Tartalomjegyzék

**Összefoglaló1**

**Abstract2**

**1 Bevezetés3**

1.1 Célkitűzés4

1.2 Választott technológiák5

1.2.1 Szerver oldal 6

1.2.2 Kliens oldal 7

1.2.2.1 Web kliens 8

1.2.2.2 Mobil kliens 9

1.2.3 Használt kifejezések 10

**2 Feladat bemutatása11**

2.1 Követelmények12

2.1.1 Funkcionális13

2.1.2 Nem-funkcionális 14

2.2 Szereplők 15

Összefoglaló

todo

Abstract

todo

1 Bevezetés

Jelen dolgozatban egy felhő alapú, microservices architektúrára épülő élelmiszerbolti vásárlást és rendelést lebonyolító alkalmazás state-of-the-art technológiákkal történő megtervezését, implementációját, telepítését és üzemeltetését vizsgálom.

* 1. Célkitűzés

A témát elsősorban a manapság az iparban előszeretettel felhasznált microservices architektúra megismerése motiválta. A webalkalmazások komplexitásának növekedésével kulcsfontosságú a rendszer funkcióinak laza csatolása annak érdekében, hogy a fejlesztők egymástól függetlenül tudjanak dolgozni a rendszer különböző pontjain. Ezen túl a választott architektúra kiválóan alkalmas a felhő alapú üzemeltetésre és karbantartásra, ezzel egyrészt csökkentve az üzemeltetés költségeit, másrészt az infrastrukturális hibákat.

* 1. Választott technológiák

Az implementációhoz felhasznált technológiák kiválasztása során igyekeztem a mai trendeknek megfelelő, egyszerű, valamint az iparban is bevált és megfelelően dokumentált megoldások közül választani.

* + 1. Szerver oldal

A szerver oldali fejlesztéshez az általam mélyrehatóan ismert JVM alapú technológiákat használtam fel, ezen belül a Spring Boot keretrendszert, ami kényelmes és robosztus fejlesztést tesz lehetővé a dependency injection mechanizmusának és projektjeinek köszönhetően. Elsődlegesen Kotlin nyelven történik az implementáció, ami magában hordozza a Java nyelv előnyeit, mint például a hordozhatóság, rengeteg library, objektum-orientáltság, illetve, hogy egy kiforrott nyelv. Ezen felül a Kotlin egy modernebb szemléletmódot nyújt a Java elavult szintaktikájával és megoldásaival szemben. A Kotlin teljes mértékben interoperabilis a Java-val, ezzel lehetővé téve még a legacy Java projekteknek is, hogy akár csak néhány funkció legyen a modernebb Kotlin-nal elkészítve.

A build-eléshez és függőségek menedzseléséhez Maven-t használok.

* + 1. Kliens oldal

1.2.2.1 Web kliens

A web kliens fejlesztéséhez alapvetően a Javascript-re épülő keretrendszerek közül érdemes manapság választani, ugyanis ezek a legelterjedtebbek és legmodernebbek. A három fő keretrendszer, az Angular, React és Vue közül nekem a React-ra esett a választásom. A React fő előnye az elterjedtsége, ezáltal a közösség jelenléte problémák esetén, ami kulcsfontosságú szempont minden technológia kiválasztásánál. Ezen túl szimpatikus a kiváló TypeScript támogatottság, ami magában hordozza a JavaScript egyszerűségét, illetve a fordítási idejű hibadetektálást, ami kliens oldali fejlesztésnél sok fejvakarástól kíméheti meg a fejlesztőket.

1.2.2.2 Mobil kliens

A mobil kliens fejlesztéséhez felhasznált technológia kiválasztásakor alapvetően azt a fontos kérdést kell feltenni, hogy natív technológiákkal akarunk dolgozni, vagy cross-platform megfelelőikkel.

A natív fejlesztésre alapvetően nincs szükség, hacsak nem valami különösen komplex alkalmazást szeretnék készíteni, amihez szükség van iOS és Android specifikus szolgáltatásokra. A cross-platform technológiák előnye, hogy egy kódbázisból tudunk több platformra is alkalmazást fejleszteni.

Az én választásom a cross-platform irányra esett, ennek egyik legnépszerűbb megvalósulása a Flutter keretrendszer. todo

2 Feladat bemutatása

Ebben a fejezetben bemutatom az elkészítendő alkalmazást, az azzal szemben támasztott követelményeket, illetve az alkalmazás felhasználóit.

2.1 Követelmények

Az alkalmazással szemben támasztott követelményeket funckionális és nem-funckionális jelleg alapján bontottam szét.

2.1.1 Funkcionális követelmények

A funckionális követelmények írják le a rendszer szolgáltatásait, hogyan kell bizonyos bemeneti adatokra reagálnia és bizonyos helyzetekben hogyan kell viselkednie.

2.1.1.2 Termékek böngészése

**Szereplők:** Felhasználó, Tulajdonos

**Leírás:** Az áruház oldalán a felhasználó tud keresgélni az árucikkek között, akár szűréssel is. Az árucikkek részletes leírásának megtekintésére is van lehetőség. Ez tartalmaz kép(eket) a termékről, árat, mértékegységet, stb.

2.1.1.3 Termékek felvitele, módosítása

**Szereplők:** Tulajdonos

**Leírás:** Az áruház tulajdonosa tudja bővíteni a termékek palettáját, illetve a meglévőket módosítani.

2.1.1.4 Felhasználók, Futárok kezelése

**Szereplők:** Tulajdonos

**Leírás:** A Tulajdonos meg tudja tekinteni, és bizonyos esetekben szerkeszteni a regiszrált felhasználók illetve futárok adatait.

2.1.1.5 Felhasználói fiók, regisztráció

**Szereplők:** Felhasználó

**Leírás:** A rendeléshez szükség van a felhasználónak saját fiókra, amiben megadta az adatait. Ehhez regisztrációra van szükség.

2.1.1.6 Futár fiók, regisztráció

**Szereplők:** Futár

**Leírás:** A szállítás third-party futárokra épül, amihez a futároknak szüksége van az adataik megadására regisztrációval.

2.1.1.7 Kosár

**Szereplők:** Felhasználó

**Leírás:** A rendeléshez a felhasználónak a rendelni kívánt árucikkeket a kosárba kell tennie, amiket külön fülön meg is tekinthet.

2.1.1.8 Rendelés

**Szereplők:** Felhasználó

**Leírás:**  Mikor a felhasználó elkészült a kosár tartalmának összeválogatásával megrendelést tehet. Ezután a szállítás időpontjának kiválasztására és a fizetésre kerül sor. Rendeléssel a kosár tartalma kiürül. A felhasználó email-ben visszajelzést kap.

2.1.1.9 Szállítás elfogadása

**Szereplők:** Futár

**Leírás:**  A futár a kliensén keresztül el tud fogadni egy szállítást, ilyenkor felveszi a szállítandó árucikkeket, majd kiszállítja azokat a felhasználóhoz.

2.1.1.10 Szállítás nyomonkövetése

**Szereplők:** Felhasználó, Futár, Tulajdonos

**Leírás:**  A rendelés kiszállításának állapotát a rendelő felhasználó nyomon követheti. A Tulajdonos nyomon követheti minden szállítás állapotát.

2.1.1.11 Szállítás késleltetése

**Szereplők:** Felhasználó, Futár

**Leírás:**  A felhasználó bizonyos ideig késleltetheti a rendelés kiszállítását, ha nem alkalmas neki az átvétel.

2.1.1.12 Kiszállítás jóváhagyása

**Szereplők:** Futár, Felhasználó

**Leírás:**  A futár a kliensén keresztül jelezni tudja, ha a felhasználónak sikeresen kézbesítette a csomagot. A felhasználó erről értesítést kap, ezután jelezheti a rendeléssel kapcsolatos érzéseit.

2.1.2 Nem-funkcionális követelmények

A nem-funkcionális követelmények írják le a rendszer teljesítményére vonatkozó előírásokat, pl: sebesség, szabványok, stb.

2.1.2.1 Skálázhatóság

**Leírás:**  Az alkalmazásnak skálázhatónak kell lennie, tehát a beérkező kéréseknek megfelelően horizontálisan képesnek kell lennie a skálázódásra, ami újabb service példányok elindítását jelenti.

2.1.2.2 Könnyű bővíthetőség

**Leírás:**  Az alkalmazásnak támogatnia kell az újabb funkciók könnyű beépíthetőségét a meglévő rendszerbe.

2.1.2.3 Robosztusság

**Leírás:**  Az alkalmazásnak robosztusnak, ellenállónak kell lennie a fellépő hibákkal szemben, ugyanis a kiesések és hibák a felhasználói élmény romlásával, sőt, felhasználók elpártolásával járhat,

2.2 Szereplők

* Felhasználó (User)
* Tulajdonos (Owner)
* Futár (Courier)