mit einem grünen Haken aufgelistet. Alle Services, die nicht funktionieren, kommen mit einem roten Ausrufezeichen. Sobald ein Service nicht läuft, gibt die Seite einen 500er Statuscode zurück, z.B. `503 Service Unavailable`, sonst ganz normal einen 200.

Dadurch ist für Mensch und Monitoring auf einen Blick ersichtlich, ob die App läuft.

Wie wird das in Rails umgesetzt? Abgesehen von den Grundlagen (Controller, View, Route) sind der Kern eine Sammlung von Checks mit Namen (Klassen in einem Modul oder ein Hash mit Namen und Blocks). Diese geben jeweils als Boolean zurück, ob der geprüfte Service gerade läuft. Die Checks werden dann aus Zeitgründen parallel in Threads ausgeführt und dann die Ergebnisse eingesammelt.

Wie ist ein Check aufgebaut? kurze Antwort: wie es sinnvoll ist.

- Die Checks für Webserver und Rails Server kann man sich sparen, ohne die wird keine Seite ausgeliefert.
- Die Datenbank lässt sich mit einem Test-Request testen. (z.B. Dummy-Tabelle anlegen, UTF-8 Content schreiben, Tabelle löschen)
- Für Redis kann man einfach einen Test-Eintrag schreiben und wieder löschen. Memcached analog.
- Hintergrundjobs kann man oft daran erkennen, dass der Prozess läuft.
- Für E-Mail Versand kann man die Verbindung zum Mailserver aufbauen. Vom Versand echter E-Mails würde ich abraten.
- Webservices haben oft schon eine `test` Funktion, die man aufrufen kann. Ansonsten muss man von Fall zu Fall entscheiden.

Am Ende hat man dann etwas Arbeit hinein gesteckt - hat es aber in Zukunft leicht(er) Probleme zu erkennen, diagnostizieren - und damit auch zu beheben.

Coding, Pattern