

Nachhaltige Softwareentwicklung Green Coding – viel diskutiert, selten praktiziert

In der Softwareentwicklung können verschiedene Massnahmen dazu beitragen, den Energieverbrauch von Anwendungen zu senken. Dazu gehören schlanke Programmiersprachen sowie die Unterteilung einer Applikation in kleine Microservices.



Da immer mehr Lebensbereiche auf digitale, vernetzte Tools angewiesen sind, wächst auch der ökologische Fussabdruck der digitalen Infrastrukturen. Green Coding bezeichnet in erster Linie die Praxis, Code so zu schreiben, so dass Anwendungen möglichst wenig Energie verbrauchen. Ein ganzheitlicher Ansatz sollte jedoch auch Green IT einbeziehen, also die Hardware. Dazu gehört beispielsweise, Rechenzentren mit erneuerbaren Energien zu betreiben (vgl. «Nachhaltige Digitalisierung», S. 14) und Hosting-Anbieter zu wählen, die sich Nachhaltigkeitszielen verpflichten (vgl. «Nachhaltigkeit in der Beschaffung», S. 24). Zertifizierungen können dabei helfen, passende Anbieter zu identifizieren.

Um den ökologischen Fussabdruck einer Anwendung zu verringern, braucht es zuerst Messungen. Auf dieser Basis lassen sich Massnahmen ableiten, die wirksam zur Reduktion beitragen. Entscheidend ist dabei der Energieverbrauch. Open-Source-Tools wie PowerJoular erfassen in Echtzeit den Stromverbrauch und zeigen die Belastung durch Anwendung und Hardware-Komponenten. Im Backend sind genaue Messungen möglich, im Frontend nur Schätzungen. Dafür stehen spezielle Online-Tools zur Verfügung.

Das richtige Werkzeug für die Aufgabe

Doch nicht nur der Betrieb, auch die Entwicklung selbst kostet mittlerweile beträchtlich Energie. Besonders die zunehmende Nutzung von KI in der Softwareentwicklung treibt den Verbrauch in die Höhe. Dabei ist zu beachten, dass grosse Sprachmodelle (LLMs, vgl. S. 58) die aktuell gängige Entwicklungspraxis abbilden. Es wäre naiv zu erwarten, dass ein LLM im Jahr 2026 die aktuellsten Best Practices für energieeffizientes Programmieren kennt.

Auch die Wahl der Programmiersprache beeinflusst die Energieeffizienz. Python etwa ist vielseitig, als Grundlage für Webserver jedoch verschwenderisch. Kompilierte Sprachen wie Go oder Rust eignen sich deutlich besser. Für die Entwicklung von kleinen Anwendungen ist es ebenfalls wichtig, die richtigen Tools zu wählen: Ein Javascript-Framework mit vielen Zusatzoptionen wäre zu ressourcenhungrig, besser eignet sich reines Javascript (vanilla JS) oder Static Site Generation (SSG).

Kleine und effiziente Container

Mehrere Ansätze helfen dabei, den Stromverbrauch von Software im laufenden Betrieb zu senken. Entwickler können etwa den Datenaustausch und die Abhängigkeiten reduzieren, indem sie für Container-Setups das kleinstmögliche, aber dennoch ausreichende Image wählen. In erster Linie sollte der Fokus darauf liegen, nur notwendige Dependencies zu verwenden und Ausschau nach weiteren leichtgewichtigen Paketen zu halten. Auch der Einsatz schlanker Webserver spart Ressourcen, insbesondere wenn Frontend und Backend in getrennten Containern betrieben werden. Das Institut Public Sector Transformation der BFH setzt beispielsweise auf Node-basierte Anwendungen. Diese laufen auf leichtgewichtigen Container-Betriebssystemen, wodurch sich die Containergrösse um rund 70 Prozent verringert – mit spürbaren Vorteilen für Geschwindigkeit und Energieverbrauch beim Hosting. Anstelle von Ubuntu kann dabei problemlos Alpine Linux verwendet werden. Darüber hinaus empfiehlt es sich, Anwendungen in Microservices aufzuteilen, die mehrfach genutzt werden können. Auf diese Weise lassen sich redundante Dienste vermeiden.

Unsere Empfehlungen



1. Grüne Infrastruktur wählen

Hosting-Anbieter auswählen, die auf erneuerbare Energien setzen (z. B. ISO 14001:2015/ Amd:2024 und/oder ISO 50001:2018).

2. Schlanke Programmierung

Container möglichst klein halten, redundante Dienste vermeiden und vorhandene Services wiederverwenden.

3. Bewusster Einsatz von KI

Spezialisierte, kleine KI-Modelle verwenden, um die Programmierung zu verbessern.

Mehr Informationen



Kontaktmöglichkeiten und weitere Informationen zu nachhaltiger Softwareentwicklung:
bfh.ch/ipst/nachhaltige-digitalisierung

Kontakt



Alexandre Thomas

Tech-Lead Digital Sustainability Lab

alexandre.thomas@bfh.ch

T +41 31 848 61 40



Aaron Sägesser

Software-Entwickler

aaron.saegesser@bfh.ch

T +41 31 848 52 65