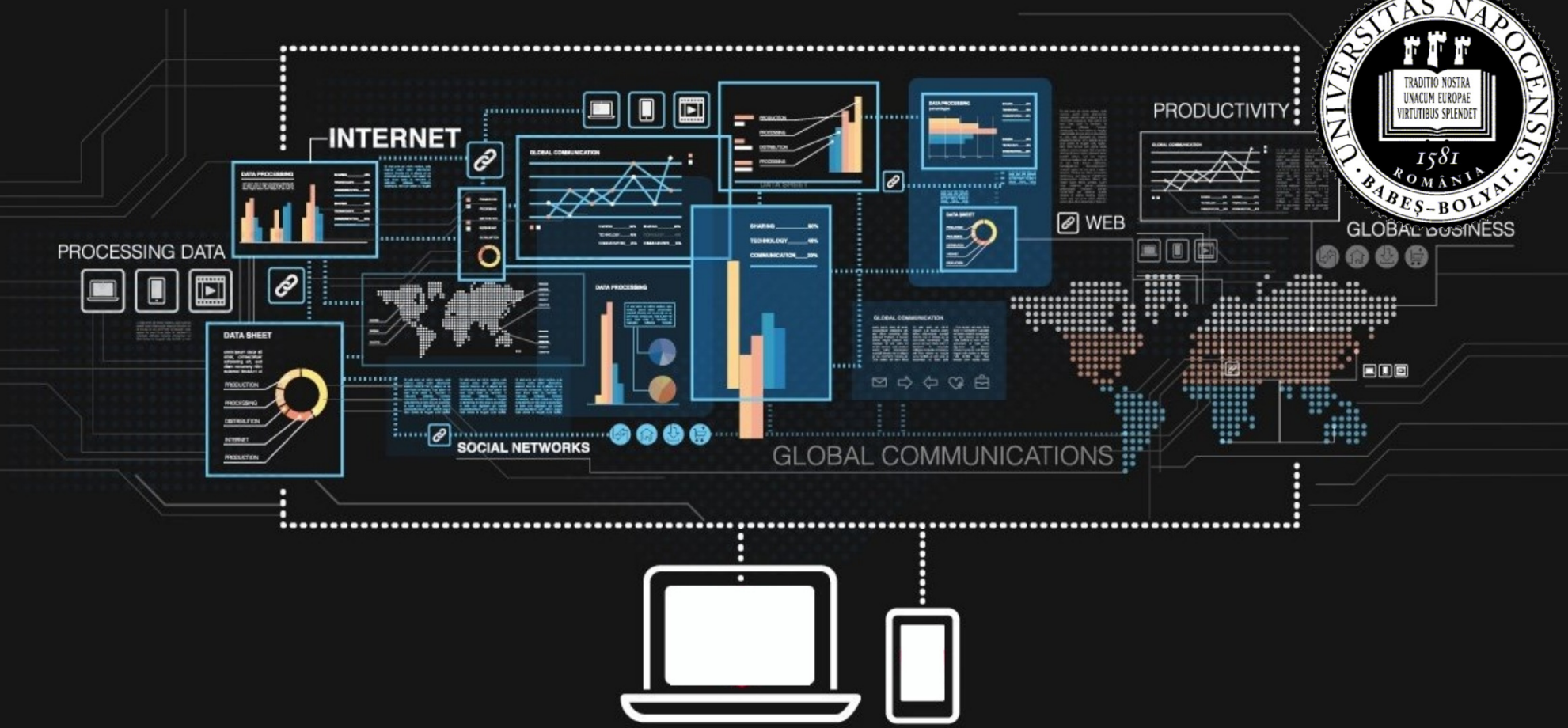


Projekt Bemutató

Community Space - A place for all your memos

Lukács Zsolt

Bevezető és CS projekt hatóköre

Tudásmegosztásra kihelyezett közösségi munkaplatform a vállalati/egyéb közösségen belüli gyors információmegosztásra és megőrzésre rövid memo-k formájában, ahol bárki könnyedén megoszthatja aktuális gondolatait/felhívásait/közléseit egy adott tematikáról vagy gondolatról csoportokon belül, amelyeket mások teljesíthetnek és elvégzésüket kvázi kötelezővé tehetik határidők megadásával, amelyekre a felhasználók értesítést kapnak sok más egyéb mellett.

Megjegyzés

Hub több memo-t foglal magába és tagok csoportjaként fogható fel, a memo egy általában rövid üzenet, amely hasonlít az e-mail-hez, gyakran valamilyen elvégzendő feladatot fog meghatározni.

Bevezető és CS projekt hatóköre

Alapvető funkcionalitások és üzleti folyamatok:

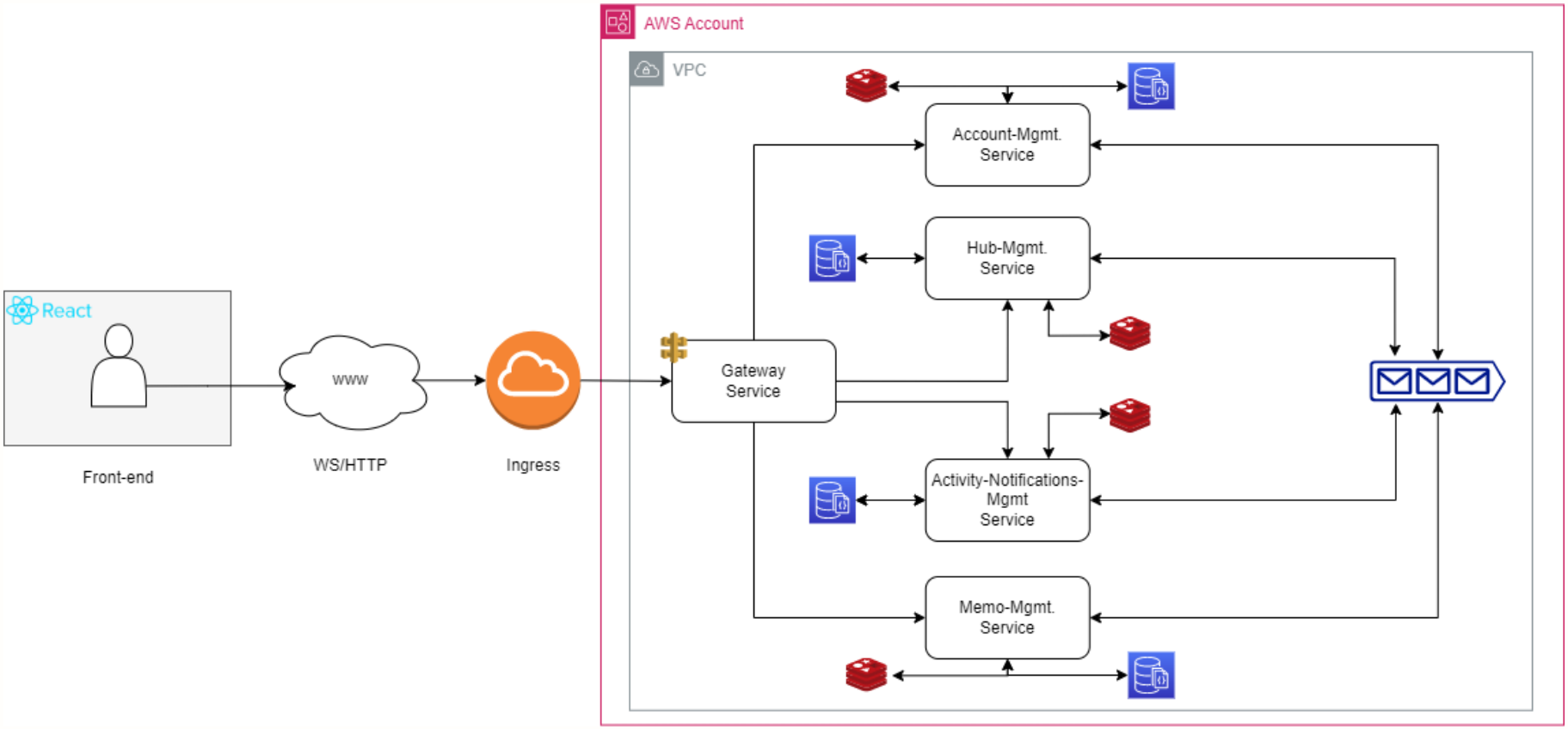
- Felhasználók képesek contókat létrehozni a platformhoz való csatlakozáshoz;
- Saját közösségeket, hub-okat létrehozni, meglévőkhöz csatlakozási kérelmeket leadni, tagokat adminisztrátor felhasználók menedzselhetik;
- Felhasználók képesek memo-kat megosztani különböző láthatósági szintekkel, prioritásokkal és határidővel kizárólag a hub-on belül, ezek elolvasni, módosítani, archívumba helyezni, törölni, kitűzni, teljesíteni, keresni közöttük, stb.;
- Aktivitási hőtérkép és interakciós (milyen tevékenységek végződtek el a múltban) listázás az aktuális hétre vetítve;
- Felhasználók képesek összegzés formájában megtekinteni az összes hub-jukat, informálódni arról, hogy hány új memo van, hány aktív tag, stb.;
- **Real-time** értesítési mechanizmus, platformbéli aktív felhasználók kilistázása.

Megjegyzés

Egyéb kiegészítő (technikai) funkcionalitások az automatizált `GitLab CI/CD`, a felhő `K8S`, annak menedzsmentje `Terraform`-al, és a `Helm Chart` kitelepítés.

CS Architektúra

Rendszer alapvető központi entitásai: felhasználók, hub-ok, memo-k (kiegészítőleg aktivitások és értesítések).



1. ábra. CS architektúra és komponens diagram

Micro-service architektúra összesen (jelenleg) 5 szolgáltatással.

CS Technológiai Stack

Front-end összefoglaló:

- Nyelv: `Typescript`;
- Keretrendszer: `Next.js`;
- Komponenskönyvtár: `MaterialUI`.

Back-end összefoglaló:

- NoSQL adatbázis: `MongoDB`;
- Cache megoldás: `Redis`;
- Általános back-end keretrendszer: `Spring Boot` (micro-service-k implementálására, pl. `Spring JPA, Spring Repository, Spring WebFlux`);
- Üzenet broker (in-cluster): `Kafka`;
- Infrastruktúra és DevOps: `Terraform` (K8S cluster létrehozása DigitalOcean-on), `K8S` és `Helm` és `GitLab CI/CD`.

Kiegészítő egyéb technológiák: `WebSocket` és `Socket.IO`, `Spring Gateway`, `useSWR` adat pulling-hoz, stb.

CS Architektúra Magyarázat

- Back-end egyetlen belépési pontja a `Gateway` szolgáltatás (HTTP/WebSocket), általános routing a többi szolgáltatáshoz; egyetlen `K8S Ingress`-t kitéve.
- Minden szolgáltatás külön `Mongo` adatbázissal rendelkezik és a legtöbb hívásukat cache-lik `Redis`-ben.
- A szolgáltatások soha nem kommunikálnak HTTP-n keresztül, kizárólag Kafka consumer-producer topic-okon keresztül, amelyeknek több szerepe is van:
  1. Amennyiben egyik szolgáltatás működése függ a másik szolgáltatásban tárolt adatoktól, a második szolgáltatásnak kötelessége egy topic-on keresztül közzé tenni ezeket a változásokat (i.e., legtöbb adatbázisváltozás valamilyen formában egy topic-ra közzé lesz téve, hogy a többi szolgáltatás erre hallgasson és a saját adatbázisába mentse el az adatokat), így függetlenné téve őket egymástól.
  2. Az `Activity-Notifications-Mgmt.` a platformbéli aktivitásokat üzenetsoron keresztül fogadja a többi szolgáltatástól, ezeket értesítésekké üzenetsoron keresztül teszi (az aktivitási üzenet fogadása után egy újabb üzenetet küld önmagának az értesítést kiküldésére/kezelésére).
  3. A memo teljesítési emlékeztetők üzenetsoron keresztül kerülnek load-balance-lésre az `Activity-Notifications-Mgmt.` példányok között (periodikusan megnézi melyek a még nem teljesített, de sürgős memo-k és üzeneteket tesz közzé minden memo-nak, amiben értesíti azokat a felhasználókat, akik még nem teljesítették az adott memo-t).

CS Valósidejűség

(Jelenleg) Két `WebSocket`-en (valójában `Socket.IO`-n, Netty bázisú szerverrel) alapuló valósidejű folyamat:

1. online státusz, értsd. aktív felhasználók;
2. értesítések (egyéni vagy csoportos).

Megvalósítások:

1. Az online státuszért az `Account-Mgmt.` (*StatusListener.java*) felel, fogadja a `Socket.IO` egy adott endpoint-jára érkező csatlakozásokat/kilépéseket karbantartva az aktív listát, amit `Redis`-ben tárol, minden változásra közölve az új listát a felhasználóknak; minden csatlakozott kliensnek periodikusan felelőssége közölni jelenlétét egy üzenet formájában (*PresenceContext.tsx*); minden 1 percben újraértékeli az aktív listát, törölve azokat, akik 1 perce nem közöltek magukról semmit.
2. Az értesítésekért az `Activity-Notifications-Mgmt.` felel (*NotificationListener.java*), minden felhasználó belépéskor `RestAPI`-val lekéri az adatbázisba mentett értesítéseit, és `WS`-n keresztül fogadja az újakat; az egyéni/csoportos izoláció `Socket.IO room`-okon keresztül van megoldva.

Megjegyzés

Minden `Socket.IO` interakció token-t igényel.

Zárószó

A kód megtalálható a gitlab.com/community-space oldalon.

Ahol a kód 3 részben van szétbontva:

- `backend/` mappa tartalmazza a projekt micro-service-einek kódját, jelenleg 5 (+ 1) ilyen projekt van ezen belül (5 service + 1 library, mind `Java Spring`-ben);
- `frontend/` mappa tartalmazza a `NextJS` front-end kódot;
- `devops/` mappa tartalmazza a `Terraform`, `Helm`, és `CI/CD` kódrészleteket, amelyek megvalósítják a folyamatos fejlesztést és kitelepítést (minden commit után automatikus build és Docker deploy folyamatok a projekt `Gitlab Registry`-jére), infrastructure-as-code a K8S cluster létrehozásához és `Helm Chart` a könnyű K8S kitelepítésért.