JavaScript in der Software-Architektur

Eine Darstellung für Manager & Entscheider

Prof. Dr. Ulrich Anders, Cologne Business School

Version 1.0.0 · 08. Oktober 2016
Version 1.1.0 · 21. Februar 2017
Version 1.2.0 · 23. Februar 2017
Version 1.3.1 · 22. Oktober 2017

1 Was ist ein Stack?

In der Software-Architektur unterscheidet man typischerweise zwischen den Technologien, die im *Frontend* und im *Backend* zum Einsatz kommen. Das Frontend beinhaltet das, was der Nutzer sieht und verwendet. Mittels des Frontends richtet der Nutzer Anfragen an das Backend, das die Antworten serviert. Das Backend umfasst daher Aufgaben wie Datenmanagement, Kommunikation zwischen Nutzern, Zahlungsverkehr oder aufwändigere Berechnungen.

Herkömmlicherweise werden im Frontend und im Backend unterschiedliche Technologien eingesetzt, die auf verschiedenen Programmiersprachen basieren. Das Frontend ist, sofern Browser zum Einsatz kommen, grundsätzlich und auf absehbare Zeit auf drei Programmiersprachen beschränkt: HTML, CSS & JavaScript. Im Backend werden hingegen sehr unterschiedliche Sprachen verwendet: PHP, Java, Ruby, Python, Go, andere und ebenfalls JavaScript.

Legt man alle Technologien, die für einen Teil einer Softwarelösung zum Einsatz kommen, auf einen gedachten Stapel, spricht man von einem Technologie-Stack für diesen Teil. Packt man alle Technologien, die über eine gesamte Softwarelösung — also sowohl im Frontend, als auch im Backend — verwendet werden, auf einen gedachten Stapel, spricht man von einem *Full-Stack*.

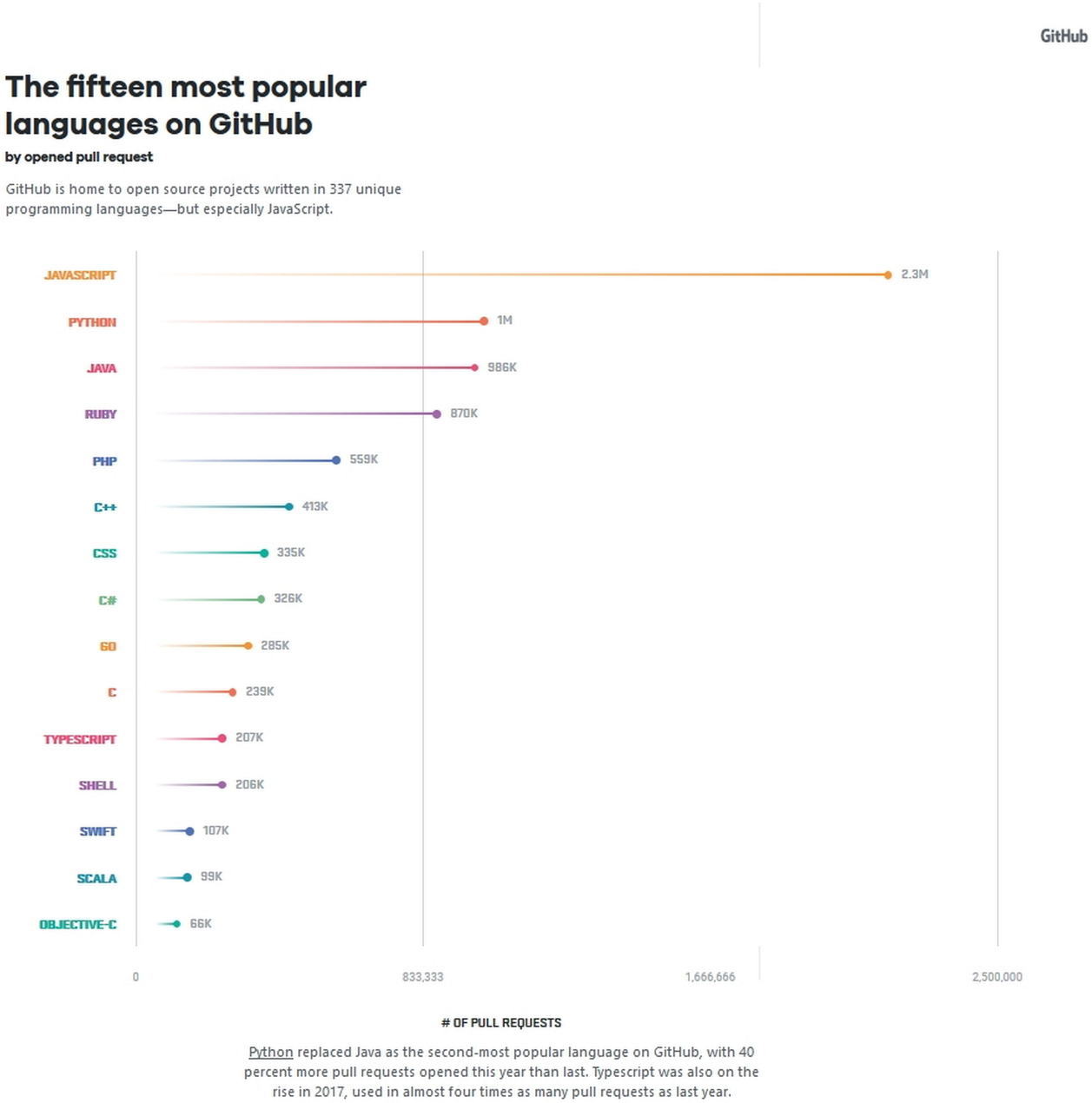
2 JavaScript

JavaScript entwickelt sich seit einigen Jahren zu der wichtigsten Technologie im Internet. Es gibt endlose Diskussionen über die Qualität der Programmiersprache im Vergleich zu anderen Sprachen.

Am Ende ist diese Diskussion jedoch müßig. JavaScript ist die einzige Sprache, mit der sich Aktionen innerhalb eines Browsers entwickeln lassen. Das wird sich auch so schnell nicht ändern. Deshalb ist sie bei einer Browser-basierten Lösung für das Frontend, also für die Benutzerseite, gesetzt. Da sich zudem auch immer mehr Inhouse-Softwarelösungen der Browser-Technologie und -Architektur bedienen, wird auch in diesem Feld JavaScript immer wichtiger.

Auf der Serverseite gibt es verschiedene Technologien. Zunächst einmal besteht die Möglichkeit, die Serverseite zu einem Cloudanbieter auszulagern. Für wen das nicht in Frage kommt, der kann im Backend unter einer großen Bandbreite an Technologien wählen, die natürlich von der Wahl der Programmiersprache, in der sie realisiert wurden, abhängen.

Noch vor einigen Jahren gehörte JavaScript nicht zu den Programmiersprachen, die auf der Serverseite zur Verfügung standen. Dann aber wurde die sogenannte JavaScript-Engine aus dem Browser auf die Serverseite portiert. Diese Portierung heißt »node«. Mit dieser Portierung der JavaScript-Engine von der Benutzerseite auf die Serverseite existiert jetzt mit JavaScript erstmals eine Programmiersprache, die durchgängig über den gesamten Technologie-Stack genutzt werden kann. Seitdem dies passiert ist, nimmt die Bedeutung an JavaScript ständig zu. Denn es erhöht die Flexibilität in jeder denkbaren Dimension ungemein, wenn die gleiche Sprache sowohl im Frontend, als auch im Backend verwendet werden kann. Betrachtet man den Vergleich auf GitHub, das in jeder Dimension die Referenz in Hinsicht auf Softwareentwicklung darstellt, zeigt sich das gleiche Bild.



Quelle: GitHub Octoverse 2017

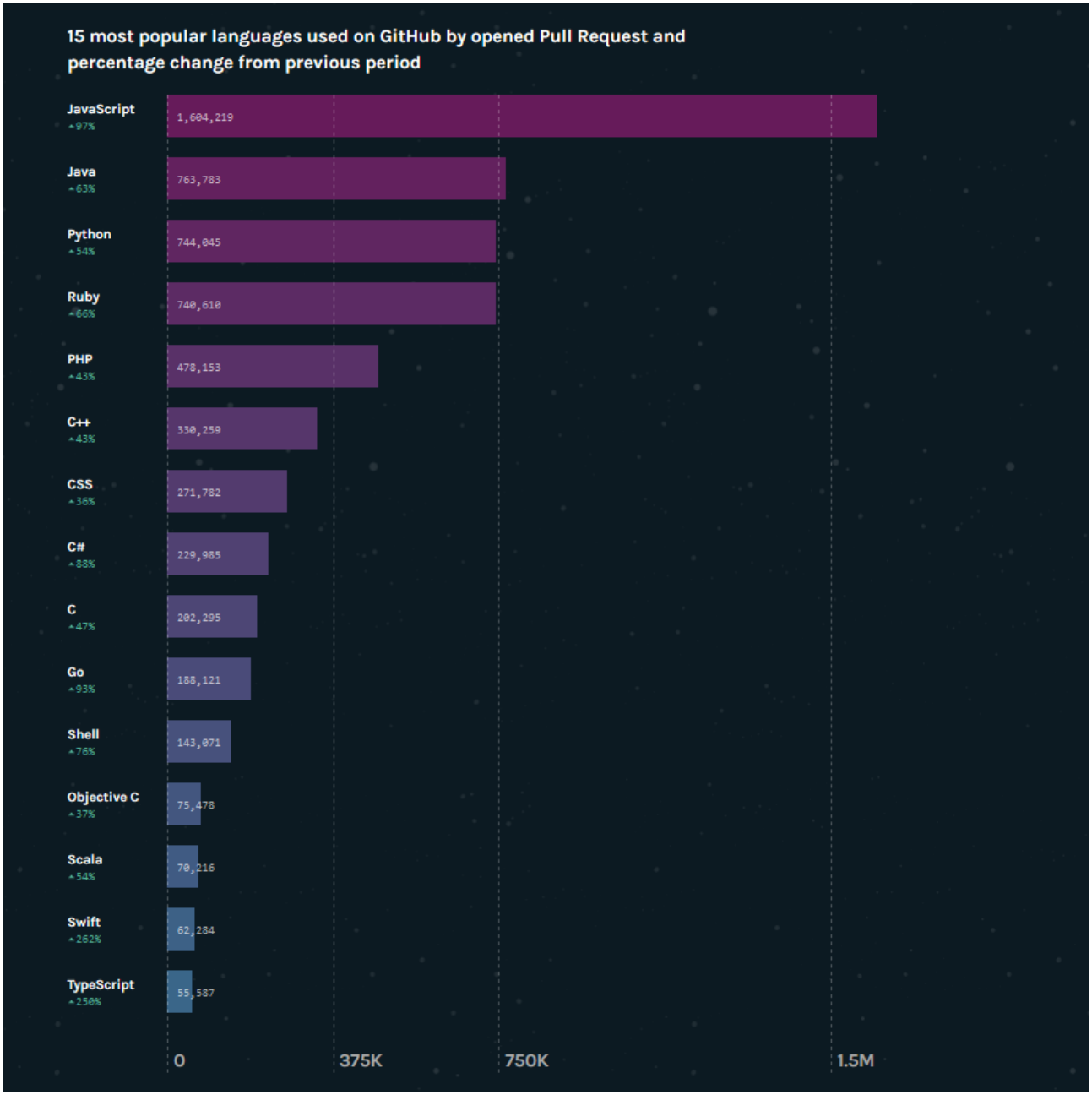
Dieser Trend wird sich noch verstärken, wenn JavaScript bald auch zur Entwicklung nativer Apps auf iOS und Android eingesetzt werden kann. An Lösungen dazu wird bereits intensiv gearbeitet.

Die prominenteste dieser Lösungen heißt React Native. Zum Zeitpunkt Oktober 2017 wird diese Bibliothek bereits über eine halbe Millionen Mal pro Monat von npm heruntergeladen. npm ist das

zentrale Register für JavaScript Pakete.

Mit React Native müssen Apps nicht mehr in ObjectiveC, Swift oder Java programmiert werden, um auf Smartphones oder Tablets zu laufen. Stattdessen kann alles in einer Programmiersprache, nämlich JavaScript, und weitgehend parallel entwickelt werden.

Die erste Version dieses Artikels stammt aus 2016. Zu diesem Zeitpunkt existierte lediglich die folgende GitHub Statistik. In 2017 hat JavaScript seinen Vorsprung vor den anderen Sprachen weiter ausgebaut. Das gilt umso mehr, wenn man TypeScript hinzuzählt. TypeScript ist JavaScript, allerdings mit Typendeklarationen. Die Popularität von TypeScript hat ebenfalls stark zugenommen. Java wurde mittlerweile von Python überrundet.



Quelle: GitHub Octoverse 2016

3 Kriterien für die Software-Technologie

Was sind die wichtigsten Kriterien für Technologie-Entscheidungen innerhalb eines IT-Projekts?

Zwei Antworten werden gern gegeben: 1. Bekanntheit und Reife der Technologie, 2. die Kosten der Entwicklung. Beide Antworten sind sicher richtig, sind aber nicht umfassend genug. Fast wichtiger sind aus Sicht eines Entscheiders Kriterien, die die Kosten und Aufwände in Zukunft überschaubar machen. Dazu gehören:

- **Modularisierung und Management von Komplexität:** Programmiercode ist komplex und wächst mit der Zeit noch an. Damit er auch langfristig noch durchschaubar ist, sollte der Programmcode in überschaubare und abgeschlossene Module und Komponenten heruntergebrochen werden. Diese sollten dann je nachdem, wo sie benötigt werden, importiert und verwendet werden können.
- **Test- und Wartbarkeit:** Programmiercode muss getestet sein und auch werden. Es sollte keinen Programmcode ohne zugehörige Tests und Dokumentation geben. Idealerweise werden Module individuell getestet und dokumentiert. Denn durch automatische Tests und Dokumentation wird der Programmcode auch in Zukunft wartbar. Gute Vorbereitung im Jetzt spart in der Zukunft Kosten, Zeit und Sorgen.
- **Zugänglichkeit der Technologie:** Zugängliche und benutzerfreundliche Technologien ziehen größere Mengen an Entwicklern an als sehr komplexe Technologien. In zugänglichen und benutzerfreundlichen Technologien zu entwickeln, macht eben auch Entwicklern viel mehr Spaß und ist deutlich motivierender.
- **Ressourcenverfügbarkeit:** Menschen verändern sich, Code bleibt. Damit Programmcode auch in Zukunft noch gepflegt und weiterentwickelt werden kann, muss es genügend Programmierer geben, die die Technologie beherrschen. Daher ist es hilfreich, wenn Software auf möglichst nur einer Programmiersprache und den in dieser entwickelten Technologien aufsetzt. Ideal ist wenn die gleichen Ressourcen für die Wartung des Frontends und des Backends eingesetzt werden können. Das ermöglicht dem Eigentümer des Softwarecodes eine größtmögliche Flexibilität in Hinsicht auf seinen Ressourcenpool.
- **Zukunftsfähigkeit:** Was nützt die beste Softwarelösung, wenn die unterliegende Technologie langsam ausstirbt und in Zukunft nicht mehr gepflegt werden kann. Software sollte also mit einer Technologie von heute entwickelt werden, von der man sicher ist, dass sie auch noch in Zukunft stark ist.

4 Die Vorteile von JavaScript

In Hinsicht auf die vorgenannten Kriterien bietet JavaScript eine große Reihe von Vorteilen.

Das Sprachsystem ist modular aufgebaut und man kann Module in kleine Einheiten kapseln und von überall her importieren. Das reduziert die Komplexität signifikant. Demzufolge steht auch eine große Anzahl von Modulen und Bibliotheken zu allen möglichen Themen zur Verfügung, die sich leicht in die eigenen Software einbinden lassen.

Zu solchen Bibliotheken gehören auch umfangreiche Test-Bibliotheken, die das Testen der Software unter verschiedenen Kriterien signifikant erleichtert oder sogar automatisiert.

JavaScript ist gut zugänglich für Programmierer und Entwickler. Zur Erleichterung, Beschleunigung und Optimierung der Entwicklung hat die Gemeinde ein große Anzahl an Tools und Werkzeugen

kreiert, die die Entwicklung komfortabel und schnell macht.

Solange JavaScript die einzige Sprache ist, die innerhalb eines Browsers genutzt werden kann, ist die Zukunftsfähigkeit von JavaScript gegeben. Daher wird sich JavaScript auch im Backend immer stärker ausdehnen und Verbreitung finden. Sicher wird irgendwann auch JavaScript einmal abgelöst werden, aber mit Sicherheit wird keine der heute im Backend zur Verfügung stehenden Sprachen den Weg in den Browser schaffen. Bis JavaScript also tatsächlich abgelöst wird, werden noch Jahre vergehen.

Die Ablösung von JavaScript wird auf der anderen Seite um so unwahrscheinlicher, je stärker die Sprache sich selbst modernisiert. Und das passiert derzeit in geradezu bespielloser Form. Während der JavaScript Sprachstandard über Jahre weitgehend unverändert geblieben ist, hat sich in den letzten zwei Jahren hier eine ungeheure Dynamik ergeben. Die Sprache wurde deutlich modernisiert und erweitert. Viele weitere sinnvolle Ergänzungen sind bereits in fortgeschrittener Diskussion und kurz vor der Vereinbarung. Zukünftig soll der Sprachstandard im Jahresturnus ergänzt und weiterentwickelt werden.

Je breiter eine Sprache eingesetzt werden kann, je leichter zugänglich sie ist, je mehr Spaß die Entwicklung in ihr macht und je schneller sichtbare Ergebnisse erreicht werden können, desto größer ist die Attraktivität für Entwickler. JavaScript punktet hier signifikant und sicher werden immer mehr Entwickler von anderen Programmiersprachen auf JavaScript umsteigen.

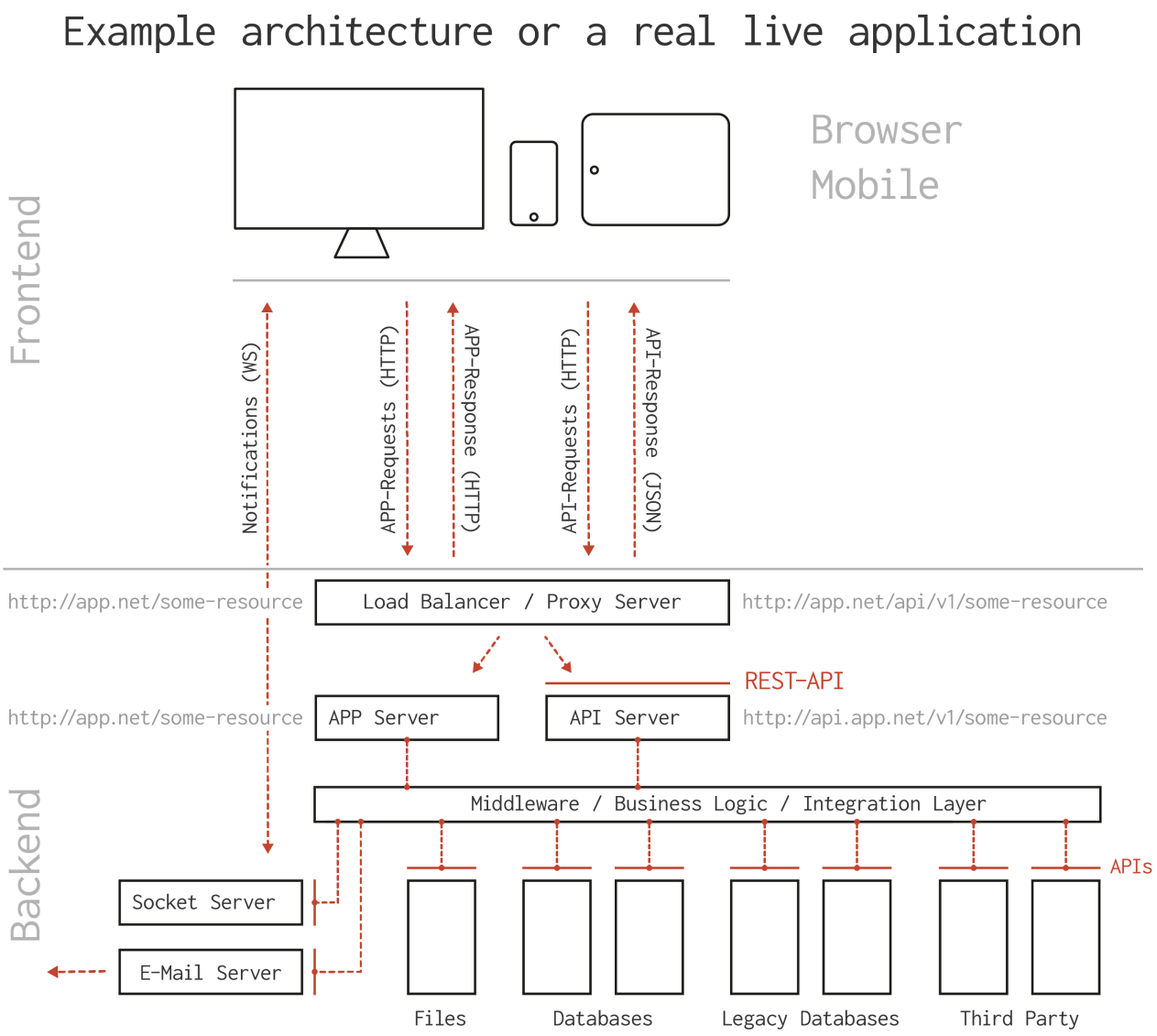
5 Ein typische Software-Architektur ohne Cloud-Dienste

JavaScript eignet sich hervorragend, um eine Browser-basierte Software-Lösung über den gesamten Stack zu realisieren. Im Browser ist JavaScript ohnehin gesetzt. Verwendet man im Browser so moderne komponentenbasierte Architekturen wie beispielsweise React, lässt sich der Frontend-Code bereits auch jetzt schon zur Erstellung von Apps portieren.[1] Alternativ muss man — weiter wie bisher — nativ auf iOS und Android entwickeln.

Im Backend stellt sich die Situation folgendermaßen dar: von den Nutzergeräten aus stellen die Nutzer Anfragen an den Server. Diese treffen zunächst auf einen zwischengeschalteten sogenannten Proxy-Server. Dieser leitet die Anfragen, je nach Art, entweder an dezidierte dahinterliegende Server weiter oder verteilt die Anfragen je nach Last auf mehrere gleichwertige parallele Server.

Sofern die Anfragen nicht an einen spezifischen Teil der App im Backend gestellten werden, sollten sie in einer möglichst allgemeinen Form an eine definierte Datenschnittstelle gerichtet werden. Ein sehr häufig verwendetes Architektur-Schema nennt sich REST API. API steht für *Application Programming Interface*. Die API definiert also, was von der Datenschnittstelle zurückkommt, und in welcher Form man dazu die Anfrage stellen muss. REST steht für *Representational State Transfer*. Dahinter verbirgt sich kurz gesagt das Prinzip, dass sich eine *Application* immer in einem Zustand (State) befindet. Zwischen zwei Zuständen findet dann der Transfer von Daten statt, um den ersten Zustand in den zweiten zu überführen. Die Schnittstelle weiß also nie etwas über den Zustand der Application, sie ist *stateless*.

[1] Diese zukunftsweisende Technologie z.B. in Form von React Native wird bereits produktiv eingesetzt.



Der Vorteil eine definierten Datenschnittstelle besteht darin, dass hinter der Datenschnittstelle alle möglichen Systeme, Server und Datenbanken in die Bereitstellung, Aufbereitung und Lieferung der Daten eingebunden werden können. Das ist die sogenannte *Middleware*, die auf der Datenhaltung aufsitzt. Die Middleware beinhaltet eigene Berechnungen und Analysen, Datenabstraktion und Datenintegration, aber auch die Anfrage an Drittsysteme, Legacy-Systeme, sowie an interne und externe Datenbanken. Die Middleware steuert im Übrigen auch das Schreiben in die Datenbanken. Ebenfalls führt sie auch die Steuerung von Kommunikationssystemen wie beispielsweise von E-Mail-Servern oder von Push-Diensten durch.