Stylers

Stylers Group

Create Value With Us

Vállalati rendszerek
microservice architektúrával

A microservice architektúra napjaink egyik legfelkapottabb témája IT körökben, mivel kiépítése számos üzleti előnnyel jár. A megoldás alkalmazásának nincsenek korlátai, viszont most kiragadunk egy platformot, az e-commerce fejlesztéseket, és azt vizsgáljuk meg, hogy integrált webshopok esetén miért éri meg a microservice architektúra mellett döntened.

Ahhoz, hogy a kérdéskört felderítsük, először érdemes szót ejtenünk magáról a fejlesztési folyamatról. Bármilyen e-commerce projektről legyen szó, általánosan érvényes, hogy három fejlesztési típus közül választhatnak a vállalatok:

Dobozos termék - Megpróbálhatod kiváltani vele azokat az igényeket, amelyekre szükséged van, ám elképzelhető, hogy előbb-utóbb akadályokba, korlátokba fogsz ütközni, mert valami nem úgy működik, ahogy te szeretnéd. Ekkor elkerülhetetlenné válik a termék átalakítása, személyre szabása.
Félkész termék továbbfejlesztése - Ennél a megoldásnál a piac általában monolitikus rendszereket (pl. Magento) nyújt, amelyek jó képességűek, viszont a tanulási idejük nagy mind a felhasználó, mind a fejlesztő számára. Ezeknél rengeteg olyan komponenst kell életben tartanod, amelyekre egyáltalán nincs szükséged ahhoz, hogy az üzleti tevékenységedet jól tudd végezni.
Egyedi fejlesztés - Amikor már nagyon szűk a nadrág és egyediek az igények, akkor érdemes az egyedi fejlesztés felé fordulni. Itt monolitikus rendszerben vagy microservice architektúrában gondolkodhatsz - mi az utóbbit szoktuk javasolni, a következő pontban kifejtük, hogy miért.

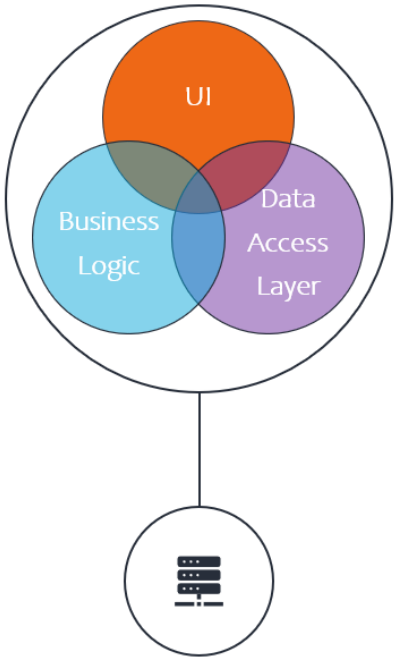
Microservice architektúra vs. Monolitikus architektúra

Egy klasszikus monolitikus rendszernél, ha bele akarunk avatkozni bármilyen folyamatba, hogy változtassunk rajta (pl. egy új komponenst akarunk elhelyezni a szoftverben), az gyakorlatilag a teljes architektúrát, a teljes szoftvert érinteni fogja. Emiatt a bevezetés mindig nehézkes, hiába csak apró módosításokat akarunk végrehajtani, hiszen rengeteg következményt von maga után minden ilyen lépés.

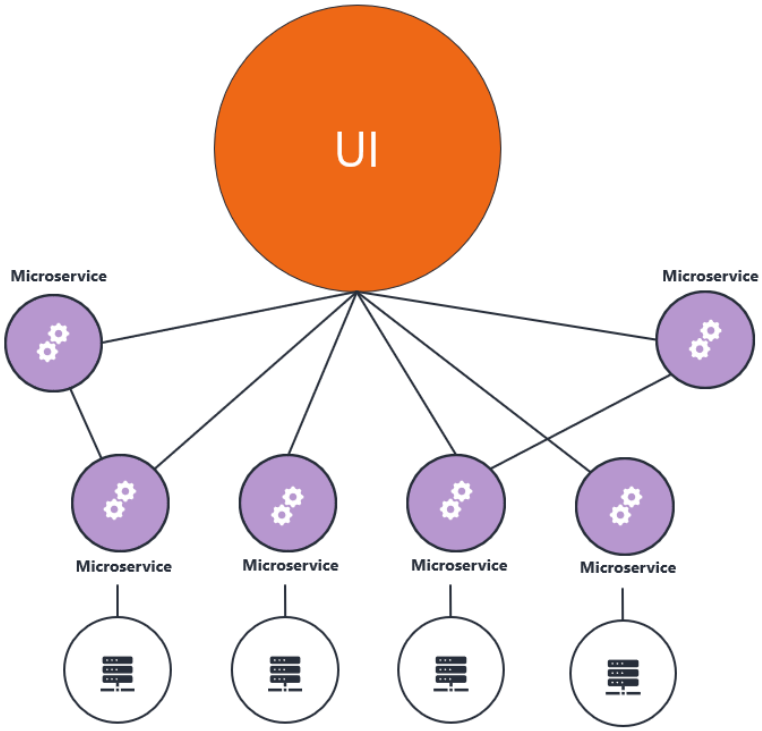
A microservice architektúránál viszont ahelyett, hogy mindent egy nagy szoftverrel próbálnánk megoldani, egy sor különböző apró szoftverek együttműködéseként hozzuk létre a rendszert. Ennek a logikájára egy jó párhuzam, ha arra gondolunk, hogy a focicsapatokban is különböző képességű játékosok játszanak különböző pozíciókon, és nem egyfajta játékosból, mondjuk a csatárból van 11 darab. Könnyen belátható, hogy egy ilyen rendszer nagyobb fokú rugalmasságra képes, és másfajta lehetőségei vannak.

Az apró service-ek, amelyek kiszolgálják a rendszer egészét, hálózati módon kapcsolódnak egymáshoz, és ezeket a hálózati tulajdonságokat viszonylag egyszerűen vagy rugalmasan fel lehet oldani - sőt, a service-eket könnyen ki lehet váltani egymástól függetlenül is. Tehát ha találunk egy jobb megoldást valamelyik rendszerre, mondjuk a felhasználókezelésünket szeretnénk megváltoztatni, akkor ez nem igazán érinti a többi komponenst, mert azok az API végpontok, azok

az interface-ek, amelyekkel egymáshoz kapcsolódnak a részelemek, nem fognak változni attól, ha A vagy B megoldást szeretnénk alkalmazni a rendszeren.



Monolitikus architektúra



Microservice architektúra

Fontos kiemelni még a microservice architektúrával kapcsolatban azt is, hogy a UI kifejezetten el van választva a backendes magoktól, így ha valamilyen trendváltozás következik be, akkor elég csak azt a réteget érinteni, nem kell a teljes rendszerünket átírni.

A microservice architektúra előnyei

Skálázhatóság: a microservice világot alapvetően a skálázhatóság miatt találták ki, létrehozásakor fontos szempont volt, hogy mind horizontálisan, mind vertikálisan rugalmas rendszereket lehessen alkotni. Ez azt jelenti, hogy az egyes komponensek terhelhetőségét önmagukban is lehetséges növelni vagy csökkenteni. Gondoljunk csak arra, hogy pl. egy ünnepi időszakban sokkal több megrendelés érkezik egy oldalra: ilyenkor nem kell az egész rendszer terhelését megváltoztatnunk, hanem elég csak azokat a kritikus pontokat alakítani, ahová a nagyobb terhelés érkezik (pl. a megrendelés folyamatában vagy a fizetési folyamattal kapcsolatos adatoknál). Ennek az az egyik következménye, hogy valószínűleg valami felhőalapú módszerrel érdemes a fejlesztést végrehajtani, de ez általában kényelmessé teszi a folyamatot. Illetve, mivel az egyes komponensek önálló módon skálázhatóak, a költségeink hosszútávon csökkennek a bevezetés után.

Fejlesztőktől vagy fejlesztő vállalattól való függetlenség: mivel a komponensek, amelyekből összeáll maga a teljes service, kicsik és függetlenek egymástól, a fejlesztőtől való függőségünk is csökken. Ez azt is jelenti, hogy a tanulási görbét, amely egy-egy komponensnek a betanulásával járna, nem

kell az ügyfélnek megfizetnie, hiszen pár óra alatt elsajátítható az egyes részeknek a logikája - így nincs az a hosszú tanulási folyamat, mint egy monolitikus rendszernél, ahol akár fél-egy év is lehet, amíg egy fejlesztő teljesen kiismeri és hatékonyan tudja kezelni az adott rendszert. Mindez azt is jelenti, hogy a microservice architektúra esetén a továbbfejlesztés is költséghatékonyabb tud lenni.

A fejlesztési idők rövidülnek: az alacsony tanulási görbe és a hálózati kommunikáció miatt a fejlesztési idők lerövidíthetőek, ami hosszú távon szintén költséghatékonyabbá teszi a fejlesztést.

Integrációbarát architektúráról beszélünk: mivel ezek a hálózati kérések könnyen bővíthetőek vagy csökkenthetőek valamilyen irányba, sokkal gyorsabban tudunk integrálni újabb megoldásokat (ERP, CRM, 3rd party megoldások). Ez is a monolitikus rendszerekkel való összehasonlítással nyer értelmet: rengetegszer találkozunk például azzal a helyzettel, hogy egy ügyfélnek egyedi marketingautomatizációs rendszere van, vagy a termékeket valamilyen egészen más rendszerben állította elő, esetleg valamilyen logisztikai lánccal kell együttműködnie a rendszernek. Ilyenkor jellemzően egy hagyományos, monolitikus e-commerce rendszernél elkezdjük megpróbálni kikapcsolni ezeket a szolgáltatásokat a rendszerből. Tehát nem csak magával az integrációval megy el idő, hanem azzal is, hogy kikapcsoljuk azokat a szolgáltatásokat, amelyeket korábban hoztak létre. Egy mai trendben, amikor általában legalább 15-20 külső szolgáltatással is együtt kell tudni működni, egy hagyományos monolit rendszernek az értelme veszik el, hiszen annyi mindent váltunk ki, hogy szinte egy teljesen más szoftverről beszélünk, mint amelyből eredetileg kiindultunk.

Növeli a felhasználói élményt: mivel rugalmasan és gyorsan ki tudunk cserélni egy-egy service-t és az azokhoz tartozó megjelenési formákat, a felhasználói élmény gyorsan változtatható és adaptálhatóak az aktuális igények, illetve a UX függősége is nagyon alacsony lesz.

A szoftverfejlesztők szeretnek dolgozni vele: rengeteg zöldmezős beruházást vagy megoldást jelent az, ha microservice architektúrával dolgozunk, hiszen a változó ügyféligények miatt csak bizonyos elemek azok, amelyek újra felhasználhatóak, és néhányat saját szájíz szerint kell kialakítani, illetve rengeteg új típusú kérés tud megjelenni a szoftverben. Nem kell a korábbi kódokat vagy programozó stílusokat átvenni, hanem nyugodtan lehet új service-eket létrehozni, mert nagyon kis költséggel jár egy ilyen folyamat.

Technológiailag független: egy átlag monolitikus rendszernél többnyire egy programnyelvvel tudunk dolgozni, és nem tudjuk kipróbálni azt, hogyan tudnánk még gyorsabban, még hatékonyabban reagálni egy-egy dologra. Ezt a microservice architektúránál könnyen meg lehet csinálni anélkül, hogy ennek nagy költsége lenne a megrendelő számára. Minden service a számára ideális technológiával kerül kivitelezésre, ami egyfelől növeli a végeredmény rugalmasságát és hatékonyságát, másrészt ez is egy olyan rugalmasság, amelyet nagyon szeretnek a fejlesztők.

Biztonságosabb: az API Gateway/BFF egységesen fejleszthető és jól el tudják fedni az adatutakat, amik jellemzően csak rajtuk keresztül érhetőek el, így könnyebben karbantartható a rendszer

biztonsági szempontból is. A microservice-alapú rendszerek komponensei több teszteléssel is járnak, így ez is segíti a teljes rendszer biztonságát.

A microservice architektúra hátrányai

Természetesen, ahogy az előnyökről, úgy a hátrányokról is kell beszélni, szerencsére azonban itt viszonylag kevés tétel jelenik meg, amiket messze túlszárnyalnak az előnyök.

Mivel sok-sok komponens együttműködéséről beszélünk és a servicek hálózati módon kapcsolódnak egymáshoz, így némi performancia csökkenés észlelhető.

A tervezési szakasza is erőforrás-igényesebb, és olyan kollégákat kell bevonni a folyamatba, akik a biztonsági és a deploying területen nagy tapasztalattal rendelkeznek.

Mindegyik service-nek meg kell írni a saját deployment-jét, ami valóban többlet erőforrást igényel a fejlesztői csapattól - ám ezek az erőforrás-befektetések később megtérülnek.

Hosszú távon viszont a hátrányok egy részéből is előny kovácsolódik. Mivel ezek a rendszerek függetlenek egymástól, valójában soha többet nem fog bekövetkezni az, hogy a teljes rendszert le kelljen cserélnünk, mindig csak az adott komponensekhez kell hozzányúlni, amelyeket az adott fejlesztések érintenek. Tehát a kezdeti nagyobb kiadás után egy hosszabb távú megtérülés következik.

Akik már microservice architektúrával dolgoznak

Mindig nagy kérdés, hogy kik azok, akik már egy adott megoldás mellett tették le a voksukat. A jó hír az, hogy a piac nagy szereplői az Amazontól az eBayen keresztül az AliExpressig mind-mind microservice architektúrát használnak.

Egy érdekes tény, hogy az Amazonnak csak a főoldal megjelenéséhez több, mint 1200 microservice együttműködésére van szükség. Persze egy átlagos vállalkozásnál nincs szüksége ilyen mértékű service-esítésre, de fontos azt látni, hogy ezek a nagy szereplők mind ilyen megoldással dolgoznak. Ennek a korábban felsorolt előnyök mellett az is az oka, hogy ezáltal a technológiai váltásokat is könnyebben meg tudják oldani, bármilyen irányba mozduljon is a piac.

Vezetői összefoglaló

Miért csináljuk?

A Stylers Group küldetése, hogy a közössége erejével folyamatos, valódi értéket teremtsen.

Hogyan csináljuk?

Folyamatosan alkalmazkodunk a külső, rendszerint üzleti és technikai igényekhez úgy, hogy közben olyan megoldásokat adunk, amelyek a felhasználóknak, ügyfeleinknek és nekünk is a megfelelő értékeket teremtsen. Szolgáltatásainkat agilis szemlélettel és módszertanokkal (scrum, kanban) végezzük.

Külföldi jelenlétünk, ügyfeleink, San Diego-ban lévő irodánk előnyeit kihasználva előre látjuk az egyes trendeket, fel tudunk készülni a várható piaci változásokra.

ISO 9001-es és ISO 14001-es tanúsítványaink által is elmondható, hogy munkafolyamataink, környezetre való odafigyelésünk tudatos.

Mit csinálunk?

Informatikai megoldásokat nyújtunk a tervezéstől kezdve a legösszetettebb fejlesztéseken át az üzemeltetésig.