***Food Delivery Appliacation  
Documentation (HU)***

Ver. 1.0.0

Tartalomjegyzék (A végére színkód is kell)

Autentikáció, jogosultságok

A szoftver JWT (JSON Web Tokens) autentikációt használ, mely technológiának a lényege, hogy a szerver, a felhasználónév/jelszó páros bejelentkezést követően egy lejáró\* tokent biztosít a kliens számára. Ezáltal a kliensnek nem kell minden kérésnél felhasználónév/jelszó párost elküldenie a szervernek, hanem csak a tokent. A token alapján történik a továbbiakban a kliens (felhasználó) azonosítása. A tokenben eltárolhatóak adatok, mint például a felhasználónév, a token kibocsájtási ideje és a lejárati ideje, illetve bármi más fejlesztő által meghatározott adat. (!!! Nem érdemes sok adatot tárolni benne.) Egy JWT token három részből áll:

* Fejléc: A tokenről tárol adatokat (Pl: Token típusa)
* Adat: A tokenen tárolt adatok (Pl: Felhasználónév)
* Ellenőrző aláírás: A szerver írja alá és az adatok sértetlenségének ellenőrzésére szolgál.

Az autentikációt a szoftver Servlet Filter segítségével végzi el. A Servlet Filter-ek azelőtt kerülnek meghívásra mielőtt a kérés elérné a Servlet-et. A Filter-ek egy Filter Chain-be vannak rendezve és számít a sorrendjük, mivel egymást hívják meg, így csak egy irányba lehetnek hatással egymásra. A Spring Boot alapból 10+ Filtert biztosít a Spring Security által, ezen Filter-ek közé került beillesztésre a UsernamePasswordAuthenticationFilter elé a JwtAuthenticationFilter. A JwtAuthenticationFilter. doFilterInternal metódusa végzi el az autentikációt és hívja meg a következő Filter-t. A metódus a kérésből kiveszi a fejlécet:

* Ha nem talál fejlécet vagy a fejlécben nem talál JWT tokent, akkor tovább küldi a kérést és ilyenkor nem történt meg az autentikáció.
* Egyébként kiveszi a kérésből a tokent és a tokenből visszafejti a felhasználónevet.
  + Ha a felhasználónév nem üres String, akkor lekérdezi a UserDetails interfészt, amelyet a User osztály imlementál.
    - Ha a JWT token érvényes JwtTokenProvider.isValidToken, akkor a SecurityContextHolder.getContext-en elmenti az autentikált felhasználót.
* Tovább küldi a kérést.

Vannak olyan végpontok, amelyek autentikáció, illetve jogosultságok nélkül is elérhetőek, például a bejelentkezés és a regisztráció.

A szoftver legtöbb végpontjának elérése nem csak autentikációhoz kötött, hanem jogosultságokhoz is. Minden regisztrált felhasználónak a User osztályon van eltárolva a jogosultsága Role enumként. A Controller osztályokban vannak a szoftvernek a végpontjai definiálva. Ezeken az osztályokon vannak a jogosultságok megkövetelve. Beszélhetünk kétféle megszorításról, az egyik az osztály szintű megszorítás, amely az osztályon alkalmazott @PreAuthorize annotációval érhető el. Ilyenkor csak az adott jogosultsággal rendelkező felhasználók érhetik el az osztályon definiált végpontokat. A másik megszorítás pedig a metódus szintű, amikor szintén a @PreAuthorize annotációval ellenőrizzük a jogosultságokat, viszont csak egy metódusra (végpontra).

Szerepkörök leírása

Kiskereskedő (Role.ROLE\_RETAILER):

Vásárló (Role.ROLE\_CUSTOMER):

Futár (Role.ROLE\_COURIER):

Adminisztrátor (Role.ROLE\_ADMIN):