Backend Services
Használati Útmutató

Tartalom

1. Data API

A data API szolgáltatás felelős az táska adatok fogadásáért, rögzítéséért illetve a megfelelő autentikációval rendelkező felhasználóknak az adatok vissza juttatásáért.

1.1. Konfiguráció

A konfigurációs config.json JSON file a service.js állomány mellet található a projekt könyvtárakban. Az alábbi táblázat minden egyes szintje, egy új JSON objektum beágyazottságot jelöl.

JSON Kulcs	JSON Kulcs	Javasolt érték	Mező funkciója
dev		false	true / false fejlesztéshez szükséges extra opciók kapcsolhatóak be vele, production környezetben mindenképpen false-al használandó!
api			
	port	8080	A port, amint az API-t el lehet érni
	timeout	1000	Mennyi idő után bontsa a kapcsolatot automatikusan az API, ha nem hall a kliensről ennyi ms-ig
	keep-alive	1000	Kapcsolat keep-alive értéke ms-ban
	pathPrefix	/web	Milyen útvonalon érhető el az API a hostname után
dal			
	cache_size	10000	Hány elemet tartson az API cache-ben, drasztikusan megnöveli az adatelérés sebességét, de cserében növeli a memória igényt
	url	mongodb://localhost	Mongo adatbázis URL-je

	db	smb	Mongo adatbázis neve
blob			
	permission	660	Adatfileok alapértelmezett jogosultsága
	encoding	utf8	Adatfileok encodeolása
	root	./data	Adatfileok tárolási útvonala
	hash_depth	2	Milyen mélységű legyen a directory hash fa. Minél nagyobb az érték, az adatok annál mélyebb könyvtárszerkezetbe lesznek rendezve
firebase			Ezek az adatok a megfelelően beállított firebase accountról letölthetőek
	type	service_account	
	project_id		Firebase Project neve
	private_key_id		Firebase Project privát kulcs azonosító
	private_key		Firebase Project privát kulcsa,
	client_email		Firebase levélküldő (SMS) szolgáltatáshoz tartozó email cím,
	client_id		Firebase levélküldő (SMS) szolgáltatás azonosítója,
	auth_uri	https://accounts.google.com/o/oauth2/auth,	Firebase OAuth URL
	token_uri	https://oauth2.googleapis.com/token,	Firebase Token URL
	auth_provider_x509_cert_url	https://www.googleapis.com/oauth2/v1/certs,	Firebase auth provider URL
	client_x509_cert_url		Firebase publikus x509 tanusítvány elérési útvonala

1.2. Környezet

1.2.1. MongoDB

A Data API-nak szüksége van egy MongoDB-re, ahova a legfrissebb adatokat tudja rögzíteni a táskával kapcsolatban. Minden lekérdezés a MongoDB-ből szolgálódik ki, mivel annak az elérése nagyságrendekkel gyorsabb, mint a fileoké.

1.2.2. Adatfile tárhely

A Data API-nak szüksége van egy írható útvonlara, ahova a folyamatosan érkező adatokat ki tudja írni. Ezen az útvonalon automatikusan létre fog hozni egy könyvtárszerkezetet.
A létrehozott könyvtárak és fileok jogosultsága konfigurálható, célszerű minimalizálni az illetéktelen hozzáféréseket a fileokhoz, mert szenzitív adat lehet bennük.

2. Tunnel Service

A tunnel service segítségével a mobilalkalmazás és a táska interneten keresztül tudnak egymással kommunikálni, akkor is ha nincsenek egymás közelében, így ha a táska távol kerül a telefontól, azon továbbra is nyomon követhető lesz.

2.1. Konfiguráció

A konfigurációs config.json JSON file a service.js állomány mellet található a projekt könyvtárakban. Az alábbi táblázat minden egyes szintje, egy új JSON objektum beágyazottságot jelöl.

JSON Kulcs	JSON Kulcs	Javasolt érték	Mező funkciója
tunnel			
	port	1337	A port, amin a tunnel szolgáltatást el lehet érni
	channel_open_timeout	3000	Mennyi idő után bontsa a kapcsolatot automatikusan az API, ha nem hall a kliensről ennyi ms-ig
	close_wait_time	1500	Hány ms türelmi időt adjon a szerver a csatorna tiszta lezárásahoz, mielőtt törölné a kapcsolatot
	ping_interval	10000	Milyen gyakorisággal küldjön a szerver ping csomagokat, ezek segítenek nyitva tartani a csatornát. Az érték ms-ben.
	client_timeout	30000	Hány ms után bontsa a tunnel service a kapcsolatot ha nem érkeznek ezen idő alatt csomagok.

3. Web API

A web API feladata, hogy a webes felület kéréseit kiszolgálja, ez tulajdon képpen a webes felület backend szolgáltatása, azonban a további bővíthetőség miatt egy különálló API serviceként működtetjük.

3.1. Konfiguráció

A konfigurációs config.json JSON file a service.js állomány mellet található a projekt könyvtárakban. Az alábbi táblázat minden egyes szintje, egy új JSON objektum beágyazottságot jelöl.

JSON Kulcs	JSON Kulcs	JSON Kulcs	Javasolt érték	Mező funkciója
dev			false	true / false fejlesztéshez szükséges extra opciók kapcsolhatóak be vele, production környezetben mindenképpen false-al használandó!
api				
	port		8080	A port, amint az API-t el lehet érni
	timeout		1000	Mennyi idő után bontsa a kapcsolatot automatikusan az API, ha nem hall a kliensről ennyi ms-ig
	keep-alive		1000	Kapcsolat keep-alive értéke ms-ban
	pathPrefix		/web	Milyen útvonalon érhető el az API a hostname után
	externalHost		https://SMBURL	A szolgáltatás külső elérése, célszerű egy url sémár és egy domaint megadni, de elvileg bármilyen értelmes URL-nek működnie kell, azonban a pathPrefix ehhez az URL-hez lesz hozzáfűzve.
dal				

	cache_size		10000	Hány elemet tartson az API cache-ben, drasztikusan megnöveli az adatelérés sebességét, de cserében növeli a memória igényt
	url		mongodb://localhost	Mongo adatbázis URL-je
	db		smb	Mongo adatbázis neve
auth				
	crypt		ANY_CHARACTER_SEQUENCE	User autentikációs adatait ezzel a kulccsal titkosítva tároljuk kliens oldalon
	ttl		900	Hány másodpercig érvényes a felhasználó bejelentkezése. Ha ezen az időintervvalumon belül nem hallunk a felhasználóról, akkor újra be kell jelentkeznie.
mail				
	host		SMTP_SZERVER	SMTP Szerver címe
	port		587	SMTP Szerver portja
	auth			
		user		SMTP szerverhez tartozó user név
		pass		A user jelszava
	sender			A feladó mezőben megjelenő e-mail cím.

3.2. Környezet

3.2.1. MongoDB

A webAPI-nak szüksége van egy MongoDB-re, ahova a regisztrált felhasználókkal kapcsolatos adatokat tudja rögzíteni, illetve a csatlakoztatott táskák azonosítóit.

4. Web UI

A Web UI az a komponens, amit a felhasználó tényleges lát a böngészőjében, a feladata, hogy a felhasználói interakciókat Web és Data API hívásokká alakítva végrehajtsa a kért műveleteket, és megjelenítse azok eredményeit.

4.1. Környezet

4.1.1. Webszerver

A web UI nem egy önálló szolgáltatás, hanem egy single page website, amelyet egy webszervernek kell kiszolgálnia. Bármilyen webszerver alkalmazható, a belépési pont a szokásos index.html.

4.1.2. Data API

A UI-nak szüksége van Data API hozzáféréshez, ezt a KISZOLGÁLÁSI_URL/data útvonalon fogja keresni. A data API segítségével fogja tudni a táska aktuális információit megjeleníteni.

4.1.3. Web API

A UI-nak szüksége van web hozzáféréshez, ezt a KISZOLGÁLÁSI_URL/web útvonalon fogja keresni. A web API a UI backend szolgáltatása itt tárol minden olyan adatot, amit a felhasználó a webes felületen beállít.

5. Környezet

5.1. Általános indítás

A szolgáltatások mindegyike a service.js-el indítható, a konfigurációs állománynak a service.js-el azonos könyvtárban kell elhelyezkednie.

```
node service.js
```

5.2. Node.js

A szolgáltatásokhoz szükség van egy telepített node.js-re. Minimum elvárt verzió v12, javasolt verzió v16

5.3. Hálózati beállítások

Minden szolgáltatás TCP protokollt használ, így a konfigurált TCP portokat a tűzfalon ki kell engedni

5.4. HTTP Reverse Proxy

A szolgáltatások fel vannak készítve arra, hogy egy HTTP Reverse proxy mögött helyezkedjenek el, így minden API elérhető ugyanazon a domainen, csak különböző HTTP útvonalon. Ehhez a konfigurációban a pathPrefix opciót kell használni.

5.5. Loggolás

Jelenleg minden service az sdtout-ra loggol, ennek átirányítása és a logrotation a futtató környezet feladata.

5.6. Leállítás

A szolgáltatások reagának a TERM signalra, ebben az esetben tisztán leállnak, zárják a kapcsolatokat, ürítik az esetleges puffereket. A futtató környezetet célszerű úgy összerakni, hogy leállás esetén első körben egy TERM signalt küldjön, majd pár másodperc türelmi idő után KILL signalt. Ezzel a megoldással csökkenthető a leállás miatti esetleges adatvesztés.