Linux-Desktopumgebung Cinnamon 4.8 veröffentlicht

[expand title="mehr lesen…"]

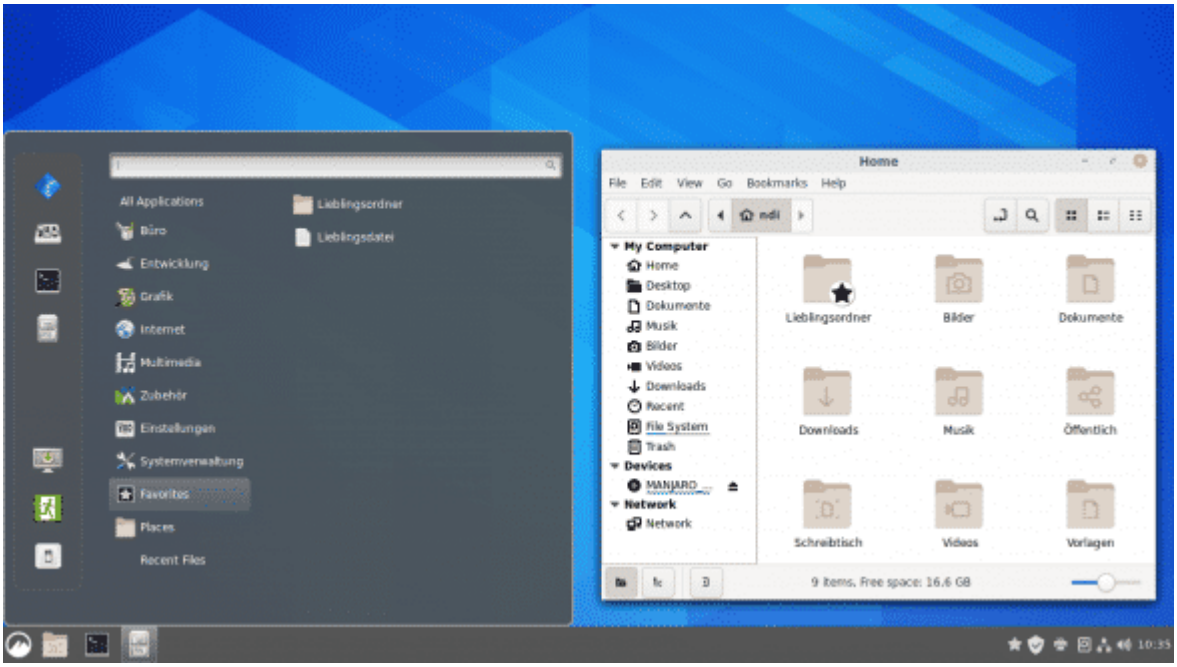
Linux-Desktopumgebung Cinnamon 4.8 veröffentlicht

Linux-Desktopumgebung Cinnamon 4.8 veröffentlicht

Die neue Version des Cinnamon-Desktops gibt einen Vorgeschmack auf das kommende Linux Mint 20.1 und hilft dabei, oft genutzte Dateien und Ordner schnell wieder zu finden.

Die Entwickler von Linux Mint und Cinnamon veröffentlichen den Cinnamon-Desktop 4.8 einige Wochen vor Linux Mint 20.1 „Ulyssa". Interessierte Nutzer können die Desktopumgebung bereits installieren und Feedback geben, bevor Cinnamon 4.8 mit dem neuen Linux Mint gegen Ende des Jahres ausgeliefert wird. Zu den Neuerungen bei Cinnamon zählt eine bessere Integration von Flatpaks.

Ist eine Anwendung als klassisches Paket und als Flatpak installiert, hängt Cinnamon 4.8 „(Flatpak)" an den Namen an. Nutzer können so unterschiedliche Versionen besser erkennen und gezielt starten. Ordner und Dateien lassen sich über das Kontextmenü im Dateimanager Nemo als „Favoriten" markieren. Zugriff auf die Favoriten besteht anschließend in der Seitenleiste von Nemo, im Hauptmenü und über das Favoriten-Applet.



In Nemo als Favoriten markierte Dateien und Ordner kann man über das Hauptmenü aufrufen.
Mit den Cinnamon „Spices“ können Nutzer den Desktop um Themes, Applets und Desklets erweitern. Ähnlich zu Gnome-Extensions sind die Spices nun in verschiedenen Versionen verfügbar. So wird die zum Cinnamon-Release passende Version der Erweiterung installiert, was Konflikte vermeidet.

Cinnamon 4.8 wechselt nach einer gewissen Zeit vom Bereitschaftsmodus in den Ruhezustand, wenn der Computer letzteren unterstützt.

Als technische Neuerung setzt der Cinnamon JavaScript Interpreter (CJS) auf Mozillas JavaScript-Engine „Mozjs78“. Dadurch soll Cinnamon schneller starten. Gleichzeitig erlaubt es anderen Distributionen den Cinnamon-Desktop leichter zu integrieren. Cinnamon 4.8 steht bereits in den Repositorys von Arch Linux zur Installation bereit. (ndi@ct.de)

OpenZFS: Linux und FreeBSD mit gemeinsamer Code-Basis

Das OpenZFS-Projekt arbeitet seit 2013 auf das Ziel hin, eine Open-Source-Alternative zu Oracles geschlossenem ZFS--Dateisystem zu entwickeln. **OpenZFS 2.0 soll den bestehenden FreeBSD-Port von ZFS und zfs-0.86 für GNU/Linux ablösen.** Die

Entwickler schaffen nun mit OpenZFS 2.0 eine geteilte Code-Basis, die auf ZFS für Linux aufbaut. Die OpenZFS-Binarys für FreeBSD und Linux können so aus dem gleichen GitHub-Repository gebaut werden. Kompatibilität mit macOS ist für ein späteres Release vorgesehen.

Darüber hinaus bietet OpenZFS 2.0 neue Funktionen. Ein fehlerhaftes Laufwerk in einem ZFS-Plattenverbund lässt sich mit dem neuen Verfahren „Sequential Resilver“ schneller wiederherstellen. Der Lese-Cache L2ARC stellt häufig genutzte Objekte bereit, die nicht mehr im Hauptspeicher vorgehalten werden. Nutzer können diesen Cache in OpenZFS 2.0 optional als persistent konfigurieren. So muss der L2ARC nicht bei jedem Neustart, Import oder Export des ZFS-Pools neu befüllt werden. Mit dem Befehl redacted zfs-send/receive können Daten beim Transfer von Datasets ausgenommen werden, um Speicherplatz zu sparen oder sensible Daten zu schützen.

Der neue Komprimierungsalgorithmus ZStandard (ZSTD) komprimiert ähnlich gut wie Gzip, aber schneller. FreeBSD-Nutzer können OpenZFS 2.0 bereits als Port installieren. Das kommende FreeBSD 13 enthält OpenZFS 2.0 als Standard. Canonical hat nach eigenen Angaben bereits in Ubuntu 20.04 LTS Funktionen aus OpenZFS 2.0 rückportiert. (ndi@ct.de)

Gnome-Projekt öffnet Türen

Softwareprojekte mussten, um Teil des Gnome-Projektes zu werden, auch dessen Infrastruktur nutzen und sich an die halbjährlichen Release-Intervalle der Desktopumgebung halten. **Mit der neuen Initiative „Gnome Circle“ möchte das Gnome–Projekt diese Barrieren für unabhängige Entwickler absenken.**

Als Anforderung gilt es weiterhin, die Gnome-Plattform-Bibliotheken zu nutzen und den Coding-Guidelines zu folgen. Im Gegenzug verspricht Gnome den Bekanntheitsgrad der Software zu steigern. Außerdem bietet die Gnome-Foundation

Reisekostenerstattungen und Zugang zu Gnome Ressourcen wie Hosting sowie Nextcloud- und GitLab-Instanzen. Entwickler können sich über die Website circle.gnome.org bewerben. (ndi@ct.de)

[/expand]

Trio: NAS-Geräte mit 16, 18 und 24 Slots

[expand title="mehr lesen…"]

Trio: NAS-Geräte mit 16, 18 und 24 Slots

Trio: NAS-Geräte mit 16, 18 und 24 Slots

Drei neue Netzwerkspeicher für die Rack-Montage und mit deftiger Ausstattung ergänzen QNAPs Angebot für den unternehmerischen Einsatz.

Das TS-h1683XU-RP, das TS-h1886XU-RP und das TS-h2483XU-RP sind QNAPs neueste Unternehmensmodelle mit vielen Slots, viel Arbeitsspeicher und Intel-Xeon-CPU. *Bild: QNAP*

NAS-Hersteller QNAP hat sein Unternehmensangebot um drei Modelle für den Einbau in 19-Zoll-Gestelle erweitert. Alle Geräte nutzen QNAPs neues NAS-Betriebssystem „QuTS hero“, das

das Dateisystem ZFS einsetzt.

Das kleinste Modell TS-h1683XU-RP besitzt 16 3,5-Zoll-Einschübe für Massenspeicher. Die Systemausstattung gleicht der des 24-Slot-Modells TS-h2483XU-RP: Ein Intel Xeon E-2236 mit sechs Kernen und 128 GByte DDR4-ECC-RAM ist bei beiden das Herzstück. Netzwerkseitig haben beide Modelle PCIe-Karten mit zwei 10-Gigabit-Kupferports sowie zwei SFP+-Slots vorinstalliert. Das 16-Slot-Modell besitzt vier PCIe-Slots, das größere fünf. Außerdem sind USB-3.2-Gen-2-Ports an Bord; vier Typ-A-Buchsen und zwei Typ-C-Buchsen.

Im TS-h1886XU-RP stecken zwölf 3,5-Zoll-, sechs 2,5-Zoll- und vier PCIe-Slots, dazu zwei USB-3.2-Gen-1-Ports sowie der Quad-Core-Xeon D-1622 mit 32 GByte DDR4-ECC-RAM. Das RAM kann man auf bis zu 128 GByte aufrüsten. QNAP installiert vorab zwei SFP+-Ports per PCIe-Karte.

Alle Modelle sind ab Werk mit redundanter Stromversorgung bestückt: zwei Netzteile mit je 550 beziehungsweise 800 Watt Spitzenleistung. Weiter gehören vier von den Erweiterungsschnittstellen unabhängige Kupfer-Gigabit-Ethernet-Ports zur Grundausstattung. Laut QNAP können die NAS per PCIe-Karte um 10-, 25- oder 40-Gigabit-Ports, Fibre-Channel-Schnittstellen, SAS-Ports für Erweiterungsgehäuse sowie M.2-Slots ergänzt werden.

Das Trio ist ab rund 3100 Euro zu haben; das TS-h1886XU-RP kostet derzeit 3130 Euro, das TS-h1683XU-RP rund 6700 Euro und das TS-h2483XU-RP 7300 Euro. (amo@ct.de)

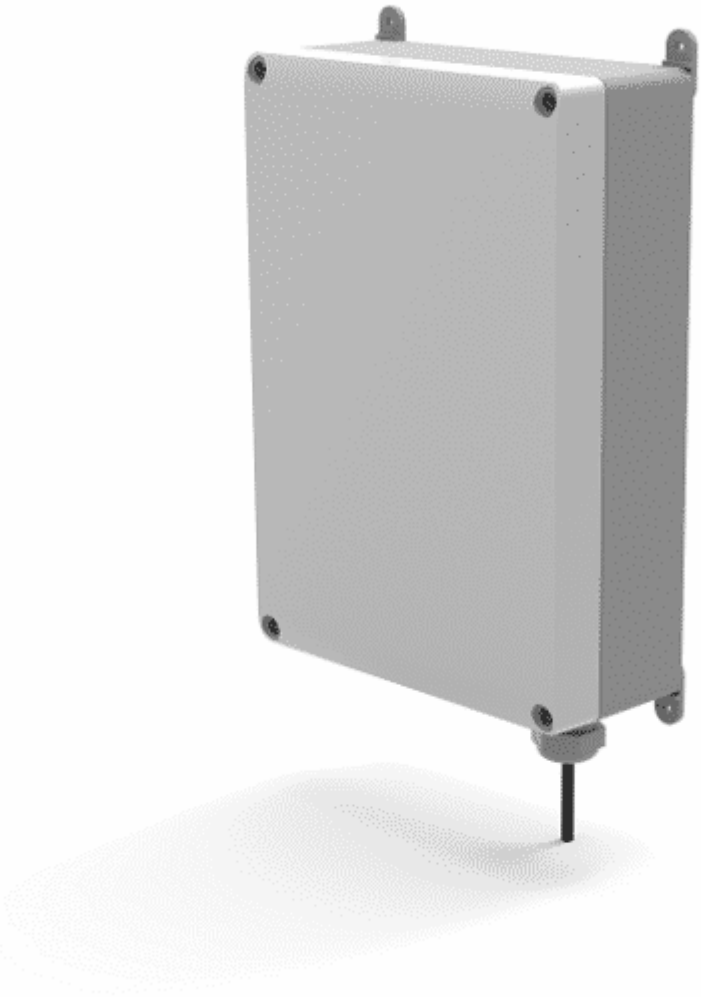
LTE-Router für Ferienhäuser und Camper

Der 4G09 AC1200 von Tenda richtet sich an Nutzer, die im Ferienhaus ohne eigenen Anschluss oder im Camper einen Internetzugang per Mobilfunk nutzen wollen. *Bild: Tenda*
Ein ordentlicher Internetzugang wird vielen auch im Urlaub oder während der Freizeit immer wichtiger. Tenda will hier mit dem 4G09 AC1200 einhaken: Der Router kommt mit einem LTE-Modem der Kategorie 6, erreicht also bis zu 300 MBit/s im Down- und 50 MBit/s im Uplink und **funkt auf allen in Europa üblichen LTE-Bändern**. Die Mobilfunkantennen sind extern per SMA ausgeführt, sodass man auch andere Varianten anschließen kann. Alternativ kann man den Internetzugang über ein externes Modem am Gigabit-Ethernet-WAN-Port aufbauen.

WLAN-seitig kommuniziert der 4G09 AC1200 als Dual-Band-Router mit zwei MIMO-Streams pro Band; auf 2,4 GHz arbeitet er nach Wi-Fi 4 (IEEE 802.11 b/g/n, max. 300 MBit/s brutto), auf 5 GHz

gemäß Wi-Fi 5 (802.11ac, 867 MBit/s). Laut Tenda kann der Router maximal 64 Geräte gleichzeitig per Funk bedienen. Über den Gigabit-Ethernet-LAN-Port lässt sich ein zusätzliches Gerät per Kabel ankoppeln oder ein Access Point für noch mehr WLAN-Clients. Die Stromversorgung läuft über ein 12-Volt-Netzteil mit Hohlstecker. Der Tenda 4G09 AC1200 ist für 99 Euro erhältlich. (amo@ct.de)

Multicell-DECT für draußen



Die DECT-Basis M900 Outdoor soll Schnurlostelefonie auf Außengeländen ermöglichen. Sie eignet sich zur Montage an Wänden, Decken und Masten. *Bild: Snom*

Der VoIP-Telefon-Hersteller Snom bietet sein Multizellen-DECT-System nun auch für draußen an: Die M900 Outdoor ist die erste **wetterfeste Basisstation** des Herstellers. Verpackt in einem

nach IK08-Norm stoßfesten und UV-resistenten Gehäuse soll sie Strahlwasser aus allen Richtungen und Temperaturen zwischen –20 und +60 °C schadlos überstehen (IP55).

Die Basis wird per Power-over-Ethernet mit Strom versorgt und entspricht auch sonst technisch der Indoor-Variante M900. Das System unterstützt bis zu 4000 Basisstationen mit maximal 16.000 DECT-Mobilteilen.

Sechs Mobilteil-Typen für unterschiedliche Anwendungen im Büro, im Krankenhaus und in rauen Umgebungen stehen zur Auswahl. Die DECT-Basis M900 Outdoor für die Versorgung von Außengeländen kostet rund 500 Euro. (amo@ct.de)

[/expand]

Kaufberatung für kompakte und günstige Netzwerkspeicher (NAS)

Kaufberatung für kompakte und günstige Netzwerkspeicher (NAS)

[expand title="mehr lesen…"]

Datenhäufchen

Kaufberatung für kompakte und günstige Netzwerkspeicher (NAS)

NAS-Fertiggeräte sind sehr flexibel nutzbar: Sie stellen nicht bloß mehrere Terabyte Daten im Netzwerk bereit, sondern lassen sich zu funktionsreichen Mini-Servern erweitern. Wir zeigen die Technik aktueller NAS ab 130 Euro und was Sie davon für Ihr Homeoffice brauchen. Von Christof Windeck

Die Abkürzung NAS steht für Network Attached Storage, also Massenspeicher mit Netzanbindung, manchmal auch lapidar Netzwerkfestplatten genannt. Die zu Preisen ab rund 100 Euro verkauften NAS-Boxen zum Selbstbestücken mit Festplatten können aber viel mehr als Daten speichern und diesen Dienst für alle Geräte im lokalen Netz bereitstellen. Denn ihre vom jeweiligen Hersteller gepflegte Firmware – gemeint ist damit fast immer ein maßgeschneidertes Betriebssystem mit Linux-Kern – hat eine Fülle eingebauter Funktionen und lässt sich mit Plug-ins komfortabel erweitern. Dann arbeiten NAS-Boxen etwa auch als Medienserver für Fotos, Videos und Musik, dienen als lokales Backup-Ziel und holen Cloud-Funktionen ins eigene Heim. So muss man vertrauliche Daten nicht mehr in die Hände mehr oder weniger vertrauenswürdiger Cloud-Dienstleister geben.

NAS-Boxen lassen sich leicht mit Festplatten bestücken, meistens braucht man kein Werkzeug dazu.
Das Angebot an NAS-Boxen ist riesig, allein von den Marktführern Synology und Qnap findet man über 100 Geräte im Handel. Wir konzentrieren uns in dieser Kaufberatung auf günstige Netzwerkspeicher mit zwei Laufwerksschächten (Drive Bays), auch 2-Bay-NAS genannt. Sie bieten ausreichend Kapazität fürs typische Homeoffice und auf Wunsch Schutz vor Festplattenausfällen. Viele sind leise und sparsam, stören also nicht bei der Arbeit (siehe Test auf S. 20). Wer mehr Platz braucht, aber keine weiteren Funktionen, findet die Technik der günstigen Geräte auch in NAS für bis zu vier Platten.

NAS oder nicht?

Ein zentraler Netzwerkspeicher lässt sich auch ohne NAS-Box einrichten, beispielsweise mit einem Raspberry Pi oder mit den NAS-Funktionen eines ohnehin vorhandenen (WLAN-)Routers. Beide Ansätze bringen aber erhebliche Einschränkungen. So eignet sich ein Raspi-NAS nur für Menschen mit ausreichendem Wissen zu Linux und Hardware. Ein Router-NAS bietet nur einen

Bruchteil der Funktionen typischer NAS-Boxen und meist viel weniger Performance. Vor allem aber muss man beim Router-NAS eine externe USB-Platte anschließen, was Bedienungsfehler begünstigt: Zieht man versehentlich das USB-Kabel ab, während noch jemand via Netzwerk auf die Platte schreibt, droht Datenverlust. Diese Ansätze taugen also nur für simple Anwendungen, beispielsweise um Kopien von andernorts sicher gespeicherten Mediendateien für den Netzwerkzugriff bereitzustellen.

Auch fast jeder PC lässt sich zum NAS umfunktionieren, etwa indem man unter Windows oder Linux Dateifreigaben einrichtet oder NAS-Software wie TrueNAS oder OpenMediaVault installiert. Mancher ältere PC schluckt aber bei Dauerbetrieb unnötig viel Strom oder rauscht laut, mit einem neu gekauften Mini-PC wiederum wird das NAS teurer als eine fertige Box. Tipps für Heimserver-Hardware liefern mehrere c’t-Artikel [1, 2, 3, 4].

Die Alternativen demonstrieren mit ihren Nachteilen die Vorzüge fertiger NAS-Boxen: Letztere schützen Festplatten mechanisch und kühlen sie leise mit einem temperaturgesteuerten Lüfter. In die NAS-Hardware der etablierten Marken fließt jahrelange Erfahrung ein, zudem pflegen die Hersteller Kompatibilitätslisten für Festplatten, was das Risiko von Pannen senkt. Man bekommt ein Komplettgerät aus der Serienfertigung inklusive Support, Dokumentation sowie mit vorhersagbarer Leistungsaufnahme. Vor allem aber erleichtern die über mehrere Generationen optimierten NAS-Firmwares die Einrichtung eines Netzwerkspeichers enorm und mindern das Risiko fataler Bedienungsfehler. Die großen Hersteller liefern auch recht zuverlässig Updates, die Sicherheitslücken schließen – allerdings gilt das nicht für jedes nachträglich installierte Plug-in.

Zur sicheren und zuverlässigen NAS-Konfiguration sind jedoch einige Grundkenntnisse nötig, vor allem sollte man sich mit den Konzepten der Nutzerverwaltung und von Zugriffsberechtigungen beschäftigen. Ausführliche Hinweise zur

sicheren NAS-Konfiguration liefern wir in einer kommenden c’t-Ausgabe.

Für den Einsatz im Homeoffice braucht ein NAS nicht mehr - Anschlüsse als Gigabit-Ethernet und USB 3.0 (rechts); ein - zweiter Ethernet-Port ist ebenso verzichtbar wie eine eSATA-- Buchse (links, rot) zur Erweiterung.

NAS-Funktionen

Jeder NAS-Hersteller pflegt sein eigenes Firmware-Ökosystem, auf das der Artikel ab Seite 26 näher eingeht. Die grundsätzlichen Funktionen gleichen sich, weil es letztlich um sogenannte File-Server geht. In der NAS-Box sitzt dazu ein sparsamer Prozessor, auf dem ein angepasstes Linux läuft. Letzteres bindet die Festplatten-, USB- sowie Netzwerkschnittstellen ein, verwaltet den Speicherplatz der

eingebauten Datenträger sowie die Rechte der Nutzer, die darauf Zugriff haben sollen.

Ein NAS arbeitet normalerweise ohne Bildschirm, Tastatur und Maus, man konfiguriert es per Browser über seine Web-Oberfläche. Dort richtet man üblicherweise sogenannte Dateifreigaben (Shares) ein, auf die anschließend bestimmte Nutzer(-gruppen) Zugriff haben – und andere nicht (Datenschutz). Jedes NAS ermöglicht Datenzugriff über das Windows-Freigabenprotokoll Server Message Block (SMB), früher auch Common Internet File System (CIFS) genannt. Windows-Rechner können eine SMB-Freigabe als Backup-Ziel nutzen, das ist mit die wichtigste Funktion fürs Homeoffice. Mit SMB kann praktisch jeder aktuelle PC umgehen, egal ob mit Windows, Linux oder Apples macOS; auch für Smartphones und Tablets mit Android oder iOS gibt es passende Apps.

Alle NAS unterstützen noch weitere Netzwerkprotokolle, etwa um Apple-Rechner mit der Backup-Funktion Time Machine anzubinden oder für WebDAV-Freigaben. Üblich sind auch Medienserver-Funktionen: Die stellen Video-, Musik- und Bilddateien so im Netz bereit, dass sie etwa Smart-TVs mit UPnP-Funktionen abspielen können.

Mit Plug-ins aus dem Online-Store des jeweiligen NAS-Anbieters lassen sich viele weitere Server-Funktionen nachrüsten, etwa smarte Backup-Dienste oder Cloud-Ersatzfunktionen wie NextCloud. Letztere sind allerdings für die meisten Nutzer nur sinnvoll, wenn sie auch außerhalb des eigenen Netzwerks funktionieren, also von unterwegs – und hier wird es kompliziert: Wer sein NAS etwa per Portweiterleitung und DynDNS-Dienst aus dem Internet erreichbar macht, reißt dabei leicht Sicherheitslücken auf. Für einen heimischen Cloud-Ersatz setzt man besser auf eine VPN-Verbindung ins Heimnetz [5, 6, 7].

RAID

Festplatten sind mittlerweile mit bis zu 18 TByte Kapazität erhältlich, was für die allermeisten Heimbüros genügt. Man kann im 2-Bay-NAS auch eine einzelne Platte betreiben: Das spart Geld und Strom und reicht oft aus, wenn man wirklich regelmäßige Backups pflegt. Weil Festplatten wie jede andere Hardware jederzeit ausfallen können, ist es aber schlauer, eine zweite Platte im NAS für einen redundanten Verbund aus zwei Laufwerken einzusetzen, ein sogenanntes RAID 1. Dabei schreibt das NAS alle Daten gleichzeitig auf beide Laufwerke; fällt eines aus, gehen keine Daten verloren. Dennoch ersetzt ein RAID kein Backup, weil es weder vor versehentlichem Löschen schützt noch vor Verschlüsselungstrojanern oder dem Ausfall der NAS-Hardware. Ein NAS ist zwar ein gutes Backup-Ziel für PCs, aber es sollte stets mindestens eine weitere Kopie auf einem anderen Speichermedium geben, das nicht ständig am Netz hängt und möglichst nicht am gleichen Ort lagert – also auf einer USB-Platte oder verschlüsselt in der Cloud [8]. Manche NAS erledigen Letzteres per Plug-in automatisch.

Geradezu leichtsinnig ist es, zwei NAS-Platten zu einem zwar schnelleren und größeren, aber auch fehleranfälligeren RAID-0-Verbund zu koppeln – dann vernichtet ein einzelner Plattendefekt *sämtliche* Daten, weil sie streifenweise auf beide Platten verteilt sind (Stripeset). Auch beim sogenannten JBOD, das den addierten Platz beider Platten zu einem einzigen logischen Volume zusammenfasst, drohen Probleme. Wird der Platz knapp, rüstet man besser eine größere Disk nach.

Alle drei großen Festplattenhersteller verkaufen Laufwerke speziell für kleine NAS; achten Sie auf ein Modell mit konventioneller Aufzeichnung, also ohne SMR-Technik wie bei - dieser WD Red.

Speed

Aktuelle 100-Euro-NAS haben Gigabit-Ethernet-(GE-)Buchsen, die bis zu 115 Megabyte Daten pro Sekunde übertragen. Eine einzelne moderne Platte schafft in ihren schnellsten Zonen noch mehr und genügt also, um große Dateien so schnell wie möglich zu kopieren. Wenn das NAS 100 MByte/s schafft, dauert der Transfer einer Backup-Datei oder eines DVD-Images mit jeweils 4 GByte Umfang weniger als 50 Sekunden. Beim Zugriff auf viele kleine Dateien sinkt die Transferrate enorm – das lässt sich aber nur bis zu einem gewissen Grad optimieren und selbst das ist teuer, weil man ein NAS mit viel RAM und SSD-Bestückung braucht. Das lohnt sich für Kleinbüros ebenso selten wie ein schnellerer Ethernet-Anschluss mit 2,5-Gigabit/s-Ethernet (2,5Base-T) oder gar 10GE: Um das

auszureizen, braucht man auch einen schnellen Switch sowie die passenden Anschlüsse am PC.

Wenn Sie Ihr NAS vor allem per WLAN nutzen, dann können Sie die volle Transferrate selbst mit Wi-Fi 6 nur dann ausschöpfen, wenn sich das Notebook nahe beim WLAN-Router befindet. Die WLAN-Transferrate sinkt mit wachsender Entfernung rasch. Anders gesagt: Wenn das Notebook meist eine schwache WLAN-Verbindung hat, brauchen Sie kein schnelles NAS. Müssen Sie sehr häufig große Dateien übertragen, dann nehmen Sie besser ein Ethernetkabel. Für Notebooks ohne Ethernet gibt es USB-GE-Adapter ab etwa 15 Euro; sie brauchen einen USB-3.0-Port, USB 2.0 ist zu lahm.

Obwohl die meisten NAS-Prozessörchen AES-Beschleuniger zum Verschlüsseln von Daten beim Schreiben auf die Platten haben, kann die Transferrate deutlich sinken. Sie müssen selbst abwägen, wie stark das stört. Wer Kundendaten auf dem NAS ablegen möchte, sollte die Verschlüsselung mindestens für eine dafür reservierte Freigabe aktivieren – zu leicht geraten Daten in falsche Hände, etwa wenn man das Gerät zur Reparatur einschickt.

Fürs Backup auf eine externe Festplatte sollte das NAS eine USB-3.0-Buchse haben (auch als USB 3.2 Gen 1 bezeichnet); das ist selbst bei günstigen Geräten inzwischen der Fall.

NAS-Festplatten

Wie viel Speicherplatz ein NAS haben soll, lässt sich nicht pauschal beantworten, weil es dabei auf Ihre individuellen Wünsche ankommt. Als Backup-Ziel fürs Homeoffice braucht man aber zunächst nicht viel mehr als die doppelte bis dreifache Kapazität der SSDs und Platten der eigenen Computer. Kaufen Sie keine extrem überdimensionierten Laufwerke, nicht bloß weil sie teuer sind, sondern auch weil sie nicht ewig leben. Sinnvoller ist es, nach drei bis vier Jahren einen Umzug auf größere und dann auch frische Platten einzuplanen. Die alten

kann man dann als zusätzliches Backup einlagern.

Zwar lassen sich grundsätzlich fast alle Platten im NAS verwenden; in die meisten passen sogar die sparsameren und leiseren 2,5-Zoll-Laufwerke für Notebooks. Es ist aber ratsam, sich an die Kompatibilitätslisten des NAS-Herstellers zu halten und davon Platten auszuwählen, die speziell für den NAS-Betrieb ausgelegt sind. Alle drei verbliebenen Festplattenhersteller Seagate, Toshiba und Western Digital (WD) bieten für kleine NAS spezielle Laufwerksfamilien an. Deren Eigenschaften (und Preise) liegen zwischen jenen für Desktop-PCs und jenen für große NAS und „Enterprise"-Server. Anders als Desktop-Typen sind die NAS-Versionen für Dauerbetrieb ausgelegt, vertragen Vibrationen durch andere Platten im gleichen System besser und arbeiten relativ sparsam. Sie drehen aber langsamer als die schnelleren Typen für größere Server, kompensieren Vibrationen nicht aktiv und sind für geringere jährliche Datentransfermengen ausgelegt. Letzteres dürfte im Homeoffice aber nicht ausschlaggebend sein.

Eine bekannte Plattenbaureihe für kleine NAS ist die WD Red, die mittlerweile WD Red Plus heißt. Allerdings verärgerte WD Mitte 2020 einige Käufer mit verwirrenden Datenangaben: So lieferte man stillschweigend WD-Red-Platten mit der Aufzeichnungstechnik Shingled Magnetic Recording (SMR), die in manchen Konstellationen zu Problemen führen kann. Besser für kleine NAS ist Conventional Magnetic Recording (CMR) – und genau diese Technik steckt in der WD Red Plus. Außerdem verkaufte WD Platten der „5400-Touren-Klasse", deren Magnetscheiben tatsächlich 7200-mal pro Minute rotieren und dafür etwas mehr Strom schlucken.

Die zur WD Red Plus vergleichbare Toshiba-Baureihe heißt schlicht „NAS Drive", Seagate empfiehlt die Serie „Ironwolf". 4-TByte-Modell der genannten Plattenfamilien kosten ab 90 Euro, 2- und 3-TByte-Modelle sind nur wenig billiger und lohnen sich deshalb nicht mehr. Bei NAS-Platten mit 6 und 8

TByte zahlt man ähnlich viel pro Terabyte wie bei den 4-TByte-Typen, ab 10 TByte wird es teurer. Platten ab 8 TByte gibt es auch mit Heliumfüllung, die bei gleicher Kapazität und Drehzahl etwas sparsamer arbeiten.

Ein NAS lässt man üblicherweise ständig laufen, die meisten schalten ihre Platten nach einiger Zeit ohne Zugriffe automatisch ab. Einfache NAS nehmen im Leerlauf mit stehenden Platten rund 5 bis 15 Watt Leistung auf. Bei einem Strompreis von 30 Cent pro Kilowattstunde summiert sich das auf 13 bis 40 Euro jährlich. Bei Zugriffen braucht der NAS-Prozessor mehr Strom und jede Platte zwischen 4 und 8 Watt, folglich stehen in diesem Beispiel zwischen 13 und 31 Watt an. Weil ein NAS im Homeoffice selten mehr als je vier Stunden an 230 Werktagen im Jahr Daten überträgt, kommen dafür bloß 4 bis 9 Euro hinzu, in der Summe ergeben sich also 17 bis 48 Euro. Würden die Platten ständig drehen, wären es stattdessen zwischen 34 und 103 Euro.

Guter Standort

Festplatten vertragen Stöße und Hitze schlecht. Auch Staub, Schmutz, Feuchtigkeit (Keller, Küche) sowie Vibrationen anderer Geräte (Drucker) können NAS-Probleme verursachen. Im Homeoffice sollte ein NAS zudem außerhalb der Reichweite von Kindern und Haustieren stehen. Wenn es im Stromnetz an Ihrem Wohnort öfters zu Störungen kommt, schützt eine unterbrechungssichere Stromversorgung (USV) vor Hardware-Defekten und daraus resultierendem Datenverlust [9].

Denken Sie an regelmäßige Backups und den Datenschutz, vor allem wenn Sie Daten von Kunden und Kollegen verarbeiten (DSGVO). Dabei hilft Verschlüsselung. *(ciw@ct.de)*

1. Literatur
2. Carsten Spille, Speicher-Quader, Flexibler Heimserver mit ECC-RAM, c’t 18/2020, S. 112
3. Carsten Spille, FAQ: Heimserver-Variationen, c’t 23/2020, S. 178

4. Christof Windeck, Netz-Zentrale, Tipps zur Server-Auswahl für kleine Netze, c’t 18/2020, S. 106
5. Ernst Ahlers, Wandelbarer Speicher, TerraMaster F2-221: NAS oder Mikroserver nach Wunsch, c’t 8/2019, S. 80
6. Urs Mansmann, Tunnel durchs Internet, Mobile Geräte mit VPN sicher ins Netz bringen,c’t 3/2016, S. 126
7. Peter Siering, Schutz ausrollen, Den eigenen VPN-Server mit WireGuard bauen, c’t 15/2019, S. 166
8. Andrijan Möcker, Schwesterkiste, Fritzbox 4040 mit OpenWrt betreiben, c’t 10/2019, S. 28
9. Holger Bleich, Auf Eis gelegt, Daten langfristig in der Cloud sichern , c’t 22/2020, S. 72
10. Rudolf Opitz, Blackout-Versicherung, Günstige USVs für den Büro-PC , c’t 3/2018, S. 110

Datenheim

Heimcloud-Grundlage: Vier NAS-Leergehäuse zum Selbstbestücken

Ein Netzwerkspeicher für Familie, WG oder Kleinfirma muss kein Vermögen kosten: Ein Terabyte RAID-gesicherten Speicherplatz kann man schon für weniger als 300 Euro ins LAN stellen, wovon etwa die Hälfte aufs NAS-Leergehäuse entfällt. Vier Exemplare dieser Klasse traten im c’t-Labor gegen ein doppelt so teures Modell an. Von Ernst Ahlers

Der Datenspeicher im LAN ist nicht nur für zentral gelagerte Dateien und PC-Backups nützlich: Er kann auch als Medienserver Musik und Videos ans Smart-TV liefern. Will man sich von Google lösen, dient das NAS mit Software-Erweiterungen (Apps aus den Hersteller-Repositorys) auf Wunsch auch als aus dem Internet erreichbarer Cloud-Server für Kontakte, Termine und

Notizen. Die zentrale Sicherung von Fotos und Videos von Familien-Smartphones per App fällt als Dreingabe ab.

Eine Übersicht der Funktionen gängiger NAS-Betriebssysteme finden Sie im folgenden Beitrag ab Seite 26, hier geht es um die Hardware-Ausstattung und Leistungsfähigkeit günstiger NAS-Leergehäuse mit zwei Massenspeicherplätzen zum Selbsteinbauen von Festplatten oder SSDs. Davon haben wir vier ins Labor geholt: Netgear ReadyNAS 212, QNAP TS-230, Synology DS220j und Western Digital (WD) My Cloud EX2 Ultra. Das TerraMaster F2-210 liegt ebenfalls in der 150-Euro-Preisklasse, seinen Kurztest finden Sie in c’t 20/2019 auf Seite 80. Ein Asustor AS5202T für rund 300 Euro gibt den Ausblick, welchen Mehrwert eine etwas größere Investition erkauft.

Die Leergehäuse bestückten wir jeweils mit zwei NAS-Festplatten Seagate ST4000VN008 zu je 4 Terabyte. Bei dieser Kapazität bekommt man zurzeit den meisten Speicherplatz fürs Geld. In der RAID-1-Zusammenstellung (Spiegelung, sichert gegen Festplattendefekt) ergibt sich eine Nettokapazität von ebenfalls 4 Terabyte. Die Geräte unterstützen zwar auch Konfigurationen für die doppelte Speichergröße, aber die sollten Sie meiden, siehe Abschnitt „RAID“ auf Seite 18.

Beim Einrichten per Browser folgten wir den Vorgaben der Software-Assistenten, so es welche gab, und maximierten gegebenenfalls den bereitgestellten Speicherplatz. Manche Geräte bieten nämlich die Möglichkeit, Kapazität für Snapshots zu reservieren. Die schützen zwar vor Datenverlust durch Fehlbedienungen oder Verschlüsselungstrojanerbefall, indem man ältere Dateiversionen „zurückholen“ kann. Aber dabei geht ein großer Teil der NAS-Kapazität für Schattenkopien drauf, was nicht jeder investieren will. Ein regelmäßig aufgefrischtes Offline-Backup auf ans NAS gesteckte USB-Massenspeicher erfüllt den Zweck meist genauso gut.

Praktische Aspekte

USB-Buchsen an der Frontseite gibt es in diesem Testfeld nur bei Asustor, Netgear und QNAP. Wer die eben erwähnte Backup-Platte anschließen will, muss bei Synology und WD das Gerät im Betrieb eventuell drehen oder verschieben. Beides birgt eine Headcrash-Gefahr für die laufenden Festplatten. Eine halbmeterlange USB-3.0-Verlängerung für ein paar Euro, deren Buchse man nach vorn legt, ermöglicht einen risikoarmen Anschluss des Backup-Mediums.

Bei den günstigen Geräten ist das RAM nicht erweiterbar. Zumindest beim TS-230 wäre das aber sinnvoll, falls man etwa per Docker-Add-on Serverfunktionen nachrüsten möchte, die es im Hersteller-Repository nicht gibt. Wer mit dieser Absicht zum AS5202T greift, der sollte gleich dessen RAM aufs Maximum von 8 GByte erweitern. Dann können auch mehrere schwergewichtige Container gleichzeitig laufen, die einige Hundert MByte belegen, wie etwa Nextcloud und seine Office-Erweiterung Collabora.

Angeschlossen

Viele einfache NAS haben nur einen Gigabit-Ethernet-Port, Netgears ReadyNAS 212 hingegen zwei. Per Link Aggregation lässt sich damit der Durchsatz steigern, aber nur, wenn zwei PCs gleichzeitig aufs NAS zugreifen und der LAN-Switch die Funktion unterstützt. Das dürfte in den meisten kleinen Netzen entbehrlich sein.

Das AS5202T hat ebenfalls zwei LAN-Buchsen, die als Multigigabit-Ethernet-Ports (NBase-T) sogar bis zu 2,5 GBit/s entsprechend 2500 MBit/s transportieren. Von der Verdoppelung profitiert aber nur, wer auch das restliche Netzwerk aufrüstet, also einen NBase-T-fähigen Switch und passende LAN-Adapter für die PCs anschafft [1].

Für den Heimeinsatz interessanter dürfte der HDMI-Videoausgang

des AS5202T sein: Durch Installieren des Software-Add-ons Asustor Portal wird das NAS zum Medienabspieler, der Videos bis zur 4K-Auflösung an Bildschirme ausliefert. Mit weiteren Plug-ins kann man auch Video-on-Demand-Filme von Amazon Prime Video, Disney+ und Netflix abspielen. Zum bequemen Bedienen der dann auf dem TV-Gerät angezeigten grafischen Oberfläche braucht es noch eine USB-Tastatur/Maus-Kombi. Solch eine Funktion bietet QNAP bei seinen teureren NAS-Modellen ebenfalls an.

Das zum Vergleich mitgetestete, teurere Asustor AS5202T hat den anderen Geräten einen HDMI-Videoausgang voraus. Seine beiden LAN-Ports arbeiten mit maximal 2,5 GBit/s (NBase-T) statt 1 GBit/s.

Den Funktionsumfang werteten wir ausgehend von „Zufriedenstellend“, für das ein Grundstock essenzieller Funktionen gegeben sein musste (Windows-Dateifreigaben per SMB/CIFS, SMART-Überwachung der Massenspeichergesundheit, Admin-Alarmierung bei Problemen mindestens per E-Mail, Unterstützung externer USB-Massenspeicher für Backups et cetera). Die Note verbessert sich, wenn die Geräte auch die flexiblere Verschlüsselung auf Freigabeebene bieten oder der Hersteller ein Synchronisationstool auch für andere Betriebssysteme als Windows bereitstellt.

Steht die besonders nützliche Docker-Erweiterung zur Verfügung, die das NAS zum universellen Server macht, klettert die Note nochmals. Einen Abzug hätte es gegeben, wenn die Verschlüsselung der gespeicherten Daten weder auf Volume-Ebene noch per einzelner Freigabe möglich ist. Diese Verschlüsselung schützt vertrauliche Daten, wenn das NAS oder seine Festplatten etwa bei einem Einbruch abhandenkommen.

Ausgemessen

Was die NAS-Gehäuse mit der von uns gewählten Festplattenbestückung zu leisten vermögen, haben wir mit dem c’t-NAS-Benchmark ermittelt [2]. Typischerweise stellt sich der höchste Durchsatz beim Zugriff auf große Dateien ein, im Test 10 × 400 MByte. Den Transfer vieler kleiner Dateien – 1000 × 256 KByte – bremsen die Latenzen in den PC- und NAS-Betriebssystemen.

Die Probanden konnten die maximale Datenrate ihrer Gigabit-Ethernet-Ports (1000 MBit/s entsprechend rund 115 MByte/s auf Anwendungsebene) bei großen Dateien erfreulicherweise weitgehend ausschöpfen (siehe Balkendiagramm auf Seite 21). Mit seinen extraschnellen Ports vermochte das teurere AS5202T sich indes ein gutes Stück nach oben abzusetzen. Welches der beim Einrichten wählbaren Dateisysteme (EXT4 oder BTRFS) man nimmt, machte dabei keinen spürbaren Unterschied. Mal hatte das eine, mal das andere die Nase vorn.

Für „sehr guten“ Durchsatz musste ein Prüfling mindestens gerundete 100 MByte/s beim Schreiben und Lesen großer Dateien schaffen. Eine gute Note resultierte, wenn er die Grenze bei einer Datenrichtung knackte. Für ein „Zufriedenstellend“ mussten es mindestens 80 MByte/s in eine Richtung sein. Eine schlechte Note hätte es gegeben, wenn dieser Wert in beiden Richtungen unterschritten worden wäre. Ver- oder entschlüsselt der NAS-Prozessor die Daten, dann sinkt dabei die Datenrate mehr oder weniger stark und damit bei manchen Geräten auch die Note.

SMB-Durchsatz eines Clients über Gigabit-Ethernet

Dateigruppe	1000 × 256 KByte		100 × 2 MByte		10 × 400 MByte	
	schreiben	lesen	schreiben	lesen	schreiben	lesen
	besser ▶	besser ▶	besser ▶	besser ▶	besser ▶	besser ▶
Unverschlüsselt						
Asustor AS5202T (EXT4)	22	12	84	37	153	169
Asustor AS5202T (BTRFS)	19	11	77	44	163	180
Netgear ReadyNAS 212	5	12	65	35	68	109
QNAP TS-230	4	6	45	34	106	107
Synology DS220j	5	10	45	36	106	111
WD My Cloud EX2 Ultra	6	9	64	30	84	95
Verschlüsselt						
Asustor AS5202T (EXT4)	15	14	55	33	131	165
Asustor AS5202T (BTRFS)	16	12	61	40	96	181
Netgear ReadyNAS 212	5	7	39	31	61	76
QNAP TS-230	4	6	44	32	99	105
Synology DS220j	5	10	46	36	108	109
WD My Cloud EX2 Ultra	10	10	47	30	75	82

alle Werte in MByte/s, gemessen mit c't-NAS-Bench aus/auf RAM-Disk, Windows-10-PC mit Core i3-6300 und 10GE-NIC Asus XG-C100C

Flotterer Massenspeicher

SATA-SSDs statt Festplatten im NAS können bei schneller LAN-Anbindung und sequenziellem Zugriff auf große Dateien sehr hohe Durchsätze bis in die Region von 1000 MByte/s ermöglichen [3]. In günstigen Netzwerkspeichern bringen sie jedoch selten einen Vorteil, was wir stichpunktartig bei den Geräten von QNAP und Synology überprüften. Einzig das Lesen mittelgroßer Dateien (100 × 2 MByte) lief mit rund 50 Prozent Gewinn so viel schneller, dass man es in der Praxis auch spürt.

Bei großen Dateien konnten die Probanden ihre Gigabit-Ports schon mit Festplatten voll auslasten. Das AS5202T hingegen vermochte mit seinen 2,5 GBit/s flotten Multigig-Ports und SSD-Einsatz auch große Dateien schneller anzunehmen und

auszuliefern: Beim Lesen kletterte der Durchsatz von rund 170 MByte/s auf etwas über 270 MByte/s. Beim Schreiben legte das Asustor-NAS mit SSDs aber nur beim Zugriff auf eine unverschlüsselte Freigabe merklich zu (202 statt 153 MByte/s).

Zufällig schneller

Geht es um zufällige Schreib- und Lesezugriffe über eine Windows-Freigabe auf eine auf dem NAS liegende Datei, beispielsweise eine von mehreren Nutzern verwendete Datenbank, dann kann eine SSD-Bestückung aber auch bei den günstigen Geräten einen deutlichen Schub bringen. Das überprüften wir mit dem Diskspd-Benchmark von Microsoft (ct.de/yd2e).

Bei 4 KByte Blockgröße und einem Schreib-/Lese-Mix von 20 zu 80 Prozent leistete das TS-230 mit Festplattenbestückung 140 Schreib- und 570 Lesevorgänge pro Sekunde (IOPS). Mit SSDs verzwölffachte sich das auf 1730 und 6920 IOPS. Beim DS220j mit seinem viel kleineren RAM (0,5 statt 2 GByte) fiel die SSD-Steigerung mit dem Faktor 44 noch weit höher aus: von 30/120 auf 1330/5300 IOPS.

Diese Werte darf man aber nur als Tendenz ansehen. Obwohl wir die mit zufälligen Daten gefüllte Testdatei mit 12 GByte so groß wählten, dass sie nur zu einem kleinen Teil in den RAM-Puffer der NAS passte, ergaben wiederholte Messungen bei einzelnen Prüflingen stark streuende Ergebnisse. Weil die Resultate so nicht zum Vergleich taugen, haben wir die IOPS-Leistung aus der Ergebnistabelle weggelassen.

Brummkreisel

Wer sich einen Netzwerkspeicher ins Wohnzimmer, Homeoffice oder Büro stellt, kriegt das Laufgeräusch der installierten Festplatten zu Gehör. Dazu kommt bei den meisten NAS das leichte Surren eines langsam drehenden Lüfters, doch unsere Prüflinge blieben dabei so leise, dass es bei einem halben Meter Messabstand für eine gute (0,5 bis unter 1,0 Sone) oder

sehr gute Geräuschnote reichte (unter 0,5 Sone). Wenn das Gerät die Festplatten schlafen legt, weil längere Zeit keine Zugriffe geschehen (einstellbarer Idle Timeout), dann wird es leiser, meistens deutlich.

Beim Asustor AS5202T müssen wir diesen Idle-Wert schuldig bleiben, weil die Platten im Test nie längere Zeit aus blieben, sondern auch ohne äußere Zugriffe nach einigen Sekunden bis einer halben Minute von selbst wieder anliefen. Außerdem traten im Idle-Zustand ebenfalls ohne äußeren Grund zeitlich unregelmäßig Kopfbewegungen mit einhergehendem Seek-Geräusch (leises Klackern) auf, sodass wir dort nur einen Bereich angeben können.

Fazit

Grobe Patzer leistete sich keiner der getesteten Prüflinge. Das preisgünstigste NAS-Gehäuse im Test, WDs My Cloud EX2 Ultra, erfüllt grundlegende Ansprüche. Wegen seines sporadisch sirrenden und damit akustisch auffälligen Lüfters stellt man das Gerät aber lieber abseits von Wohn- und Arbeitsbereichen auf.

Das ReadyNAS 212 von Netgear ist schon etwas besser ausgestattet und hinterließ einen soliden Eindruck, ist aber vergleichsweise teuer und dürfte wegen seiner überdurchschnittlichen Idle-Leistungsaufnahme höhere Stromkosten zeitigen als der Rest.

Um die NAS-Krone streiten wie üblich QNAP und Synology. In Sachen Performance und Energieeffizienz schenken sich TS-230 und DS220j nichts. Doch das QNAP-Modell lockt mit viermal so großem Hauptspeicher und Docker-Unterstützung; es ist auch ein Quäntchen leiser, aber nicht mal teurer als der Synology-Konkurrent.

Was mit mehr Kapitaleinsatz möglich ist, demonstriert das AS5202T: höherer Durchsatz dank Multigigabit-Ethernet,

Mediaplayer-Option per HDMI-Ausgang, RAM-Aufrüstbarkeit für große Docker-Container. Wenn Asustor noch an den Details feilt, könnte das Gerät Ihr nächstes NAS werden. *(ea@ct.de)*

Asustor AS5202T

Das AS5202T spielt preislich in einer anderen Liga als das restliche Testfeld. Das merkt man an der Hardware-Ausstattung: Ein x86-Prozessor nebst Docker-Unterstützung bahnt den Weg zu einer riesigen Auswahl von Funktionserweiterungen abseits des Herstellerangebots. Das RAM ist passend dazu vergrößerbar, die NBase-T-LAN-Ports liefern mehr als die doppelte Datenrate und per HDMI-Ausgang taugt das Gerät auch als Filmabspieler.

Auch bei der Software kommt Asustor immer näher an die NAS-Giganten QNAP und Synology heran, doch bei Details hakelt es: Im Test legten sich die Festplatten nicht zuverlässig schlafen. Lässt man sie der unmittelbaren steten Verfügbarkeit halber ohnehin durchlaufen, ist das aber einerlei.

- schnelle LAN-Ports
- RAM erweiterbar
- Idle-Timeout unzuverlässig

Netgear ReadyNAS 212

Netgear ist mit der ReadyNAS-Serie schon lange im Markt. Das merkt man am vergleichsweise großen Add-on-Angebot im Hersteller-Repository (apps.readynas.com). In Sachen Energiebedarf ist das RN212 jedoch nicht auf aktuellem Stand, bringt aber immerhin zwei LAN-Ports für Durchsatzsteigerung per Link Aggregation mit, worauf das restliche LAN eingerichtet sein muss.

Vom Aufbau her wirkt das ReadyNAS mit seinem Blechgehäuse und den stabilen Festplatten-Trays solider als die anderen, in Kunststoff gekleideten Geräte. Bei der Performance hält es gut mit, kommt aber nicht an das Niveau des Spitzenfeldes heran.

- breites Add-on-Angebot
- guter Durchsatz
- energiehungrig

QNAP TS-230

Mehr kann man für den vergleichsweise kleinen Preis kaum verlangen: Das TS-230 ist leise, energieeffizient, performant

und per Add-ons um viele Funktionen flexibel erweiterbar, mit Docker auch herstellerunabhängig. Wäre das RAM noch erweiterbar und würde das Gerät Verschlüsselung auf Freigabeebene anbieten, hätten NAS-Bastler eine perfekte Grundlage.

Wie beim Konkurrenten rechts gibt es ein breites Software-Angebot für Clients (PCs, Mobilgeräte), doch wer für Zugriffe aus dem Internet nicht auf QNAPs Dyndns-Dienst setzen will, schaut in die Röhre. Immerhin ist ab Werk schon ein DLNA-Server fürs Streaming lokal gespeicherter Medien an Bord.

- sehr leise und performant
- breites Add-on-Angebot
- Verschlüsselung nur pro Volume

Synology DS220j

Synology hat bei seinem DS220j an jeder Ecke aufs Sparen geachtet: Die Festplatten muss man – etwas umständlich, aber nur selten nötig – in eine Halterung einbauen und der Hauptspeicher ist mit 0,5 GByte (512 MByte) der kleinste im Test. Doch das hinderte das NAS nicht daran, eine sehr gute Leistung auch beim Zugriff auf verschlüsselte Freigaben

abzuliefern.

Der Lüfter war deutlicher wahrnehmbar als beim Konkurrenten links (0,4 Sone bei ruhenden Platten), dürfte so aber die Hardware kühler halten, was die Zuverlässigkeit fördert. Softwareseitig bleiben keine Wünsche offen, es gibt Add-ons zum Nachrüsten vieler Funktionen und Software für zahlreiche Client-Betriebssysteme.

- flott trotz kleinem RAM
- große Software-Palette
- keine Docker-Unterstützung

WD My Cloud EX2 Ultra

Western Digital (WD) verkauft seine My-Cloud-NAS normalerweise mit Festplatten aus eigener Herstellung. Die als Leergehäuse beschaffte My Cloud EX2 Ultra lief im Test problemlos mit Seagate-Platten und erfüllt die grundlegenden Anforderungen an ein NAS, doch in Sachen Erweiterbarkeit sieht es mager aus.

Das Einrichten der Online-Anbindung hat WD dafür besonders einsteigertauglich gestaltet: Wenn sich das Gerät nicht selbst per UPnP-Automatik eine Router-Freigabe einrichten kann, erreicht man es nach Registrierung alternativ über einen Herstellerserver, der als Proxy agiert. Den innen versteckten, manchmal anlaufenden Lüfter wünschen wir uns etwas leiser.

- einfaches Setup
- wenig erweiterbar
- sporadisch sirrender Lüfter

NAS-Benchmark-Tools: ct.de/yd2e

1. Literatur
2. Andrijan Möcker, Bitautobahn, Hardware fürs Multigigabit-LAN, c’t 16/2020, S. 56
3. Ernst Ahlers, FAQ NAS/Netzwerkspeicher, c’t 21/2019, S. 170
4. Ernst Ahlers, Presto-NAS, Schnelle Netzwerkspeicher: Vier NAS für 10-Gigabit-Ethernet, c’t 25/2020, S. 124

2-Bay-NAS-Leergehäuse – Technische Daten und Testergebnisse					
Modell	**AS5202T**	**ReadyNAS 212**	**TS-230**	**DS220j**	**My Cloud EX2 Ultra**
Hersteller/Marke	Asustor, www.asustor.com	Netgear, www.netgear.de	QNAP, www.qnap.com	Synology, www.synology.com	WD, shop.westerndigital.com
getestete Firmware / Linux-Kernel	3.5.3.RBH1 / 4.14.x	6.10.3 / 4.4.190	4.5.1.1495 / 4.2.8	DSM 6.2.3-25426 Up. 2 / 4.4.59+	5.06.115 / 4.14.22
Hardware					
Prozessor / Takt (Burst)	Celeron J4005 / 2 × 2,0 GHz (max. 2,7 GHz)	ARM Cortex-A15 / 4 × 1,4 GHz (k. A.)	ARM Cortex-A53 / 4 × 1,4 GHz (k. A.)	ARM Cortex-A53 / 4 × 1,4 GHz (k. A.)	ARM v7 / 2 × 1,3 GHz (k. A.)
RAM / erweiterbar bis auf	2 GByte / 8 GByte	2 GByte / –	2 GByte / –	0,5 GByte / –	1 GByte / –
Massenspeicher-Montage / werkzeuglos	Tray / ✓	Tray / ✓	Tray / ✓	Halterung / –	Schacht / ✓
Gigabit-LAN-Ports / Multigig-Ports (max. Datenrate)	– / 2 (2,5 GBit/s)	2 / –	1 / –	1 / –	1 / –
USB-Ports (Typ A)	3 × 5 GBit/s	3 × 5 GBit/s	2 × 5 GBit/s, 1 × 480 MBit/s	2 × 5 GBit/s	2 × 5 GBit/s
weitere Anschlüsse	HDMI 2.0a, IR-Empfänger für Fernbedienung	eSATA	–	–	–
Bedienelemente	Ein, Copy, Reset	Ein, Backup, Reset	Ein, Copy, Reset	Ein, Reset	Reset
Anzeigen	9 Leuchten	9 Leuchten	7 Leuchten	5 Leuchten	5 Leuchten
Maße (B × H × T)	17 cm × 11 cm × 22 cm	10 cm × 14,5 cm × 23 cm	9 cm × 19 cm × 15,5 cm	10 cm × 16,5 cm × 22,5 cm	10 cm × 17 cm × 15,5 cm
Sharing-Funktionen					
max. SMB-Version / Samba-Server-Version	3.11 / 4.4.3 (Mai 2016)	3.11 / 4.8.0 (März 2018)	3.11 / 4.10.18 (Sep. 2020)	3.11 / 4.4.16 (Sep. 2017)	3.11 / 4.9.5 (März 2019)
Webdav(s) / FTP(S) / NFS	✓ / ✓ / ✓	✓ / ✓ / ✓	✓ / ✓ / ✓	(✓)[1] / ✓ / ✓	– / (nur FTP) / ✓
AppleShare (AFP) / TimeMachine	✓ / ✓	✓ / ✓	✓ / ✓	✓ / ✓	✓ / ✓
Rsync / abschaltbar / iSCSI-Target	✓ / ✓ / ✓	✓ / ✓ / –	(✓)[1] / ✓ / ✓	✓ / ✓ / ✓	– / – / ✓
Printserver / Protokoll	✓ / IPP, Remote-USB	– / –	✓ / IPP	✓ / IPP, LPR, Apple Wireless Printing	– / –
Medienserver	(DLNA, iTunes)[1]	DLNA, iTunes	DLNA, weitere[1]	(DLNA, iTunes)[1]	(DLNA, iTunes)[1]
weitere vorinstallierte Server-Dienste	Web, TFTP, SFTP, Reverse Proxy	–	Web, LDAP, SQL, Syslog, Radius, TFTP	Reverse Proxy, TFTP	–
Nutzer-Auth. gegen Active Directory / LDAP	✓ / ✓	✓ / –	✓ / ✓	✓ / ✓	✓ / –
Active-Directory-Server (Domaincontroller)	–	–	✓	–	–
Wartung und Logging					
SSH / Root-Shell möglich	✓ / ✓	✓ / ✓	✓ / ✓	✓ / ✓	✓ / ✓
SNMP-Versionen / Traps	1,2c,3 / ✓	1,2c,3 / ✓	1,2c,3 / ✓	1,2c,3 / –	1,2c,3 / ✓
Logging: Syslog-Client / Alarme	✓ / E-Mail, SMS, App-Push	– / E-Mail	✓ / E-Mail, SMS, IM, App-Push	(✓) [1] / E-Mail, SMS, App-Push	– / E-Mail, SMS

2-Bay-NAS-Leergehäuse – Technische Daten und Testergebnisse					
Modell	**AS5202T**	**ReadyNAS 212**	**TS-230**	**DS220j**	**My Cloud EX2 Ultra**
NTP-Client / beliebige Quelle / Sommerzeit / Server	✓ / ✓ / ✓ / –	✓ / ✓ / ✓ / –	✓ / ✓ / ✓ / ✓	✓ / ✓ / ✓ / ✓	✓ / ✓ / ✓ / –
Port-Forwards per UPnP / Dyndns (Anzahl Dienste)	✓ / ✓ (11)	– / –	✓ / –	✓ / ✓ (19)	✓ / ✓ (2)
herstellereigener Dyndns-Dienst / mit IPv6-Auflösung	✓ / –	– / –	✓ / ✓	✓ / ✓	– / –
Massenspeicher					
Idle-Timeout / SMART-Wächter / Test per Zeitplan	✓ (5–60 min) / ✓ / ✓	✓ (5–45 min) / ✓ / ✓	✓ (5–60 min) / ✓ / ✓	✓ (10–300 min) / ✓ / ✓	✓ (5–30 min) / ✓ / –
internes Dateisystem vorgeschlagen / alternativ	EXT4 / BTRFS	BTRFS / –	EXT4 / –	EXT4 / –	EXT4 / –
externe Dateisysteme (USB-Massenspeicher)	EXT4, EXT3, NTFS, FAT32, HFS+	EXT4, EXT3, NTFS, FAT32, HFS+	EXT4, EXT3, NTFS, FAT32, HFS+, exFAT[2]	EXT4, EXT3, NTFS, FAT32, HFS+, exFAT[2]	EXT4, EXT3, NTFS, FAT32, HFS+
Verschlüsselung pro Freigabe / Volume	✓ / –	– / ✓	– / ✓	✓ / –	– / ✓
Extras					
Betrieb nach Zeitplan / Wake on LAN	✓ / ✓	✓ / ✓	✓ / ✓	✓ / ✓	✓ / ✓
Fernzugriff per App für Android / iOS	✓ / ✓	✓ / ✓	✓ / ✓	✓ / ✓	✓ / ✓
Sync-Tool für Clients / Betriebssysteme	Asustor EZ Sync / Windows	ReadyCloud / Windows, macOS, Android, iOS	QSync / Windows, macOS, Linux, Android, iOS	Synology Drive / Windows, macOS, Linux, Android, iOS	– / –
Sicherung/Mirroring übers Netz mit	rsync, FTP, Cloud[1]	rsync, Cloud[1]	QSync-Add-on[1]	rsync, Synology Drive ShareSync[1], Cloud[1]	rsync, Cloud[1]
USV-Kopplung per USB / SNMP / NUT / NUT-Server	✓ / ✓ / – / ✓	✓ / ✓ / ✓ / ✓	✓ / ✓ / ✓ / ✓	✓ / ✓ / ✓ / ✓	✓ / – / ✓ / ✓
Container/VM-Unterstützung	✓ (Docker/LXC, VirtualBox)[1]	–	✓ (Docker/LXC)[1]	–	–
Messwerte (mit 2 × ST4000VN008, RAID 1, Schreiben / Lesen)					
SMB-Durchsatz große Dateien **unverschlüsselt**	153 / 169 MByte/s	68 / 109 MByte/s	106 / 107 MByte/s	106 / 111 MByte/s	84 / 95 MByte/s
verschlüsselt	131 / 165 MByte/s	61 / 76 MByte/s	99 / 105 MByte/s	108 / 109 MByte/s	75 / 82 MByte/s
Geräuschentwicklung Idle / Platten aus	0,3–0,5 Sone / k. A.[3]	0,5 / < 0,1 Sone	0,4 / < 0,1 Sone	0,6 / 0,4 Sone	0,4 / < 0,1 Sone
Leistungsaufnahme Idle / Platten aus	ca. 14 Watt (30 VA) / k. A.[3]	18,1 Watt (33 VA) / 10,1 Watt (22 VA)	11,6 Watt (26 VA) / 4,2 Watt (14 VA)	12,7 Watt (28 VA) / 3,7 Watt (11 VA)	12,3 Watt (27 VA) / 5,2 Watt (14 VA)
Bewertung					
Funktionsumfang					

Durchsatz große Dateien (verschlüsselt)

(

)	

(

)	

(

)	

(

)	

(

)					
Geräusch					
Energieeffizienz					
Preis (ohne Platten)	299 €	193 €	157 €	160 €	138 €

sehr gut gut zufriedenstellend schlecht

sehr schlecht ✓ vorhanden – nicht vorhanden k. A. keine Angabe
[1] nachrüstbar aus Hersteller-App-Store [2] exFAT per gebührenpflichtiger Lizenz [3] Im Idle immer wieder Seek-Geräusche, Platten gingen im Test nie lang genug schlafen.

Speicherverwalter

NAS-Betriebssysteme beleuchtet

Die Rolle des NAS als reiner Netzwerkspeicher ist überholt: Die Geräte können viel mehr und haben oft riesige App-Angebote. Um die vielen Funktionen jedoch gut und sicher konfigurieren zu können, braucht es eine durchdachte Firmware.

Wir vergleichen die Konzepte von fünf NAS-Herstellern. Von Andrijan Möcker

Brauchbare Netzwerkspeicher fürs Homeoffice oder als Zweitgerät zur Sicherung bekommt man ohne Platten schon ab 100 Euro – nach oben sind die Preise offen. Große Unterschiede zwischen verschiedenen NAS-Boxen gibt es aber nicht nur bei der Hardware, sondern vor allem bei der Firmware. Bevor Sie also das nächstbeste NAS-Schnäppchen kaufen, sollten Sie prüfen, was sein Betriebssystem bietet und ob Ihnen das genügt. Denn jeder Hersteller kocht sein eigenes Süppchen und einfach wechseln kann man das Betriebssystem nicht.

Bei den NAS-Firmwares gibt es große Unterschiede bei Nutzerfreundlichkeit und Funktionsumfang, aber beispielsweise auch bei der Struktur der Weboberflächen, die in der Geräteklasse als primäre Konfigurationswerkzeuge dienen, bei der Dokumentation sowie bei eingebauten Hilfen wie Assistenten und Kurzbeschreibungen. Wir haben die Systeme der ab Seite 20 getesteten Günstig-NAS unter die Lupe genommen und vor allem auf die Grundeinrichtung, die Bedienbarkeit und die Backup-Funktionen als wichtigste Merkmale geachtet. Auch einen kleinen Blick in die App-Auswahl haben wir geworfen. Einen Test der Videoüberwachungsfunktionen von QNAP und Synology lesen Sie in [1].

Grundsätzliches

Derzeit bietet kein NAS-Hersteller einen Schritt-für-Schritt-Assistenten im Betriebssystem an, der am Ende nicht doch eine wichtige Kleinigkeit bei der Konfiguration auslässt. Die grundlegende Einrichtung deckt meist die Passwortänderung des Administrator-Accounts, die Einstellung der Zeitzone und die Konfiguration der Speicher ab. Darüber hinaus sind es eher separate Assistenten, die dem Nutzer bei den Einstellungen der spezifischen Menüpunkte helfen – beispielsweise bei Backups. Das setzt aber voraus, dass der Nutzer selbst auf die Idee kommt, dass bestimmte Dinge für ihn wichtig sind, etwa

Datensicherheit und Verbindungsverschlüsselung im Webinterface.

Abhängig von Ihrem persönlichen Vorwissen und Ihren Konfigurationswünschen müssen Sie somit einige Zeit in die Einrichtung des NAS investieren. Als ersten Schritt zum Kennenlernen ist es hilfreich, sich neugierig durch die Menüs zu klicken, um ein Gefühl für die Struktur zu bekommen.

Wer nicht die Zeit dafür findet, sollte vorerst bei USB-Speichermedien bleiben. Die Netzwerkanbindung des NAS ist gleichzeitig Vor- und Nachteil: Wer hier Konfigurationsfehler macht und das System nach außen, also zum offenen Internet, freigibt, legt unter Umständen sein komplettes Privatleben offen und riskiert große finanzielle Schäden.

Asustor holt den Benutzer direkt zu Beginn mit einer Systemeinführung ab. Allerdings hapert es im gesamten System bei den Übersetzungen, die entweder holprig sind oder ganz fehlen.

Sicherheit

Daten, die man nicht in der Öffentlichkeit oder bei Dritten

wissen möchte, sollte man auch auf dem NAS schützen. Dabei ist nicht nur ein Angriff über die Internetverbindung ein realistisches Szenario, sondern auch ein Einbruchdiebstahl. Lösen kann man letzteres mit einer Festplatten- oder Ordnerverschlüsselung. Alle NAS-Betriebssysteme im Test (siehe S. 20) beherrschen mindestens eine Variante. Schlagen Sie unbedingt das Angebot aus, das Verschlüsselungspasswort auf dem NAS selbst zu speichern. Denn dann bekommt ein Dieb mit hoher Wahrscheinlichkeit doch Zugriff auf vermeintlich -geschützte Daten. Es ist zwar unbequem, aber zwingen Sie sich dazu, das Passwort bei den seltenen Neustarts von Hand einzugeben – und verwahren sie es sicher, am besten auch eine Kopie auf einem zweiten Zettel.

Auch ein Einbruch ins lokale Netz, etwa über das WLAN, ist nicht auszuschließen. Die beste Abwehr dagegen ist neben komplexen und langen Passwörtern die Verschlüsselung der Netzwerkverbindung, damit Schlüssel nicht einfach mitgelesen werden können. Alle getesteten NAS-Betriebssysteme brachten die Möglichkeit mit, das Webinterface über HTTPS zu nutzen und optional SMB3 inklusive Verschlüsselung zu aktivieren.

Wo es wirklich notwendig ist, ein NAS aus dem Internet zu erreichen, sollte dies ausschließlich mit einem gültigen TLS--Zertifikat geschehen. Asustor, Synology und QNAP haben dafür einen Assistenten an Bord, der automatisch ein Let’s-Encrypt-Zertifikat beschafft und regelmäßig erneuert.

Backups

Ein NAS zählt nur als Backup, wenn es ausschließlich zur Sicherung von Daten eingeschaltet wird und der Nutzer auf seinem System im Blick hat, dass alles in Ordnung ist. Wenn das NAS permanent läuft und als alleiniger Speicherplatz für bestimmte Daten dient, sind diese stark bedroht; ein Verschlüsselungstrojaner auf einem verbundenen Rechner kann Daten auch auf dem NAS chiffrieren und so Ihrem Zugriff entziehen. Deshalb ist es unumgänglich, regelmäßig eine

„Offlinekopie“ der eigenen Daten anzulegen.

Netgear, Asustor und WD boten im Test die einfachste Form eines Offline-Backups: Die System-Tools kopieren einfach sämtliche Daten aus ausgewählten Ordnern auf ein angestecktes USB-Speichermedium. Das ist zwar sehr simpel einzurichten, hat allerdings den zeitfressenden Nachteil, dass alle Dateien unabhängig davon kopiert werden, ob sie geändert wurden oder nicht. Weiter kann auch hier ein Verschlüsselungstrojaner zuschlagen, wenn man den Angriff nicht bemerkt.

Besser machen es QNAP und Synology mit ihren Backup-Assistenten: Sie können inkrementell sichern, also nur Änderungen seit dem letzten Backup mitnehmen, und bieten außerdem umfangreiche Versionierungseinstellungen: Das NAS behält vorherige Stände der Dateien, sodass diese bei Bedarf wiederhergestellt werden können. Dafür kann man beispielsweise einfach ein Versionslimit festlegen, also die maximale Anzahl von Dateiversionen unabhängig von der Zeit, in der sie entstehen. Die Versionierung lässt sich auch so einstellen, dass nur Kopien unterschiedlichen Alters – beispielsweise tages-, wochen- und monatsweise – aufbewahrt werden. So steigt die Wahrscheinlichkeit, dass man nach einem Trojanerbefall noch an zwar alte, aber verwertbare Daten kommt.

QNAP und Synology (im Bild) bieten als einzige Hersteller im Test Sicherheitsscanner für ihre NAS-Betriebssysteme. Sie suchen nach sicherheitsrelevanten Schwachstellen in der Konfiguration und melden diese. Synology nennt die Funktion „Sicherheitsberater“ („Security Advisor“), QNAP rein englisch „Security Counselor“.

Cloud-Synchronisierung

Alle Betriebssysteme im Test bringen Clients für mehrere Cloud-Services mit. Dazu müssen die Zugangsdaten eingegeben und ein Ordner festgelegt werden, dessen Inhalt der Client auf den Clouddienst spiegelt. Die Synchronisierung erfolgt dann permanent ohne Ihr Zutun im Hintergrund.

Wozu ein Clouddienst, wenn man ein NAS hat? Er kann, nach genauer Prüfung der Datenschutzrechtslage und der allgemeinen Geschäftsbedingungen, als zusätzliche Sicherungsebene für wichtige Dateien dienen. Viele Clouddienste unterstützen zumindest einfache Versionierung und sind besser gegen Ausfälle geschützt als das heimische NAS.

Wer regelmäßig größere Dateien mit anderen teilt, profitiert noch auf andere Weise: Man kopiert die große Datei mit Gigabit-Geschwindigkeit in den Cloud-Synchronisationsordner auf dem NAS, danach kümmert es sich automatisch um das Hochladen und der Rechner kann heruntergefahren werden. Gleichzeitig freut sich der Empfänger über einen schnelleren Download.

Synology bietet als einziger NAS-Hersteller einen eigenen Cloudspeicher an, genannt „Synology C2“. 100 GByte kosten 10 Euro, 300 GByte 25 Euro und 1 TByte 60 Euro pro Jahr. Den Lagerort der Daten muss man im Angebot vorher auswählen, zur Wahl stehen Frankfurt und Seattle. Der Synology-Client kann sowohl für den eigenen Dienst als auch für Drittanbieter das Backup verschlüsseln, sodass es nicht vom Cloud-Anbieter gelesen werden kann.

NAS-App-Erweiterungen

Wenn es darum geht, den „Network Attached Storage“ zum „Network Attached Multitalent“ zu machen, also weitere Funktionen auf dem NAS per App nachzurüsten, sind Synology, Asustor und QNAP mit ihren App Stores ganz vorne dabei. Im Test überraschen alle positiv mit einer umfangreichen Auswahl von aktuell gehaltenen eigenen Drittanwendungen und eigenen Erweiterungen.

WD und Netgear schneiden an dieser Stelle nicht so gut ab; viele Anwendungen sind veraltet, was Sicherheitslücken aufreißen kann, und einige der eigenen Erweiterungen wirkten hemdsärmelig umgesetzt. Wer das NAS zu mehr als einem reinen Datenspeicher aufrüsten will, orientiert sich besser bei QNAP, Asustor und Synology.

Fazit

Alle getesteten Systeme sind mindestens für ihren grundlegenden Zweck als Netzwerkspeicher brauchbar. Den größten Funktionsumfang bieten zweifelsohne Synology und QNAP, als Einsteiger muss man sich aber erst einmal einarbeiten. Asustor ist den beiden NAS-Marktführern dicht auf den Fersen und verfolgt ein ähnliches Bedienkonzept, muss aber an der Übersetzung und den Beschriftungen allgemein arbeiten. Western Digital und Netgear zeigen eher das klassische, zweckgebundene NAS-Betriebssystem, das Einsteiger nicht gleich in einen riesigen Funktionswald schickt und größtenteils fertig konfiguriert ist, aber keine Multitalente aus den Geräten macht – was auch nicht jeder braucht. *(amo@ct.de)*

QNAP QTS 4

Direkt nach der fünfschrittigen Grundeinrichtung und dem ersten Blick auf den NAS-Desktop im Browser öffnet QTS trommelfeuerartig neue Fenster. Bevor man sich versieht, ist der Speichermanager im Vordergrund und verlangt Anweisungen

zur Konfiguration der eingesteckten Festplatten. Die -Einstellungen sind zwar passend vorbesetzt, aber dass man sie unverändert verwenden kann, signalisiert QNAP nicht – hier braucht man also Vorwissen oder Hilfe. QNAPs Desktop sieht dem des direkten Konkurrenten Synology sehr ähnlich und auch die Menüstruktur weicht nur wenig ab. Wer Synology kennt, kommt auch mit QNAP klar. Allerdings fällt auf, dass es mit der Übersetzung bei QNAP noch an einigen Stellen geringfügig hapert.

Die nützliche Backup-Funktion „HBS 3 Hybrid Backup Sync“ will zwar händisch nachinstalliert werden, integriert sich jedoch nahtlos ins Betriebssystem und nimmt den Benutzer bei der Erstellung von Backup-Aufträgen gut an die Hand. Die Software bringt umfangreiche Versionierungs- und Filterregeln mit, sodass das Anpassen an die eigenen Erfordernisse kein Problem ist.

Für das restliche Betriebssystem stellt QNAP das Hilfecenter mit einer ausführlichen Dokumentation sowie den „Security Counselor“ bereit. Weshalb letzterer per App Center nachinstalliert werden muss und nicht von Haus aus an Bord ist, erschloss sich uns nicht.

QTS kann eine Hürde für Einsteiger darstellen; das System ist umfangreich und benötigt zur gründlichen Einrichtung Wissen und Zeit. Doch wer sich einarbeitet, wird den großen Funktionsumfang schnell zu schätzen wissen.

Synology DSM 6

Der „Disk Station Manager“, kurz DSM, von Synology bringt ab Werk ein Minimalsystem zur Installation mit. Dieses lädt die neueste Betriebssoftware herunter und installiert sie. Die Grundeinrichtung beschränkt sich auf Wesentliches; sie fordert direkt auf, ein neues Benutzerkonto mit sicherem Passwort zu erstellen. Der Hersteller empfiehlt zudem einige Pakete zum schnellen Einstieg, unter anderem einen Backup- und einen

Foto-Manager, und installiert diese auf Wunsch. Auf dem desktopartigen System angekommen empfängt DSM den NAS-Admin mit Tipps und einem dezenten Hinweis auf den „Sicherheitsberater“, der sich gern auch direkt präsentieren dürfte. Nach einem System-Scan erscheinen sinnvolle Hinweise zur Sicherheit; nur die Unterscheidung zwischen privater und geschäftlicher Verwendung hat uns gestört, denn die zusätzlichen Scans, beispielsweise ob die automatische HTTPS-Weiterleitung aktiviert oder das Passwort sicher ist, steigern auch daheim die Sicherheit.

Darüber hinaus muss man sich selbst mit dem System vertraut machen, etwa indem man die auf dem Desktop verlinkte Hilfe liest. Einzelne Erweiterungen kommen oft wieder mit Assistenten für die Grundeinrichtung – das Sicherungsprogramm Hyper Backup führte uns beispielsweise sehr leicht durch die Konfiguration eines Backups auf einer Vielzahl von Zielen von USB-Stick bis Cloudspeicher.

Insgesamt bietet Synology ein rundes und ausgereiftes NAS-Betriebssystem, das lesewilligen Einsteigern mit Assistenten, Schnelltipps und einer umfangreichen Dokumentation einen leichten Start ermöglicht.

Netgear ReadyNAS 6

Der ReadyNAS-Assistent fällt direkt am Anfang positiv wie negativ auf: Er fragt nach einem E-Mail-Konto für Warnmeldungen – besonders praktisch, um zeitig über einen drohenden Festplattendefekt informiert zu werden. Allerdings ist der Benutzername „admin“ nicht änderbar und somit ein Punkt weniger, den ein Angreifer erraten muss.

ReadyNAS hat ein klassisches, in Registerkarten sortiertes Webinterface. Auf den ersten Blick erscheint das System dadurch deutlich übersichtlicher als die desktopartigen Systeme von Asustor, QNAP oder Synology. Im Lauf des Tests empfanden wir die Menüstruktur durch Inkonsistenz, wenige In-

System-Hilfen und mehr oder minder versteckte Untermenüs aber als labyrinthisch. Beispielsweise verwirren unscheinbare Zahnrad-Icons oder klickbare Statussymbole mit vielen zusätzlichen Einstellungen; die Fensterstruktur anderer Systeme merkt man sich leichter. Auch manche Bezeichnungen erschienen uns nicht passend übersetzt.

Die Sicherungsfunktion ist leicht konfigurierbar und kann praktischerweise pauschal, also ohne genaue - Laufwerkszuordnung, auf beliebige USB-Medien sichern. Definiert ist nur der USB-Anschluss, an dem zum Zeitpunkt der Sicherung ein Speicher stecken muss. Versionierung fehlt jedoch, was das Risiko erhöht, Daten durch einen - Verschlüsselungstrojaner zu verlieren.

Netgears ReadyNAS könnte mit etwas Überarbeitung und Erweiterung der Software deutlich besser bei Heimnutzern ankommen, die einfach nur eine Datenhalde suchen und den großen Funktionsumfang anderer Systeme nicht benötigen. Insgesamt ist das System jedoch gut benutzbar, wenn man sich in der gewöhnungsbedürftigen Menüstruktur einmal orientiert hat.

Western Digital My Cloud OS 5

Das My Cloud OS von Western Digital tut direkt am Anfang etwas für die eigene Sicherheit, wenn auch erfahrene Nutzer nicht unbedingt Fan dieser Selbsttätigkeit sein dürften: Das Betriebssystem registriert einen DNS-Eintrag mit seiner lokalen IPv4-Adresse und fordert für diesen Hostnamen ein Let’s-Encrypt-Zertifikat an. HTTPS-Verbindungen sind also sofort sicher verschlüsselt.

Nach der Grundeinrichtung erreicht man eine Weboberfläche mit Registerkarten, die für eine tolle erste NAS-Erfahrung sorgt: Die Menüs sind logisch und konsistent strukturiert und Menüpunkte sind größtenteils mit kleinen Ausrufezeichen versehen, die kurze Erläuterungen bereithalten. Zwei Mankos

sind uns dennoch aufgefallen: Die Zeiteinstellungen (korrekte Zeitzone, Aktivieren der Sommerzeitumschaltung) fehlen im Einrichtungsassistenten und die Passwortrichtlinie beschränkt sich auf eine Mindestanzahl von acht Zeichen – mehr sind besser.

Die Backup-Funktion auf USB überzeugte uns nicht: Auf USB-Geräte können Dateien entweder „kopiert“ oder „synchronisiert“ werden; ersteres bedeutet, dass alle Dateien, die schon im Backup liegen, nach Änderung des Originals nicht aktualisiert werden, zweiteres, dass alles immer wieder zeit- und stromfressend überschrieben wird. Wer ein echtes Backup braucht, kommt um das Synchronisieren nicht herum.

Das My Cloud OS punktet mit leichter Einrichtung sowie einem einsteigerfreundlichen Webinterface. Wenn Western Digital die Sicherungsfunktion überarbeitet und fest integriert, kann das System zur guten Wahl für Nutzer werden, die lieber ein übersichtliches, auf das Wesentliche beschränktes NAS haben wollen als ein Funktionsmonster.

Asustor ADM 3

Asustor sorgt bei der Grundeinrichtung für Verwirrung: Sätze wie „Nutzt die optimierten Einstellungen entsprechend Ihren computer-Einstellungen und installierten Festplatten.(empfohlen)“ und „Was beschreibt Ihre Datenspeicheranforderungen am besten? Maximale Kapazität oder Balanciert“ konnten wir nur durch Ausprobieren verstehen; ersteres meint die Schnelleinrichtung mit den empfohlenen Einstellungen, zweiteres RAID 0 oder RAID 1.

Doch Asustor gibt sich sonst Mühe, den Nutzer gut abzuholen: Auf dem Browser-Desktop empfängt einen der „Data Master“, der mit kurzen Beschreibungen erläutert, wo was zu erledigen ist – inklusive Menüverlinkung in der Erklärung, sodass das Suchen entfällt. Die Systemstruktur ähnelt insgesamt stark der von QNAP und Synology und bietet einen ähnlichen Umfang. Die Apps

im „App Central“ erschienen uns größtenteils aktuell. Eine leicht einzurichtende Backup-Funktion hat Asustor ebenfalls integriert.

Leider ziehen sich fehlende, verwirrende oder gar falsche Übersetzungen durchs gesamte System. Längere Texte wirken wie automatisch übersetzt und dort, wo Übersetzungen fehlen, ist der Text auf Englisch. Englischsprachige Einsteiger haben es womöglich einfacher, wenn sie das System auf Englisch benutzen.

Asustor orientiert sich zweifelsohne an Synology und QNAP und versucht aufzuholen. Mit der grundlegenden Systemstruktur, den Hilfen und dem Funktionsumfang hat Asustor bereits den Fuß in der Tür. Allerdings hapert es noch sprachlich bei der Dokumentation und den In-System-Hilfen; die Backup-Funktion könnte ausgeklügelter sein.

1. Literatur
2. Andrijan Möcker, Videoschlucker, Heimüberwachung: Videostreams mit dem NAS aufzeichnen, c’t 5/2019, S. 102

[/expand]

Amazon als Handelsplattform

Amazon als Handelsplattform

[expand title=“mehr lesen…“]

Amazon als Handelsplattform

Beim Produktsuchriesen

Johannes Ungerer

Die Stimmen der Onlinehändler, die sich von Amazon gegen die Wand gedrückt fühlen, werden lauter. Gleichzeitig sehen immer mehr Verkäufer den Amazon-Marktplatz als Chance. Als maßgebliche Vorteile gelten der riesige Kundenstamm und die Möglichkeit der vollautomatisierten Versandabwicklung.

iX-TRACT

Onlinehändler können ihre Produkte statt über eigene Shops auch über Amazon vertreiben und die Logistik dorthin auslagern.

Dadurch erreicht man schnell große Sichtbarkeit ohne aufwendiges Marketing, muss sein Angebot jedoch auf Amazons Suchalgorithmus anpassen.

Händler können mit Fulfillment by Amazon ortsunabhängig werden und von den Werkzeugen des Unternehmens profitieren.

Amazons Regelwerk bietet eingeschränkten Freiraum bei der Produktpräsentation und schirmt einen Teil des Geschäftsprozesses vom Anbieter ab.

Obwohl Amazon versucht, eine Monopolstellung im Onlinehandel zu erreichen, existieren noch immer Tausende sehr profitabler unabhängiger Onlineshops. Viele Händler stellen sich allerdings inzwischen die Frage, ob sich der Aufwand eines eigenen Shops überhaupt noch lohnt.

Pauschal lässt sich das nicht beantworten. Neuen Shops fehlt häufig sowohl die Reichweite als auch das Vertrauen der Kunden

in die Marke. Dafür sind zunächst große Investitionen in Marketing und Suchmaschinenoptimierung nötig. Außerdem muss man eine Infrastruktur mit Domain und ein Warenwirtschaftssystem einrichten.

Für Marken, die bereits einen gewissen Bekanntheitsgrad erreicht haben, kommt ein eigener Onlineshop durchaus in Betracht. Anders als auf öffentlichen Marktplätzen wie Amazon oder eBay haben Unternehmen dann die vollständige Kontrolle über ihre Geschäftsprozesse. Ein eigener Shop bietet außerdem deutlich bessere Möglichkeiten, Produkte individuell und auf die Zielgruppe angepasst zu vermarkten und die Vision des Unternehmens zu verbreiten, während öffentliche Marktplätze in der Regel sehr eintönig gestaltet sind.

Insgesamt erfordert ein profitabler Onlineshop mehr als lediglich Software und ein paar Produkte. Vor allem für noch unbekannte Marken ist der Start kostspielig und zeitaufwendig. Eine attraktive Alternative kann es daher sein, seine Produkte zunächst über bestehende Marktplätze zu vertreiben. Da Amazon in den letzten Jahren ein immenses Wachstum zu verzeichnen hat und der Marktplatz sehr lukrativ zu sein scheint, soll dessen Angebot im Folgenden vorgestellt werden.

Riesiger Kundenstamm, großes Vertrauen

Umsatzentwicklung von Amazon gegenüber dem Vorjahr in den Jahren 2007 bis 2015 (Abb. 1) *Quelle: Statista.com*

Die Zahlen sprechen für sich: 44 Millionen Kunden – davon 17 Millionen mit Prime-Abonnement – hat Amazon allein in Deutschland. Der wohl größte Vorteil für die Händler ist die Tatsache, dass all diese Kunden von sich aus auf die Seite strömen. Teure Marketingmaßnahmen, um Kunden in den eigenen Shop zu bringen, können entfallen. Durch eine geschickte Vermarktung und ein Händchen für die Suchmaschinenoptimierung auf Amazon können Händler in kurzer Zeit viel Traffic auf ihre Angebote leiten.

Dazu kommt, dass die Kunden mittlerweile mit Amazons Zahlungs- und Bestellvorgängen vertraut sind. Viele Kunden haben bereits ihre Daten (Zahlungsmethode, Anschrift) hinterlegt und wissen, dass sie dort bequem und unkompliziert einkaufen können.

Es scheint, als würde die Ausweitung des Unternehmens immer weiter voranschreiten. Amazons Wachstum ist konstant und das Unternehmen erschließt ständig neue Bereiche, inzwischen sogar den Handel mit Lebensmitteln.

Amazon wird zur Suchmaschine

Wenn es um die Beschaffung von Informationen geht, ist Google im Internet schon lange die Nummer eins. Wenn es allerdings um produktbezogene Daten wie die Akkulaufzeit eines Notebooks geht, tendieren Interessenten immer häufiger dazu, Google zu umgehen und direkt auf Amazon zu suchen. Das Unternehmen entwickelt seit 2003 für diesen Anwendungsbereich eine eigene Suchmaschine, spezialisiert auf das Vergleichen von Produkten. Die Tochterfirma A9, deren Aufgabe die Entwicklung des Algorithmus ist, sagt über sich selbst: „We manage product search and advertising technologies that are scalable, highly available, and cross-platform for our parent company, Amazon, and other clients.“

Den Fokus bei der Entwicklung legt das Unternehmen darauf, dass Kunden einen Bogen um den großen Konkurrenten Google machen. Amazon nutzt für seinen Algorithmus interne Daten und kann so spezifische, optimierte Abfragen verwenden. Bei der Google-Suche werden Millionen von Webseiten durchforstet und bewertet, wodurch sich die Suche deutlich aufwendiger gestaltet.

Die Studie „Cross-Channel im Umbruch – das Informations- und Kaufverhalten der Konsumenten Vol. 7“, durchgeführt von ECC Köln in Zusammenarbeit mit SAP Hybris, zeigt, dass Kaufinteressierte immer häufiger Amazon als erste Instanz für die Suche nutzen (siehe „Alle Links“ am Ende des Artikels).

Der Studie zufolge informiert sich inzwischen etwa ein Drittel der Befragten auf Amazon, bei Google nur 14,3 Prozent.

Mit dem Programm Fulfillment by Amazon (FBA) erhalten Händler die Möglichkeit, den Versand sowie Retouren- und Kundenservice an Amazon auszulagern. Dazu sendet der Händler seine Ware in der Regel an eines der Logistikzentren, die überall in Deutschland verteilt sind. Dadurch wird zudem der Prime-Versand für die jeweiligen Produkte freigeschaltet.

Vollautomatisierte Abwicklung

Sobald ein Kauf getätigt wurde, versendet Amazon das Produkt automatisch aus dem eigenen Lager. Die Kosten für den Service setzen sich aus der Versandgebühr pro Einheit und der monatlichen Lagergebühr zusammen. Erstere berechnet sich aus Abmessung und Gewicht, Letztere beträgt (pro Kubikmeter) 12,50 Euro von Januar bis September und 18 Euro von Oktober bis Dezember.

Selbstverständlich gibt es auch externe Angebote mit ähnlichen Dienstleistungen. RHIEM Fulfillment, Baur Fulfillment Solutions und Rhenus Logistics etwa sind als Fulfillment-Anbieter eine Alternative zu Amazon.

Viele kleine Händler nutzen mittlerweile Amazon FBA, um sich ein ortsunabhängiges Unternehmen aufzubauen. Dazu suchen sie sich einen passenden Hersteller und lassen die Waren direkt an Amazons Logistikzentrum senden. Sie selbst kümmern sich um die Suchmaschinenoptimierung und Vermarktung – Aufgaben, die keine physische Präsenz erfordern.

Durch gezielte On- und Off-Page-Optimierung soll nämlich das Produkt sichtbar werden. Da neu eingestellte Artikel für Kunden kaum zu finden sind, müssen die Anbieter künstlich die Verkäufe ankurbeln, etwa mit Pay-per-Click-Anzeigen oder Gutscheinen für Tester. Dadurch rangieren die Produkte mit der Zeit weiter vorne in den Suchergebnissen und mehr potenzielle

Kunden finden das Produkt direkt über die Amazon-Suchleiste.

Daran schließt die Conversion-Rate an: Wie viele der potenziellen Kunden werden tatsächlich auch Käufer? Einen großen Einfluss haben die Kundenrezensionen. Häufige negative Bewertungen sind der Conversion-Killer schlechthin. Das lässt sich nur bedingt steuern, wichtig sind daher die Produktqualität und das Preis-Leistungs-Verhältnis.

„Premium“-Kunden und viele Impulskäufe


Aussagekräftige Fotos sind ein wichtiger Bestandteil der Produktvermarktung (Abb. 2).
Auch relevant sind die Bilder: möglichst scharfe und helle Fotos, die die Alleinstellungsmerkmale des Produktes hervorheben. Die Größe sollte außerdem 1000 Pixel nicht unterschreiten, damit die Zoomfunktion aktiviert ist.

Im Vergleich zu vielen Onlineshops und Marktplätzen wie eBay ist Amazon teuer. Häufig lassen sich woanders günstigere Angebote finden. Wieso ist Amazon trotzdem so beliebt?

Offensichtlich ist, dass viele Kunden auf Amazon gerne bereit sind, den einen oder anderen Euro mehr zu bezahlen, um möglichst schnell und unkompliziert ein qualitativ hochwertiges Produkt zu bestellen. Das Design der Seite ist auf diese Kunden angepasst: strukturiertes Sortiment, Übersichtlichkeit, strenge Richtlinien für Händler. Vergleicht man das Prinzip mit Supermärkten, so ist Amazon wohl kein Discounter, sondern der Edeka unter den Onlinemarktplätzen.


Durch Einblenden passender Produkte versucht Amazon, Kunden zu Impulskäufen anzuregen (Abb. 3).
Das führt dazu, dass sich die Kunden gerne auf der Plattform aufhalten und durch Impulskäufe dazu verleitet werden, deutlich mehr Geld auszugeben und mehr Produkte zu kaufen als im Vorhinein geplant. Um das zu begünstigen, bietet Amazon

seinen Kunden mögliche Ergänzungsprodukte an (siehe Abbildung 3).

Ihr Sortiment durch eine Premiumpositionierung auf genau diese Kunden abzustimmen, kann für viele Händler die richtige Strategie sein, um sich gegen etablierte Marken und Billiganbieter aus Asien in Stellung zu bringen. Der Ansatz dabei ist, qualitativ sehr hochwertige Produkte im oberen Preissegment anzubieten. Somit entgeht man dem Preiskampf der Masse zu einem gewissen Maße und kann höhere Margen erzielen.

Einfaches Expandieren auf EU-Marktplätze

Sofern man seine Ware bereits in Amazons Logistikzentrum lagert, sind es nur noch wenige Klicks, um auf allen EU-Markplätzen zu verkaufen. Große Märkte befinden sich außer in Deutschland vor allem in England, Frankreich und Italien. Amazon kümmert sich um die gesamte Abwicklung und verteilt die Ware auf alle Länder. Für Händler sind hauptsächlich die Sprachbarrieren und das Umsatzsteuerrecht in den jeweiligen Ländern zu beachten. Mittlerweile verkauft Amazon in folgenden Ländern: USA, Deutschland, Österreich und Schweiz, Großbritannien, Frankreich, Kanada, Italien, Spanien, Niederlande, Australien, Brasilien, Japan, China, Indien und Mexiko.

Differenzierte Gebührenstaffelung

Selbstverständlich hat es seinen Preis, die Reichweite nutzen zu dürfen und eigene Produkte auf Amazon zu platzieren. Die Gebühren für Händler hängen von verschiedenen Faktoren ab. Für diejenigen, die nur sehr wenig auf Amazon verkaufen möchten (weniger als 40 Artikel pro Monat), genügt ein kostenloser Basis-Account. Bei diesem Account fallen keine monatlichen Gebühren an, sondern es wird eine Verkaufsgebühr von 99 Cent für jeden verkauften Artikel berechnet.

Bei größerem Verkaufsvolumen empfiehlt sich das

„professionelle Anbieterkonto“ (39 Euro/Monat zuzüglich Umsatzsteuer). Alle weiteren Gebühren sind abhängig von der Kategorie, in der das jeweilige Produkt gelistet wird. Je nach Kategorie werden 7 bis 15 % auf den Verkaufspreis erhoben.

Eine weitere Gebühr von 25 Cent je verkauftem Artikel fällt an, wenn der Händler nicht in die osteuropäischen Länder liefert. Damit möchte Amazon seine Händler motivieren, diese Marktplätze zu erschließen, um das Produktangebot in diesen Ländern zu vergrößern und dort schneller Fuß zu fassen.

Auch bei den Versandgebühren ist Amazon gnadenlos: Ob ein Kunde Prime nutzt oder selbst die Versandkosten trägt, macht für den Verkäufer keinen Unterschied. Die einheitliche Versandgebühr zahlt der Händler bei jedem Verkauf, wenn er die Fulfillment-Dienstleistung in Anspruch nimmt. Nicht-Prime-Kunden sind für Amazon also doppelt lukrativ.

Größter Konkurrent: Amazon

Solange man bei Amazon verkauft, sollte man zwei Dinge immer im Hinterkopf haben:

– Amazon hat detaillierte Einblicke in alle Geschäftsvorgänge.

– Mit der Marke AmazonBasics verkauft Amazon selbst alle möglichen Produkte.

Hat man es geschafft, sein Produkt überdurchschnittlich gut zu platzieren, kann man sich über hohe Umsätze freuen. Bedenken sollte man aber immer, dass sich auch Amazon dessen bewusst ist und profitable Produkte theoretisch immer selber verkaufen kann, solange keine Patentrechte verletzt werden. Amazon sollte daher von Anfang an auch als potenzieller Konkurrent wahrgenommen werden.

Doch die Eigenmarke ist nicht alles, womit Amazon den Händlern Konkurrenz macht. Manche großen Marken möchte Amazon selbst auf seiner Plattform verkaufen und erlaubt keine unabhängigen

Anbieter. Ein Beispiel sind die Produkte des Hi-Fi-Anbieters Bose.

Viele Richtlinien und wenig Freiheiten

Amazons Ziel ist es, seinen Marktplatz strukturiert und aufgeräumt zu halten. Für die Händler heißt das: Richtlinien und Regeln bis ins letzte Detail. Von der Gestaltung der Fotos und dem Produkttitel bis zur Beschreibung – alles wird genauestens vorgeschrieben. Die Vermarktung speziell auf ein Produkt anzupassen, ist schwer.

Auf Amazon gibt es ein Verwarnungssystem, das Verstöße des Händlers gegen die Richtlinien speichert. Einsehen kann aber selbst der Verkäufer dieses System nicht. Grundsätzlich muss man immer im Kopf haben, dass der Verkäufer-Account bei Nichteinhaltung einer Regel jederzeit suspendiert werden kann.

Verschlüsselung der Kundendaten

Verkäufer dürfen nicht außerhalb von Amazons Plattform mit den Kunden kommunizieren. Verständlicherweise möchte der Marktplatzbetreiber seine wertvolle Kundschaft halten und verhindern, dass Händler die Kunden von Amazon weg auf ihren eigenen Shop leiten.

Zu erkennen ist das auch an den strikten Richtlinien: So verbietet Amazon Händlern explizit, extern Kontakt mit ihren Kunden aufzunehmen oder diese durch Rabatte oder Aktionen auf den eigenen Onlineshop zu verweisen. Die Bemühungen des Unternehmens zielen darauf ab, dass es für den Händler schwer ist, seine Marke und Vision an den Kunden weiterzugeben.

Fazit

Ein eigener Onlineshop bietet wertvolle Freiheiten und ist für etablierte Marken definitiv der lukrativere Absatzmarkt. Im eigenen Shop sind die Gebühren geringer und der Händler erhält

sämtliche Kundendaten, die für das After-Sales-Marketing wichtig sind, um nach einem erfolgreichen Kauf Kunden zu mehr Geschäftsabschlüssen zu bringen.

Wer klein beginnt und möglichst schnell wachsen möchte, sollte sich das Errichten eines eigenen unabhängigen Shops jedoch gründlich überlegen. Ohne Reichweite und Vertrauen bedarf es großer Investitionen, bevor der Shop erste Früchte tragen kann. Amazon bietet seinen Händlern von Anfang an die nötige Reichweite und trägt dabei die Verantwortung für die Vermarktung und Verbreitung der Internetseite. Hinzu kommt, dass der Versand sowie der Kunden- und Retourenservice (mitunter die zeitaufwendigsten Aufgaben) zu sehr guten Konditionen an Amazon outgesourct werden können. (jab) Johannes Ungerer handelt über Onlinemarktplätze und Onlineshops unter seinem eigenen Label. Sein Spezialgebiet ist dabei die On-Page- und Off-Page-Suchmaschinenoptimierung auf Amazon.

Literatur

- [1] Sabine Buschmann, Dr. Eva Stüber, Sabrina Klinksiek-Rumpf; Cross-Channel im Umbruch 2015: Informationsverhalten, Kaufverhalten, Cross-Channel-Effekte, Click & Collect; Köln 2015

Beim Produktsuchriesen

- Cross-Channel im Umbruch
- Amazon-Blog des Autors
- Amazon Podcast
- Interview mit dem Autor
- Verkaufsgebühren bei Amazon
- Versand bei FBA

[/expand]

Bessere Websites

Bessere Websites

[expand title="mehr lesen…"]

Prinzipien der Webentwicklung

Bessere Websites

Jens Oliver Meiert

Professionelle Webentwicklung bedeutet mehr, als bloß funktionierenden Code zusammenzuhacken. Es lohnt sich, gelegentlich mal einen Schritt zurückzutreten und darüber nachzudenken, wie man zu besseren Ergebnissen kommt.

Nach rund 20 Jahren ist die Webentwicklung zu einem reifen Berufsfeld geworden. Permanente Veränderung ist dabei eine Konstante: Immer neue technologische Standards, Möglichkeiten und Praktiken haben professionelle Webentwickler begeistert, aber auch frustriert. Der Artikel auf Seite 38 erklärt, wie man die Performance moderner Websites analysiert und verbessert. Ab Seite 46 wird beschrieben, wie man durch die clevere Trennung von lesenden und schreibenden Datenbankzugriffen (Command Query Responsibility Segregation, CQRS) Web-Anwendungen erheblich beschleunigen kann

Doch es gibt grundlegende Prinzipien, die die Arbeit in der Vergangenheit geleitet haben und die auch in Zukunft als

Leitfaden dienen können [a]. Als ein Entwickler, der an kleinen und großen Projekten gearbeitet, zu Standards beigetragen und zu empfohlenen Entwicklungspraktiken publiziert hat, will ich hier eine konkrete Sicht auf diese Prinzipien liefern.

Fokus auf den Nutzer

Auch auf die Gefahr hin, es zum tausendsten Mal zu sagen: Webentwicklung muss sich immer auf den Nutzer konzentrieren. Das ist aus gutem Grund ein Google-Prinzip, das sich jeder Entwickler zu eigen machen sollte. Einige Beispiele sollen die enge Verknüpfung zwischen Endnutzern und Entwicklern verdeutlichen. Barrierefreiheit zum Beispiel dreht sich ausschließlich um Interaktionsmöglichkeiten der Endnutzer, muss aber bei der Entwicklung immer mitgedacht werden. Auch Codeperformance wird normalerweise von der Endnutzerseite ermittelt. Tatsächlich sind höchstens Designfragen von dieser engen Verknüpfung ausgenommen: Usability-Tests betreffen Endnutzer, aber Entwickler nicht unbedingt.

Zwei Dinge helfen bei der Konzentration auf den Nutzer. Das eine ist das Auf- und Durchsetzen von Coderichtlinien. Einen guten Einstieg bieten styleguides.io, cssguidelin.es oder das kurze Buch „The Little Book of HTML/CSS Coding Guidelines“ des Autors [b]. Das andere ist ein klarer Qualitätsanspruch.

Fokus auf Qualität

Der Fokus auf die Qualität ist, was den Experten vom Amateur trennt, den Profi vom Hobbyentwickler. Jeder kann einen Webauftritt zusammenhacken – das Internet ist voll mit den Ergebnissen –, aber wenige können eine qualitativ hochwertige Website bauen. Es geht darum, eine Einstellung für Qualität zu kultivieren, Qualitätsziele zu setzen und Werkzeuge zu implementieren, die Qualität messen oder verbessern [c]. Das ist alles leider leichter gesagt als getan.

Letztlich geht es um die Frage: Wie kann man beurteilen, wie gut die eigene Arbeit ist, und wie lässt sie sich verbessern? Der qualitätsbewusste, ambitionierte Entwickler mag sogar auf Exzellenz abzielen und damit auf das Qualitätsprinzip noch eins draufsetzen. Unser Arbeitsfeld weist eine Tendenz zur Schludrigkeit auf, die sich vielleicht ablegen lässt, wenn wir uns zwischendurch auf Exzellenz besinnen.

„Keep it simple“

Es gibt kaum ein besseres Prinzip als das der Einfachheit. Es entspringt der Beobachtung, dass weniger tatsächlich mehr ist. Auch wenn nicht alle Entwickler die Kraft von „keep it simple“ sofort durchdringen und sie effektiv einzusetzen wissen: Erst mal verstanden, ist diese Kraft nicht zu unterschätzen. Sie ergibt sich letztlich aus dem Wissen, was wirklich wichtig ist und was nicht. Wer die Dinge nicht einfach hält, sagt damit, dass alles wichtig ist; das mag manchmal zutreffen, ist aber meistens falsch.

Ähnlich dem Fokus auf Qualität ist auch das Einfachhalten abstrakt. Was wichtig ist, hängt vom Kontext und den Zielen ab. Die Dinge einfach zu halten, erfordert Fokus und Erfahrung. Letztlich helfen Fokussierung und das Sammeln von Erfahrung zusammen mit dem Wissen von Kontext und Zielen dabei, dem wichtigen Prinzip der Einfachheit gerecht zu werden.

Langfristiges Denken (und Vermeidung von Hypes)

Ebenfalls auf der Liste bewährter Prinzipien befinden sich langfristiges Denken und Nachhaltigkeit. Auch das ist nicht ganz so einfach zu fassen – vielleicht funktionieren Prinzipien genau deshalb, weil sie zum Nachdenken über die eigene Arbeit zwingen? Langfristiges Denken bedeutet, Projekte in geplanter, nachhaltiger Manier anzugehen. Konkret heißt

das:

– das Anstreben qualitativ hochwertiger Inhalte und Dienste (und die Angewohnheit, über Kampagnenseiten und Landingpages hinauszublicken);

– das Schaffen benutzbarer und ansprechender Designs (und ihre stetige Verbesserung mittels iterativer Prozesse);

– das Schreiben von robustem Code und seine Wartung sowie das Vermeiden von Code, der mit großer Wahrscheinlichkeit kurz- und mittelfristig schon wieder hinfällig sein wird [d];

– das absolute und leidenschaftliche Ablehnen von „Fire-and-Forget“ [e].

Es gibt einen einfachen Weg, sich langfristiges Denken anzueignen und andere davon zu überzeugen: indem man genau notiert, wie teuer kurzfristiges Denken ist durch all die Extra-Arbeit, die es verursacht. Das führt dann auch gleich zum nächsten Prinzip:

„Don’t repeat yourself“ (oder: jeder Code braucht Wartung)

DRY, „don’t repeat yourself“, ist vermutlich das wichtigste Prinzip für wartbaren Code. Es ist deshalb so wichtig, weil es die Menge an Code reduziert und die Zahl der Stellen vermindert, die man im Blick behalten muss [f]. Weniger Code bedeutet weniger Zeit, die auf Wartung verwendet werden muss.

Wiederholung lässt sich auf zwei Arten vermeiden: keinen Code duplizieren und keine Dateien kopieren. Wenn etwas bereits existiert, muss es einen Weg geben, es wiederzuverwenden. Das erfordert Selbstdisziplin, Bewusstsein, Fleiß und Automatisierung. Schwierig ist die Situation bei der automatisierten Vermeidung von Wiederholungen.

Verantwortungsbewusst entwickeln

Was die Kampagne „Code Responsibly“ (vom Autor einstmals initiiert) schon auf Jahre bewirbt, stellt ein wichtiges Leitprinzip dar: Webentwicklung bedeutet Verantwortung [g]. Das heißt: ständiges Lernen, Achten auf Barrierefreiheit und Performance, semantisch korrekte Auszeichnung der Inhalte, Codevalidierung, Fokus auf Wartbarkeit, Zusammenarbeit mit anderen, gute Dokumentation, Achten auf Qualität sowie die Bereitschaft, anderen etwas beizubringen.

Verantwortungsbewusste Webentwicklung beginnt aber vielleicht schon mit dem Bekenntnis zu Qualität – und zu Professionalität. Dazu gehören Wissen und Fähigkeiten, Urteilsvermögen und Kompetenz sowie natürlich Integrität und angemessenes Verhalten.

Den Überblick behalten

Überblick bedeutet, sein Arbeitsfeld, dessen Grenzen und auch seine Nachbarn zu kennen. Das Arbeitsfeld sollte durch Webstandards [h] und darauf aufsetzende Empfehlungen [i] gut markiert sein. Die Nachbarn umfassen alles, was der Nützlichkeit und Qualität von Sites und Apps dient: Design, Usability und User Experience, Barrierefreiheit, Psychologie und Mensch-Maschine-Interaktion sowie die Inhalte der Sites und Apps. Eine technisch gesunde Website ist unabhängig von ihren Inhalten, eine großartige Website kann diese Grenze unscharf machen.

Es gibt aber auch Grenzen, derer sich Entwickler gewahr sein müssen, insbesondere bei User-Agents. Webentwickler haben einige unnötige und deshalb eher blöde Abenteuer unternommen, wenn sie probiert haben, sich sämtlicher Barrierefreiheits- [j] und User-Experience-Probleme [k] anzunehmen. Wir müssen auch in der Lage sein, zu erkennen, wann etwas nicht mehr unser Problem ist.

Obwohl all diese Prinzipien eher abstrakt sind, muss man aufpassen, was „absolute Wahrheiten“ anbelangt: Oft kommt es halt doch darauf an. Diese abstrakten Prinzipien geben jedoch eine Richtung vor und helfen dabei, bessere Entscheidungen zu treffen. Welche das letztlich sind, liegt in der Hand des Entwicklers: Webentwicklung hat immer einen Zweck. (odi) Jens Oliver Meiert ist Philosoph und Entwickler (Google, W3C, O’Reilly). Er ist spezialisiert auf maßgeschneiderten Code für komplexe internationale Websites.

Bessere Web-Apps

- Jeremy Keith, Design Principles
- Jens Oliver Meiert, The Little Book of HTML/CSS Coding Guidelines
- Jens Oliver Meiert, The Little Book of Website Quality Control
- Einführung in Wartbarkeit
- The Problem of „Fire and Forget“ in Web Design
- CSS, DRY, and Code Optimization
- Code Responsibly
- W3C Standards
- Web Fundamentals
- Joe Clark, When accessibility is not your problem
- window.scrollTo() or: When to Stay Clear of User Agents

[/expand]

Webanwendungen beschleunigen

mit CQRS

Webanwendungen beschleunigen mit CQRS

[expand title="mehr lesen…"]

Webanwendungen beschleunigen mit CQRS

Getrennt siegen

Arne Blankerts, Sebastian Heuer

Typische PHP-Anwendungen verbringen einen Großteil ihrer Zeit damit, Daten aus einer Datenbank abzurufen, zu verarbeiten und als HTML auszugeben. Command Query Responsibility Segregation (CQRS) kann die Zahl der Datenbankabrufe drastisch reduzieren und so die Anwendung erheblich beschleunigen.

iX-TRACT

CQRS (Command Query Responsibility Segregation) trennt lesende und schreibende Zugriffe auf den Datenbestand strikt voneinander. Dadurch lassen sich datenbankbasierte Webanwendungen deutlich beschleunigen.

Der für Schreibzugriffe zuständige Prozess erzeugt bei Änderungen eine neue HTML-Repräsentation des aktuellen Zustands der geänderten Daten.

Bei Lesezugriffen muss die Webanwendung lediglich die

vorproduzierten HTML-Schnipsel ausliefern, statt die Daten aus der Datenbank abzufragen und selbst für die Darstellung im Browser aufzubereiten.

Da das zeitaufwendige Generieren der HTML-Repräsentation nicht mehr bei jedem Zugriff stattfindet, sondern nur noch bei Änderungen der Daten, profitieren vor allem Anwendungen mit deutlich mehr Lese- als Schreibzugriffen.

PHP gehört nach wie vor zu den gängigsten Sprachen bei der Webentwicklung. Stand früher besonders die einsteigerfreundliche Lernkurve im Vordergrund, werden inzwischen auch umfangreiche Webanwendungen mit der Skriptsprache entwickelt. Die aktuelle Version 7 brachte PHP eine deutlich optimierte Laufzeitumgebung – Grund genug, sich näher anzusehen, wie sich heutzutage hochperformante Webseiten mit PHP umsetzen lassen.

Anders als Java oder Node.js folgt PHP dem „shared nothing“-Prinzip: Die einzelnen Requests an den Server teilen sich erst einmal keinerlei Daten. Auch wenn moderne PHP-Frameworks komplexe Implementierungen möglich machen, läuft es technisch doch erschreckend häufig auf die folgenden Schritte hinaus: den für die URL zuständigen Controller ermitteln, mit SQL aus der Datenbank die für das Model notwendigen Daten abfragen, den View mit HTML erzeugen und das Ergebnis ausliefern.

Was hier so einfach klingt, ist in der Praxis natürlich deutlich aufwendiger und wird bei komplexen Datenbankabfragen und steigender Systemlast schnell zum Problem. Auf den ersten Blick wäre dann ein dauerhaft laufender Application Server wie bei Java oder Node.js von Vorteil, der bereits geholte Daten im Speicher hält. Ob und wann geänderte Daten in der Persistenz aktualisiert werden, wäre dann bloß ein Implementierungsdetail.

Problemfall zentrale Datenbank

Der so erzielte Performancegewinn wird jedoch im wahrsten Sinne des Wortes teuer erkauft: Nimmt der Besucherandrang zu, muss man den Server immer weiter aufrüsten. Irgendwann geht es nicht mehr sinnvoll weiter und es bleibt nur eine horizontale Skalierung. War bislang ein einziger Serverprozess für die Datenhaltung verantwortlich, müssen sich jetzt mehrere Prozesse und Maschinen synchronisieren, was entweder eine ausgefeilte Interprozess-Kommunikation oder doch wieder eine zentrale Persistenz etwa in Form einer permanent aktualisierten Datenbank erfordert.

Bei einer derartigen Architektur, egal ob mit PHP oder einer anderen Plattform genutzt, hat selbst eine deutliche Beschleunigung der Laufzeitumgebung nur eine marginale Auswirkung auf die Gesamtperformance: Wenn die Anwendung einen signifikanten Anteil der Zeit mit Warten auf Antworten der Datenbank verbringt, beschleunigt das, nüchtern betrachtet, lediglich das Warten.

Vielleicht liegt die Lösung des Problems also weniger im Beschleunigen der Ausführung als im Vermeiden derselben. Beachtet man zudem, dass die meisten Webseiten deutlich häufiger gelesen als neu geschrieben werden, bietet es sich an, hier zuerst anzusetzen.

Mehr Tempo durch Caching

Wenig überraschend ist die gängige Reaktion auf Performanceprobleme bei Lesezugriffen: die Einführung von Caches. Sie sollen das erneute Generieren einer Antwort unnötig machen, sofern die Anfrage innerhalb eines bestimmten Zeitfensters schon einmal beantwortet wurde.

Naheliegend und technisch am einfachsten umzusetzen ist das Vorschalten eines Reverse Proxy wie Varnish, der die HTML-Ausgabe der Applikation zumeist im Arbeitsspeicher

zwischenspeichert und sie bei ähnlichen Requests direkt an den Client zurücksendet, ohne dass die Applikation Arbeit damit hat. Das sorgt zwar für eine sehr schnelle Beantwortung von Requests, für die bereits eine passende Antwort im Cache liegt, bringt aber neue Probleme mit sich.

Daten in einem Cache sind immer potenziell veraltet, da sie der Reverse Proxy lediglich anhand ihres Alters aus dem Cache löschen kann – die Anwendung soll ja gerade nicht in die Beantwortung eingebunden werden. Zudem kommt es schnell zu Inkonsistenzen, wenn verschiedene Ansichten der gleichen Daten zu unterschiedlichen Zeiten abgerufen wurden und so in unterschiedlichen Ständen im Cache zwischengespeichert sind.

Auch die Auslieferung personalisierter Inhalte unter der gleichen URL ist ein Problem, da die HTML-Ausgabe dann eben gerade nicht identisch ist. Um dennoch vom Caching zu profitieren, müsste die Antwort in personalisierte und nicht personalisierte Fragmente zerlegt werden. Die personalisierten Bereiche müssten dabei für jeden eingehenden Request von der Applikation neu geliefert werden.

Varnish bietet mit Edge Side Includes (ESI) eine in diese Richtung gehende Option. Allerdings bringt das einen zusätzlichen Frontend-Layer außerhalb der Applikation mit sich, was zu neuen Problemen führt. Und gerade die Fragmente mit dynamischen Inhalten, deren Erzeugung teuer ist, werden nicht vom Cache abgedeckt, sodass das Performanceproblem letztlich bestehen bleibt.

Nicht zuletzt erschwert die Abhängigkeit der Anwendung von einem gefüllten Cache auch noch das Deployment neuer Versionen. In der Praxis erweist es sich als sehr kompliziert, zu ermitteln, welche Daten veraltet sind; daher löscht man meist einfach alle Daten aus dem Cache. Das bedeutet allerdings, dass die Anwendung jetzt wieder alle Anfragen bearbeiten muss, was zwangsläufig zu einem Performanceengpass führt.

Caching kann nicht die Antwort sein

Caching lindert also nur Symptome, ohne jedoch die eigentliche Ursache von Performanceproblemen zu adressieren. Die vermeintlich einfache und schnelle Verbesserung der Antwortzeiten wird mit zusätzlicher Komplexität, neuen Ausfallszenarien und der Auslieferung potenziell veralteter Daten erkauft. Es lohnt sich daher, einige Schritte zurückzutreten und erneut einen Blick auf die Softwarearchitektur zu werfen.

Eines der wichtigsten Ziele jeder Website ist die schnellstmögliche Beantwortung von Anfragen. Um das zu erreichen, muss der Server mit der ohnehin schon knappen Ressource Zeit extrem sparsam umgehen. Geht es um Antwortzeiten von 100 Millisekunden und weniger, wird die Luft beim Einsatz gängiger Fullstack-Frameworks auch und gerade im PHP-Umfeld schnell dünn: Die schiere Größe der Codebasis, die zur Laufzeit viele Entscheidungen treffen muss, zusammen mit nur selten benötigter Flexibilität fordert ihren Preis.

Doch auch ein bewusst schlankes, im Zweifel selbst geschriebenes Framework verschenkt noch viel Potenzial. Betrachtet man beispielsweise die Produktseite zu einem Artikel in einem E-Shop, beginnt deren Aufbau meist damit, die Artikelstammdaten – Bilder, Texte, tagesaktuelle Preise – unter Einsatz eines ORMs wie Doctrine aus der Datenbank zu ziehen und die zugehörigen Models zu instanziieren. Die so vorbereiteten Daten gehen dann an eine Template-Engine zur Umwandlung in HTML.

Aber sind all diese Schritte überhaupt notwendig? Aus Sicht des Browsers hat das empfangene HTML mit dem Model „Produkt“ nichts mehr zu tun. Nimmt man hinzu, dass sich die Artikelstammdaten nicht bei jedem Request ändern, folgt daraus: Das Erzeugen der HTML-Repräsentation muss gar nicht während der Beantwortung der Anfrage geschehen, sondern nur dann, wenn sich die zugrunde liegenden Daten ändern.

CQRS to the Rescue

CQRS (Command Query Responsibility Segregation) beschreibt ein Architekturmuster, das lesende und schreibende Operationen strikt voneinander trennt. Konkret bedeutet das, dass ein generisches Model aufgeteilt wird in eines für die lesende und eines für die schreibende Seite. Aus einer Klasse, die eine API zum Abrufen und eine zum Verändern von Daten anbietet, werden also zwei Klassen mit spezialisierten APIs. Das Read-Model ist dann nur noch eine unveränderbare Repräsentation eines Zustandes, während das Write-Model Methoden zu dessen Veränderung bereitstellt.

Diese explizite Trennung eröffnet spannende Möglichkeiten für die Performanceoptimierung. Da lediglich Schreiboperationen den Zustand verändern können, reicht es, wenn sie die Neugenerierung der Zustandsrepräsentation triggern. Bei Abfragen muss lediglich die bei der letzten Änderung erzeugte Repräsentation ausgeliefert werden.

Interessanterweise findet sich diese konzeptuelle Trennung schon im HTTP-Standard: Der lesende *GET*-Request verändert den Zustand der Applikation nicht, während die schreibenden Request-Methoden wie *DELETE, PATCH, POST* und *PUT* eine Zustandsänderung bewirken. CQRS und das Web scheinen also bestens zusammenzupassen.

Für einen E-Shop heißt das: Die HTML-Repräsentation einer Produktseite muss nur dann neu generiert werden, wenn beispielsweise neue Artikeldaten vom ERP-System kommen. Daher kann ein von den eingehenden Requests unabhängiger Prozess diese Daten entgegennehmen und neue HTML-Repräsentationen erzeugen. Listen mehrerer Artikel beispielsweise einer Kategorie kann man aus HTML-Schnipseln für die einzelnen Artikel zusammensetzen.

Key-Value-Store statt SQL

Diese Schnipsel müssen nun so persistiert werden, dass sie sich bei der Verarbeitung eines Requests möglichst schnell laden lassen. Dafür eignet sich ein Key-Value-Store, der beliebige Daten unter einem eindeutigen Schlüssel ablegt. Statt eine SQL-Query an die Datenbank zu schicken, lädt die Anwendung die Daten über den Schlüssel, was viel schneller geht.

Der quelloffene Key-Value-Store Redis beispielsweise beantwortet bei geringem Ressourcenverbrauch mühelos mehrere Tausend Requests pro Sekunde. Dass der Key-Value-Store die generierten Daten dabei nicht wie eine relationale Datenbank in normalisierter Form, sondern im Gegenteil mehrfach in verschiedenen Varianten speichert, gehört dabei zum Konzept. Als Datenquelle für die Generierung kann natürlich weiterhin eine relationale Datenbank dienen.

Das Frontend liefert lediglich HTML-Snippets aus, die beim Anlegen und Ändern von Einträgen in der Produktdatenbank generiert werden. Die Suchmaschine ermöglicht beliebige Kombinationen der Schnipsel (Abb. 1).
Erfahrungen aus Projekten der Autoren zeigen, dass auch der Arbeitsspeicherbedarf geringer ausfällt, als man vielleicht vermuten würde. Und wenn der Key-Value-Store in einer sehr großen Anwendung einmal mehr als ein paar Gigabyte belegt, lässt sich der HTML-Code leicht komprimiert ablegen, was den Speicherbedarf drastisch verringert – um den Preis einer etwas höheren CPU-Belastung durch das Dekomprimieren beim Lesen.

Ein Beispiel aus der Praxis

Im Folgenden zeigen wir die Funktionsweise von CQRS in PHP am Beispiel der E-Commerce-Plattform hinter www.kartenmacherei.de, einem Shop für personalisierbare Print-Produkte wie Einladungskarten oder Fotokalender. Das Team der

kartenmacherei hat 2016 ein komplett neues Shop-Frontend vor eine existierende Standardsoftware gesetzt, das konsequent den Grundsätzen von CQRS folgt. Die Anwendung auf Grundlage dieser Architektur erreicht ohne vorgeschaltetes Caching Antwortzeiten von unter 35 Millisekunden und ermöglicht eine problemlose horizontale Skalierung.

Die Applikation ist in mehrere Komponenten aufgeteilt. Das in PHP geschriebene Frontend ist für die Beantwortung von Requests zuständig und soll daher möglichst wenig zu tun haben. Bei Lesezugriffen ermittelt es anhand des Request-URI die XHTML-Snippets, die zum Aufbau der Seite erforderlich sind, und lädt sie aus dem Key-Value-Store. Anschließend werden sie nach einer vorgegebenen Logik in ein XHTML-Template eingefügt und als HTML an den Client geschickt.

Eine Kategorienseite ist aus fertigen HTML-Snippets zusammengesetzt (Abb. 2).

Listing 1: Kategorienseite

```
<?php
class CategoryPage {
  private $template;
  private $searchEngine;
  private $snippetStore;

   public function __construct(SearchEngine $searchEngine, SnippetStore $snippetStore,  PageTemplate $template) {
    $this->searchEngine = $searchEngine;
    $this->snippetStore = $snippetStore;
    $this->template = $template;
  }

   public function asHtml(CategoryName $categoryName, FilterCollection $filters): HTML {
                                $productSkus = $this->searchEngine->getProductsInCategory($categoryName, $filters);
```

```
                                        $snippets        =
$this->snippetStore->getListTiles($productSkus);
     $template = $this->template->inject('category-items',
$snippets);
    return $template->asHtml();
  }
}
```

Die zu ladenden Produkt-Snippets beispielsweise für eine Kategorieseite im Shop ermittelt eine Suchmaschine. Bei kartenmacherei.de ist das Elasticsearch, eine Alternative wäre Apache Solr. Beide Open-Source-Projekte basieren auf der Lucene-Bibliothek und beantworten Anfragen auch unter hoher Last innerhalb weniger Millisekunden. Elasticsearch gibt für eine Kategorie unter Berücksichtigung von Filterkriterien („nur die Farben Rot und Pink“) eine Liste von Artikelnummern zurück, mit denen das Frontend die HTML-Snippets aus dem Key-Value-Store lädt. Listing 1 zeigt, wie wenig Code im Frontend dafür nötig ist.

Es mag so klingen, als sei der Key-Value-Store nur ein besserer Cache, doch es gibt fundamentale Unterschiede. Aus Sicht des Frontend ist der Store als primäre Datenquelle an die Stelle der zuvor genutzten relationalen Datenbank getreten. Fehlt dort ein Eintrag, verhält sich die Anwendung nicht anders, als wenn ein Eintrag in einer MySQL-Tabelle fehlt. Somit haben die Daten im Key-Value-Store auch keine TTL, da sie per Definition immer den aktuellen Datenstand repräsentieren. Dass eine separate Komponente diese Daten von Zeit zu Zeit durch neuere Versionen ersetzt, weiß das Frontend nicht. Auch das bei Caches übliche Verdrängen älterer Einträge durch neuere gibt es nicht: Geht dem Key-Value-Store der Arbeitsspeicher aus, quittiert er schlichtweg den Dienst oder muss sich mit Swappen behelfen.

Um das Frontend möglichst einfach zu halten, werden schreibende Requests aufgrund von Benutzeraktionen im Frontend an Microservices delegiert. Diese kapseln die gesamte Geschäftslogik und sind unter anderem auch für die inhaltliche

Validierung zuständig. So prüft der Warenkorb-Service beispielsweise, ob ein Artikel überhaupt verfügbar ist, bevor er im Warenkorb landet. Diese Zustandsänderungen werden in der Regel synchron ausgeführt, das Frontend wartet also, bis der Service die Aufgabe erledigt hat. Da Nutzer bei solchen Aktionen eine gewisse Verarbeitungszeit erwarten, sind hier etwas längere Antwortzeiten vertretbar. Tatsächlich gibt es Fälle, in denen eine zu schnelle Beantwortung einer abgesendeten Bestellung zu der Annahme führte, etwas habe nicht funktioniert.

Das Backend erzeugt die HTML-Snippets

Listing 2: Einfügen eines neuen Produkts

```
class ProductSnippetRenderer {
  private $snippetStore;

  public function __construct(SnippetStore $snippetStore)  {
    $this->snippetStore = $snippetStore;
  }

  public function render(Product $product) {
    $this->snippetStore->startTransaction();
$this->snippetStore->addProductListSnippet($this->renderProduc
tListSnippet($product));
$this->snippetStore->addProductDetailPageSnippet($this->render
ProductDetailPageSnippet($product));
$this->snippetStore->addCartItemSnippet($this->renderCartItemS
nippet($product));
    // (...)
    $this->snippetStore->commit();
  }
}
```

Listing 3: Schnittstelle zu Elasticsearch

```
class ElasticsearchProductSearchIndexer {
  private $elasticsearchClient;
```

```
    public function __construct(ElasticsearchClient
$elasticsearchClient) {
      $this->elasticsearchClient = $elasticsearchClient;
  }

  public function index(Product $product) {
$this->elasticsearchClient->index($this->mapProductToElasticSe
archDocument($product));
  }
}
```

Das ebenfalls in PHP geschriebene Backend befüllt den Key-Value-Store mit den vom Frontend benötigten HTML-Snippets und sonstigen Datenstrukturen (Listing 2). Dazu nimmt es neue Produktdaten entgegen (Push) oder fragt regelmäßig nach Änderungen etwa im ERP-System (Pull). Zusätzlich wird die Suchmaschine mit neuen Daten versorgt: Neue Artikeldaten gehen als JSON-Objekte an die REST-API von Elasticsearch (Listing 3).

Die Idee der vorgenerierten HTML-Snippets lässt sich auf die Pflege von Content in einem CMS übertragen: Beim Veröffentlichen von Inhalten generiert das CMS als Backend die passenden Snippets und schreibt sie in den zentralen Key-Value-Store. Das Frontend muss nur noch wissen, unter welcher URL welche Snippets auszuliefern sind. Diese Information kann das CMS als zusätzlichen Eintrag in den Key-Value-Store schreiben.

Dabei muss das CMS gar nicht über das Internet erreichbar sein, da es an der Auslieferung von Seiten nicht beteiligt ist – in Anbetracht der häufigen Sicherheitslücken in den populären Content-Management-Systemen dürfte das manchem Administrator einen ruhigeren Schlaf bescheren. Zudem lässt sich das CMS als interne Komponente leichter austauschen: Die einzige Anforderung ist die Fähigkeit, veröffentlichte Seiten oder Seitenbestandteile als (X)HTML zu rendern und inklusive der dazugehörigen URL zu exportieren.

Da Frontend und Backend nicht voneinander abhängig sind, lassen sie sich problemlos getrennt skalieren. So genügt es, bei steigender Besucherzahl einfach zusätzliche Frontend-Instanzen hochzufahren.

Jeder Onlineshop hantiert mit Session-abhängigen Daten wie der Anzahl der Artikel im Warenkorb eines Benutzers oder personalisierten Preisen. Solche Informationen lassen sich nicht ohne Weiteres als fertiges HTML ablegen: Sämtliche HTML-Schnipsel mit allen möglichen Preisen für jeden existierenden Benutzer zu rendern, ist nicht praktikabel. Daher kann es notwendig sein, einige Informationen weiterhin während der Verarbeitung eines Requests zu ermitteln. Doch auch hier können optimierte Datenstrukturen, etwa einfache Listen mit sämtlichen Preisen je User, vorbereitet und im Key-Value-Store bereitgestellt werden, um dem Frontend die Arbeit zu erleichtern.

Personalisierung im Shop

Listing 4: Personalisierte Preise

```
class PriceInjector {
  private $priceStore;
  private $template;

  public function __construct(PriceStore $priceStore, Template
$template) {
    $this->priceStore = $priceStore;
    $this->template = $template;
  }

  public function updatePrices(User $user) {
    if (!$user->hasCustomPrices()) {
      return;
    }
                                $updatedTemplate       =
$this->template->inject($this->priceStore->getUserPrices($user
));
    return $template;
```

```
  }
}
```

Personalisierte Preise kann das Frontend nach dem Laden der HTML-Snippets abfragen und mittels DOM-Operationen anstelle der Standardpreise in die Snippets einsetzen. Der Aufwand dafür ist minimal und fällt bei einer Performancemessung kaum ins Gewicht. Da das HTML-Snippet die Standardpreise enthält, ist die zusätzliche Arbeit nur erforderlich, wenn der Benutzer eingeloggt ist und abweichende Preise haben kann. Darüber hinaus kann es keine fehlerhafte Artikeldarstellung geben, die wegen einer fehlgeschlagenen Ersetzung keinen Preis enthält: Im Zweifelsfall bleibt einfach der Standardpreis stehen (Listing 4).

Auch bei sich schnell ändernden Daten wie Lagerbeständen kann es sinnvoll sein, diese dynamisch in die Snippets zu injizieren, wenn das Backend ansonsten die Snippets zu häufig neu generieren müsste. Hier ist eine Abwägung auf Basis der spezifischen Eigenheiten der Anwendung nötig.

Fazit

Die hier vorgestellte Softwarearchitektur ist nicht auf den Einsatz von PHP beschränkt, sondern mit jeder Sprache umsetzbar. Sogar die Verwendung eines Key-Value-Store wie Redis ist im Grunde ein Implementierungsdetail. PHP mit seinem „shared nothing“-Prinzip eignet sich jedoch besonders gut für die Trennung der Verarbeitung von Requests und Änderungen an den zugrunde liegenden Daten nach dem CQRS-Ansatz.

Patterns wie CQRS sind dann besonders vorteilhaft, wenn ein starkes Ungleichgewicht zwischen lesenden und schreibenden Operationen herrscht, da sich jede der beiden Seiten unabhängig von der anderen optimieren lässt. In typischen Webanwendungen gilt es, die Performance bei Lesezugriffen zu verbessern, während schreibende Requests deutlich seltener vorkommen und nicht so zeitkritisch sind.

Zustandsrepräsentationen vorzugenerieren, um das Frontend zu entlasten, ist keine neue Idee: Konzerne wie Facebook oder Twitter nutzen dieses Prinzip schon lange, um User Generated Content an Millionen Nutzer in aller Welt ausliefern zu können.

Die Anwendung von CQRS ist in der Regel mit einem Umdenken verbunden, das einige Zeit brauchen kann. Auch einige gewohnte Entwicklungs-Workflows ändern sich. So werden Änderungen an Template-Dateien erst nach dem Neugenerieren der Snippets im Frontend sichtbar. Dafür erhält man eine modular skalierbare, hochperformante Applikation, die durch eine konsequente Aufteilung in verschiedene Komponenten auch langfristig beherrsch- und wartbar bleibt. (odi) Arne Blankerts ist Mitbegründer und Principal Consultant der thePHP.cc Consulting Company. Er berät kleine und große Unternehmen unter anderem bei Fragen zum Aufbau und Betrieb von hochperformanten PHP-Umgebungen. Sebastian Heuer ist Developer Advocate bei der kartenmacherei GmbH. Zusätzlich unterstützt er als freiberuflicher Consultant Teams bei der Entwicklung langlebiger und wartbarer Software.

[/expand]

Performantere Webapplikationen entwickeln

Performantere Webapplikationen entwickeln

[expand title="mehr lesen…"]

Performantere Webapplikationen entwickeln

Richtig schnell

Sebastian Springer

Egal, ob klassische Website oder moderne Single-Page-App: In Zeiten schnellen Internets sinkt die Bereitschaft der Anwender, auf Webseiten zu warten. Wer zufriedene Nutzer (und gute Rankings bei Google) haben will, ist daher gut beraten, Ladezeiten zu minimieren und seinen JavaScript-Code auf Trab zu bringen.

iX-TRACT

Trotz guter Inhalte haben viele Websites ein Problem: Die Benutzer müssen zu lange auf die erste Interaktionsmöglichkeit warten.

Das kleine Einmaleins der Performanceoptimierung: erkennen, verstehen und beheben. Dabei helfen die Analysewerkzeuge der Browser.

Die aktuellen, hoch optimierten JavaScript-Engines bieten gerade bei Single-Page-Applikationen großes Potenzial für gute Performance.

Haben Besucher einer Seite früher noch Wartezeiten von einer

halben Minute mit der Erklärung „das Internet ist langsam“ verziehen, springen sie mittlerweile schon nach wenigen Sekunden ab. Untersuchungen zeigen: Je länger die Ladezeit einer Seite, desto geringer die Konversionsrate bei E-Commerce-Sites, die Zahl der weiteren Klicks auf der Site, die Zufriedenheit der Besucher und die Wahrscheinlichkeit, dass sie zurückkommen [a]. Auch in das Google-Rating geht die Ladezeit mit ein.

Wenig Geduld mit lahmen Sites ist verstärkt bei Webapplikationen und Websites für Mobilgeräte zu beobachten. Hier erwarten die Benutzer vor allem ein flüssiges und responsives Bedienkonzept.

Will man die Performance einer Webanwendung verbessern, muss man die genutzte Technik verstehen, über Analysewerkzeuge verfügen und wissen, wie man Performanceprobleme behebt. Auch wenn Web-Apps und Websites unterschiedliche Ansätze verfolgen können, die Grundlagen sind doch immer die gleichen: Webserver und Browser kommunizieren über HTTP. Das Frontend ist mit HTML strukturiert, Styling und Animationen erfolgen mit CSS und die Applikationslogik im Frontend ist in JavaScript implementiert. Je nach den Anforderungen an die Applikation muss man in unterschiedlichen Bereichen optimieren. Außerdem gibt es einige Best Practices, die Entwickler kennen sollten.

Die Entwicklerwerkzeuge der Browser helfen dabei, Webapplikationen und -seiten zu analysieren und potenzielle Schwachstellen zu finden. Dieser Artikel konzentriert sich auf die Chrome Developer Tools. Die vorgestellten Funktionen finden sich unter etwas anderer Bezeichnung aber auch in den anderen Browsern.

Der Lebenszyklus einer Webseite

Nach Eingabe der URL in die Adresszeile des Browsers versucht dieser, die angegebene Datei und daraus referenzierte Ressourcen (HTML-, CSS-, JavaScript- und Mediendateien)

herunterzuladen, zu verarbeiten und ihren Inhalt darzustellen. Die Gesamtheit der Aktionen, die der Browser bis zur initialen Darstellung der Applikation unternimmt, heißt Critical Rendering Path.

Die Entwicklertools im Browser bieten zahlreiche Werkzeuge zur Analyse der Performance von Webanwendungen. Eine lange Wartezeit bis zum Beginn der Datenübertragung deutet auf ein Problem beim Server hin (Abb. 1).
Der Download der benötigten Ressourcen ist das erste Hindernis auf dem Weg zu einer performanten Applikation. Der Netzwerk-Tab in den Entwicklertools zeigt eine Übersicht über alle Downloads der aktuellen Webseite. Die initial angefragte HTML-Seite kann auf weitere Ressourcen verweisen, die selbst weitere Ressourcen anfordern können – das führt zu einer Baumstruktur von Downloads mit entsprechenden Abhängigkeiten. So kann ein Stylesheet mit *@import* zusätzliche Stylesheets referenzieren (Abbildung 1).

JavaScript-Code kann weitere Downloads initiieren, wenn er beispielsweise ein neues Image-Element in den DOM-Baum einfügt oder ein *script*-Tag für eine weitere JavaScript-Datei erzeugt. Eine weitere, nicht zu unterschätzende Quelle von Downloads sind Mediendateien, Werbung und Tracking. Alle diese Elemente erzeugen zusätzliche Anfragen beim Server und können ein schnelles Laden der Seite verhindern.

In der Liste der Downloads sieht man neben der angeforderten Ressource und dem Initiator der Verbindung auch den Statuscode des Servers. 200 bedeutet, dass der Server die Anfrage ordnungsgemäß verarbeitet und eine Antwort geschickt hat. Ressourcen mit dem Statuscode 304 konnte der Browser aus seinem Cache laden, sodass sie keine Netzwerklast erzeugt haben. Ebenfalls relevant ist die Anzahl der übermittelten Bytes von Header und Inhalt der Antwort.

Weniger Daten sind schneller geladen

Bei der benötigten Zeit für den Download gibt es zwei Kennzahlen: die Gesamtzeit und den Latenzwert. Erstere gibt die Zeitspanne vom Absenden der Anfrage bis zum kompletten Download der Antwort an. Die Latenz oder Antwortzeit umfasst den Verbindungsaufbau und die Wartezeit bis zum Empfang des ersten Bytes der Antwort. Bei Downloads, die viel Zeit benötigen, lohnt ein Blick in die Details. Eine lange Wartezeit deutet auf ein Serverproblem hin, da hier das Erzeugen der Antwort lange gedauert hat.

Einige Best Practices im Bereich Netzwerk verhindern hier bereits Probleme. Natürlich sollte der Server nur die Informationen an den Client senden, die dieser wirklich benötigt. Die Webanwendung darf nicht zu viele Verbindungen zum Server aufbauen, da der Browser nur eine bestimmte Menge an parallelen Verbindungen zulässt. Weitere Anfragen müssen warten, bis wieder eine Verbindung frei wird, sodass sich die Verbindungen gegenseitig ausbremsen.

Die Lösung liegt im Zusammenfassen von Ressourcen. Werden mehrere JavaScript-Dateien zu einer großen Datei verbunden, fällt der Overhead für den mehrfachen Verbindungsaufbau weg. Auch CSS-Dateien lassen sich zusammenfassen. Für Bilder gibt es eine ähnliche Methode: Sprites kombinieren mehrere Bilder zu einem großen Bild und zeigen dann per CSS nur den relevanten Ausschnitt an.

Sowohl für JavaScript als auch für CSS gibt es Werkzeuge wie UglifyJS und CSSmin, die unnötige Leerzeichen und Kommentare entfernen und so die zu übermittelnde Datenmenge reduzieren. JavaScript-Code lässt sich weiter verkleinern, indem beispielsweise Variablennamen verkürzt werden. UglifyJS kann JavaScript-Dateien so auf die Hälfte bis ein Drittel eindampfen. Bei Bildern lässt sich die Dateigröße reduzieren, indem eine möglichst geringe Auflösung ausgewählt wird. Mit dem *srcset*-Attribut kann man je nach Bildschirmgröße

unterschiedliche Auflösungen ausliefern, ohne mit JavaScript basteln zu müssen.

Schließlich sorgt die Aktivierung von gzip-Komprimierung beim Server für eine weitere Reduzierung der Datenmenge. Die Komprimierung auf dem Server und die Dekomprimierung im Browser erzeugen zwar Rechenlast, die Einsparung auf Netzwerkebene ist allerdings in den meisten Fällen deutlich größer.

Der Critical Rendering Path

Um eine Webseite dazustellen, muss der Browser immer die gleichen grundlegenden Schritte abarbeiten:

– HTML-Datei herunterladen und in das DOM umwandeln;

– anhand der Stylesheets das CSSOM erzeugen;

– DOM und CSSOM zum Render Tree kombinieren;

– für jeden sichtbaren Knoten Position und Größe berechnen;

– die Elemente darstellen.

Für eine schnelle Seite muss jeder Abschnitt dieses Critical Rendering Path optimiert werden.

Das Tortendiagramm zeigt, dass der Browser bei dieser Seite die meiste Zeit mit dem Abarbeiten von JavaScript verbringt (Abb. 2).
Der Timeline-Tab der Entwicklertools zeigt die Ereignisse an, die im Browser auftreten. Zur Analyse des Critical Rendering Path startet man eine Aufnahme, lädt seine Webanwendung und beendet die Aufzeichnung, wenn die Seite komplett geladen ist. Einen ersten Überblick gibt das Tortendiagramm auf dem Summary-Tab unten, das die zeitliche Verteilung der wesentlichen Schritte – Laden des Codes, Ausführen der Skripte, Rendern von DOM und CSSOM, Darstellung der Seite –

zusammenfasst (Abbildung 2).

Das DOM bildet die Hierarchie der HTML-Elemente und deren Abhängigkeiten ab; wie lange sein Aufbau gedauert hat, zeigt das Event „Parse HTML“ auf dem Reiter „Event Log“ an. Wenn dabei viel Zeit vergeht, kann dies auf einen großen DOM-Baum hinweisen. Dann stellt sich die Frage, ob zum Start der Applikation bereits alle Knoten nötig sind oder ob diese nachträglich geladen und per JavaScript eingefügt werden können.

Nicht mehr als unbedingt nötig

Für den ersten Kontakt mit einer Seite ist der sichtbare Bereich entscheidend. Dieser muss schnell geladen sein, damit der Benutzer interagieren kann. Nun kann beispielsweise ein Loading-Indicator signalisieren, dass die restlichen Elemente nachgeladen werden. Ziel sollte stets sein, den Benutzer informiert zu halten – und das lässt sich auf keinen Fall mit der weißen Seite erreichen, die der Browser anzeigt, bis der HTML-Code verarbeitet und der DOM-Baum aufgebaut ist.

Auch während des Ladens und Verarbeitens der CSS-Styles zum CSSOM kann der Browser noch nichts anzeigen. In diese Baumstruktur fließen sowohl die definierten Styles als auch die Standard-Styles des Browsers ein, die sogenannten User Agent Styles. Dabei werden zunächst die allgemeinen Styles für ein Element angewandt und diese dann von spezifischeren Styles überschrieben.

Das Parsen des CSS-Quellcodes und der Aufbau des CSSOM tauchen in der Timeline der Browser-Tools zusammengefasst als Ereignis „Recalculate Style“ auf. Auch hier gilt: Je umfangreicher das Stylesheet, desto länger dauert die Verarbeitung – und desto länger die Wartezeit der Benutzer. Daher sollte das Stylesheet nur die wirklich erforderlichen Angaben enthalten. Außerdem sollte man das Einbinden zusätzlicher Stylesheets über *@import* vermeiden, da das zu zusätzlichen Requests führt. Zeit lässt

sich auch durch die Verwendung von Media-Queries und Media-Types sparen, die angeben, für welches Medium welches Stylesheet gilt. Hier lassen sich sowohl Auflösungen als auch Bildschirmausrichtung oder Ausgabetyp wählen.

In Verbindung mit JavaScript entsteht ein weiteres Performanceproblem. *script*-Tags blockieren das Rendering des DOM, bis der Browser dessen Inhalt ausgeführt hat. Das *async*-Attribut im *script*-Tag signalisiert dem Browser, dass DOM-Erzeugung und JavaScript-Code entkoppelt sind, sodass das JavaScript nicht blockiert.

DOM + CSSOM = Render Tree

Aus DOM und CSSOM erzeugt der Browser den Render Tree mit den sichtbaren Knoten. Nicht sichtbare Elemente wie *meta*-, *script*– und *link*-Tags finden sich hier nicht. Außerdem fallen alle Knoten weg, die per CSS ausgeblendet werden. Anhand der Regeln aus dem CSSOM kann der Browser jetzt die Positionen und Größen der Knoten im Render Tree berechnen. Der Aufbau des Render Tree und die Berechnung der Dimensionen ist in der Timeline unter dem Event „Layout“ zusammengefasst. Für die Optimierung des Render Tree gelten die gleichen Regeln wie für DOM und CSSOM: möglichst schlanke HTML- und CSS-Strukturen, die sich auf das beschränken, was zur initialen Darstellung nötig ist.

Schließlich stellt der Browser die Inhalte der Seite für den Benutzer sichtbar dar. Dieser Vorgang triggert das Event „Paint“. Damit ist die initiale Darstellung abgeschlossen, die Applikation befindet sich im nächsten Abschnitt ihres Lebenszyklus.

Damit sich eine Seite möglichst schnell anfühlt, zeigt man zunächst nur so viele Daten an, dass sich der Besucher weiter mit der Seite beschäftigt. Der Rest lässt sich dann asynchron weiter aufbauen. Wenn beispielsweise am Anfang nur ein Teil des CSS geladen wurde, kann JavaScript-Code jetzt weitere *link*-Tags erzeugen, die zusätzliche Stylesheets nachladen.

Sind rechenintensive JavaScript-Funktionen per Callback an das *load*-Event des Browsers gebunden, laufen sie erst, nachdem die Seite inklusive aller weiteren Ressourcen fertig geladen und aufgebaut ist.

Best Practices

Auch die Ausnutzung des Browser-Cache steigert die Performance einer Webseite. Dazu sind serverseitig die korrekten HTTP-Header zu setzen. Häufig reicht es schon, den Cache-Control-Header auf einen passenden Zeitwert zu setzen. Die Feinheiten beim Thema Caching und Webserver-Konfiguration würden allerdings den Umfang dieses Artikels sprengen, daher müssen wir hier auf weiterführende Literatur verweisen [b].

Eine weitere Methode zur Performancesteigerung sind progressive Web-Apps, also Webapplikationen, die mit Progressive Enhancement arbeiten. Progressive Apps starten mit einer sehr einfachen Darstellung, die auch auf Mobilgeräten und älteren Browsern gut funktioniert. Dieses leichtgewichtige Gerüst wird mit asynchron nachgeladenen Inhalten erweitert, die die Fähigkeiten des konkreten Endgeräts möglichst gut ausnutzen. Der Performancegewinn resultiert aus den geringen initialen Dateigrößen, dem Einsatz des Browser-Cache sowie Service Workern [1].

Analyse und Verbesserung zur Laufzeit

Nachdem die Seite im Browser geladen ist, interagieren die Benutzer mit der Oberfläche. Auch hierbei gibt es Optimierungspotenzial – nicht so sehr für traditionelle Multi-Page-Anwendungen, wohl aber für länger laufende Single-Page-Applikationen, die sich ähnlich wie Desktop-Anwendungen anfühlen. Die Navigation zwischen unterschiedlichen Modulen erfolgt hier, indem lediglich Teile der Anzeige ausgetauscht werden. Das Neuladen kompletter Seiten geschieht nur noch in Ausnahmefällen.

Dadurch lässt sich einiges an Overhead sparen, da der Browser zur Laufzeit nur die Daten laden muss, die dem Benutzer gerade angezeigt werden sollen. Die statischen Bestandteile sind meist bereits geladen; es ist nicht nötig, weitere HTML-, CSS- und JavaScript-Dateien zu laden. Zudem greifen einige Optimierungen der JavaScript-Engine erst, wenn bestimmte Strukturen häufiger verwendet werden. Die folgenden Ausführungen beziehen sich vor allem auf die V8-Engine des Chrome-Browsers. Andere JavaScript-Engines verfügen jedoch über ähnliche Optimierungen.

Listing 1: JavaScript-Optimierung

```
console.time('first');
let arr2 = ['Mary', 'Peter', 'Paul', 'Henry'];
for (let i = 0; i < arr2.length; i++) {
  console.log(arr2[i]);
}
console.timeEnd('first');

console.time('second');
let arr = ['Peter', 'Paul', 'Mary'];
for (let i = 0; i < arr.length; i++) {
  console.log(arr[i]);
}
console.timeEnd('second');
```

Wie deutlich sich diese Optimierung auf die Performance auswirkt, lässt sich an einem sehr einfachen Beispiel demonstrieren. Der Code in Listing 1 iteriert zweimal über ein JavaScript-Array. Die Iteration über das erste Array dauert 1,7 Millisekunden, beim zweiten Array aber nur noch 0,08 Millisekunden.

Hidden Classes beschleunigen den Speicherzugriff

Eine häufige Operation in JavaScript ist der Zugriff auf Eigenschaften und Methoden von Objekten. Dazu muss die

JavaScript-Engine den passenden Speicherbereich im Objekt finden. Da diese Adressauflösung relativ kostspielig ist, kommen sogenannte Hidden Classes zum Einsatz. Hidden Classes sind Verzeichnisse für den Speicher eines Objekts und helfen der Engine bei der Lokalisierung von Eigenschaften des Objekts. Listing 2 zeigt, wie die Hidden Classes für ein Objekt funktionieren.

Listing 2: Hidden Classes

```
class User {
  constructor(firstname, lastname) {
    this.firstname = firstname;
    this.lastname = lastname;
  }
}

console.time('first object');
let klaus = new User('Klaus', 'Müller');
console.log(klaus.firstname);
console.timeEnd('first object');
console.time('second object');
let peter = new User('Peter', 'Meier');
console.log(peter.firstname);
console.timeEnd('second object');
```

Wenn der Code eine erste Instanz *klaus* der Klasse *User* anlegt, erzeugt die JavaScript-Engine eine initiale Hidden Class für das *User*-Objekt. Werden die Eigenschaften *firstname* und *lastname* gesetzt, entstehen zwei weitere Hidden Classes, die auf der ersten basieren und den Speicher-Offset dieser Eigenschaft enthalten. Außerdem erhält die initiale Hidden Class Verweise auf die weiteren Hidden Classes. Wenn man jetzt, wie im Beispiel, auf eine Eigenschaft des Objekts zugreift, findet die Engine mithilfe der Hidden Classes die Speicherstelle wesentlich schneller.

Beim Anlegen des zweiten *User*-Objekts *peter* erkennt die JavaScript-Engine, dass es sich um den gleichen Objekttyp handelt. Sie verwendet die gleichen Hidden Classes wie beim

ersten Objekt, sodass keine neuen Hidden Classes mehr erzeugt werden müssen. Der Overhead für das Anlegen der Hidden Class, der beim ersten Objekt anfiel, entfällt also, das Objekt ist schneller angelegt. Entwickler sollten daher für ähnliche Objekte eine gemeinsame Klasse definieren, um der Engine die Optimierung der Applikation zu ermöglichen. Als Nebeneffekt wird so auch der Quellcode besser strukturiert.

Garbage Collector

Die relativ lange Laufzeit von Single-Page-Applikationen ohne Neuladen von Seiten hat allerdings nicht nur Vorteile. Die Objekte der Applikation können – je nachdem, in welchem Gültigkeitsbereich sie sich befinden – mitunter recht lange im Speicher liegen. Ein stetig ansteigender Speicherverbrauch führt dazu, dass die Applikation immer langsamer wird und irgendwann nicht mehr benutzbar ist – der Anwender muss die Browser-Sitzung beenden.

Web-Apps tendieren dazu, mit der Zeit immer mehr Speicher zu belegen. Von Zeit zu Zeit räumt der Garbage Collector nicht mehr benötigte Objekte aus dem Speicher (Abb. 3).
Die Speichernutzung lässt sich ebenfalls mit den Entwicklertools des Browsers überwachen. Einen ersten Überblick liefert die Timeline, die neben den bereits erwähnten Events auch den Speicherverbrauch aufzeichnet und als hellblaue Fläche darstellt (Abbildung 3). Meist wächst sie im zeitlichen Verlauf, bricht aber immer wieder ein.

Dafür ist der Garbage Collector der JavaScript-Engine verantwortlich, der nicht mehr benötigte Objekte – solche, auf die keine Referenz mehr verweist – aus dem Speicher entfernt. Referenzen lassen sich explizit löschen, beispielsweise mit dem *delete*-Operator, und werden automatisch entfernt, wenn der Gültigkeitsbereich eines Objekts verlassen wird: Eine mit *var* oder *let* in einer Funktion definierte Variable existiert nur innerhalb einer Funktion oder eines Blocks. Sobald die

Abarbeitung der Funktion beendet ist, wird der dafür allozierte Speicher wieder freigegeben.

Der Garbage Collector wird immer wieder aktiv; seine Aktivität ist in der Timeline als „Minor GC“ und „Major GC“ verzeichnet. Wie die unterschiedlichen Bezeichnungen nahelegen, gibt es verschiedene Arten der Garbage Collection. Die V8-Engine von Chrome teilt den Speicher in zwei Bereiche auf: einen für kurzlebige und einen für langlebige Objekte. Neue Objekte landen zunächst im Speicherbereich für kurzlebige Objekte, der in einzelne Sektionen unterteilt ist.

Ist eine Sektion voll, verschiebt die JavaScript-Engine die noch verwendeten Objekte in eine neue Sektion und leert die vollgelaufene Sektion. Wird ein Objekt zum zweiten Mal verschoben, verlagert es die Engine in den Speicher für langlebige Objekte. Dieser Vorgang taucht als Minor GC im Event Log auf. Der auch als Scavenge bezeichnete Minor-GC-Durchlauf eignet sich gut für kleine Speicherbereiche, da er zwar sehr schnell ist, aber relativ viel Speicher benötigt: Kurzzeitig werden ja der alte und der neue Speicherbereich belegt.

Der Speicher für langlebige Objekte wird durch einen sogenannten Mark-and-Sweep-Algorithmus aufgeräumt: Noch verwendete Objekte werden markiert, alle Objekte ohne Markierung anschließend gelöscht. Diese Major Garbage Collection passiert wesentlich seltener als die Minor GC, da sie die Ausführung des Skripts kurz unterbricht. Das führt bei JavaScript-Animationen zu unschönem Ruckeln und unruhigem Verlauf, weshalb die Garbage Collection Cycles möglichst in die Idle Time der CPU verlagert werden.

Speicher sparen

Wenn möglich sollten Anwendungen Objekte wiederverwenden, um dem Garbage Collector Arbeit zu ersparen, und Referenzen auf Objekte nur solange halten wie nötig, damit der Speicher

schnell wieder freigegeben werden kann. Nicht freigegebener Speicher führt zu einem Memory Leak. Einer der häufigsten Gründe ist die Verwendung von (bewusst oder versehentlich erzeugten) globalen Variablen: Da diese überall in der Applikation erreichbar sind, kann der Garbage Collector ihren Speicher nicht freigeben.

DOM-Knoten sind eine weitere Ursache für Speicherverlust. Normalerweise werden nicht mehr benötigte Knoten gelöscht. Allerdings können Applikationen Referenzen auf DOM-Knoten in Variablen speichern, um sie beispielsweise aus dem DOM auszuhängen, zu bearbeiten und an einer anderen Stelle wieder einzuhängen. Bleiben die Variablen allerdings nach dieser Operation gültig, kann der durch die DOM-Knoten belegte Speicher nicht freigegeben werden. Eine ebenfalls nicht ganz offensichtliche Ursache für Memory Leaks sind verkettete Listen von Referenzen. Listing 3 gibt hierfür ein Beispiel.

Listing 3: Verkettete Listen (HTML)

```
<html>
  <head>
    <script>
      let value;

      function leak() {
        let prevValue = value;
        function debug() {
          if (prevValue) {
            console.log('already hava a value');
          }
        }
        value = {
          val: new Array(10000).join('Memory'),
          getVal: function () {
            return this.val;
          }
        }
      }
```

```
      let count = 0;
      let interval = setInterval(() => {
        leak();
        if (count > 20) {
          clearInterval(interval);
        }
      }, 500);
    </script>
  </head>

  <body>
    <p>A real memory leak!</p>
  </body>
</html>
```

Die globale Variable *value* soll als Zwischenspeicher dienen. Die Funktion *leak()* speichert den aktuellen Inhalt von *value* in der temporären Variablen *prevValue*, auf die wiederum die Hilfsfunktion *debug()* zugreift. Letzteres sorgt dafür, dass der Garbage Collector den Speicher nicht ohne Weiteres freigeben kann. Schließlich wird der globalen Variablen in *leak()* ein neuer Wert zugewiesen.

Wird diese Funktion nun in einer Schleife oder wie im Beispiel über einen Timer aufgerufen, sieht man in der Timeline einen stetigen Anstieg der Speichernutzung. Ohne ein Abbruchkriterium (Variable *count*) würde der Speicher mit der Zeit volllaufen und der Browser abstürzen.

Die Grafik des Speicherverbrauchs ist ein erster Schritt in der Analyse von Memory Leaks. Genauere Informationen bieten die Speicherprofile, die sich im Profiles-Tab anfertigen lassen. Sie zeigen, welche Objekte wie viel Speicherplatz belegen und welche Funktion für die Belegung verantwortlich ist. Mit diesen Informationen lässt sich der Quellcode so optimieren, dass der Speicher korrekt freigegeben wird.

Performante Animationen

Moderne Webapplikationen nutzen Animationen, um die Aufmerksamkeit des Benutzers an eine bestimmte Stelle zu lenken und ihn bei der Interaktion zu unterstützen. JavaScript-Code kann Bewegungen über Modifikationen des CSSOM erzeugen: Erhöht man den *left*-Wert eines Objekts kontinuierlich, wandert es von links nach rechts über den Bildschirm. Nachteil: Die Rendering-Engine kann solche Animationen nicht optimieren, da sie von außerhalb – mit JavaScript – erzeugt werden. Sie belasten daher die CPU.

Listing 4: CSS-Animation

```
<style type="text/css">
.outer {
  position: relative;
  width: 510px;
  height: 10px;
  border: 1px solid black;
}
.inner {
  transition: all 5s;
  position: absolute;
  top: 0;
  left: 0;
  width: 10px;
  height: 10px;
  background-color: red;
}
.move {
  transform: translateX(500px);
}
</style>

<div id="outer" class="outer">
  <div id="inner" class="inner"></div>
</div>

<script>
  document.getElementById('outer').onclick =  function () {
```

```
    document.getElementById('inner').classList.add ('move');
  };
</script>
```

Wesentlich eleganter und auch performanter ist die Verwendung von CSS-Animationen. Für eine so einfache Animation wie das Verschieben eines Elements reicht eine Transformation aus. Die Angabe *transform: translateX(500px)* sorgt dafür, dass das Element durch die Veränderung seiner x-Koordinate nach rechts verschoben wird. Die Animation wird dabei nicht durch die CPU ausgeführt, sondern auf der Grafikkarte. CSS-Animationen sind daher ruckelfreier und um einiges ressourcenschonender als ihre Gegenstücke in JavaScript. Listing 4 zeigt ein Beispiel für eine solche CSS-Animation.

Komplexere Animationen sollte man allerdings nicht mit Transitionen umsetzen, sondern lieber auf *@keyframes* zurückgreifen. Damit lassen sich wesentlich mehr grafische Aspekte steuern. Das bedeutet allerdings auch mehr an Aufwand beim Programmieren der Animation. Es gibt jedoch Generatoren, die den Quellcode fürs Stylesheet erzeugen, zum Beispiel cssanimate.com.

Reflow und Repaint

Jede Manipulation von DOM und CSSOM mit JavaScript führt dazu, dass der Render Tree angepasst und die Seite teilweise neu dargestellt werden muss. Einzelne Eingriffe in diese Strukturen stellen kein Problem dar. Wenn man jedoch in einer Schleife neue DOM-Knoten einfügt, müssen in jedem Schleifendurchlauf Teile der Seite neu dargestellt werden. Man spricht von Reflow (Neuberechnung des Render Tree sowie der Abmessungen und Position der Knoten) und Repaint (Berechnung der visuellen Eigenschaften und Darstellung der Elemente).

Wie auch der initiale Seitenaufbau tauchen Reflows in der Timeline als Layout-Events auf. Jeder Reflow zieht einen Repaint nach sich. Nicht nur das Einfügen neuer Knoten in das

DOM, sondern auch zahlreiche weitere Aktionen wie das Verändern des Inhalts der Seite, das Ändern der Fenstergröße, Scrollen durch den Benutzer und sogar das Auslesen und Schreiben bestimmter CSS-Eigenschaften lösen einen Reflow aus. Unverhältnismäßig viele Layout- und Paint-Events in der Timeline sind ein Hinweis auf Performanceengpässe in einer Anwendung.

Einige Best Practices können die Zahl der Reflows und Repaints reduzieren. Statt CSS-Eigenschaften einzeln zu ändern, fasst man besser mehrere Änderungen in CSS-Klassen zusammen. Die Browser-Engine kann so die Layoutänderungen optimieren – ein Reflow reicht für mehrere Änderungen aus. Für Anpassungen am DOM sollte man den betroffenen Teil des Baums aushängen, die Änderungen abseits des sichtbaren Bereichs durchführen und den veränderten Subbaum in einer Aktion wieder einhängen. So wird lediglich je ein Reflow für das Ein- und Aushängen ausgelöst. Beim Bearbeiten der Styles von Elementen mit JavaScript empfiehlt es sich, diese in Variablen zu speichern, um das wiederholte Auslesen von Werten zu vermeiden.

Fazit

Das Thema Performance in Webapplikationen ist vielfältig, sowohl was die Problemstellungen angeht als auch hinsichtlich der Lösungsansätze und Hilfsmittel. Es empfiehlt sich, gängigen Best Practices zu folgen, von denen einige hier vorgestellt wurden.

Allerdings sollten Entwickler ihre Webanwendung nicht zu früh einseitig auf Performance optimieren: Eine verlässliche Analyse lässt sich erst im Betrieb mit realen Daten durchführen. Eine vorzeitige Optimierung bedeutet möglicherweise viel Aufwand an der falschen Stelle. Es kann auch durchaus in Ordnung sein, bei selten verwendeten Features nicht die letzte Millisekunde herauszuholen: Letztlich muss man immer Kosten und Nutzen von Verbesserungen abwägen. (odi) Sebastian Springer versucht als Dozent für JavaScript,

Sprecher auf zahlreichen Konferenzen und Autor, die Begeisterung für professionelle Entwicklung mit JavaScript zu wecken.

Literatur

- [1] Sebastian Springer; Dienst am Kunden; Progressive Web Apps: Service Worker und mehr; iX 6/2016, S. 120

Richtig schnell

- Caching
- Site Performance For Webmasters

[/expand]

RESTful APIs mit Python und Flask entwickeln

RESTful APIs mit Python und Flask entwickeln

[expand title="mehr lesen…"]

RESTful APIs mit Python und Flask entwickeln

Kommunikationshelfer

Sebastian Bindick

Microservices kommunizieren über standardisierte APIs. Die Python-Frameworks Flask und Flask-RESTPlus ermöglichen Entwicklern, REST-APIs einfach zu erstellen.

-tract

- Unternehmen setzen verstärkt Microservices ein, die über APIs in Kontakt stehen.
- Für die Koordination der einzelnen Dienste untereinander verwenden Entwickler REST.
- Mit Flask und seiner Erweiterung Flask-RESTPlus lassen sich REST-APIs für Enterprise-Anwendungen entwickeln.
- Ein Anwendungsbeispiel zeigt die Funktionsweise von RESTful Webservices mit Flask-RESTPlus: Aufbau und Struktur der Anwendung, Ressourcen und Routing, Validieren von Modellen, Fehlerbehandlung, API-Testing, Umgebungen sowie Dokumentation.

Qualitativ hochwertige APIs sind so wichtig wie noch nie, denn immer mehr Unternehmen setzen auf Microservices, die über Schnittstellen miteinander kommunizieren. Das Zusammenspiel der einzelnen Dienste bestimmt dabei maßgeblich die Qualität des gesamten Systems. Für diesen Zweck nutzen Softwareentwickler mittlerweile vorwiegend den Architekturstil REST (Representational State Transfer). Dieser orientiert sich an den Prinzipien des World Wide Web und beschreibt die leichtgewichtige Kommunikation zwischen Services im Netzwerk.

Für die Programmiersprache Python stehen mit Flask und Flask-RESTPlus umfangreiche Frameworks zur Verfügung, mit denen das

Entwickeln von REST-APIs leicht gelingt. Dieser Artikel stellt anhand einer Beispielanwendung zur Verwaltung von Brettspielsammlungen vor, wie RESTful Webservices auf Basis von Flask-RESTPlus funktionieren.

Flask ist ein schlankes, in Python geschriebenes Webframework, das einen einfachen Einstieg ermöglicht und sich zudem gut erweitern lässt. Im Kern enthält es Basisfunktionen, die sich durch Erweiterungen etwa für Authentifizierung, Object-Relational Mapping, Caching oder RESTful APIs ergänzen lassen. Flask macht hierbei und bei der Projektstruktur wenig Vorgaben und lässt Entwicklern viele Freiheiten. Es zählt mittlerweile zu den populärsten Python-Webframeworks und Unternehmen wie Pinterest, LinkedIn, Netflix und Red Hat setzen es ein (siehe ix.de/zhjd).

Die Installation von Flask erfolgt über das Python-Paketverwaltungsprogramm pip mit dem Kommando pip install Flask. Listing 1 zeigt eine einfache Flask-Anwendung, die nach dem Import der Flask-Bibliothek eine Instanz der Applikation mit dem Namen der App als Argument erstellt. Der Decorator @app.route erzeugt die Route /hello. Jeder Routenaufruf führt die Funktion hello() aus, die den String "Hello World!" zurückgibt.

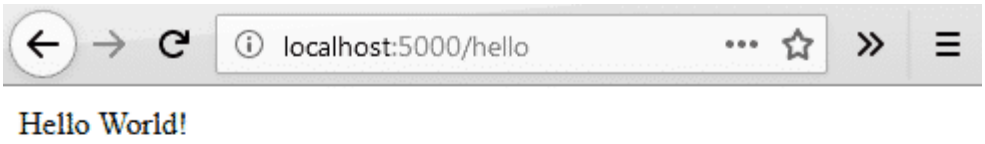
Listing 1: Einfacher Flask-Webservice (hello.py)

```
from flask import Flask

app = Flask("my hello app")

@app.route("/hello")
def hello():
    return "Hello World!"

app.run()
```

Das Ausführen der Datei hello.py auf der Konsole startet den Webservice mit folgender Ausgabe:

```
python hello.py
* Serving Flask app "my hello app"
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Flask verwendet hierzu standardmäßig einen integrierten einfachen Entwicklungsserver, der nicht für produktive Umgebungen geeignet ist. Beim Aufruf des Links im Browser erscheint der String "Hello World!" (Abbildung 1).



Die Anwendung lässt sich im Browser unter http://localhost:5000/hello aufrufen (Abb. 1).

In Kooperation

Flask-RESTPlus ist eine Erweiterung für das Flask-Framework, mit der sich REST-APIs einfach entwickeln lassen. Sie stellt eine Reihe von Werkzeugen zur Beschreibung der API bereit und generiert automatisch einen Endpunkt für Swagger UI – ein Framework für die interaktive Dokumentation und Visualisierung von APIs.

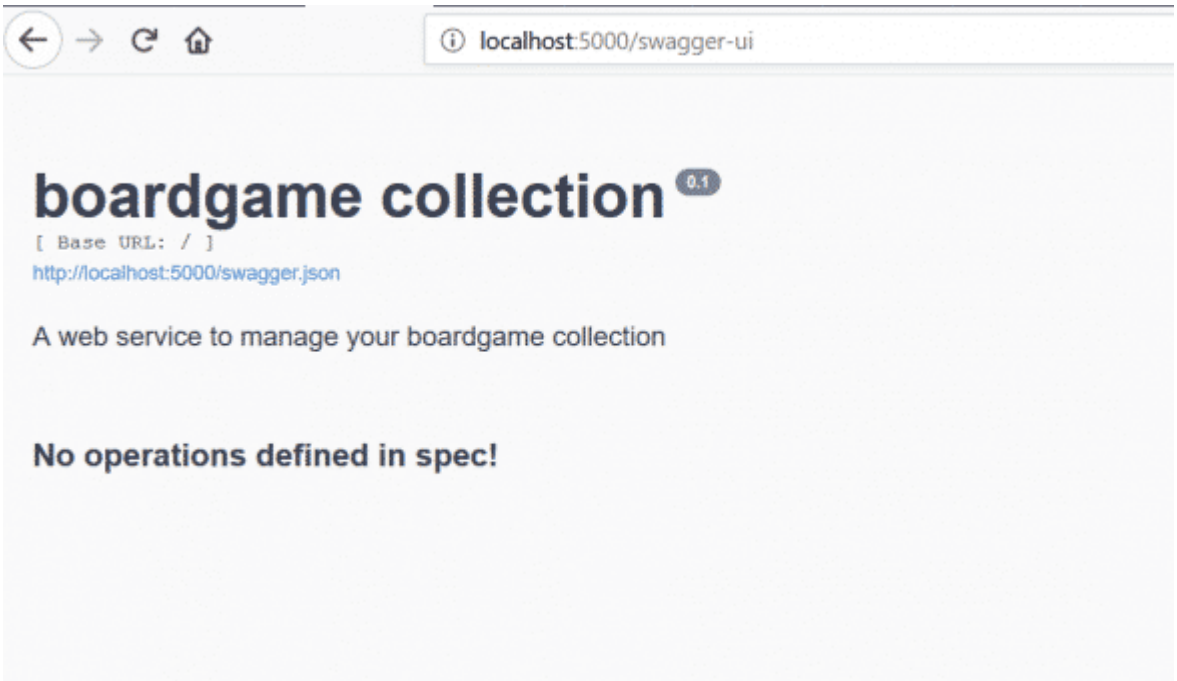
Zum Installieren startet man pip mit dem Kommando pip install flask-restplus. Der Haupteinstiegspunkt einer Flask-RESTPlus-Anwendung ist die Klasse Api, die sich mit einer Flask-Applikation initialisieren lässt (Listing 2). Die Konfiguration von Api wie Version, Titel, Beschreibung, Pfad zur Dokumentation, Kontaktinformationen und Lizenz erfolgt über den Konstruktor. Als Beispiel dient hier ein Webservice zum Verwalten einer Brettspielsammlung (boardgame collection),

die in den folgenden Abschnitten um weitere Funktionalitäten ergänzt wird.

Listing 2: Haupteinstiegspunkt der API (main.py)

```
from flask import Flask
from flask_restplus import Api

app = Flask("boardgame collection")
api = Api(app,
          version='0.1',
          title='boardgame collection',
           description='A web service to manage your boardgame
collection',
          doc='swagger-ui')

if __name__ == '__main__':
    app.run(debug=True, port=5000)
```

Der Aufruf der run-Methode startet den Service. Die automatisch generierte API-Dokumentation lässt sich im Browser aufrufen, die Informationen hierzu stammen aus der Konfiguration (Abbildung 2).



Swagger UI zeigt die API-Dokumentation unter http://localhost:5000/swagger-ui (Abb. 2).
Zur besseren Übersicht ist die Anwendung in einer Baumstruktur

entsprechend ihren funktionalen Aufgaben organisiert (Listing 3). Hierzu legt man für jede API-Ressource eine Datei im Paket resources an. Die zugehörigen fachlichen Modellbeschreibungen finden sich im Paket models in jeweils eigenen Dateien. Alle Tests der API sind im Paket tests abgelegt. Die folgenden Abschnitte erläutern Ressourcen, Modelle und Tests für die App boardgame collection.

Listing 3: Struktur und Aufbau der Anwendung

```
├── resources
│       └── __init__.py
│       └── boardgames.py
├── models
│       └── __init__.py
│       └── boardgame.py
│       └── boardgame-expansion.py
├── tests
│       └── __init__.py
│       └── test_boardgames_api.py
├── environments.py
└── main.py
```

Die spezifische Konfiguration verschiedener Zielumgebungen wie Produktiv- oder Entwicklungssystem erfolgt über environments.py (Listing 4).

Listing 4: Verschiedene Zielumgebungen konfigurieren (environments.py)

```
import os

class DevelopmentConfig:
    port = 5000
    debug = True
    log_path = "boardgames.log"
    documentation_path = "/swagger-ui"
    ...

class ProductionConfig:
    port = 8000
```

```
    debug = False
    log_path = "boardgames.log"
    documentation_path = None
    ...

configurations = {
    "dev":  DevelopmentConfig,
    "prod": ProductionConfig }

environment = os.environ.get("BG_CONFIG", "dev")
config = configurations[environment]
```

Hierdurch lässt sich das jeweilige Einstellungsset für Port oder Debug-Modus über eine Umgebungsvariable von außen für unterschiedliche Deployments steuern. Ein Python Dictionary speichert die Konfigurationen ProductionConfig und DevelopmentConfig. Das Programm liest beim Importieren der -environments.py die Umgebungsvariable BG_CONFIG ein und setzt die entsprechende Konfiguration für die Zielumgebung. Der Import der Konfiguration mit

```
from environments import config
```

stellt die Einstellungen in Komponenten wie der main-Methode bereit (Listing 5). Die Methode run erhält die Einstellungen für den Debug-Modus und den Port für die Zielumgebung. Außerdem übernimmt die Anwendung den Pfad zu Swagger UI aus config und initialisiert einen Python-Standard-Logger mit dem entsprechenden Pfad aus der Konfiguration. Vor dem Ausführen der Anwendung lässt sich die Zielumgebung über die Kommando-zeile setzen (unter Windows mit set BG_CONFIG=prod; unter Linux mit ENV BG_CONFIG=prod).

Für den produktiven Betrieb von Flask-Anwendungen eignen sich diverse Webserver wie Apache Web Server, nginx, Cherokee und Gunicorn, da Flask auf die WSGI-Standardschnittstelle (Web Server Gateway Interface) für die Kommunikation zwischen Webserver und Webframework setzt. Für Details zur Skalierung und zum Betrieb produktiver Flask-RESTPlus-Anwendungen siehe ix.de/zhjd.

Listing 5: Haupteinstiegspunkt der API um Konfiguration aus environments ergänzt (main.py)

```
from flask import Flask
from flask_restplus import Api
from environments import config
import logging

app = Flask("boardgame collection")
api = Api(app,
          ...
          doc=config.documentation_path)

if __name__ == '__main__':
         logging.basicConfig(filename=config.log_path,
level=logging.DEBUG)
    logging.info("start boardgame collection service ...")

    app.run(debug=config.debug, port=config.port)
```

Ressourcen und Routing

Zentraler Bestandteil jeder REST-API sind Ressourcen. Hierzu stellt Flask-RESTPlus eine entsprechende Basisklasse (Resource) zur Verfügung, die das Routing für verschiedene HTTP-Methoden (GET, PUT, DELETE) für einen gegebenen Endpunkt (URL) ermöglicht.

Die Ressource Boardgame in Listing 6, abgeleitet von Resource, stellt die Basis-CRUD-Operationen Create, Read, Update und Delete für ein einzelnes Brettspiel bereit. Der Decorator @api.route('/<id>') weist die Route zu. Mit der Ressource BoardgameList lässt sich auf Brettspiellisten zugreifen. Beide Ressourcen fasst man in einem gemeinsamen Namespace /boardgames zusammen. Namespaces gruppieren in Flask-RESTPlus Ressourcen unter einem Endpunkt; mit der Methode add_namespace(...) fügt man sie einer Api hinzu (Listing 7).

Listing 6: Boardgame-Ressource (resources/boardgames.py)

```
from flask_restplus import Namespace, Resource
from flask import abort

bg_collection = {} # boardgames stored in memory
api = Namespace('boardgames', description='boardgame related
operations')

@api.route('/<id>')
class Boardgame(Resource):
    def get(self, id):
      '''Gets a boardgame by id'''
      if id in bg_collection:
        return bg_collection[id], HTTPStatus.OK
      else:
            abort(HTTPStatus.NOT_FOUND, 'Boardgame {0} not
found.' .format(id))

    def delete(self, id):
      '''Removes a boardgame from the collection'''
      if id in bg_collection:
        del bg_collection[id]
        return '', HTTPStatus.NO_CONTENT
      else:
        abort(HTTPStatus.NOT_FOUND, 'Boardgame {0} not found.'
.format(id))

    def put(self, id):
      '''Updates a boardgame'''
      boardgame = request.json
      if id in bg_collection:
        bg_collection[id] = boardgame
        return boardgame, HTTPStatus.OK
      else:
        abort(HTTPStatus.NOT_FOUND, 'Boardgame {0} not found.'
.format(id))

@api.route('')
class BoardgameList(Resource):
```

```
    def get(self):
      '''Lists all boardgames in collection'''
      bg_collection_list = list(bg_collection.values())
      return bg_collection_list, HTTPStatus.OK

    def post(self):
      '''Adds a boardgame to collection'''
      boardgame = request.json
      bg_collection[boardgame['id']] = boardgame
      return boardgame, HTTPStatus.CREATED
```

Listing 7: Namespace boardgame registrieren (main.py)

```
from flask import Flask
from flask_restplus import Api
from app.resources.boardgames import api as boardgames_api_namespace

app = Flask("boardgame collection")
api = Api(...)

api.add_namespace(boardgames_api_namespace)
...
```

Somit ergeben sich die folgenden URLs für den Brettspiel-Webservice:

- GET http://localhost:5000/boardgames/1 gibt das Brettspiel mit der id 1 zurück.
- DELETE http://localhost:5000/boardgames/1 löscht das Brettspiel mit der id 1.
- PUT http://localhost:5000/boardgames/1 aktualisiert das Brettspiel mit der id 1.
- GET http://localhost:5000/boardgames gibt die Liste von Brettspielen zurück.
- POST http://localhost:5000/boardgames erzeugt ein neues Brettspiel in der Liste.

Der Aufruf dieser URLs etwa über curl http://localhost:5000/boardgames/1 -X GET triggert die

entsprechende Methode in der Python-Ressource (wie get der Klasse Boardgame). Die Methode verarbeitet dann den Input, generiert eine Antwort und schickt diese als HTTP-Response an den Aufrufer zurück. Die Methode erhält die Inputparameter wie id zur Identifikation eines Objekts aus dem URL-Pfad als Parameter.

Für die Antwortnachricht wandelt die Methode die Rückgabewerte automatisch in ein Flask-Response-Objekt um. Sie kann bis zu drei Rückgabewerte enthalten. An erster Stelle steht der im HTTP Response Body verwendete Content. Optional lassen sich über die weiteren Rückgabewerte der HTTP-Statuscode – als Default wird hier 200 OK verwendet – und eigene HTTP Response Header setzen. Die Hilfsmethode abort generiert im Fehlerfall eine HTTPException abhängig vom jeweiligen HTTP-Statuscode.

Das Python Dictionary bg_collection hält die Daten, hier Brettspiele, im Speicher. Auf das Anbinden einer Datenbank wird zugunsten der besseren Lesbarkeit verzichtet. Somit lassen sich jetzt Brettspiele über die API der Sammlung hinzufügen und abrufen (Listing 8).

Listing 8: Brettspiel erstellen und Sammlung über curl ausgeben

```
$ curl -X POST "http://localhost:5000/boardgames" -d "{ "id":
"1",
"name": "Wingspan", "designer": "Elizabeth Hargrave",
"playing_time": "60 Min",
"rating": 8.1, "expansions": [{"name": "European Expansion",
"rating": "8.5"}]}"

$ curl -X GET "http://localhost:5000/boardgames"
[{
    "id": "1",
    "name": "Wingspan",
    "designer": "Elizabeth Hargrave",
    "playing_time": "60 Min",
    "rating": 8.1,
    "expansions":
```

```
      [{
        "name": "European Expansion",
        "rating": 8.5
      }]
}]
```

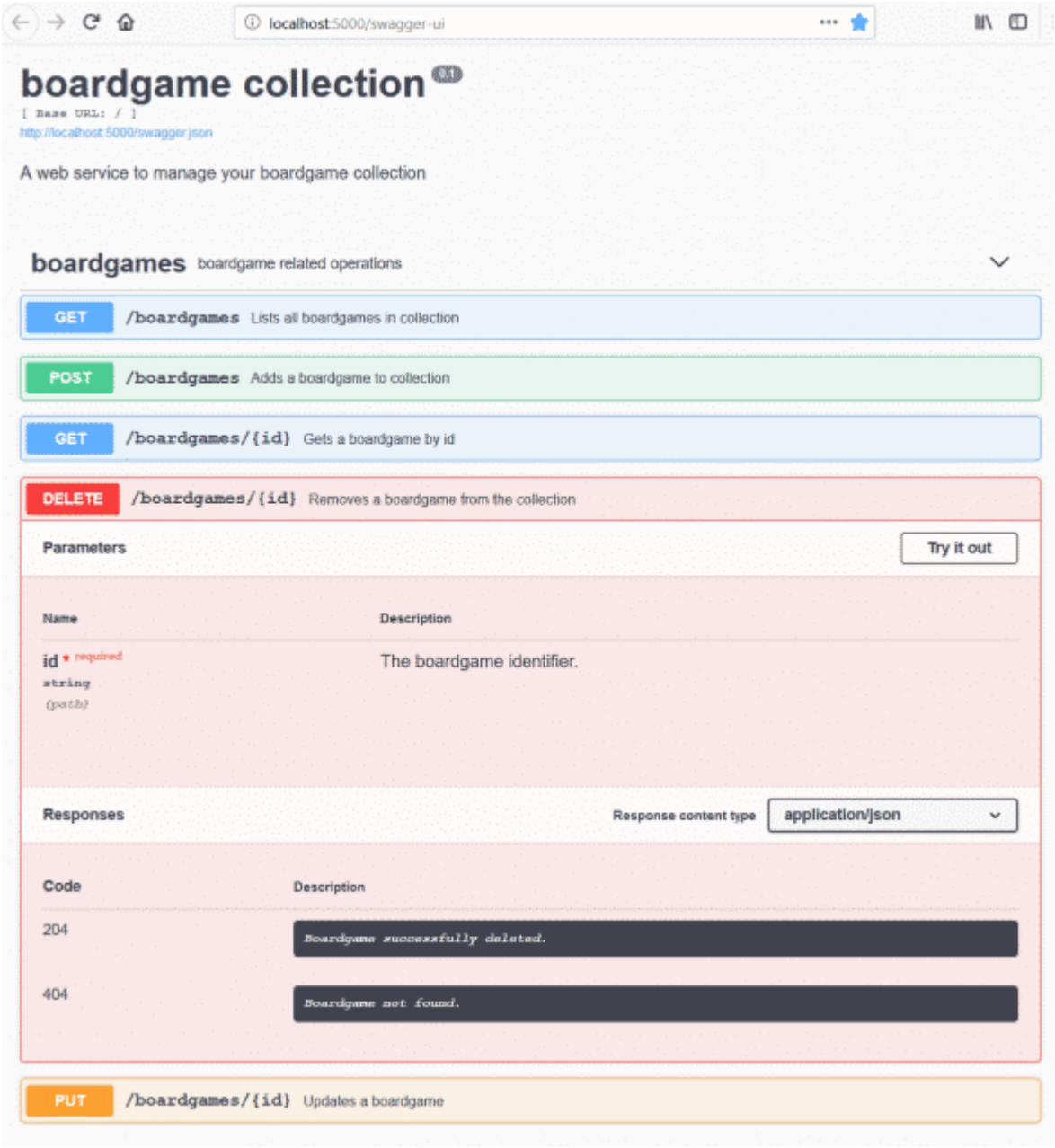
Kleine Helferlein

Verschiedene Werkzeuge von Flask-RESTPlus dienen dazu, die automatisch generierte Dokumentation auf Basis der OpenAPI Specification individuell anzupassen. So ermöglicht es der Decorator @api.response(), mögliche Antworten einer Ressource oder Methode festzulegen und sie in der Dokumentation unter der Response der jeweiligen Route aufzuführen (Listing 9 und Abbildung 3). Alternativ kann die Dokumentation der Parameter aus dem URL-Pfad über den Decorator @api.param() erfolgen und damit global an der Ressource (gilt dann für alle Methoden) oder individuell je Methode.

Listing 9: Boardgame-Ressource um Dokumentation erweitert (resources/boardgames.py)

```
...
@api.route('/<id>')
@api.param('id', 'The boardgame identifier.')
@api.response(404, 'Boardgame not found.')
class Boardgame(Resource):
    ...
    @api.response(204, 'Boardgame successfully deleted.')
    def delete(self, id):
      ...

    @api.response(200, 'Boardgame successfully updated.')
    def put(self, id):
      ...

@api.route('')
class BoardgameList(Resource):
    @api.response(200, 'All boardgames successfully fetched!')
    def get(self):
```

```
    ...
@api.response(201, 'Boardgame successfully created.')
def post(self):
    ...
```



Swagger UI stellt das Routing für die Ressource boardgames dar (Abb. 3).

Flask-RESTPlus ermöglicht es, die Struktur der Payload von HTTP-Anfragen und Antworten durch Modelle zu beschreiben. Das Framework rendert auf Basis dieser Modelle die HTTP-Payload automatisch aus den internen Daten wie ORM-Modellen und eigenen Klassen. Über das als Marshalling bezeichnete Verfahren lässt sich steuern, welche Daten die API in welchem Format kommuniziert.

Mit der Factory model() lassen sich Modelle auf dem Namespace der API erstellen und registrieren. Listing 10 legt durch api_namespace.model('Boardgame',{...}) ein Modell für das Objekt Brettspiel (boardgame) an. Ein Dictionary beschreibt das Modell. Jeder Schlüssel im Dictionary repräsentiert hierbei ein in der HTTP-Payload gerendertes Attribut. Der zugehörige Wert definiert das Format des zu rendernden Attributs, beschrieben durch das Modul fields. Flask-RESTPlus stellt unter diesem Modul verschiedene Klassen für Datentypen wie Zeichenketten (String), Fließkommazahlen (Float), Datum (DateTime) und Objektlisten (List) bereit. Die Datentypen besitzen eine Reihe optionaler Argumente, die das Feld detailliert beschreiben. So lassen sich zum Beispiel eine Ober- und Untergrenze für einen Wert festlegen, Defaultwerte vorgeben, eine Beschreibung (description) ergänzen und ein Wert als erforderlich (required) definieren.

Mit Modellen arbeiten

Listing 10: Boardgame-Modell (model/boardgame.py)

```
from flask_restplus import fields
from models.expansion import create_expansion_model

def create_boardgame_model(api):
    boardgame_model = api.model('Boardgame', {
        'id': fields.String(description='unique boardgame
identifier',
                            required=True),
      'name': fields.String(description='boardgame name',
                            min_length=3,
                            max_length=128,
                            required=True),
         'designer': fields.String(description='boardgame
designer',
                                   required=True),
        'playing_time': fields.String(description='aprox.
playing time',
                                      enum=["15 Min", "30 Min",
```

```
"60 Min"],
                                    required=True),
      'rating': fields.Float(description='rating [0 to 10]',
                             min=0.0,
                             max=10.0,
                             required=False),
                                                'expansions':
fields.List(fields.Nested(create_expansion_model(api),
description='list of expansions',
                                                    required=False))
    })

    return boardgame_model
```

Die Modelle der hier vorgestellten Anwendung zur Verwaltung einer Brettspielsammlung sind im Paket models abgelegt. Listing 10 und 11 erstellen die Modelle Brettspiel (boardgame) und Brettspielerweiterung (expansion). Ein Brettspiel erhält das Attribut expansions, das eine Liste der zugehörigen Erweiterungen speichert. Flask-RESTPlus stellt die Klasse fields.Nested zum Anlegen verschachtelter Strukturen bereit, der sich zuvor angelegte Modelle wie expansion_model übergeben lassen. Die so definierten Modelle sind in der API-Dokumentation dargestellt (Abbildung 4).

Listing 11: Expansion-Modell (model/expansion.py)

```
from flask_restplus import fields

def create_expansion_model(api):
    expansion_model = api.model('Expansion', {
       'name': fields.String(description='boardgame expansion
name',
                             min_length=3,
                             max_length=128,
                             required=True),
      'rating': fields.Float(description='rating [0 to 10]',
                             min=0.0,
                             max=10.0,
                             required=False)
    })
```

```
    return expansion_model
```

Swagger UI zeigt die Modellbeschreibung für Boardgame und Expansion (Abb. 4).
Damit sich diese Modelle innerhalb einer Ressource verwenden lassen, stellt Flask-RESTPlus verschiedene Decorators zur Verfügung. Der Decorator @api.marshal_with(model) definiert für eine Ressourcenmethode, dass die Daten in der HTTP-Antwort automatisch auf Basis des entsprechenden Modells gerendert werden (Listing 12). @api.marshal_list_with(model) rendert die Rückgabe als Objektliste des Modells.

Listing 12: Boardgame-Ressource um Modelle ergänzt (resources/boardgames.py)

```
...
from models.boardgame import create_boardgame_model

bg_collection = {} # boardgames stored in memory
api = Namespace('boardgames', description='boardgame related operations')

boardgame = create_boardgame_model(api)
```

```
@api.route('/<id>')
@api.param('id', 'The boardgame identifier.')
@api.response(404, 'Boardgame not found.')
class Boardgame(Resource):
    @api.marshal_with(boardgame)
    def get(self, id):
      ...

    @api.response(200, 'Boardgame successfully updated.')
    @api.expect(boardgame, validate=True)
    def put(self, id):
      ...


@api.route('')
class BoardgameList(Resource):
    @api.response(200, 'All boardgames successfully fetched!')
    @api.marshal_list_with(boardgame)
    def get(self):
      ...

    @api.response(201, 'Boardgame successfully created.')
    @api.expect(boardgame, validate=True)
    def post(self):
      ...
```

Der Decorator @api.expect(model) erlaubt es, die Payload einer HTTP-Anfrage auf Basis des Modells zu spezifizieren. Das Setzen des optionalen Parameters validate auf True aktiviert die Inputvalidierung. So gleicht Flask-RESTPlus das in der Payload enthaltene Objekt mit der Modellspezifikation ab und generiert im Fall von Abweichungen eine Antwort mit dem Status 400 (Bad Request). Diese Antwort enthält außerdem Details zu den gefundenen Abweichungen (Listing 13).

Listing 13: Fehlermeldung aufgrund fehlgeschlagener Inputvalidierung für die post-Route

```
$ curl -X POST "http://localhost:5000/boardgames" -d "{ "id": "2", "rating": 11.0
```

```
"name": "King of Tokyo", "playing_time": "30 Min",
"expansions": []}"

400 BAD REQUEST
{
    "errors": {
      "designer": "'designer' is a required property",
      "rating": "11.0 is greater than the maximum of 10.0"
    },
    "message": "Input payload validation failed"
}
```

Sortieren und testen

Für API-Endpunkte, die Listen von Elementen zurückliefern, ist es üblich, die maximale Anzahl abzurufender Elemente zu begrenzen. Dies bezeichnet man als Pagination. Andernfalls kann es bei großen Datenmengen zu Performanceproblemen kommen. Der API-Aufrufer übergibt bei der Pagination in der Regel ein Offset (Startelement in der Liste) und ein Limit (die maximale Anzahl abzurufender Elemente) als URL-Parameter. Darüber hinaus lässt sich über URL-Parameter auch das Sortieren und Filtern der Listenelemente realisieren.

Vor diesem Hintergrund implementiert Listing 14 entsprechende Mechanismen für die get-Route der Ressource BoardgameList, die eine Liste von Brettspielen ausliefert. Hierzu definiert der Decorator @api.param() die URL-Parameter für Pagination (offset und limit) und Sortierung (sort_by und sort_order). Der URL-Paramerter sort_by gibt vor, nach welchem Attribut die Liste der Brettspiele zu sortieren ist. Über den Decorator @api.param() lassen sich neben einer Beschreibung des Parameters auch der Datentyp und ein Defaultwert festlegen. So ist der URL-Parameter sort_order als Aufzählungstyp (enum) realisiert und definiert die Reihenfolge der Sortierung als auf- oder absteigend (asc oder desc).

Listing 14: Ressource BoardgameList um URL-Parameter für Pagination und Sortierung erweitert (resources/boardgames.py)

```
@api.route('')
class BoardgameList(Resource):
    @api.param('offset',
               'The offset of the first boardgame in the list
to return.',
               type=int,
               default=0)
    @api.param('limit',
               'The maximum number of boardgames to return.',
               type=int,
               default=100)
    @api.param('sort_by',
               'Sort the returned boardgames by key.',
               default="id",
                enum=["id", "name", "designer", "rating"])
    @api.param('sort_order',
               'Ascending or descending sort order.',
               default="asc", enum=["asc", "desc"])
    @api.response(200, 'All boardgames successfully fetched!')
    @api.marshal_list_with(boardgame)
    def get(self):
        '''Lists all boardgames in collection'''
        offset = request.args.get('offset')
        ...
```

Über die Anfrageargumente (request.args) greift man auf die URL-Parameter innerhalb der get-Methode der Ressource zu. Der folgende curl-Aufruf ruft die ersten hundert Brettspiele aufsteigend nach der ID sortiert ab:

```
curl -X GET
"http://localhost:5000/boardgames?sort_
order=asc&sort_by=id&limit=100&offset=0"
```

Die vorgestellte Boardgame-Collection-API testet man über das Python-Unit-Testing-Framework unittest. Im Paket tests ist hierzu die Datei test_boardgames_api.py angelegt, die eine

Klasse BoardgamesApiTest enthält – abgeleitet von unittest.TestCase. Das Initialisieren der Tests findet in der Methode setUp statt, die automatisch vor jedem Test ausgeführt wird. Die Methode test_client() des Flask-Frameworks erstellt innerhalb von setUp einen Testclient für die API. Für die Kommunikation mit der API stellt der Client Schnittstellen für die verschiedenen HTTP-Methoden bereit. So lässt sich mit response = self.app.get('/boardgames') die Route boardgames get triggern und die HTTP-Antwort im Rahmen der Tests auswerten. Listing 15 zeigt exemplarisch einige Tests zur Absicherung der API-Grundfunktionen. Man startet die Tests über die Kommandozeile mit

```
python -m unittest tests/test_boardgames_api.py
```

Listing 15: Boardgame-API testen (tests/test_boardgames_api.py)

```
import unittest
from main import app

class BoardgamesApiTest(unittest.TestCase):
    def setUp(self):
      # 1. ARRANGE
      self.app = app.test_client()
      # initialize app with first boardgame
      self.app.post('/boardgames', json={
          "name": "Wingspan",
          "designer": "Elizabeth Hargrave",
          "playing_time": "60 Min",
          "rating": 8.1,
          "id": "1",
            "expansions": [{"name": "European Expansion",
"rating": 8.5}]
      })
    def test_get_boardgame_by_id(self):
      # 2. ACT
      response = self.app.get('/boardgames/1')
      # 3. ASSERT
      self.assertEqual('200 OK', response.status)
```

```
      self.assertEqual('Wingspan', response.json['name'])

    def test_get_boardgame_for_unknwon_id(self):
      # 2. ACT
      response = self.app.get('/boardgames/2')
      # 3. ASSERT
      self.assertEqual('404 NOT FOUND', response.status)
         self.assertTrue("Boardgame 2 not found." in
response.json['message'])

    def test_delete_boardgame_by_id(self):
      # 2. ACT
      response = self.app.delete('/boardgames/1')
      # 3. ASSERT
      self.assertEqual('204 NO CONTENT', response.status)
      response = self.app.get('/boardgames')
      self.assertEqual(0, len(response.json))

    def test_post_new_boardgame(self):
      # 2. ACT
      response = self.app.post('/boardgames', json={
          "name": "King of Tokyo",
          "designer": "Richard Garfield",
          "playing_time": "30 Min",
          "rating": 7.2,
          "id": "2",
          "expansions": []
      })
      # 3. ASSERT
      self.assertEqual('201 CREATED', response.status)
      self.assertEqual('King of Tokyo', response.json['name'])
```

Fazit

Flask und Flask-RESTPlus bieten einen beachtlichen Funktionsumfang und erlauben die Entwicklung robuster REST-APIs auch für Enterprise-Anwendungen. Hierbei ermöglicht der ressourcenorientierte Aufbau von Flask-RESTPlus, APIs nah am REST-Standard zu realisieren. Die automatische HTTP-Payload-Validierung durch das Framework auf Basis der Modellbeschreibungen reduziert die Fehleranfälligkeit der

Anwendung. Mit wenigen Zeilen gut lesbarem Code entstehen umfangreiche APIs, die sich einfach testen und weiterentwickeln lassen. API-Konsumenten hilft die automatisch generierte API-Dokumentation auf Basis der OpenAPI Specification und das Bereitstellen über Swagger UI.

Wer sich für die Sicherheit von Flask-Anwendungen interessiert, findet in der Flask-Dokumentation weitere Informationen (siehe ix.de/zhjd). (nb@ix.de)

1. Quellen
2. Listings, Informationen zu Flask im Unternehmen, Skalierung und Sicherheit: ix.de/zhjd

Dr. Sebastian Bindick

ist IT-Architekt in der Volkswagen-IT. Seine Schwerpunkte liegen in den Bereichen moderne Architekturen, Webtechnologien, Testautomatisierung und Computational Engineering.

[/expand]

Webbasierte und interaktive Darstellung von Kartendaten

webbasierte und interaktive

Darstellung von Kartendaten

[expand title="mehr lesen…"]

Für eine webbasierte und interaktive Darstellung von Kartendaten eignet sich die Kombination aus SVG-Standard und React-Framework.

-tract

- Um eine Karte interaktiv zu gestalten, sind mehrere Schritte nötig. Das Open-Source-Webframework React dient als Grundlage für das Erstellen komponentenbasierter Webanwendungen.
- Das Tool overpass turbo des freien Geodatenprojekts OpenStreetMap hilft, die Kartendaten zu extrahieren. gpx2svg wandelt sie in SVG um.
- Die Bibliothek Emotion bringt CSS an die React-Komponenten.

Berlin als Großstadt überzeugt mit vielfältigem Leben in den Bezirken. Um dieses Spektrum im Web abzubilden, bieten sich interaktive Karten an. Die geografischen Umrisse der Stadtteile lassen sich so mit Statistiken und Metriken verbinden. Dies nutzen vor allem Immobilienfirmen in Analysedashboards oder stadteigene Anwendungen.

Die Beispiele des Artikels sind mit dem Webframework React erstellt, das sich für das Programmieren interaktiver Webanwendungen eignet. Es nutzt HTML direkt in JavaScript mit JSX (JavaScript XML).

Neben React kommt der Grafikstandard SVG zum Einsatz, der sich für Kartenanwendungen gut eignet. 1998 von der W3C-Arbeitsgruppe als offener Standard entwickelt, beschreibt er als Vektoren dargestellte Grafiken. Eine Grafik lässt sich damit ohne Darstellungsverluste vergrößern. SVG ist ein von

Maschinen und Menschen lesbares Format, das in HTML eingefügt und so ein integraler Bestandteil einer Webseite wird. Die SVG-Grafik ist zudem animierbar.

Die Umrissdaten der Bezirke von Berlin liefert das freie Geodatenprojekt OpenStreetMap. 2004 von Steve Coast initiiert, verfügt es über umfangreiches globales Kartenmaterial. Alle Daten aus OpenStreetMap stehen unter der Open Database License – ODbL frei zur Verfügung. Mit dem OpenStreetMap-Werkzeug overpass turbo können Anwender die Overpass-API abfragen – eine schreibgeschützte API, die benutzerdefinierte ausgewählte Teile der OpenStreetMap-Kartendaten bereitstellt. So extrahieren und filtern sie gewünschte Daten mit einer Abfragesprache. Die gewonnenen Daten lassen sich in verschiedenen Dateiformaten exportieren.

Die Abfragesprache von overpass turbo nennt sich Overpass Query Language, kurz Overpass QL. Jede Angabe schließen Anwender mit einem Semikolon ab. Listing 1 zeigt die Overpass-QL-Abfrage für eine Umrisszeichnung der Bezirke von Berlin. Am einfachsten generieren Anwender das Listing mit dem Wizard, indem sie boundary=administrative and admin_level=9 eingeben und ausführen. Die Variable admin_level mit dem Wert 9 bezeichnet in deutschen Städten die Bezirksebene.

Listing 1: Overpass-QL-Abfrage der Berliner Bezirke

```
[out:json][timeout:25];
{{geocodeArea:Berlin}}->.searchArea;
    (
      // query part for: "boundary=administrative and admin_level=9"
node["boundary"="administrative"]["admin_level"="9"](area.searchArea);
way["boundary"="administrative"]["admin_level"="9"](area.searchArea);
relation["boundary"="administrative"]["admin_level"="9"](area.searchArea);
```

```
);
// print results
out body;
>;
out skel qt;
```

Leider steht im Exporter von overpass turbo das benötigte SVG-Format nicht zur Verfügung. Deshalb sind zunächst die Daten als GPS Exchange Format, GPX, zu exportieren und herunterzuladen. Zum Umwandeln der Daten von GPX in SVG bietet sich das Werkzeug gpx2svg von Tobias Leupold an (siehe ix.de/zxex). Zum Ausführen benötigt man Python 3 und die GPX-Datei. Das folgende Kommando wandelt die Bezirksumrissdatei von GPX in SVG um:

```
python3 gpx2svg -i berlin.gpx -o berlin.svg
```

Nun kann die SVG-Karte schon mit gängigen Bildbearbeitungsprogrammen wie Inkscape oder Adobe Illustrator angesehen und wenn gewünscht weiterverarbeitet werden. Für das Web braucht es aber ein wenig Vorarbeit.

Vorbereitungen treffen

Zunächst gilt es, Node.js und npm auf dem Rechner zu installieren. Mit npm können Entwickler benötigte JavaScript-Softwarebibliotheken verwalten. Es empfiehlt sich, Node.js und npm mit einem Node Version Manager zu installieren. Versionsupdates von Node.js sind damit durchführbar. Auch zum Testen der Anwendung unter verschiedenen Node.js-Versionen eignet sich ein Versionsmanager gut: für Linux und Mac OS X etwa nvm, für Windows nvm-windows. Das verwendete Beispiel hier nutzt die neueste LTS-Version von Node.js, 12.18.3. LTS steht für Long Term Support und ist die stabile und empfohlene Version von Node.js, die auch für den produktiven Einsatz geeignet ist.

Als Nächstes ist der Paketmanager Yarn in Version 1 zu installieren, der ähnlich wie NPM arbeitet und damit

kompatibel ist. Man muss nur aufpassen, NPM und Yarn nicht in einem Projekt zusammen zu verwenden.

Eine gute Möglichkeit, schnell eine neue lauffähige React-Anwendung zu erstellen, bietet das Paket create-react-app. Es fertigt automatisch ein gutes Grundgerüst der neuen Anwendung an. Darauf aufbauend lässt sich die Anwendung um eine neue Komponente erweitern. Der folgende Befehl erstellt eine neue React-Anwendung mit dem Namen react-district-map-berlin:

```
npx create-react-app react-district-map-berlin
```

npx führt das angegebene Paket create-react-app automatisch aus, in diesem Fall mit dem Parameter react-district-map-berlin, dem Namen der Anwendung. Die Bibliothek Emotion setzt das Styled-Components-Muster in React um. Damit fügen Entwickler an React-Komponenten den CSS-Stil hinzu. Die benötigten Pakete der Bibliothek, @emotion/styled und @emotion/core, installieren Entwickler mit Yarn:

```
yarn add @emotion/styled @emotion/core --save
```

Die nun erzeugte Anwendung hat den Eintrittspunkt in der Datei src/index.js, die Hauptkomponente App.js ist erstellt. Es gilt jetzt, eine neue Komponente BerlinMap im Verzeichnis src/components anzufertigen. Diese Komponente wird später in die Datei App.js importiert, um sie im Programm nutzen zu können. Listing 2 zeigt einen Ausschnitt der Berlin-Kartenkomponente.

Listing 2: Die Anwendung mit Funktionskomponente und eingefügter SVG-Datei

```
import React from "react";
import styled from "@emotion/styled";

function BerlinMap(props) {
    const handleMouseOver = (evt) => {
      for (const attr of evt.target.attributes) {
        if (attr.name === "name") {
```

```
          props.callback(attr.value);
        }
      }
    };
    return (
      <>
        <svg
          xmlns="http://www.w3.org/2000/svg"
          id="svg626"
          viewBox="0 0 3001.18 2470"
          version="1.1"
          style={{ width: "50%" }}
        >
          <defs id="defs630" />
          <g id="g624">
            <Borough
              id="path598"
                 d="m 575.41788,646.71475 -4.2765,12.94153
-2.30212,3.5527 -2.4876,3.44405           -0.83693,3.14533
0.15517,3.36692  1.00458,2.55306  1.38493,1.90799  1.83438,
2.20371  4.82182,4.28924  2.10949,3.35063  1.67163,3.41958
1.00726,3.61913   d-1.2846,9.32649   -0.22294,8.49046
3.96037,10.2116   2.79393,3.71342 4.05623,5.07077
d...
   d3.29064,6.03348   1.75457,7.10791   -1.68679,6.8105
-3.12522,10.42868 z"
              name="Reinickendorf"
              onMouseOver={handleMouseOver}
            />
[...]
</g>
        </svg>
      </>
    );
}

const Borough = styled.path`
    fill: #000080;
    stroke: yellow;
    :hover {
      fill: red;
```

```
    }
`;
```

Die Datei besteht aus einem Importteil, der Funktionskomponente BerlinMap und einer CSS-Stildefinition als Styled Component (mit Emotion). Entwickler fügen die generierte SVG-Datei in die React-Komponente ein und passen sie an die Bedürfnisse der Anwendung an. Dabei bauen sie den Inhalt der SVG-Datei zwischen einem Fragment (<></>) ein. Um das Fragment mit dem SVG-Code in der Anwendung nutzen zu können, müssen sie manuelle Änderungen daran vornehmen. Der SVG-Code besteht aus dem Tag svg und einem Containerelement g. Letzteres dient dazu, weitere gleichartige Elemente zu verbinden. Solche Elemente sind im vorliegenden Fall die Pfadelemente path, die als verbundene Punkte die Form eines Bezirksumrisses beschreiben.

Als Nächstes fügen Entwickler im Element svg das Attribut viewBox hinzu und entfernen die ursprünglichen Attribute width und height. Das Attribut viewBox setzt das initiale Koordinatensystem fest und ist wichtig, weil es eine skalierte Darstellung der Karte ermöglicht. Die einzelnen path-Elemente definieren jeweils einen Bezirk. Um diese Bezirke richtig zu stylen, kommen die Styled Elements aus der Emotion-Bibliothek zum Einsatz. Sie lassen sich wie React-Komponenten verwenden.

Bezirke einfärben

Ein JavaScript-Tagged-Template-Literal beschreibt den Stil als CSS und ermöglicht Entwicklern CSS in JavaScript zu schreiben, ohne andere Methoden wie das Setzen eines Inline-Styles oder CSS-Klassen verwenden zu müssen. In der Komponente Borough (englisch für Bezirk) ist das CSS-Attribut fill die Füllfarbe des Bezirks, stroke bezeichnet die Farbe der Umrisslinien. Der Pseudoselektor :hover tritt in Aktion, wenn der Nutzer mit dem Mauscursor über dem jeweiligen Bezirk liegt. Dann färbt sich der Bezirk rot.

Alle path-Elemente können Entwickler nun durch die React-Styled-Komponente Borough ersetzen. Mit dem Attribut name weisen sie der Komponente den richtigen Namen der Bezirke zu. Bewegt sich der Mauscursor über den Bezirk, stellt das ein Ereignis dar, auf das reagiert werden soll. Bei einem Mouse-over ruft der Browser die angegebene Funktion auf, hier handleMouseOver. Darin iteriert man über alle Attribute der jeweiligen Bezirkskomponenten, bis das name-Attribut gefunden ist. Eine aufgerufene Funktion callback gibt das Elternelement als Prop mit. Der Parameter der callback-Funktion ist der Inhalt des Attributs name, also der Bezirksname.

Logik aus Komponenten recyceln

Im Elternelement lässt sich die callback-Funktion bearbeiten, indem Entwickler mit setBorough den Status der Variablen borough setzen (Listing 3). React aktualisiert automatisch die Variable borough im DOM (Document Object Model) und der jeweilige Bezirksname erscheint.

Listing 3: Status der Variablen borough festlegen

```
import React, { useState } from "react";
import BerlinMap from "./components/BerlinMap/BerlinMap";
import styled from "@emotion/styled";

function App() {
    const [borough, setBorough] = useState("");

    const handleBorough = (name) => {
      setBorough(name);
    };

    return (
      <>
        <Container>
          <h1>Berliner Bezirke</h1>
          <BerlinMap callback={handleBorough} />
          <h1>{borough}</h1>
        </Container>
```

```
      </>
    );
}

const Container = styled.div`
    margin-left: auto;
    margin-right: auto;
    width: 800px;
`;

export default App;
```

Listing 3 verwendet den React-Hook useState(""). Mit React-Hooks lassen sich Statusfunktionen in Funktionskomponenten nutzen. Entwickler können eine statusbezogene Logik aus Komponenten extrahieren und wiederverwenden. Der Parameter von useState("") ist der Initialwert der Variablen. Er liefert als Ergebnis ein Array zurück, dessen erstes Element die Variable zum Lesen des Wertes ist (hier borough). Das zweite Element des Arrays ist eine Funktion zum Setzen des Wertes, hier setBorough. Beide werden durch Array Destructuring extrahiert. Ruft man nun die Anwendung mit yarn start auf, sieht man im Browser die Anwendung wie in der Abbildung.

Der Name des Bezirks unter der Karte aktualisiert sich

dynamisch, je nachdem wo sich der Mauscursor befindet. *OpenStreetMap*
Die Kartenanwendung lässt sich mit Statistiken und Metriken zu den Bezirken erweitern. Alternativ sind Bezirke je nach Höhe einer Metrik anders einfärbbar. Die fertige Anwendung findet sich unter ix.de/zxex. (nb@ix.de)

1. Quellen
2. Informationen zu gpx2svg, Listings und die fertige Anwendung: ix.de/zxex

Thomas Derflinger

ist freiberuflicher Softwareentwickler mit Schwerpunkt auf der Frontend-Entwicklung von modernen Webanwendungen.

[/expand]

Werkzeuge zur automatischen Codeanalyse

Werkzeuge zur automatischen Codeanalyse

[expand title="mehr lesen…"]

Werkzeuge zur automatischen Codeanalyse

Quelltext im Fokus

Andreas Wiegenstein

Unternehmen sind gut beraten, auf Tools zurückzugreifen, die Entwickler und Tester durch automatisierte Qualitätsanalysen entlasten. Doch worauf sollten Unternehmen achten, wenn sie einen Codescanner einsetzen wollen?

-Tract

- Code automatisch zu scannen und zu testen, gewinnt in immer mehr Unternehmen an Bedeutung, gerade bei zugekaufter, fremder Software in der eigenen Codebasis.
- Kosten und Ressourcenaufwand wachsen, je später Tests ausgeführt werden. Integration und Automatisierung sowie hohe Erkennungsraten erweisen sich als Kernfaktoren für gute Codescanner.
- Bei der Anschaffung sollten Unternehmen genau darauf achten, was sie brauchen, und sich nicht vom Funktionsumfang blenden lassen. Eine Evaluation vor der Anschaffung spart später viel (Nach-)Arbeit.

Hacken ist eine Wachstumsbranche. Und weil fast alle Sicherheitslücken auf Programmierfehlern basieren, kommt jeder, der Software schreibt, unweigerlich früher oder später mit dem Thema Applikationssicherheit in Kontakt – je früher, desto besser. Bei der Entwicklung von Software konkurrieren allerdings Hunderte von Qualitätsanforderungen aus unterschiedlichen Bereichen um die Aufmerksamkeit und die Ressourcen der Entwickler. Und praktisch niemand kann diese alle kennen, geschweige denn richtig umsetzen, insbesondere wenn das Programmierprojekt einem gewissen Zeit- und

Budgetdruck unterliegt. Gute Codeanalyse kann da sowohl finanziell als auch zeitlich einiges einsparen helfen, vom Ärger ganz zu schweigen.

Die Informationen in dieser Marktübersicht stammen zum großen Teil aus öffentlich verfügbaren Beschreibungen der Scanner im Internet, wobei sich die Suche nach genauen technischen Angaben zu den unterstützten Sprachen, Testfällen, Testverfahren und Plattformen bisweilen schwierig gestaltet. Auf den folgenden Seiten finden Sie eine Tabelle mit Kurzbeschreibungen zu 35 Tools zur Codeanalyse sowie eine ausführliche Tabelle mit vier exemplarisch ausgewählten, sehr unterschiedlichen, aber typischen Vertretern für das, was der Markt anbietet: CAST Application Intelligence Platform, Perforce Klocwork, Micro Focus Fortify und Veracode. Die Tabelle „Sprachvergleich“ zeigt die Sprachunterstützung der vier Werkzeuge im direkten Vergleich.

Sprachenwirrwarr

An erster Stelle steht dabei natürlich die Frage, ob ein einziges Tool alle vom Unternehmen verwendeten Programmiersprachen untersuchen kann. Manche Werkzeuge sind auf nur eine Sprache ausgerichtet, andere versprechen, mehrere Sprachen untersuchen zu können. Es liegt daher nahe, dass große Unternehmen in der Regel die Tools mit den breiten Sprachpaletten bevorzugen. Sie liefern eine zentrale Plattform zur Analyse und lassen sich auch einfacher warten. Allerdings sind diese generalistischen Tools häufig nicht so gut auf die einzelnen Sprachen ausgerichtet, was sich in der Benutzbarkeit und Testqualität niederschlagen kann.

Kosten sind ein weiterer wichtiger Faktor. Dabei sind aber nicht nur Lizenzen und Support zu berücksichtigen – sofern keine Open-Source-Software zum Einsatz kommt. Auch die laufenden internen Kosten für den Betrieb, wie etwa dedizierte Hardware und das Ausrollen von Updates sowie damit verbundene Downtime, sollten in die Entscheidung einfließen.

Aber es entstehen noch weitere Kosten, die den meisten Unternehmen zunächst nicht bewusst sind. Jedes Tool zur statischen Codeanalyse (SCA) verwendet bestimmten Techniken, um Schwachstellen zu identifizieren.

Dies können einfache Mustersuchen sein, um festzustellen, ob eine bestimmte gefährliche Funktion verwendet wird. Es können aber auch komplexe Verfahren wie Datenflussanalysen sein, die zu ermitteln versuchen, ob während der Verarbeitung auch wirklich externe Eingaben an die bekannten gefährlichen Funktionen gelangen können.

Falsch ist nicht gleich falsch

Bei solchen Untersuchungen kann es aber in der statischen Analyse zu Fällen kommen, in denen nicht eindeutig feststellbar ist, ob tatsächlich ein Risiko vorliegt. In so einem Fall werden die Ergebnisse – je nach Strategie des Herstellers – entweder verworfen, trotzdem als Fehler gezeigt oder als potenzielle Fehler eingestuft.

Alle drei Wege verursachen Aufwand und Kosten beim Kunden. Wenn etwa bekannt ist, dass das Tool bei bestimmten Tests falsch negative Ergebnisse liefert, dann muss das Unternehmen ein ergänzendes Testverfahren aufbauen, um diesen Mangel zu kompensieren, zumindest wenn diese Tests für das Unternehmen hohe Priorität haben.

Kommt es zu falsch positiven Ergebnissen, dann entstehen Kosten durch die unnötige Analyse des als fehlerhaft gemeldeten Codes, Diskussionen mit entnervten Entwicklern oder durch übereifrige Korrekturen, die eigentlich nicht nötig sind.

Kommt es zu Ergebnissen, die nur als potenzielle Fehler eingestuft sind, so muss das Unternehmen eine Strategie entwickeln, bis zu welchem Grad solche Ergebnisse zu ignorieren sind und welche manuell nachuntersucht werden

müssen. Diese Managemententscheidung kann einen großen Unterschied machen und bringt erhebliche Konsequenzen mit sich: Je nachdem, wie umfangreich der eigene Code ist und wie viele Testfälle das eingesetzte Tool verwendet, laufen durch solche zusätzlichen Analysen unter Umständen enorme Kosten auf.

Das gilt insbesondere wenn ein Unternehmen große Mengen an Code untersucht, der aus einer Zeit ohne prüfbare Sicherheitsvorgaben stammt oder durch den Erwerb von Firmen oder Produkten zur Codebasis hinzukam, wo die Entwickler schlimmstenfalls keine oder abweichende Standards bei der Softwareerstellung befolgten.

Testfälle auflisten

Unternehmen sollten von der Analyse möglicher Tools eine Liste unverzichtbarer Testfälle erstellen. Diese Liste muss alle Programmierfehler enthalten, die dem Unternehmen großen Schaden zufügen würden. Für alle möglichen Programmierfehler aus der Liste sollten Programme mit verschiedenen Ausprägungen der Fehler erstellt werden, wobei vorher festgelegt wird, welche Ausprägungen ein Tool als Fehler einstufen müsste und welche nicht. Auf dieser Basis lässt sich abschätzen, ob und wo es zu relevanten falsch positiven oder falsch negativen Ergebnissen kommen kann.

Außer der Ergebnisqualität gibt es aber noch andere Anforderungen an die technischen Fähigkeiten des gewählten Werkzeugs. Manche Unternehmen versprechen sich von Tools insbesondere einen umfassenden Satz an Know-how in Form von standardmäßig enthaltenen Testfällen. Niemand sollte jedoch blind das Tool mit den meisten ausgelieferten Testfällen kaufen. Stattdessen braucht es ein Tool, das genau die Fehler aufspüren kann, die im eigenen Unternehmen und in den untersuchten Sprachen den größten Schaden anrichten können.

In jedem Fall sollten Unternehmen eine qualifizierte Auswahl

treffen, welche Tests im produktiven Betrieb den größten Sicherheitsnutzen mit möglichst geringem Aufwand bringen. Und es kann durchaus sein, dass im Laufe der Zeit eigene Tests hinzukommen müssen, weil bestimmte Anforderungen zu speziell sind, als dass sie ein allgemeines Tool abdecken würde, oder weil das Unternehmen schnell auf neue Anforderungen reagieren muss. Es ist vorteilhaft, die Standardeinstufungen der Fehler im Tool anpassen zu können, falls im Unternehmen abweichende Richtlinien gelten.

Zu den technischen Fähigkeiten gehören aber auch Angaben, wie schnell eine bestimmte Menge Code untersucht werden kann. Das ist wichtig, wenn beispielsweise Scans, bei denen es um große Mengen an Code geht, nachts oder innerhalb bestimmter Zeitfenster stattfinden müssen. Oder vielleicht wenn geplant ist, die Ergebnisse stündlich an ein SIEM weiterzugeben. Aber auch wenn Entwickler ihre Arbeit einer Zwischenprüfung unterziehen möchten, sollte der Einsatz des Tools nicht zu einer erweiterten Kaffeepause führen.

Haptik und Zielgruppe

Das führt direkt zur Handhabung des Tools. Auf welche Zielgruppen ist es ausgerichtet? Testteams benötigen Berichte mit Kritikalitätseinstufungen und gut nachvollziehbaren Fehlerbeschreibungen, aus denen sie gegebenenfalls weiterführende manuelle Tests ableiten können. Manager brauchen eine zentrale Sicht aller Risiken und der Fortschritte, die durch den Einsatz des Tools erzielt wurden. Auditoren benötigen exportierbare Berichte, die revisionssicher verfasst sind, also nicht manipuliert werden können. Entwickler hingegen haben den größten Nutzen, wenn das Tool direkt in ihre IDE integriert ist, also während des Programmierens bereits Feedback zu Fehlern geben kann, entweder auf Knopfdruck oder – besser – fortlaufend, ähnlich einer interaktiven Rechtschreibprüfung bei Textverarbeitungssoftware.

Mit einem Scanner, der direkt in der IDE aktiv ist, erzielen Firmen den besten Wirkungsgrad. Je früher eine Sicherheitslücke im Code erkannt wird, desto kostengünstiger kann sie behoben werden. Fehler, die direkt beim Schreiben des Codes auffallen, verursachen keinen organisatorischen Aufwand, keine Nachtests, keine ungeplanten Sonderschichten oder sonstigen Änderungen am aktuellen Projekt. Sie erzielen außerdem unmittelbar einen Lerneffekt beim Entwickler. Dafür müssen dann in der IDE natürlich Hinweise abrufbar sein, was genau das Problem ist und wie es vermieden werden kann. Die Qualität der Beschreibungen und Verbesserungsvorschläge ist für alle Beteiligten elementar.

Rekursiv? Nicht immer

Eine weitere interessante Erkenntnis aus der Praxis ist, dass nicht alle Scanner die von ihnen vorgeschlagenen Lösungen auch erkennen, nachdem der Entwickler den Code korrigiert hat und diesen dann erneut scannt. Auch das manuelle Unterdrücken falsch positiver Ergebnisse oder die Behandlung von Ausnahmen funktionieren nicht immer wie erwartet – sofern diese Features überhaupt vorhanden sind. Bei einer Evaluierung sollte man daher auf jeden Fall prüfen, was passiert, wenn man Code ändert, der mit der unterdrückten Schwachstelle nicht in Zusammenhang steht, aber in der gleichen Funktion vorkommt, in der die Ausnahme angelegt wurde. Als ergänzender Test empfiehlt sich, in einer solchen Funktion eine zweite Schwachstelle gleichen Typs anzulegen, um zu ermitteln, ob eventuell sogar mehrere Schwachstellen gleichen Typs in derselben Funktion durch den Mechanismus verdeckt werden. Es gibt tatsächlich Tools mit solchen Designfehlern.

Auch aus operativen Gesichtspunkten gibt es Anforderungen an SCA-Tools. Soll das Werkzeug nur im eigenen Unternehmensnetz (on Premises) laufen oder ist auch der Einsatz eines Cloud-Produkts denkbar? Dies hängt sicher davon ab, wie viel geistiges Eigentum im Quellcode steckt und was der

Datenschutzbeauftragte dazu sagt, wie weit man einem (ausländischen) Anbieter diesen Code und insbesondere auch die darin befindlichen Sicherheitsfehler anvertrauen möchte. Ja, Hacken ist eben die angesprochene Wachstumsbranche. Ebenso wichtig ist auch die Frage, ob das Tool selbst bei einer lokalen Installation regelmäßig Daten mit dem Hersteller austauscht. Solche „Features“ muss der Kunde klären, um später böse Überraschungen zu vermeiden. Nicht zuletzt sollten sich Unternehmen auch informieren, wie viele Sicherheitspatches für das Tool bisher veröffentlicht wurden. Niemand sollte der Illusion verfallen, dass Sicherheitstools automatisch sicher sind. Da kann es lehrreich sein, mal beim Hersteller nachzufragen, wie der Prozess für das eigene Schwachstellenmanagement aussieht.

Neben den rein funktionalen Auswahlkriterien für SCA-Tools spielen noch andere Aspekte beim produktiven Einsatz von Scannern eine Rolle. Bei den meisten Unternehmen, die dem Autor bekannt sind, wurde nach dem Kauf eines Scanners zunächst eine Menge alter Code gescannt, der noch nie konsequent und flächendeckend auf Schwachstellen untersucht worden war. Dabei wurden möglichst viele Testfälle des Tools aktiviert, um den Istzustand und die Leistungsfähigkeit des Tools zu ermitteln. Dieses Vorgehen ist verständlich, kann jedoch Komplikationen mit sich bringen.

Fallout vom Wirtschaftsprüfer

Denn falls der untersuchte Code zu einer Software gehört, die im Unternehmen Finanzdaten verarbeitet, wie etwa in SAP üblich, dann müssen die Ergebnisse solcher Scans auch dem Wirtschaftsprüfer vorgelegt werden. Nun sind Wirtschaftsprüfer aber nur selten Experten für Applikationssicherheit. Es kann also passieren, dass deren Prüfbericht verlangt, dass schlichtweg *alle* Meldungen, die dem Wirtschaftsprüfer vorliegen, behoben werden müssen. Das kann schnell zu hohen Kosten und ungeplanten, bisweilen vermeidbaren Arbeiten

führen.

Nicht nur deshalb sollten Unternehmen klein anfangen, zumindest was die Analyse von Bestandscode angeht. Sie sollten langsam und in mehreren Phasen vorgehen. Vor dem Einsatz des Tools muss feststehen, welche Tests für das Unternehmen relevant sind und welche Arten von Schwachstellen es in einer ersten Phase beheben will und kann. Ebenso muss ermittelt werden, welche Tests mit einem hohen manuellen Prüfungsaufwand verbunden sind.

Projektteams sollten darauf achten, dass sie während einer Korrekturphase keine Updates des Scanners einspielen. Falls der Hersteller Veränderungen an der Testlogik vorgenommen hat, kann das auch Meldungen beeinflussen, die in der aktuellen Phase bearbeitet werden und so zu ungeplantem Mehraufwand führen.

Code verwerfen oder doch korrigieren?

Bevor Korrekturen an fehlerhaftem Code durchgeführt werden, ist immer zu prüfen, ob der überhaupt noch gebraucht wird. Code löschen oder stilllegen ist die bei Weitem sicherste und günstigste Art, Fehler zu beheben. Manche Tools bieten auch automatisierte Codekorrekturen an. Passiert das in einer IDE, kann der Entwickler sofort erkennen, ob die Korrektur seinen Anforderungen entspricht. Passieren die Korrekturen aber im Hintergrund als Massendatenverarbeitung, kann dies sehr unschöne Effekte haben.

Viele Probleme im Umgang mit komplexen Tools werden leider erst nach längerer Nutzung ersichtlich. Beispielsweise könnte die Qualität des Supports unzureichend sein. Oder der Hersteller wird von einem größeren (ausländischen) Unternehmen aufgekauft, das ganz andere Ziele verfolgt. Unternehmen sind daher gut beraten, von Anfang an über eine Strategie nachzudenken, wie man den verwendeten Scanner irgendwann wieder ablösen könnte.

Fazit

Es gibt nicht den einen idealen Codescanner. Jedes Unternehmen sollte solche Produkte nach den Eigenschaften auswählen, die für den eigenen Einsatzzweck am besten geeignet sind. Dazu zählen vor allem die verwendeten Programmiersprachen und Entwicklungsplattformen. Zusätzlich ist es erforderlich, technische Grenzen und Schwächen der Scanner zu kennen, um diese durch ergänzende Maßnahmen möglichst weitgehend zu kompensieren. Erst nach dieser tiefen Analyse ist es möglich, zusätzliche Kosten in die Gesamtkostenbetrachtung einzukalkulieren. Wer hier die Kosten minimieren will, kommt nicht umhin, vor der Anschaffung Arbeitszeit in sinnvolle Tests zu investieren. (mfe@ix.de)

Hersteller, Lizenzen und Sprachen				
Hersteller	Scanner	Lizenz	Sprache(n)	Kommentar
http://awap.sourceforge.net/	WAP	Open Source	PHP	Verwendet Datenflussanalyse mit 8 Testfällen für gängige Schwachstellen; bietet eine automatisierte Korrektur von Fehlern.
http://clang-analyzer.llvm.org/	Clang Static Analyzer	Open Source	C, C++, Objective-C	Führt eine Codeanalyse während des Build-Prozesses durch; einfache Mustererkennung für Sicherheitsfehler, sucht aber auch funktionale Fehler.
http://sparrow.fasoo.com/en/	SPARROW	proprietär	C/C++, Android Java, Java, Objective-C, JSP, HTML, C#, SQL, XML, ABAP, PHP, ASP.NET, Python, VB.NET, Swift, JavaScript, APEX, VBScript, Visualforce, XSL	Bietet Integration in die IDE, Build-Systeme und Versionskontrollsysteme; umfangreiche Testfälle, basierend auf Compliance-Standards von CWE, OWASP und anderen.
https://github.com/Microsoft/DevSkim	DevSkim	Open Source	C, Objective C, C++, C#, COBOL, Go, Java, JavaScript/TypeScript, PHP, PowerShell, Python, Ruby, Rust, SQL, Swift, Visual Basic	Liefert ein plattformübergreifendes Command-Line-Interface und IDE-Plug-ins für Visual Studio und Visual Studio Code. Die Plug-ins liefern dem Entwickler direktes Feedback zu Fehlern, inklusive Erklärungen und Korrekturvorschlägen.
https://brakemanscanner.org/	Brakeman	Open Source	Ruby	Brakeman ist ein kostenloser Schwachstellenscanner, der speziell für Ruby-on-Rails-Anwendungen entwickelt wurde.
https://discotek.ca/deepdive.xhtml	Deep Dive	Freeware	Ear, War, Jar, APK	Statisches Codeanalysetool für JVM Deployment Units. Hat einen erweiterbaren Satz an Testfällen.
https://discotek.ca/sinktank.xhtml	Sink Tank	Freeware	Java	Verwendet Datenflussanalyse und eine nicht genannte Zahl an Testfällen. Kann nur Bytecode analysieren, inkl. kompilierter JSPs.

Hersteller, Lizenzen und Sprachen				
Hersteller	Scanner	Lizenz	Sprache(n)	Kommentar
https://dotnet-security-guard.github.io/	Roslyn Security Guard	Open Source	.NET	Analysiert .NET- und .NET-Core-Projekte im Hintergrund (IntelliSense) oder während eines Builds. Hat nur begrenzte Testfälle und verwendet einfache, intraprozedurale Analysen.
https://github.com/ajinabraham/nodejsscan	nodejsscan	Open Source	Node.js	Open-Source-Scanner für Node.js-Anwendungen
https://github.com/designsecurity/progpilot	Progpilot	Open Source	PHP	Open-Source-Scanner für PHP-Anwendungen
https://github.com/PyCQA/bandit	Bandit	Open Source	Python	Bietet ca. 70 Testfälle für Python-Anwendungen, verwendet dabei Mustersuchen. Läuft stand-alone und erzeugt Berichte.
https://github.com/securego/gosec	Gosec	Open Source	Go	Bietet ca. 30 Testfälle für Go-Anwendungen, verwendet dabei Mustersuchen.
https://github.com/wireghoul/graudit/	Graudit	Open Source	ActionScript, Android, ASP, C, COBOL, .NET, exec, fruit, Go, iOS, Java, js, Perl, PHP, Python, Nim, Ruby	Open-Source-Tool, das mit einfacher Mustererkennung arbeitet. Erfordert manuelle nachträgliche Analyse der Ergebnisse.
https://huskyci.opensource.globo.com/	HuskyCI	Open Source	Ruby, Go, Python, JavaScript, Java	Verwendet verschiedene Testssets für die Fehlersuche, wie z. B. gosec, Bandit und Brakeman.
https://pumascanpro.com/	Puma Scan	proprietär	C#, ASPX, .cshtml	Bietet umfangreiche Testfälle für C#-Anwendungen, verwendet dabei Mustersuchen und Datenflussanalyse. Arbeitet als Plug-in für Visual Studio und für .NET Frameworks.
https://rubygems.org/gems/dawnscanner	Dawnscanner	Open Source	Ruby	Verwendet Datenflussanalyse und eine nicht genannte Zahl an Testfällen. Liefert auch Lösungsvorschläge.
https://www.securityreviewer.net/	Security Reviewer	proprietär	C#, VB.NET, VB6, ASP, ASPX, Java, JSP, JavaScript, TypeScript, eScript, Java Server Faces, APEX, Ruby, Python, R, Go, Kotlin, Groovy, Flex, ActionScript, PowerShell, Lua, HTML5, XML, XPath, JSON, C, C++, PHP, Scala, Rust, IBM Streams SPL, Objective-C, Objective-C++, Swift, COBOL, JCL, RPG, PL/I, ABAP, SAP HANA, UiPath, BPMN, BPEL, SAIL, PL/SQL, T/SQL, U-SQL, Teradata SQL, SAS-SQL, ANSI SQL, IBM DB2, IBM Informix, MySQL, FireBird, PostGreSQL, SQLite, MongoDB	Verwendet komplexe Analyseverfahren mit mehr als 1100 Prüfregeln. Breite Palette an Schnittstellen und Plug-ins für IDEs und CI/CD-Lösungen. Kann auch Binärcode untersuchen. Basiert auf mehreren innovativen und patentierten Verfahren.
https://sourceforge.net/projects/cppcheck/	Cppcheck	Open Source	C, C++	Verwendet besondere Analyseverfahren zur Erkennung von Schwachstellen in C-Code. Liefert Plug-ins für viele Entwicklungswerkzeuge.
https://sourceforge.net/projects/visualcodegrepp/	VisualCodeGrepper (VCG)	Open Source	C++, C#, VB, PHP, Java, PL/SQL, COBOL	freier statischer Codescanner für 32-Bit-Windows-Plattformen
https://spotbugs.github.io/	SpotBugs	Open Source	Java	Verwendet mehr als 400 Tests zur Fehlererkennung. Integration in Ant, Maven, Cradle und Eclipse möglich.
https://www.adacore.com/codepeer	CodePeer	proprietär	Ada	

Hersteller, Lizenzen und Sprachen				
Hersteller	Scanner	Lizenz	Sprache(n)	Kommentar
https://www.adacore.com/sparkpro	SPARK tool set	proprietär	Spark	Verwendet verschiedene komplexe und mathematische Analyseansätze. Vollständige Integration in GNAT Studio.
https://www.blueclosure.com	BlueClosure BC Detect	proprietär	JavaScript	Untersucht Code, der mit JavaScript Frameworks wie Angular.js, jQuery, Meteor.js oder React.js verwendet wird. Kann auch verschleierten Code untersuchen.
https://www.codacy.com/	Codacy	proprietär	APEX, Bash, C, CoffeeScript, C++, C#, Crystal, CSS, Dockerfile, Elixir, Go, Groovy, Java, JavaScript, JSON, JSP, Kotlin, LESS, Markdown, PHP, PLSQL, PowerShell, Python, Ruby, SASS, Scala, Swift, TSQL, TypeScript, Velocity, Visual Basic, Visualforce, XML	Unterstützt eine breite Palette an Sprachen. Als On-Premises- und Cloud-Lösung verfügbar. Arbeitet direkt in GIT Repositories.
https://www.codescan.io/	CodeScan Cloud	proprietär	Salesforce	Scanner für die Salesforce-Plattform; als On-Premises- und Cloud-Lösung verfügbar.
https://www.grammatech.com/products/codesonar	CodeSonar	proprietär	C/C++, Java, C#, Intel- und ARM-Binärdateien	CodeSonar unterstützt die Standards MISRA-C, MISRA-C++, AUTOSAR C++-14, CERT, DISA STIG, OWASP, CWE. Schwachstellen werden persistent verwaltet und über Builds hinweg verfolgt, selbst wenn sich der Code ändert. Sie können mit Anmerkungen versehen, in eine Rangfolge gebracht, zugewiesen, gesucht und verglichen werden. Unterstützung für Teamtools wie Jenkins, Visual Studio, Eclipse, Jira, GitLab und Docker.
https://www.kiuwan.com/	Kiuwan	proprietär	ABAP, ActionScript, ASP.NET, C, C#, C++, COBOL, Go, HANA SQL Script, HTML, Informix, Java, JavaScript, TypeScript, JCL, JSP, Kotlin, Natural, Objective-C, Oracle Forms, PHP, PL-SQL, PowerScript, Python, RPG4, Scala, SQL, Swift, Transact-SQL, VB.NET, Visual Basic 6, XML	Deckt gängige Sicherheitsprobleme ab. Compliance-Berichte decken folgende Standards ab: OWASP, CWE, MISRA, NIST, PCI und CERT. Unterstützt die IDEs von Eclipse, Visual Studio, IntelliJ IDEA, PhpStorm, PyCharm and WebStorm. Benötigt eine Internetverbindung.
https://www.mathworks.com/products/polyspace-bug-finder.html	Polyspace Bug Finder	proprietär	C, C++	Überprüft laut Hersteller die Einhaltung von Programmierstandards wie z. B. MISRA-C, MISRA-C++, JSF++, CERT C, CERT C++.
https://www.mathworks.com/products/polyspace-code-prover.html	Polyspace Code Prover	proprietär	C, C++	Analysiert Quellcode auf eine Reihe üblicher Fehler, verwendet dabei Mustersuchen und Datenflussanalysen. Arbeitet mit Eclipse.
https://www.parasoft.com/products	Parasoft Test	proprietär	Java, C, C++, C#	Verwendet über 2500 Regeln für die Analyse. Compliance-Berichte automatisieren die Dokumentationsanforderungen für Standards wie MISRA, AUTOSAR und CERT. Die Analyselogik verwendet Machine-Learning-Algorithmen.

Hersteller, Lizenzen und Sprachen				
Hersteller	Scanner	Lizenz	Sprache(n)	Kommentar
https://www.reshiftsecurity.com	reshift	proprietär	Java	Deckt OWASP Top 10 ab. Optimiert für Entwickler und den SDL. Schwachstellen werden nach Risikostufen in der modernen IDE gruppiert und haben detaillierte Beschreibungen. Generiert Codevorschläge für die Behebung der Fehler inklusive GIT Request.
https://www.viva64.com/en/pvs-studio/	PVS-Studio Analyzer	proprietär	C, C++, C#, Java	Verwendet verschiedene Analyseverfahren zur Erkennung von Schwachstellen. Wertet Konstanten und Variablen aus, um Pufferüberläufe zu ermitteln. Trial Version verfügbar.
https://www.xanitizer.com/xanitizer/	Xanitizer	proprietär	Java, JavaScript/TypeScript, Angular	Laut Hersteller hat das Tool über 100 Testfälle und konnte die OWASP Benchmark Test Suite mit 100% Erfolg analysieren. Der Hersteller merkt allerdings an, dass man diese Genauigkeit nicht in echtem Code erwarten darf. Analysiert nur Bytecode.

Vier Hersteller im Featurevergleich				
	CAST Application Intelligence Platform	Perforce Klocwork	Micro Focus Fortify	Veracode
Scan-Engine				
Wie hoch ist die Falsch-positiv-Rate (Schätzung)?	<5%	<10%	k. A.	<5%
Wie hoch ist die Falsch-negativ-Rate (Schätzung)?	<5%	<10%	k. A.	k. A.
Wie viele Testfälle hat das Tool?	k. A.	1000	800	k. A.
Können Kunden eigene Testfälle entwickeln?	–	✓	✓	–
Funktioniert die Analyse auch mit Code, der (noch) nicht kompiliert werden kann?	✓	✓	✓	–
Können Binaries oder Objektcode ebenfalls untersucht werden?	✓	–	teilweise	✓
Kann man falsch positive Ergebnisse für künftige Scans unterdrücken? Wenn ja, ist dieser Mechanismus stabil genug, um auch Änderungen im umgebenden Code zu verkraften?	–	✓ / ✓	✓ / ✓	✓ / ✓
Erkennt das Tool die Implementierung der von ihm vorgeschlagen Methoden der Fehlerbehebung zuverlässig?	✓	✓	✓	✓
Hat das Tool Einschränkungen bezüglich des untersuchten Codes, verwendeter Libraries oder Frameworks, der Binaries oder des Objektcodes?	keine	keine	keine	keine
Kann der Kunde die Standardbewertungen der Testfälle ändern?	–	✓	✓	✓

Vier Hersteller im Featurevergleich				
	CAST Application Intelligence Platform	Perforce Klocwork	Micro Focus Fortify	Veracode
Worauf basiert die Analyse? Kommen nur simple Pattern-Matching-Methoden oder auch komplexere Daten- und Kontrollflussanalysen zum Einsatz?	komplexe Analysen	komplexe Analysen	komplexe Analysen	komplexe Analysen
Installation und Wartung – Installation and Maintenance				
Wird das Tool on Premises installiert oder ist es eine Cloud-Lösung?	beides möglich	beides möglich	beides möglich	Cloud
Kann das Tool (auf einem Server) im Continuous-Monitoring-Betrieb laufen / Code automatisiert untersuchen? Gibt es SIEM-Support?	✓ / ✓	✓ / k. A.	✓ / ✓	✓ / ✓
Ist das Tool in die IDE integrierbar? Falls ja, bietet es automatische Korrekturen für Fehler an?	✓ / k. A.	✓ / ✓	✓ / ✓	✓ / k. A.
Wird das Tool zentral installiert/gewartet oder auf jedem System einzeln?	lokale Installationen, zentrale Sammlung der Ergebnisse	beides möglich	beides möglich, in jedem Fall zentrale Sammlung der Ergebnisse	–
Auf welchen Betriebssystemen läuft das Tool?	Windows, Linux	Windows, Linux, SUN Solaris, IBM Power	Windows, Linux, macOS	–
Welche Versionen der Programmiersprache/Runtime werden unterstützt?	32x und 64x	k. A.	siehe Dokumentation auf Webseite	siehe Dokumentation auf Webseite
Wartung				
Wie häufig kommen neue Features/Versionen?	wöchentlich	alle 3 Monate	2 Major-Produkt-Releases und 4 Rulepack-Updates pro Jahr. Die Rulepack-Updates aktualisieren die für den Scan genutzte Schwachstellendatenbank.	mindestens einmal pro Monat
Gibt es eine (relevante) Downtime bei Updates oder Upgrades?	Downtime < 3 Minuten	Downtime kann vermieden werden	Nein, da es möglich ist, zwei verschiedene Scan-Engines parallel zu installieren. Für den SaaS-Bereich gibt es ein Upgrade ohne Downtime.	Nur in seltenen Fällen handelt es sich um eine Downtime-Release – in diesen Fällen werden die Anwender darauf aufmerksam gemacht.
Lassen sich Updates automatisiert einspielen und benötigen sie mehrere manuelle Schritte?	Updates sind manuell	Updates sind manuell.	Bei dem Cloud-Dienst sind Updates automatisch. On-Premises-Installationen benötigen manuelle Updates.	automatisch (in der Cloud)
Ergebnispräsentation				
Werden gefundene Fehler ausführlich erklärt?	✓, inklusive Lösungsvorschlägen	✓, inklusive Lösungsvorschlägen	✓, inklusive Lösungsvorschlägen	✓

Vier Hersteller im Featurevergleich				
	CAST Application Intelligence Platform	Perforce Klocwork	Micro Focus Fortify	Veracode
Kann man die Ergebnisse einzeln exportieren / in ein externes Ticketing-System überführen? Automatisch?	✓, z. B. Jira	✓ (mittels REST-API)	✓	✓, z. B. Jira, Azure DevOps, GitLab oder GitHub (API)
Wird die Fehlerbehebung ausführlich erklärt?	✓	✓	✓ Es gibt zusätzlich noch die Möglichkeit, das Thema in E-Learning-Videos zu vertiefen.	✓ Veracode bietet zusätzlich die Möglichkeit, Fragen mit Experten via WebEx zu klären.
Verfügt das Tool über ein interaktives Dashboard?	✓	✓	✓	✓
Lassen sich gefundene Fehler revisionssicher dokumentieren?	✓	✓	✓	✓
Sicherheit/Security				
Benötigt das Tool eine Onlineverbindung zum Hersteller?	–	–	–	✓
Übermittelt das Tool (regelmäßig) Daten an den Hersteller? Wenn ja, welche?	–	–	–	✓, aber nur den zu testenden Binärcode
Wie viele Sicherheitspatches wurden für das Tool bisher herausgegeben?	k. A.	<5	k. A.	k. A.
Kosten				
Lizenz (OSS oder proprietär; bei OSS, welche Lizenz?)	proprietär	proprietär	proprietär	proprietär, jährliche Subscription
Wie wird lizenziert – nach Entwicklern / nach Codevolumen / nach untersuchten Systemen / nach Firmengröße / …?	Anzahl Entwickler und LOCs	Anzahl Benutzer	on Premises: nach Anzahl von Applikationen und Installationen oder nach Anzahl der Entwickler; on Demand: Pay-per-Use-Modell bzw. nach der Anzahl der Scans	Lizenzkosten richten sich nach der Anzahl der Application Profiles.
Wie hoch sind die jährlichen Wartungskosten (in Prozent zum Kaufpreis)?	20%	20%, wenn eine Lizenz erworben wurde	on Premises: 28%; on Demand: Support und Wartungskosten sind bereits im Preis enthalten.	drei Stufen für 10, 20 und 27 Prozent des Lizenzwertes

Sprachenvergleich																			
Hersteller	ABAP	C	C++	C#	Delphi/Object Pascal	Haskell	Java	JavaScript	Kotlin	Lua	PHP	Objectiv-C	Python	Ruby	Rust	Swift	Visual Basic	Visual Basic .NET	
Veracode	–	✓	✓	✓	–	–	✓	✓	✓	–	✓	✓	✓	✓	–	✓	✓	✓	vollständige Liste der unterstützten Sprachen und Frameworks auf Webseite

Sprachenvergleich																		
Hersteller	ABAP	C	C++	C#	Delphi/Object Pascal	Haskell	Java	JavaScript	Kotlin	Lua	PHP	Objectiv-C	Python	Ruby	Rust	Swift	Visual Basic	Visual Basic .NET	
Micro Focus	✓	✓	✓	✓	–	–	✓	✓	✓	–	✓	✓	✓	✓	–	✓	✓	✓	Plus: COBOL, ActionScript, APEX, ASP.NET, ColdFusion, Go, JSP, TypeScript, PL-SQL, T/SQL, Scala, ASP VBScript
Perforce	–	✓	✓	✓	–	–	✓	–	–	–	–	–	–	–	–	–	–	–	
CAST	✓	✓	✓	✓	✓	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	vollständige Liste der unterstützten Sprachen und Frameworks auf Webseite

Andreas Wiegenstein

ist Geschäftsführer beim SAP-Cybersecurity-Unternehmen SERPENTEQ.

[/expand]