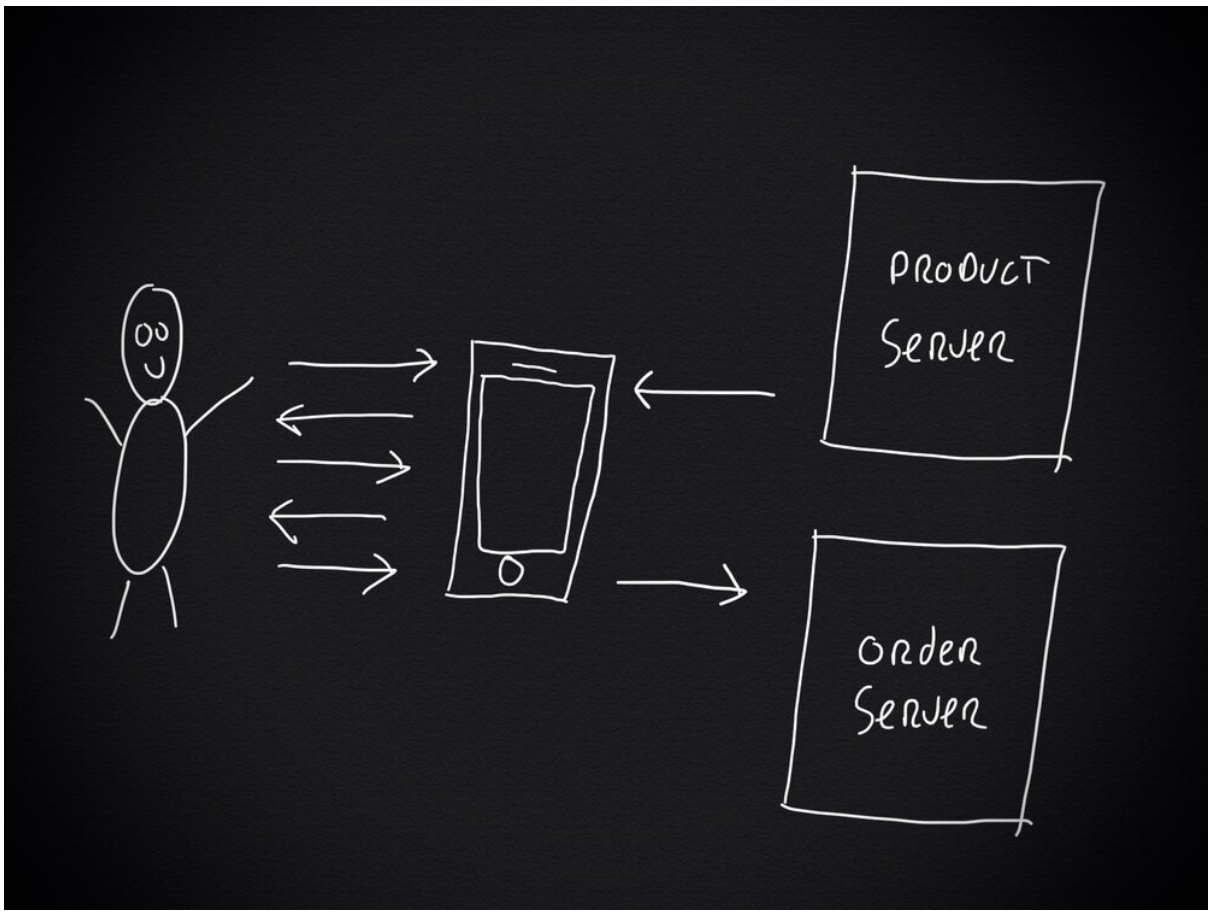
# Skalierbares Anwendungsdesign ohne Magie

*Nutzung der Client-Rechenleistung für hohe Performance bei vielen Benutzern*

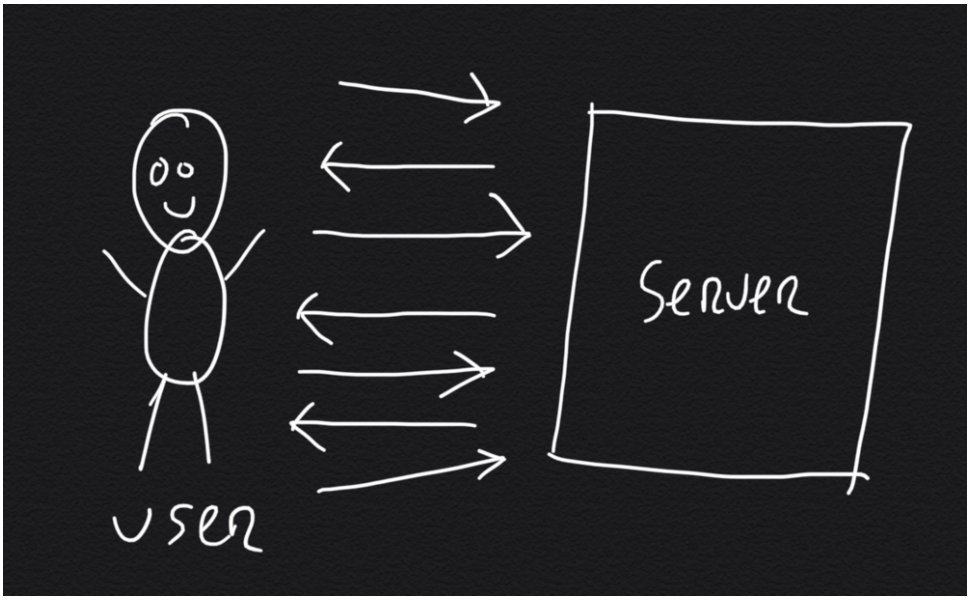
Willem L. Middelkoop
May 11, 2020




**Im Rahmen der Online-Bestell-App, die ich gerade entwickle, musste ich eine skalierbare Backend-Infrastruktur entwerfen, die viele gleichzeitige Benutzer verarbeiten kann. Skalierbarkeit gilt als ein schwer zu lösendes Problem. Oft wird sie so dargestellt, als wäre sie etwas Magisches, das von millionenschweren Unternehmen mit geheimen Werkzeugen bewerkstelligt wird. Aber so etwas wie Magie gibt es nicht, oder doch?**

## Was ist Skalierbarkeit?

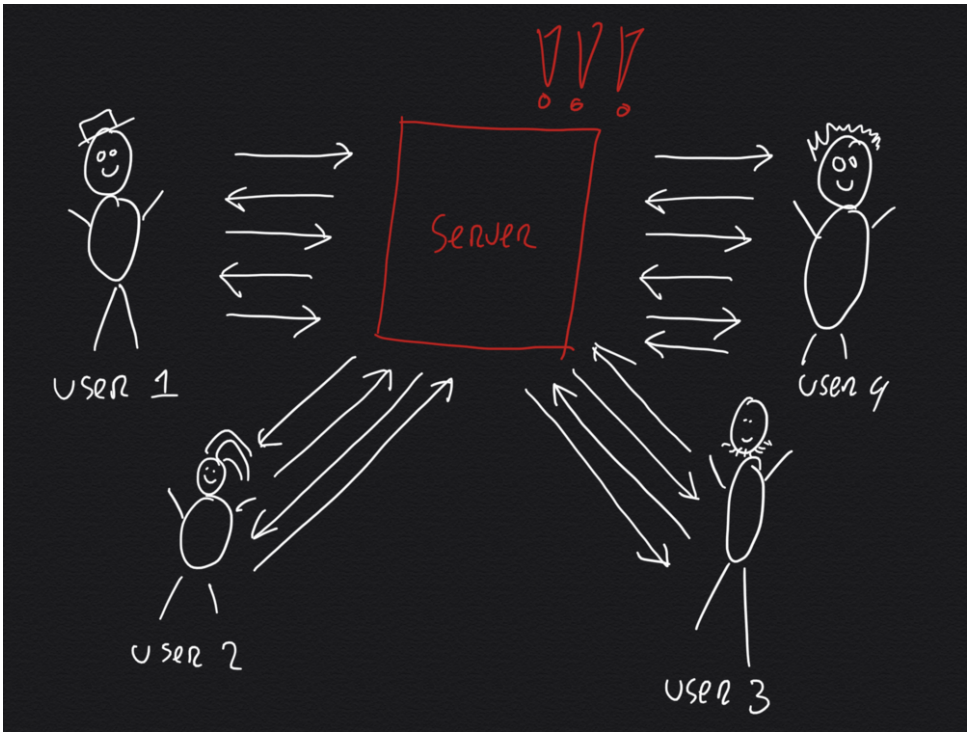
Wenn Sie eine App entwickeln, fangen Sie wahrscheinlich sehr klein an. Nur ein paar Benutzer (oder nur Sie, der Entwickler) arbeiten mit der App. Alles wird bestens funktionieren: Die App funktioniert gut, schnell und es gibt keine Probleme. *(Genießen Sie dieses Gefühl!)*




*Ein Benutzer interagiert mit einem Server*

In diesem frühen und kleinen Szenario kann der einzelne Server, der die App ausführt, die Interaktion des Benutzers damit bewältigen. Jedes Mal, wenn der Benutzer eine Taste tippt oder klickt, erledigt der einzelne Server etwas Arbeit. Wenn Sie gut programmieren, können diese Arbeitslasten in effiziente Arbeitsabschnitte optimiert werden. Dies ist ein typisches Beispiel für eine monolithische Software Architektur .

Auch wenn Ihr Projekt oder Ihre App eher klein ist, kann es irgendwann zu einem Zustrom von Benutzern kommen. Wenn Ihr System hohe Lasten nicht bewältigen kann, besteht eine hohe Wahrscheinlichkeit eines Ausfalls.




*Server unter hoher Last, der mehrere Benutzer gleichzeitig bedient*

Aufgrund des monolithischen App Designs interagieren alle Benutzer mit demselben Server. Ihre gesamte Interaktion (Tippen, Klicken, Eingabe) wird von diesem Server verarbeitet. Bei erhöhter Last muss er viel härter arbeiten. An einem bestimmten Punkt werden Sie feststellen, dass sich die Dinge verlangsamen, weil der Server mit der Menge an Arbeit, die er zu erledigen hat, nicht Schritt halten kann. Das bedeutet, dass Ihre App nicht gut skaliert, mit anderen Worten: Sie sind in Schwierigkeiten.

### Ausfall im "entscheidenden Moment"

Eine schlechte Skalierung ist ein großes Problem, das niemand unterschätzen sollte. Die meisten Apps (und deren Geschäftsmodelle) sind auf hohe Volumina mit geringen Einnahmen pro einzelnem Benutzer angewiesen. Wenn Sie möchten, dass Ihre App ein (finanzieller) Erfolg wird, müssen Sie sicherstellen, dass sie gut funktioniert, wenn der Traffic plötzlich ansteigt. Wenn Ihre App abstürzt, wenn sie plötzlich die Aufmerksamkeit der Massen auf sich zieht, verpassen Sie eine einmalige Gelegenheit für (organisches) Wachstum.

In gewisser Weise ist das Entwerfen für Skalierbarkeit wie das Anlegen des Sicherheitsgurts, wenn Sie in Ihrem selbstgebauten Raketenschiff mit experimentellen Triebwerken sitzen. Man weiß nie genau, wann es zünden wird, aber wenn es soweit ist, sollten Sie sich besser gut festhalten (und die Fahrt genießen!).

## Gängige Wege zur Skalierbarkeit

Es wurde viel über die Entwicklung skalierbarer Apps geschrieben. Gängige Wege zur Verbesserung der Skalierbarkeit sind:

- **Erhöhung der Serverkapazität**: Hinzufügen von mehr Speicher, CPU-Leistung,

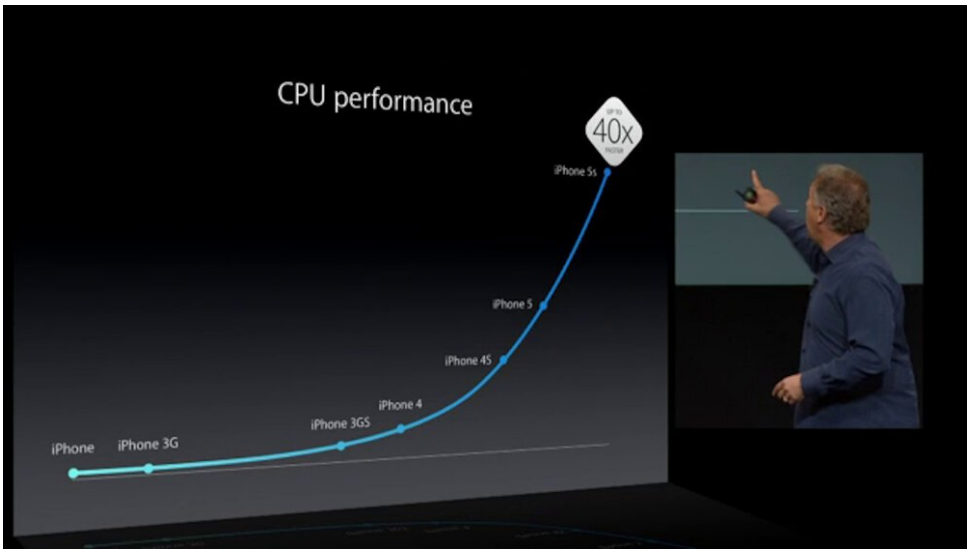
Speicherplatz. Das wird funktionieren, aber es wird Sie nur "bis zu einem gewissen Punkt" bringen.

- **Hinzufügen weiterer Server**: Anstatt eines einzelnen Servers, der als Flaschenhals fungiert, können Sie weitere Server hinzufügen, die sich die Arbeitslast teilen. Das ist leichter gesagt als getan, da es Load Balancing und Datenfreigabe erfordert, was schwierig sein kann (z. B. wenn Sie einen Bestellzähler haben und zwei Server zählen, welcher bestimmt dann die nächste Nummer?).
- **Optimierung des Programmcodes**: Verwenden Sie die richtigen Werkzeuge! Wählen Sie eine leistungsorientierte Programmiersprache und entsprechende Serversoftware. Schauen Sie sich die funktionale Programmierung an, damit Ihr Code asynchron auf mehreren Kernen laufen kann. Machen Sie ihn zustandslos. Benchmarken Sie ihn. Optimieren Sie ihn. Jede Millisekunde, die bei einer einzelnen Anfrage gewonnen wird, summiert sich schnell, wenn Sie große Volumina verarbeiten.
- **Verwenden Sie die Datenbank und trennen Sie sie vom Anwendungsserver**: Wenn Sie eine Datenbank verwenden, nutzen Sie deren Leistung! Nutzen Sie Query Caching, Indizes und Suchfunktionen. Die meisten Datenbanklösungen bieten bewährte Clustering-Optionen; versuchen Sie nicht, das Rad neu zu erfinden, das andere bereits perfektioniert haben.

Das Beste, was Sie tun können, ist, all diese Optionen in Betracht zu ziehen. Auch wenn Sie nicht sofort weitere Server hinzufügen, sollten Sie Ihren Code so schreiben, dass er später unterstützt wird. Das Benchmarking und die Optimierung Ihres Codes sollten ein integraler Bestandteil Ihrer Arbeit sein, nicht nur ein nachträglicher Gedanke.

## Nutzung der Client-Rechenleistung

Wenn Sie alle gängigen Wege zur Skalierbarkeit in Betracht gezogen haben, gibt es noch "eine Sache".

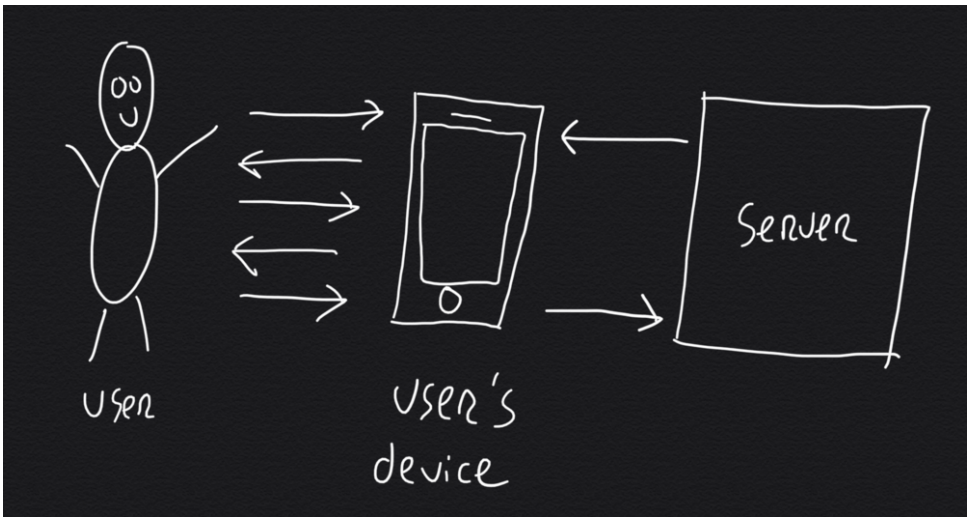



*Smartphones (und Tablets, PCs) sind im Laufe der Jahre leistungsfähiger geworden (image: Apple)*

Nur wenige Entwickler bedenken dies: Das Skalierbarkeitsproblem enthält die Lösung! Wenn Sie 1000 Benutzer haben, haben Sie 1000 Computer mit leistungsstarken CPUs

und viel Speicher. Moderne Computer und Smartphones sind immer leistungsfähiger geworden.

Sie müssen einfach einen Weg finden, die Rechenleistung des Clients zu nutzen. Auch wenn es sich nach einer bösen Sache anhört (die Rechenleistung des Benutzers zu nutzen), wird es ihm eine viel bessere und schnellere Erfahrung bieten. Das Beste daran ist, dass *seine* Rechenleistung für *seine* fantastische Erfahrung aufgewendet wird, wodurch Ihr Skalierbarkeitsproblem als glücklicher Nebeneffekt gelöst wird.



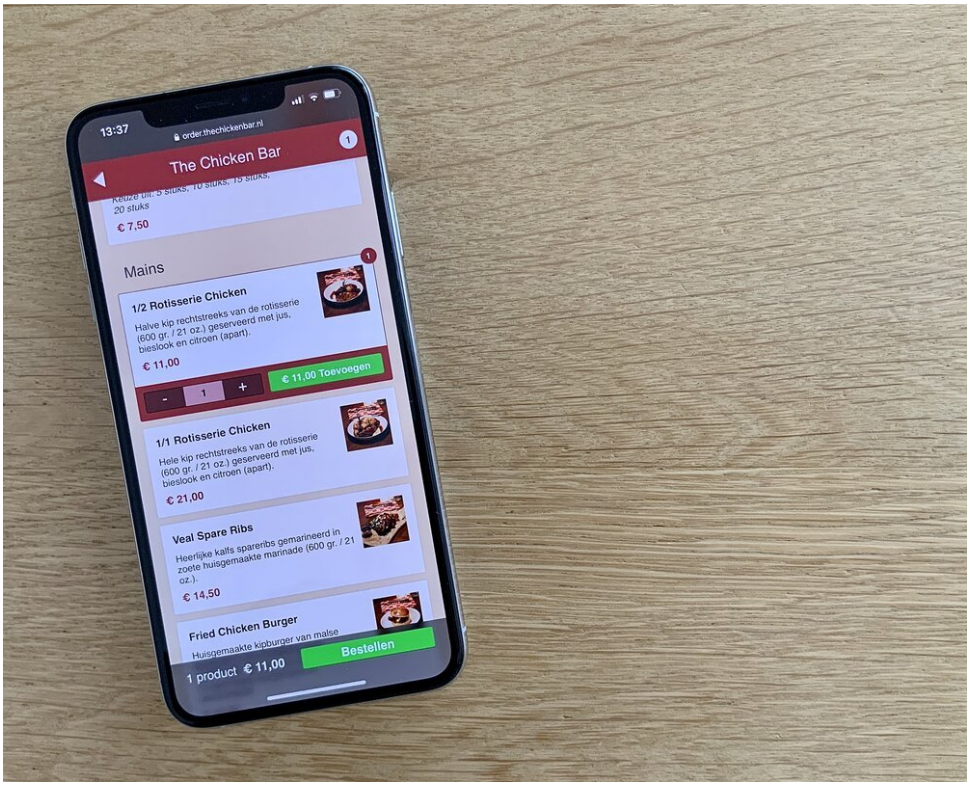

*Die Leistung des Benutzergeräts nutzen: Das Telefon des Benutzers übernimmt den Großteil der Arbeit, die ursprünglich vom Server erledigt wurde*

Indem Sie die Rechenleistung des Benutzers nutzen, reduzieren Sie die Menge an Arbeit, die der Server zu erledigen hat. Anstatt dass der Server *jedes* Mal, wenn der Benutzer eine Taste tippt oder klickt, ein wenig Arbeit erledigt, wird diese Arbeit nun größtenteils vom Gerät des Benutzers erledigt. Dies ist effizienter als die Kommunikation über eine WLAN-/4G-Verbindung mit einem Server (spart Akkulaufzeit und Zeit).

Es erfordert ein wenig Umdenken, aber dieses Prinzip kann auch auf Webanwendungen angewendet werden. Anstatt alle Dinge auf dem Server zu erledigen, können Sie viele Arbeiten auf dem Client ausführen. Dinge wie Suchen, Browsen, Filtern, Navigieren und sogar das Hinzufügen von Artikeln zu einem Warenkorb können mit clientseitigem JavaScript erledigt werden. Sie - lieber Entwickler - müssen nach Wegen suchen, dies zum Funktionieren zu bringen, aber wenn Sie Erfolg haben, haben Sie nahezu unendliche Skalierbarkeit in Ihren Händen.

## In der Praxis: Lebensmittelbestell-App

Für die Lebensmittelbestell-App habe ich nach Wegen gesucht, die Rechenleistung des Clients zu nutzen, um eine hohe Leistung zu erzielen. Die größte Herausforderung bei Liefer- und Abholbestellungen besteht darin, dass sie zur Essenszeit ihren Höhepunkt erreichen. Viele Leute nutzen die App ungefähr zur gleichen Zeit, dies ist ein Rezept für Skalierbarkeitsprobleme.

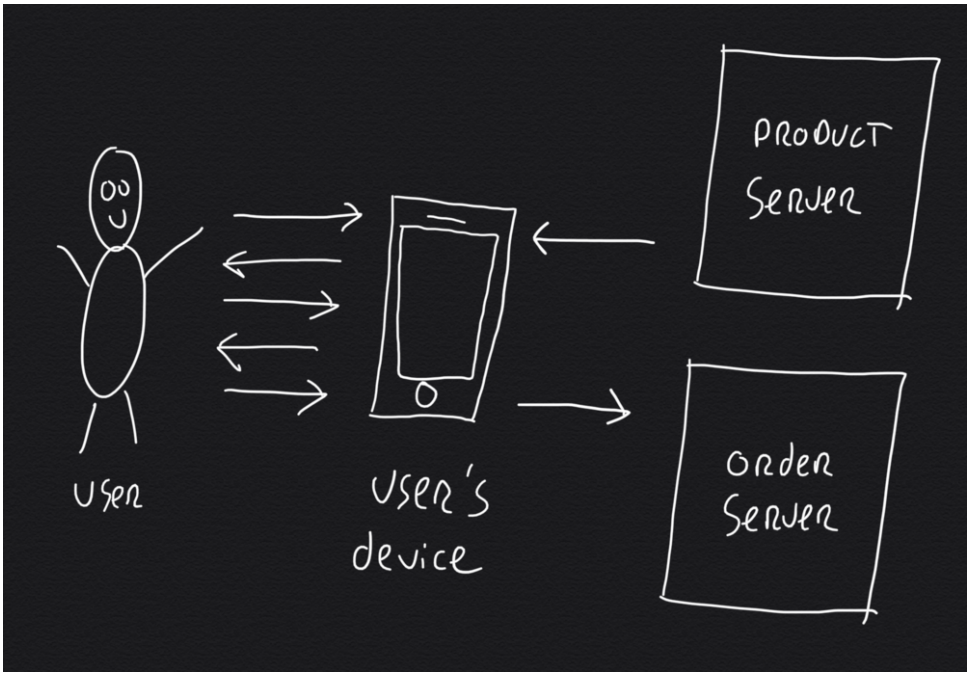



*Online Essen bestellen*

Wenn Sie an die Online-Bestellung von Lebensmitteln denken, können Sie den gesamten Prozess in kleinere Abschnitte unterteilen:

- **1) Öffnen der Seite**: Laden der App/Seite
- **2) Durchsuchen der verschiedenen Optionen**: Auflisten der Produkte, Anzeigen ihrer Beschreibungen, Preise, Fotos usw.
- **3) Suche nach etwas Bestimmtem**: Suche nach Kategorie, Produktname usw.
- **4) Auswahl eines Produkts**: Hinzufügen zur Bestellung
- **5) Anpassen eines Produkts**: Auswahl einer Sauce, Beilage, Topping usw.
- **6) Eingabe Ihrer Daten**: Ihr Name, Telefonnummer, Lieferdetails usw.
- **7) Bezahlen Ihrer Bestellung**: Auswahl einer Zahlungsmethode, Verbindung zu Ihrer Bank oder Kreditkarte
- **8) Schließen der App**

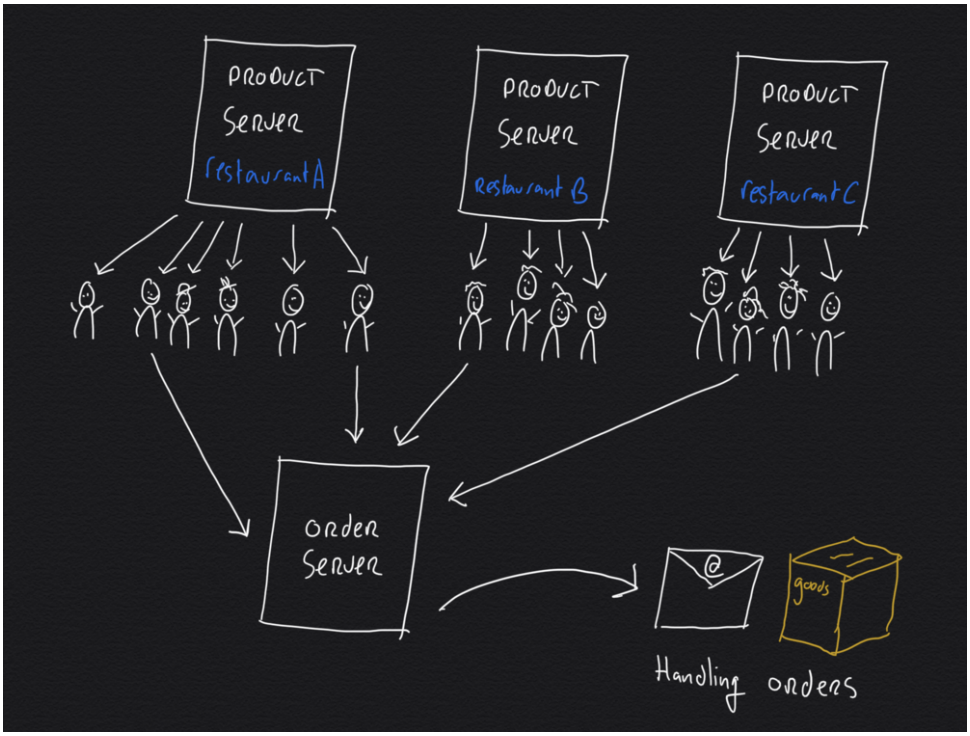
Die meisten dieser Schritte können so programmiert werden, dass der Server *nicht notwendig* ist. Nur Schritt 1 (**Laden**) und Schritt 7 (**Bezahlen**) erfordern Kontakt mit der Backend-Infrastruktur. Es ist wichtig zu erkennen, dass die Schritte 2 bis 5 mehrmals wiederholt werden, da es üblich ist, dass Benutzer mehrere Produkte zu ihrer Bestellung hinzufügen.




*Die verbleibende Serverlast trennen: Produkte anbieten und Bestellungen bearbeiten*

Durch die Trennung der verbleibenden Arbeitslast nach Aufgaben kann die Software weiter optimiert werden. Der **"Produktserver"** ist optimiert für die Bereitstellung statischer Assets (Texte, Preise, Bilder), die es dem Client ermöglichen, das Produkterlebnis zu erstellen. Sie können diese Art von Server stark optimieren, indem Sie Caching, HTTP/2, Komprimierung usw. nutzen.

Nur die Benutzer, die den Bestellvorgang abschließen, werden an den **"Bestellserver"** weitergeleitet, der die Zahlung abwickelt. Er wird sich mit dem Zahlungsanbieter verbinden. Sobald eine Zahlung abgeschlossen ist, wird der Zahlungsanbieter den Bestellserver kontaktieren, um ihn über den Zahlungsstatus zu informieren. Dies ist etwas, worüber Sie die volle (serverseitige) Kontrolle übernehmen möchten, so wie ich es getan habe, als ich zuvor ein Zahlungssystem entworfen habe . Die verbleibende Arbeitslast auf dem "Bestellserver" ist viel geringer, da nicht jeder eine Bestellung aufgeben wird, und ein Großteil der Auftragsabwicklung kann im Hintergrund erfolgen.




*Umgang mit hohem Traffic durch verteiltes Rechnen*

Um die Skalierbarkeit zu maximieren, können Sie problemlos mehrere Server hinzufügen (oder ein Content Delivery Network verwenden), um die statischen Assets bereitzustellen. Für die Lebensmittelbestell-App verwende ich *pro Restaurant* einen separaten "Produktserver". Dies ermöglicht eine hohe Leistung und reduziert die Ladezeiten. Die Leute merken, dass es schnell ist, manche können nicht glauben, dass es Webtechnologie ist, es ist *fast* magisch.

## Fazit

Das Entwerfen einer skalierbaren Anwendung kann ohne Magie oder Millionen-Dollar-Budgets erfolgen. Man muss es nur durchdenken und nach Möglichkeiten suchen, die sich zusammen mit den Herausforderungen bieten.

Das Problem enthält die Lösung. Der Umgang mit vielen gleichzeitigen Benutzern ist nicht nur ein Problem: Sie bringen ihre eigene Rechenleistung mit, es ist die Lösung! Sie müssen sie nur nutzen.