

Hoe als SaaS-bedrijf jezelf futureproof maken

13 oktober 2022, 11:24

Nick Boucart

Als Sirris-expert krijg ik regelmatig vragen van SaaS-ondernemers rond technische issues. Deze zijn voor heel wat startups en scaleups dagelijkse realiteit, en daarom kunnen de antwoorden en oplossingen die ik bied ook voor andere bedrijven nuttig zijn. Enkele recente voorbeelden van zulke vragen en hoe ze aan te pakken.

"Ik maak me zorgen of we wel goed bezig zijn."

Aan het woord één van de oprichters van een veelbelovende SaaS-scaleup: 'Ik merk dat we de laatste tijd minder snel vooruit gaan, onze klanten vinden geregeld nieuwe bugs en eerlijk, ik stel me soms wat vragen of we wel de juiste technologie gebruiken voor ons platform'.

Dit soort vragen krijg ik wel vaker. Een SaaS bouwen is niet simpel en de spanning tussen de business (sales, marketing, klanten) enerzijds en het technische team dat alles moet bouwen anderzijds, is soms best pittig. De eersten willen liefst tonnen functionaliteit, hier en nu, terwijl developers vaak erg diep in hun code zitten.

Twee werelden die botsen

Als ik zo'n vraag krijg, stel ik meestal voor om een halve dag samen met het team aan tafel te zitten: ik stel dan open vragen over hoe het team werkt, welke technologie ze gebruiken en vooral, hoe ze die gebruiken. Hoe worden requirements opgesteld? Hoe wordt er aan planning gedaan?

Worden er unit-tests of andere (geautomatiseerde) testen ontwikkeld? Etc.

Bijna altijd lukt het me wel om na zo'n gesprek 4-5 concrete aanbevelingen te doen om business en technisch team dichter bij mekaar te brengen, om zo een aanzet te geven het vertrouwen tussen business en development te verbeteren. Op het einde van de rit geldt, toch wat mij betreft, dat hoe beter business en technisch team elkaar begrijpen, hoe beter ze mekaars gevoeligheden en metier kennen, hoe groter de kans op slagen.

"Hoe krijgen we onze software opgeschaald?"

"Onze klanten draaien veel meer simulaties dan vroeger en onze server krijgt het moeilijk," aldus het hoofd van R&D bij een ingenieursbureau dat de afgelopen jaren sterk gegroeid is, en een deel van zijn knowhow in zijn domein heeft omgezet in een SaaS-product.

"Onze ingenieurs doen onderzoek in ons domein, en ontwikkelen en verfijnen onze algoritmes en berekeningsmethodes de hele tijd. Deze Python-scripts dienen als basis voor onze technische partner, die deze integreert in een webapp/SaaS-platform dat we sinds enige tijd succesvol vermarkten. Wat we nu merken, is dat onze huidige architectuur en infrastructuur het moeilijker krijgen om tijdig onze simulaties te draaien en we zoeken naar manieren om dat robuuster te maken. Zou de cloud ons daar kunnen helpen en zo ja, hoe doen we dat?"

Het is zeker niet de eerste keer dat ik die vraag krijg. Technische teams kijken wel vaker naar de cloud als een mogelijke oplossing om hun SaaS-platform beter te doen schalen. Maar eerlijk is eerlijk, de drempel om met de cloud te beginnen, is best wel hoog. Kijk gewoon even naar de productpagina's van AWS of Azure, en je wordt overspoeld met buzz-words, lijsten van services, inspirationeel bedoelde cases waarmee een gewone sterveling zich zelden kan identificeren, en prijspagina's waarvoor je hogere studies in de wiskunde moet gedaan hebben om die te begrijpen. Daar een beetje wegwijs in geraken en een gevoel krijgen over hoe een combinatie van de verschillende compute- en storage-opties eruit zou kunnen zien voor je eigen use case, dat vraagt vaak het overwinnen van een serieuze leercurve.

Wat ik dan meestal doe, is aan de hand van voorbeeldarchitecturen verschillende mogelijkheden schetsen van hoe zo'n probleem zou kunnen aangepakt worden, om dan ook stil te staan bij de verschillende implicaties van die keuzes. Ik doorloop dan samen met het team de pro's en contra's van bepaalde technische keuzes, en laat hen zeker ook stilstaan bij operationele aspecten zoals monitoring, (auto-)scaling en security. Tijdens zo'n proces blijkt vaak dat het technische team niet zo mee is met trends zoals 'serverless' of containers, en daarom niet kan inschatten wat dat voor hen zou kunnen betekenen.

In bovenstaand geval hebben we aan de hand van een proof-of-concept bekeken hoe AWS Lambda, AWS S3 en AWS Step Functions de rekenmotor een pak schaalbaarder en transparanter kon maken. Door via een praktische proof-of-concept te werken, kreeg het team ook inzicht in aspecten zoals IAM, infrastructure-as-code, AWS Cloudwatch, ... allemaal dingen die sowieso vroeg of laat aan bod komen, maar die wel maken dat de leercurve om met de cloud te beginnen, stijl is.

Organiseren voor verandering

Wanneer ik met een development team van een startup of scaleup praat, probeer ik altijd uit te zoeken hoe ze zich organiseren voor verandering. Want als er nu één constante is wanneer het

gaat over software, dan is het wel verandering: elk contact met klanten, prospects of gebruikers geeft nieuwe inzichten en ideetjes voor verbetering van het product, nieuwe teamleden komen erbij en soms vertrekt er wel eens iemand, nieuwe cloud/open-source-bouwblokken bieden potentieel nieuwe mogelijkheden. De vraag is dus niet zozeer "wat gaat er morgen weer veranderen?", de vraag zou eerder moeten zijn "hoe kunnen we onze softwareontwikkeling zo organiseren dat we de veranderingen van morgen het beste de baas kunnen?"

Zo vraag ik altijd om in grote lijnen de architectuur van de software uit te leggen. Mijn vuistregel om 'veranderingsproof' te zijn, is: 'the simplest solution that could possibly work'. Niet dat het per se fout is, maar een team van drie dat microservices bouwt, maakt het zichzelf niet gemakkelijk, zeker niet wanneer in de praktijk voor quasi elke aanpassing ook elke microservice aangepast moet worden.

Een ander hulpmiddel om beter om te kunnen met verandering, zijn geautomatiseerde tests. Denk hierbij in de eerste plaats aan unit tests, eventueel een aantal functionele en/of end-to-end tests. Er zijn meerdere redenen waarom ik geautomatiseerde tests hoog inschat:

- Ze zorgen voor een vangnet voor de developers: een degelijke unit test coverage geeft ontwikkelaars snel feedback over of wat gisteren nog werkte, vandaag ook nog werkt.
- Geautomatiseerde tests zijn ook een vorm van documentatie: ze tonen hoe code werkt en gebruikt kan worden, zowel nu als in de toekomst.

Geautomatiseerde tests helpen voor mij zowel met het 'onderhoudbaar' houden van de bestaande code, als met het 'veranderproof' maken van het team.

Herkenbaar? Altijd bereid om eens van gedachten te wisselen!

Check ook onze Masterclass Cybersecurity - SaaS, Digital Platforms, Software

Auteurs



Nick Boucart