**Twelve-Factor App**

A **Twelve-Factor App** egy olyan ajánlásgyűjtemény, melyet a ***Heroku*** platform

**fejlesztői dolgoztak ki** arra, hogy **hogyan lehet olyan alkalmazást fejleszteni**, amely a **felhőbe telepíthető és üzemeltethető**. A Twelve-Factor App tehát nem más, mint **szabályok és javaslatok gyűjteménye**. Ha ezeket betartjuk, akkor az alkalmazásunkat egyszerű lesz felhőbe telepíteni. Nem csak felhőben futó alkalmazás esetén érdemes ezeket betartani, hanem klasszikus alkalmazások esetén is, hiszen ezzel az üzemeltetési költségeket biztosan csökkenthetjük, valamint a hibázási lehetőségek is minimalizálhatóak.

A kidolgozásakor a *cloud*-ot, a *Platform as a Service (PaaS)*-t és a *continuous deployment*-et tartották szem előtt, hiszen a Heroku nem más, mint egy PaaS-szolgáltató.

A *Platform as a Service* azt jelenti, hogy a szolgáltató számunkra nem egy virtuális gépet ad, tehát nem nekünk kell rá az operációs rendszert, az adatbázisokat, az alkalmazásszervereket telepíteni, hanem **a szolgáltató ezt elvégzi helyettünk**. Nagyon sok ilyen szolgáltató létezik (pl. Google App Engine, Redhat Open Shift, Pivotal Cloud Foundry, Heroku, AppHarbor, Amazon AWS, stb.)

A Spring fejlesztése mögött a Pivotal nevű cég áll. Ők dolgozták ki **a *cloud native*** fogalmat. Ez egy jelző, azokat a cégeket, szervezeteket jellemzi, amelyek gyorsan képesek az automatizálás előnyeit kihasználva olyan alkalmazásokat fejleszteni, amelyek megbízhatóak és skálázhatóak. A fogalom megalkotásakor a ***continuous delivery*-t, a *DevOps*-ot és a *microservice*-**eket tartották szem előtt. Egy *cloud native* jelzővel rendelkező cég által kifejlesztett alkalmazás jellemzői:

* Tipikusan valahol a felhőben fut (PaaS-en, amely lehet publikus vagy privát is).
* Rendelkezik az *elastic* tulajdonsággal, vagyis ha maga az infrastuktúra észreveszi, hogy hirtelen megnövekedett a felhasználók száma, ezért automatikusan skálázódik, hogy az alkalmazás ki tudja szolgálni a megnövekedett igényeket.

A Twelve-Factor App a következő 12 szabályt tartalmazza:

1. **Verziókezelés**: az adott alkalmazáshoz tartozó összes forráskódnak és erőforrás állománynak egy repository-ban kell lenniük, ugyanabban a verziókezelő rendszer repository-ban.
2. **Függőségkezelés**: az alkalmazás külső függőségeit explicit módon deklarálni kell és **külön kell tárolni ezeket az alkalmazás forráskódjától.**
3. **Konfiguráció**: az adott környezetekre jellemző konfigurációk nem az alkalmazás részét képezik, hanem az **adott környezet részének** kell lenniük.
4. **Háttérszolgáltatások**: el kell gondolkozni azon, hogy milyen külső háttérszolgáltatásokat akarunk igénybe venni, és ezeket külön kell kezelnünk. Ezek **ne legyenek az alkalmazásunk részei,** hanem ezek külön jelenjenek meg, és az alkalmazásunk ehhez kapcsolódjon. Ilyen például az adatbázis. Itt fontos megjegyezni, hogy a többi alkalmazás is, amihez a mi alkalmazásunk kapcsolódik, háttérszolgáltatásnak minősül.
5. Vegyük **külön** a **build**, a **release** és a **futtatás** lépéseit! Ezek ne keveredjenek, hanem legyenek jól szeparálva!
6. **Folyamatok: Az alkalmazás állapotmentes folyamatokból álljon**. Az állapotmentesség nagyon fontos, hiszen az alkalmazásunk csak így lehet tetszőlegesen skálázható.
7. **Port hozzárendelés:** fontos, hogy meg tudjuk mondani, hogy **az alkalmazásunk melyik porton induljon** el. Ez is a skálázhatóság miatt fontos, hogy akár több alkalmazáspéldányt is el lehessen indítani párhuzamosan, ha arra van szükség.
8. **Párhuzamosság**: fontos arra figyelni, hogy az alkalmazást a felhasználók párhuzamosan is teljes értékűen igénybe vehessék, ne legyenek a párhuzamosságból eredő problémák.
9. **Mivel felhőkörnyezetben dolgozunk, fontos,** hogy az alkalmazás gyorsan tudjon elindulni és gyorsan tudjon leállni is.
10. **Próbáljunk arra törekedni, hogy az éles és a fejlesztői környezet a lehető legjobban hasonlítson egymásra.** Így minimalizálható a hibák elkövetésének lehetősége.
11. **Naplózás**: fontos, hogy az alkalmazás naplózzon. A Twelve-Factor App azt mondja, hogy **lehetőleg a konzolra naplózzon**, mert az alkalmazásnak nem feladata a naplózás céljának meghatározása, hanem az üzemeltető és az infrastruktúra dolga az, hogy ezt a naplót oda továbbítsa/mentse, ahova szükséges. Tehát a naplózás céljáról ne az alkalmazás döntsön, hanem ezt kívülről lehessen megadni standard, operációs rendszer szintű eszközök használatával.
12. **Felügyeleti folyamatok:** az alkalmazás mellett tároljuk és ugyanúgy verziókezeljük azokat a felügyeleti folyamatokat, amelyeket ad-hoc módon kell lefuttatni. Ez azért fontos, hogy ne lehessen ilyeneket kívülről, egyszeri esetekben lefuttatni.

**Ezenkívül van egy könyv is**, amelyet szintén a **Pivotalnál** írtak. A címe: ***Beyond the Twelve-Factor App***. Ez főleg példákat hoz arra, hogy hogyan lehet a fenti 12 szabályt megvalósítani. Ezek közül néhányat érdemes kiemelni:

* A könyv azt mondja, **hogy először alakítsuk ki az API-t és utána fejlesszük le az alkalmazást.** (Erről megoszlanak a vélemények, sokan vannak, akik ezt pontosan fordítva látják jónak.)
* A **konfigurációnál kiemeli a jelszavak és tanúsítványok használatának fontosságát.**
* Azt írja, hogy különös figyelmet kell fordítani az autentikációra és az autorizációra, vagyis arra, hogy hogyan történik az alkalmazásba való bejelentkezés és a hozzáférések szabályozása. Ha állapotmentes alkalmazást írunk, akkor **nem tárolhatjuk el magában az alkalmazásban a felhasználók adatait,** hanem valamilyen más módon kell ezt megoldani.
* **Telemetria**: ezeket az alkalmazásokat **monitorozni** kell tudni, azaz **az alkalmazás állapotát lehessen lekérdezni**, és az így kapott adatokat visszamenőlegesen tárolni és akár grafikusan megjeleníteni is lehessen.

**Kérdések**

*Mi a 12Factory App, miket fogalmaz meg és kik dolgozták ki?*

*Ezen ajánlások betartásáva mi célt szolgál?*

*Mi az a PaaS?*

*Mire képes egy cloud native jelzővel rendelkező cég? Milyen szemléleteket követ? Milyen alkalmazásokat fejleszt?*

*Mi a 12Factory App 12 szabálya?*

*A Pivotal által írt Beyond the Twelve-Factor App könyv még milyen plusz ajánlásokkal szolgál?*