

GÁBOR DÉNES FŐISKOLA

MÉRNÖKINFORMATIKUS ALAPKÉPZÉS

**Helpdesk rendszer megvalósítása
mikroszerviz alapú elosztott
alkalmazással**

Bőle Balázs

Konzulens:

Dr. Nagy Elemér Károly

Szoftverfejlesztés szakirány



2020 november

Helpdesk rendszer megvalósítása mikroszerviz alapú elosztott alkalmazással

készítette

Bőle Balázs

Neptun kód: DXQRPJ

Elérhetőség: bolebalazs@gmail.com

Konzulens: Dr. Nagy Elemér Károly

A dolgozat elektronikus változata elérhető a <https://github.com/balazsBole/> címen.



Budapest, 2020 november.

Kivonat

Dolgozatomban ismertetem egy mikroszerviz alapú elosztott alkalmazás felépítését, a tervezés során fellépő általános problémákat, valamint ezekre a problémákra adható megoldásokat.

Nagy vonalakban és feladatspecifikusan áttekintem a felhasznált technológiákat és módszertanokat.

Ezek tükrében bemutatom a létrehozott szoftvert, infrastruktúrát és az üzemeltetéséhez szükséges eszközöket.

Tartalomjegyzék

Tartalomjegyzék	iii
Ábrák jegyzéke	iv
Bevezetés	1
1. Üzleti igények	2
1.1. Funkcionális igények	2
1.1.1. E-mail fogadása és küldése	2
1.1.2. E-mail szálak kezelése	2
1.1.3. Több felhasználó	2
1.2. Nem funkcionális igények	3
1.2.1. Skálázhatóság	3
1.2.2. Granuláris felosztottság	4
1.2.3. Mérhető indikátorok	4
1.2.4. I18N	5
2. Technológiai áttekintés	6
2.1. Mikroszerviz architektúra	6
2.2. Hexagonális architektúra	7
2.3. Event stream processing	8
2.4. Rétegek szeparálása	8
2.5. Angular	8
2.6. Spring Boot?	9
3. Architekturális áttekintés	10
3.1. Component Diagram	10
3.2. Sequence UML Diagram	11
3.3. Adatbázis UML ábra	11
3.4. Átfogó áttekintés az implementálás bemutatása előtt	11

4. Implementáció	12
4.1. Mikroszerviz infrastruktúra	12
4.1.1. Nginx	12
4.1.2. Docker konténerizáció	12
4.1.3. metrikák	12
4.2. E-mail kliens	13
4.2.1. E-mail szabvány	13
4.3. helpdesk backend	13
4.3.1. springBoot	13
4.3.2. data persistance layer	13
4.3.3. egyéb eszközök	14
4.4. helpdesk frontend	14
4.4.1. RxJs store	14
4.4.2. Komponensek	14
4.4.3. deploymnet	14
4.5. keycloak	14
4.5.1. JWT-token	14
4.5.2. role-ok	14
4.5.3. admin felületről valami?	14
5. Alkalmazás bemutatása	15
5.1. Load test	16
5.2. Multiple instance	16
5.3. Egy e-mail útja	16
5.4. Felhasználói felület funkciói	16
5.5. Kafka topicok?	16
6. Továbbfejlesztési lehetőségek	17
Irodalomjegyzék	20

Ábrák jegyzéke

1.1. Az e-mailszálak státuszváltozásai	3
--	---

1.2. Elérhető funkciók jogosultság szerint csoportosítva	4
2.1. Hexagonális alkalmazások felépítése	8
3.1. A legfontosabb komponensek	10
5.1. Deployment diagram	15

Bevezetés

Ahogy az O'Really által az év elején készített felmérésből [1] is látszik, a mikroszerviz alapú alkalmazások egyre nagyobb népszerűségnek örvendenek. Egyre több cég szeretné lecserélni meglévő monolit rendszerét, vagy a szükséges új funkciókat a régebbi rendszertől függetlenül, hibrid rendszerben valósítana meg.

Mint az a wiredelta cikkéből [2] is látszik, a mikroszerviz architektúrának számtalan előnye van. Míg a nagyvállalati környezetben sokszor a folyamatos szállítási igény, vagy az egymástól függetlenül fejleszthető alrendszerek miatt döntenek emellett a technológia mellett, az én esetemben a legfontosabb szerepet a skálázhatóság, az újrafelhasználhatóság, és az alacsony fenntartási költség játszotta.

Úgy gondolom, hogy nincs olyan technológia, ami minden problémára megoldást nyújtana. De úgy érzem hogy az ilyen elvek mentén kialakított alkalmazások, természetükből adódóan időtállóbbak lesznek. Ha el tudjuk érni, hogy egy alkalmazás valóban csak egy funkcióért kell hogy felelős legyen, azzal a problémamegoldás analitikus oldalát emeljük rendszerszintre.

Éppen ezért, a mikroszerviz architektúra legnagyobb előnye szerintem a rendszerezésből következik.

1. fejezet

Üzleti igények

Ebben a fejezetben szeretném bemutatni a Helpdesk alkalmazás felé megfogalmazott üzleti igényeket.

1.1. Funkcionális igények

1.1.1. E-mail fogadása és küldése

Az ügyfelektől érkező e-maileket az alkalmazás képes fogadni, hosszú távra megőrizni. Számukra formázott válasz e-mail küldhető.

A rendszernek képesnek kell lennie több e-mail cím kezelésére. A beérkező új üzeneteket a címzettnek megfelelő előre definiált sorhoz kell hozzárendelni.

1.1.2. E-mail szálak kezelése

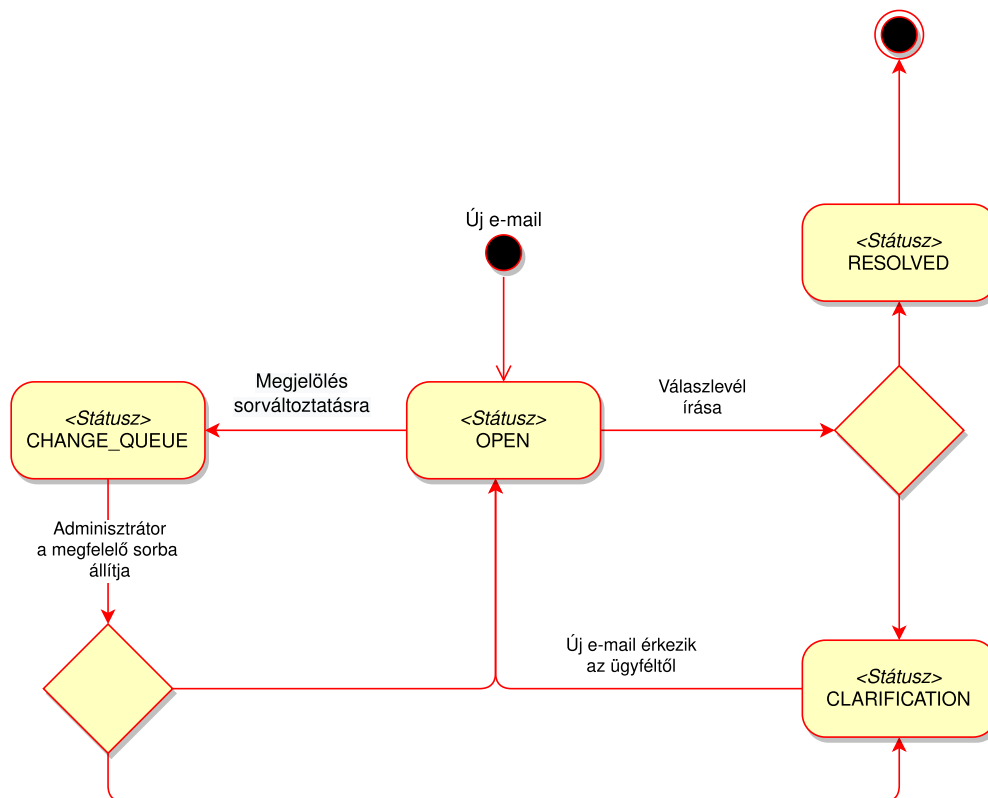
A rendszer által kezelt üzenetek szálakba rendezve érhetőek el. Egy szál az ügyfél és a felhasználó közötti üzenetváltásokból épül fel.

Az üzenetszálakra vonatkozó összes adat historikusan lekérdezhető, státuszuk az [1.1](#) ábrán definiált útvonalaknak megfelelően változtatható.

1.1.3. Több felhasználó

A rendszert egyszerre több felhasználó használhatja. Minden felhasználó csak a saját emailszállait kezelheti, csak azokra válaszolhat.

Minden felhasználó pontosan egy az [1.1.1](#) fejezetben említett sorhoz tartozik. Csak az ugyanabba a sorba tartozó e-mail szál rendelhető hozzá. A számára kijelölt szálakat képes –a saját során belül– más felhasználóhoz rendelni.



1.1. ábra. Az e-mailszálak státuszváltozásai

A felhasználók eltérő jogkörökkel rendelkezhetnek. Az adminisztratív jogkörrel rendelkező felhasználó végzi az új emailszál felhasználóhoz rendelését, valamint a *change queue* státuszban (1.1 ábra) lévő üzenetszálak új sorba irányítását.

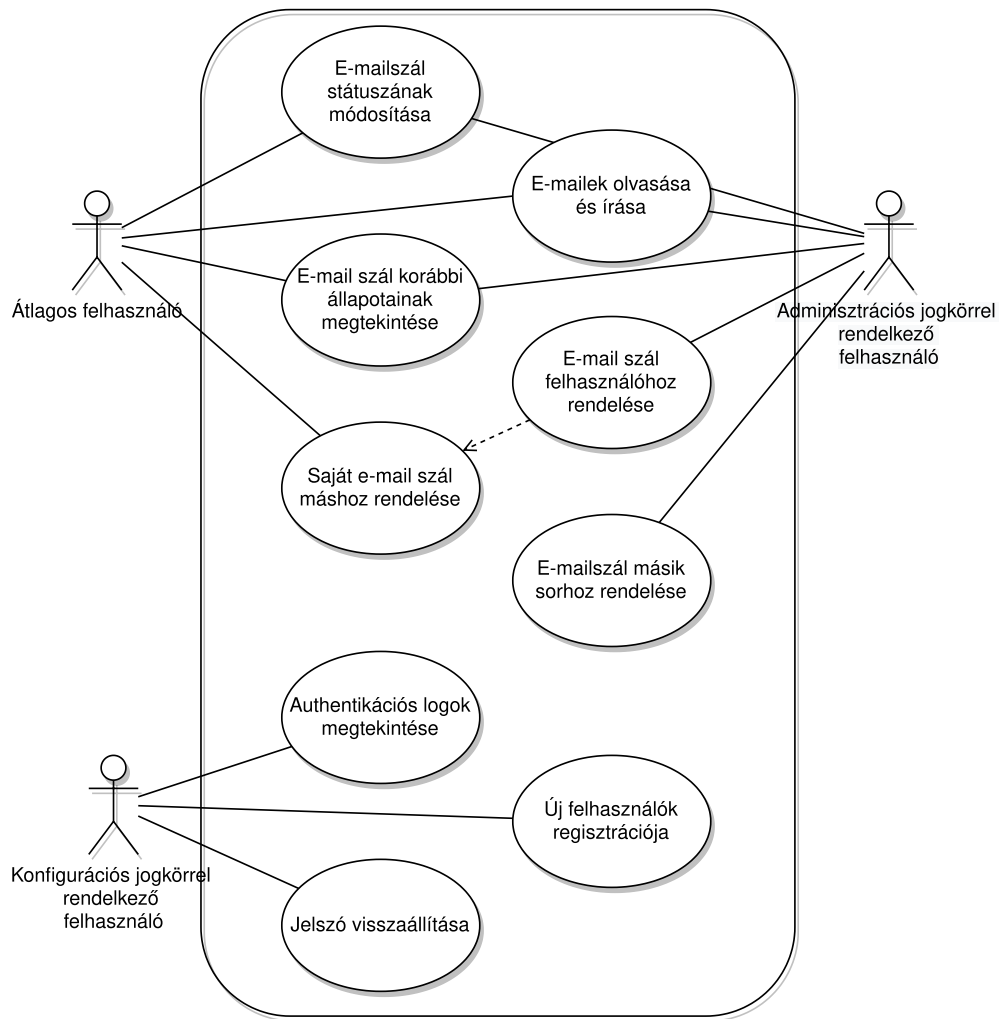
A konfigurációs jogkörrel rendelkező felhasználó feladata más felhasználók regisztrálása, valamint az alkalmazásban használt jogkörök (*role-ok*) kezelése. Lehetősége van továbbá autentikációs logok megtekintésére, jelszó visszaállítására és más felhasználók megszemélyesítésére (*impersonate*).

A felhasználói felületen elérhető funkciókat az 1.2 ábra foglalja össze.

1.2. Nem funkcionális igények

1.2.1. Horizontális skálázhatóság

A kiszolgáltató kliensek száma napi és havi szinten is eltérő. Az év egyes időszakaiban nagyobb volumenű ügyfél-interakció prognosztizálható. A hibátűrés javítása, és a megnövekedett forgalom érdekében –ezekben az előre meghatározott időszakokban– horizontális skálázódás szükséges.



1.2. ábra. Elérhető funkciók jogosultság szerint csoportosítva

1.2.2. Granuláris felosztottság

A helpdesk alkalmazást használó ügyfélszolgálat munkaórákban a legaktívabb, míg az e-maileket küldő ügyfelek hétvégente és hétköznapi munkaórákon kívül a legaktívabbak.

A hosszútávú tervekben szerepel a helpdesk alkalmazás és a belső céges levelezés integrálása.

A fenti két szempont miatt célszerű a megvalósítandó funkciók minél nagyobb mértékű szeparálására törekedni.

1.2.3. Mérhető indikátorok

A rendszernek átlagosan 100 felhasználót kell kiszolgálnia másodpercenként. A várható csúcsteljesítmény 10 000 lekérdezés másodpercenként. A tolerálható legnagyobb válaszidő 1 másodperc/lekérés.

1.2.4. I18N

A felhasználói, adminisztratív és karbantartói felületek angol nyelven érhetőek el. Több nyelv kezelése nem szükséges.

2. fejezet

Felhasznált technológiák

Az alkalmazás rendszer szinten mikroszerviz, a modulok szintjén hexagonális architektúrába rendezve készült el. A kód szervezése során felhasznált technológiák a 2.4 pontba vannak összeszedve.

2.1. Mikroszerviz architektúra

Bár a kifejezés már régóta ismert, nincs egy központilag elfogadott, egységes definíció arra nézve, miket nevezünk mikroszervizeknek. A legtöbb szerző [3] jobb híján a visszatérő karakterisztikus tulajdonságuk alapján sorolja be az alkalmazásokat ebbe a kategóriába. Egy tipikus mikroszerviz a következő tulajdonságoknak felel meg:

- pontosan egy üzleti funkció köré szerveződik
- más szervizekkel laza, általában hálózaton keresztül megvalósuló kapcsolatban áll
- ha szüksége van adatbázisra, akkor sajáttal rendelkezik
- önmagában is működőképes
- decentralizált, tehát nincs egy a munkáját befolyásoló központi irányítórendszer

A hasonló felépítésükből adódóan, számos olyan eszköz van, ami –nem kötelezően, de legtöbbször– együtt fordul elő a mikroszerviz architektúrával. A legfontosabb ilyen fogalmak a:

skálázhatóság a rendszer képessége az áteresztőképességének növelésére. Létezik vertikális¹ és horizontális skálázhatóság².

¹több processzor vagy memória bevonása

²újabb példányok futtatása

konténerizálás a szerviz futtatása saját elszeparált környezetében hardveres virtualizáció segítségével nélkül.

erőforrás felderítés a rendszer által nyújtott erőforrások automatikus felfedezhetősége³.

loadbalancer az a folyamat, ami a bejövő feladatokat erőforrásokhoz rendeli. Legegyszerűbb megvalósítása a *round robin* algoritmus, célja a terhelés egyforma elosztása.

monitorozás az önálló szervizek állapotának felügyelése. A monitorozás során nyújtott metrikák kiterjedhetnek a felhasznált memória mennyiségére, processzorigényére, vagy processzeire is.

2.2. Hexagonális architektúra

A hexagonális architektúra –vagy más néven portok és adapterek architektúrája– egy Alistair Cockburn által létrehozott [4] szoftvertervezési minta. Nevét a cikkben felrajzolt hatszögletű rendszerábrázolásról kapta (2.1 ábra), ami szembenegy a korábban elterjedt réteges elrendezéssel.

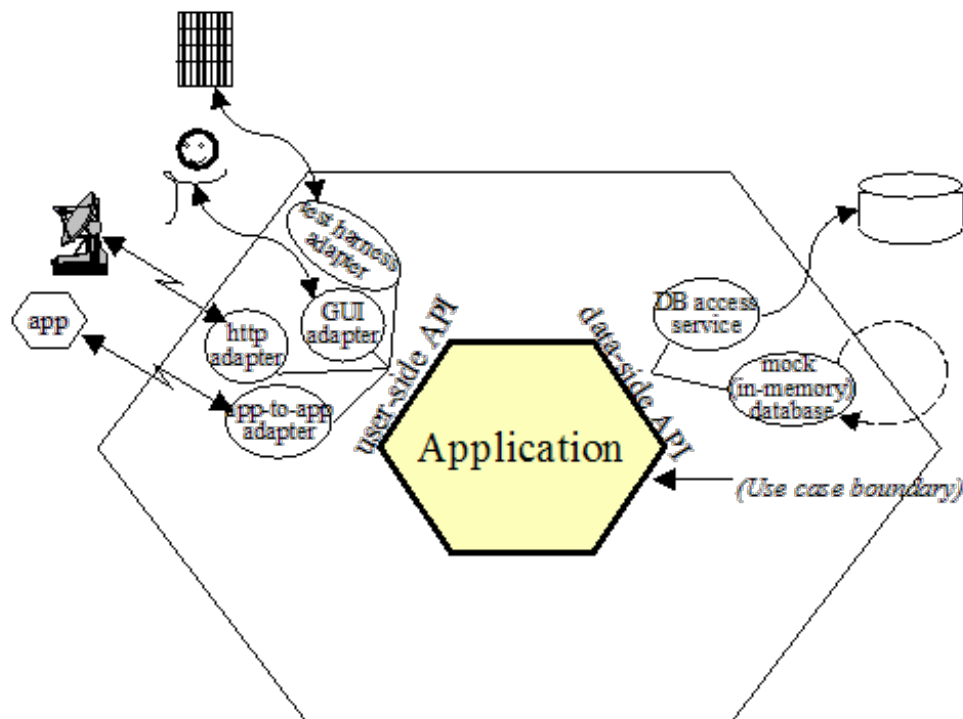
Az eredeti szándék mögöttese az alkalmazás függetlenítése mindennemű külső függőségtől⁴, így lehetővé téve az üzleti és a technikai igények nagy mértékű szeparálását. Egy absztrakt port feladata kell legyen a külvilággal való kapcsolat, így az üzleti logika csak az üzenet tartalmáért felelős, az üzenetküldés módjáért már nem.

Ahogy Robert C. Martin a *The Clean Architecture* cikkében [5] összeszedte, a portadapter és a hasonló architektúrával készülő alkalmazások mind:

- Könnyen, és önmagukban is tesztelhetőek. Mivel az üzleti szabályoknak, nincs külső függőségük.
- Függetlenek a külső tényezőktől. Így az alkalmazás által használt felület vagy adatbázis könnyen cserélhető.
- Keretrendszerrel függetlenül is megvalósíthatóak. A megvalósítás nem függ semmilyen könyvtártól vagy egyéb tulajdonságtól.

³angolul *service discovery*-nek hívják

⁴például adatbázis, felhasználók, automatizált tesztek



2.1. ábra. Alistair Cockburn által [4] felrajzolt ábra a hexagonális alkalmazásról. Cockburn célja a külső függőségek elszeparálásának bemutatása.

2.3. Event stream processing

publish subscribe schema kafka event processing általánosan?

2.4. Rétegek szeparálása

DI, SOLID, based springboot javában Mapstruct, Lombok, JPA angular mvc

2.5. Angular

Az Angular egy a Google által fejlesztett TypeScript alapú platform és keretrendszer[6]. A segítségével létrehozott kód erősen modularizált, így könnyű vele újra felhasználató és az MVC-elveit követő alkalmazást létrehozni.

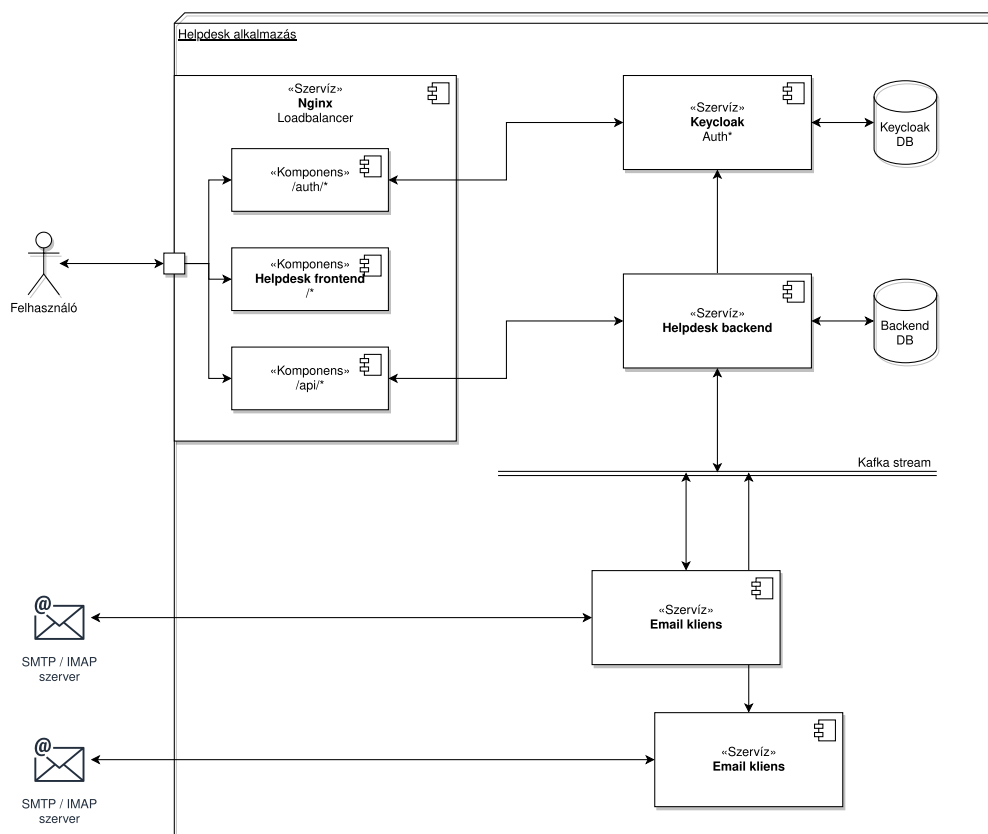
A segítségével készített honlap teljes mértékben a kliens oldalon fut, így a szervertől elegendő egy egyszerű, statikus HTML-oldalt visszaadó alkalmazáserver használat.

2.6. Spring Boot?

3. fejezet

Architektúrális áttekintés

3.1. Component Diagram



3.1. ábra. A legfontosabb komponensek

3.2. Sequence UML Diagram

3.3. Adatbázis UML ábra

3.4. Átfogó áttekintés az implementálás bemutatása előtt

4. fejezet

Implementáció

4.1. Mikroszerviz infrastruktúra

4.1.1. Nginx

Az nginxnek három elkülönülő szerepe van:

- A helpdesk frontend alkalmazásszervereként működik (lásd 2.5 pont)
- Routingot valósít meg, rajta keresztül érhető el a helpdesk backend és a keycloak szerviz
- HTTP cache-ként működik a frontend és a backend között, illetve a frontend és a keycloak között

A loadbalancer funkcionalitás a docker round-robin DNS-én (4.1.2) keresztül valósul meg.

4.1.2. Docker konténerizáció

Az alkalmazás összes szervize saját docker konténerben fut. A docker konfigurációs leírása a *docker-compose.yml* állományban van. A *docker-compose* parancs ez alapján indítja el az alkalmazást, hozza létre a saját alhálózatát, valósítja meg a hálózaton belüli DNS-funkciót.

A konténerek skálázása is a dockeren keresztül (*docker-compose -scale*) valósul meg.

4.1.3. metrikák

A springes alkalmazásaim egy-egy HTTP endpointon keresztül érhetőek el a prometheus számára (*/actuator/prometheus*) és induláskor beregisztrálják magukat az eureka

szerverbe¹.

A prometheus² az eurekán keresztül találja meg az instanceokat, és gyűjti össze a metrikákat. Az alkalmazások információt küldenek a Kafka konnektorukról, REST interfészükről és az adatbázis kapcsolatukról³.

A Prometheus által összegyűjtött adatokat grafanában⁴ ábrázolom.

4.2. E-mail kliens

Az e-mail kliens szerepe az üzenetek küldése és fogadása egy meghatározott e-mail címről. Feladata a külső protokollok leválasztása az alkalmazástól. Irányítja és karbantartja az IMAP és SMTP szerverrel való kapcsolatot.

A két irányú kömmunikáció megvalósulása:

1. az IMAP-on keresztül fogadott e-mailt az *email.in.v1.pub* kafka topicba írja
2. a saját –e-mail cím specifikus– topic-jából kiolvassa az üzenetet és továbbítja az SMTP szerver felé

4.2.1. E-mail szabvány

rfc5322: messageId replyTo refereneces

4.3. helpdesk backend

Class Diagram

4.3.1. springBoot

default behavior, DI security,

4.3.2. data persistance layer

hibernate, liquibase, Hibernate envers (Auditlog)

¹Az Eureka a Netflix által fejlesztett *discovery server*. Feladata az összes kliens port és ip adatának nyilkvántartása.

²A Prometheus egy open source monitorozó eszköz. 15 másodpercenként lekérdezi a szervizek állapotát.

³HikariCP-t használok JDBC kapcsolathoz

⁴A Grafana egy open source elemző és megjelenítő web alkalmazás

4.3.3. egyéb eszközök

openApi dokumentáció, mapstruct, hibernate,ombok

4.4. helpdesk frontend

MVC szerint van szeparálva a kód

4.4.1. RxJs store

4.4.2. Komponensek

material, quill

4.4.3. deploymnet

static html-lé fordul a kliens oldalán fut a frontend

4.5. keycloak

4.5.1. JWT-token

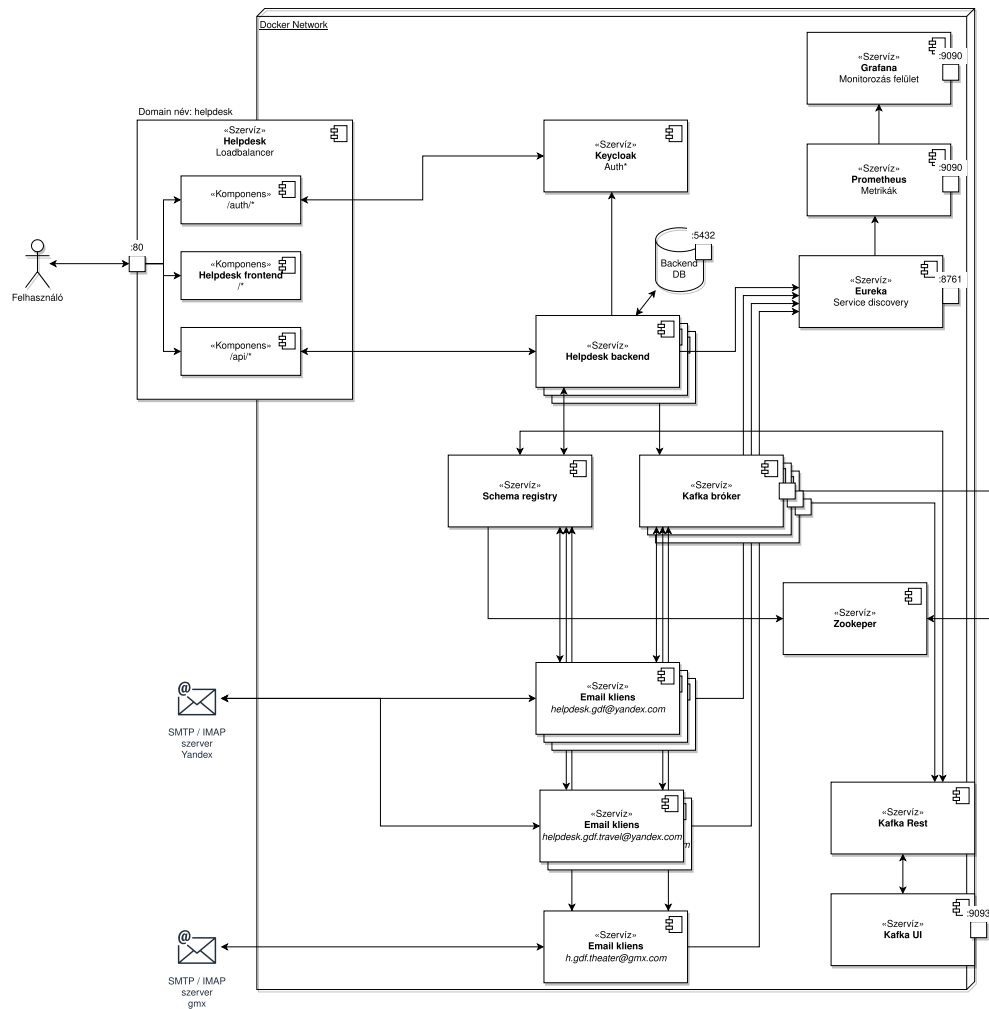
4.5.2. role-ok

4.5.3. admin felületről valami?

5. fejezet

Alkalmazás bemutatása

a három e-mailcím amivel már működik



5.1. ábra. Deployment diagram

5.1. Load test

Jmeter load test

5.2. Multiple instance

Grafana dashboard

5.3. Egy e-mail útja

message flow diagram

5.4. Felhasználói felület funkciói

képenyőképek? valami how to dokumentáció

5.5. Kafka topicok?

6. fejezet

Továbbfejlesztési lehetőségek

docker swarm kubernetes

Diszkusszió

Diszkusszió a végén

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt

unt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Irodalomjegyzék

- [1] Mike Loukides és Steve Swoyer. Microservices adoption in 2020, Júl. 15 2020. URL: <https://www.oreilly.com/radar/microservices-adoption-in-2020/>.
- [2] Andzhela Angelova. 10reasons why microservices are the future, Jún. 20 2020. URL: <https://wiredelta.com/10-reasons-why-microservices-are-the-future/>.
- [3] Matt McLarty Mike Amundsen Irakli Nadareishvili, Ronnie Mitra. *Microservice Architecture: Aligning Principles, Practices, and Culture*, chapter 1, The Microservices Way. O'Reilly, 1 edition, 2016.
- [4] Dr. Alistair Cockburn. Hexagonal architecture, Ápr. 1 2005. URL: <https://web.archive.org/web/20180822100852/http://alistair.cockburn.us/Hexagonal+architecture>.
- [5] Robert C. Martin. The clean architecture, Aug. 13 2012. URL: <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>.
- [6] Introduction to angular concepts. Letöltve: 2020. Nov. 16. URL: <https://angular.io/guide/architecture>.