

# O'ZGARUVCHAN MUHITDA SHAXS YUZINI TANIB OLISH TIZIMINING DASTURIY-TEXNIK TA'MINOTI

*Nishanov Akram Hasanovich<sup>1</sup>, Babadjanov Elmurod Satimbayevich<sup>2</sup>, Narziyev Narzullo  
Baxshilloyevich<sup>3</sup>*

<sup>1,3</sup>Muhammad al-Xorazmiy nomidagi Toshkent axborot texnologiyalari universiteti

<sup>2</sup>Nukus davlat texnika universiteti

E-mail: n.b.narziyev@gmail.com

## KEYWORDS

tizim arxitekturasi,  
ma'lumotlar bazasi,  
mikroservislar, edge  
computing, biometrik  
identifikatsiya, RBAC, API

## ABSTRACT

Maqolada manzil-koloniyalarda mahkumlarni masofaviy nazorat qilish tizimining dasturiy-texnik ta'minotini taqdim etilgan. Taqsimlangan hisoblash muhitida ishlaydigan uch darajali arxitektura (edge-fog-cloud) ishlab chiqilgan. Ma'lumotlar bazasi modeli, dasturiy modullar tuzilishi, foydalanuvchi rollari va huquqlar tizimi batafsil tavsiflangan. Tizim xavfsizligi, autentifikatsiya mexanizmlari va tashqi tizimlar bilan integratsiya masalalari ko'rib chiqilgan. Taklif etilgan arxitektura cheklangan tarmoq ulanishi sharoitida ishonchli ishlash, ma'lumotlar yaxlitligini ta'minlash va kengaytiriluvchanlikni ta'minlaydi.

## 1. KIRISH

Oldingi tadqiqotimizda [1] o'zgaruvchan muhitda shaxs yuzini tanib olishning gibril usuli taklif etilgan edi. Ushbu usul klassik preprocessing (CLAHE, adaptive gamma correction) va lightweight neyron tarmoqlarni (MobileFaceNet + ArcFace) birlashtirgan holda, manzil-koloniya sharoitlarida 94.7% aniqlikni ta'minladi. Eksperimental natijalar shuni ko'rsatdiki, taklif etilgan gibril usul yuz holatining o'zgarishi (quyoshda qorayish, ifloslanish, shikastlanish, soqol) sharoitlarida bazaviy usullarga nisbatan 4-7% yuqori aniqlik ko'rsatadi. Inference vaqti mobil qurilmada 58ms ni tashkil etdi – bu real vaqt talabini qondiradi.

Biroq algoritmik yechimning o'zi amaliy tatbiq uchun yetarli emas. Birinchi maqolada taklif etilgan SCRFD detektor (0.5MB, 18ms), MobileFaceNet modeli (1MB, 25ms) va HNSW qidirish algoritmi (3ms, N=5000) samarali ishlashi uchun ularni birlashtiruvchi, ma'lumotlarni boshqaruvchi va foydalanuvchilarga xizmat ko'rsatuvchi dasturiy-texnik infratuzilma zarur. Bu infratuzilma bir qator murakkab muammolarni hal qilishi kerak: taqsimlangan hisoblash, ma'lumotlar sinxronizatsiyasi, xavfsizlik va kengaytiriluvchanlik.

Manzil-koloniyalarda mahkumlarni nazorat qilish tizimi o'ziga xos talablarga ega. O'zbekistonda 40 dan ortiq jazoni ijro etish muassasasi mavjud bo'lib, ular mamlakatning turli hududlarida joylashgan [2]. Bu muassasalar o'rtasida tarmoq ulanishi sifati turlicha – poytaxt atrofida optik tolali aloqa mavjud, uzoq hududlarda esa faqat mobil internet yoki hatto sun'iy yo'ldosh aloqasi ishlatiladi. Shu sababli tizim tarmoq uzilishlariga bardoshli bo'lishi va offline rejimda ham ishlashi kerak.

Oxirgi qurilmalar – Face-ID planshetlar va kuzatuv kameralari – cheklangan hisoblash resurslariga ega. Birinchi maqoladagi eksperimentlar Samsung Galaxy Tab A7 (Snapdragon 662, 3GB RAM) qurilmasida o'tkazildi. Bu qurilma 58ms da identifikatsiya qilish imkonini beradi, ammo murakkab server-side logikani ishlatish mumkin emas. Shu sababli hisoblash yukini qurilma va server o'rtasida to'g'ri taqsimlash zarur.

Biometrik ma'lumotlar maxfiy hisoblanadi. O'zbekiston Respublikasining "Shaxsga doir ma'lumotlar to'g'risida"gi qonuni (2019) [3] shaxsiy ma'lumotlarni yig'ish, saqlash va qayta ishlashda maxfiylik va xavfsizlik talablarini belgilaydi. Biometrik ma'lumotlar (yuz embeddingleri) maxsus kategoriyaga kiradi va

ularni himoya qilish uchun qo'shimcha choralar talab etiladi.

Tizim kengaytiriluvchan bo'lishi kerak. Hozirda har bir koloniyada o'rtacha 300-800 mahkum mavjud [2], ammo bu son o'zgarishi mumkin. Yangi koloniyalar qo'shilishi, mavjud koloniyalar kengayishi yoki qisqarishi mumkin. Shuningdek, yangi qurilmalar (planshetlar, kameralar) va yangi foydalanuvchilar (nazoratchilar, rahbarlar) qo'shilishi oddiy bo'lishi kerak.

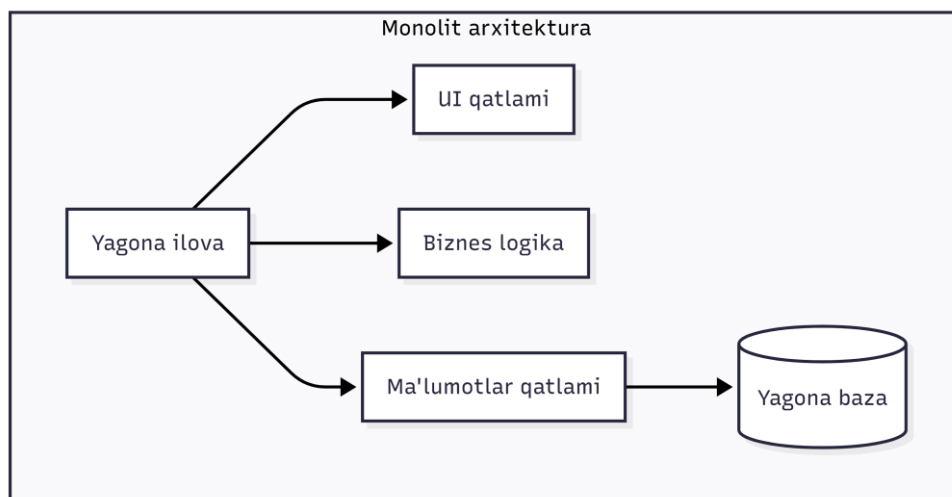
Ushbu maqolaning maqsadi birinchi maqolada taklif etilgan gibrid yuzni tanib olish usulini amalda qo'llash uchun to'liq dasturiy-texnik arxitekturaning ishlab chiqish. Maqolada quyidagi ilmiy masalalar ko'rib chiqiladi: (1) taqsimlangan muhitda real vaqtda biometrik identifikatsiya arxitekturasi; (2) embedding vektorlarni samarali saqlash va qidirish uchun ma'lumotlar bazasi modeli; (3) cheklangan resursli qurilmalarda ishlash uchun dasturiy modullar tuzilishi; (4) maxfiy ma'lumotlarni himoya qilish uchun xavfsizlik arxitekturasi.

## 2. MATERIALLAR VA METODLAR

### 2.1. Arxitektura yondashuvlari tahlili

Dasturiy tizimlar arxitekturasini loyihalashda bir nechta asosiy yondashuvlar mavjud. Har bir yondashuvning o'ziga xos afzalliklari va kamchiliklari bor, shuning uchun tizim talablariga qarab to'g'ri tanlash muhim ahamiyatga ega.

**Monolit arxitektura** barcha tizim komponentlarini yagona deploy birligida birlashtiradi. Bass va boshqalar [4] monolit arxitekturaning "single deployment unit" sifatida ta'riflaydilar – barcha modullar bir jarayonda ishlaydi va ular o'rtasidagi aloqa funktsiya chaqiruvi orqali amalga oshiriladi. Monolit arxitekturaning afzalliklari: ishlab chiqish sodda (bir til, bir loyiha), deploy oson (bitta artifact), debugging qulay (yagona jarayon), tranzaksiyalar sodda (lokal ACID). Biroq monolit arxitekturaning jiddiy kamchiliklari bor: gorizontall kengaytirish qiyin (butun tizimni ko'paytirish kerak), texnologik qulflanish (bir til/freymvork), mustaqil deploy mumkin emas (kichik o'zgartirish uchun butun tizim qayta deploy), katta kod bazasi murakkabligi oshadi. Netflix kompaniyasi 2008-2009 yillarda monolit arxitekturadan mikroservislarga o'tdi, chunki million foydalanuvchiga xizmat ko'rsatishda monolit tizim kengaytirilmas bo'lib qoldi [5].



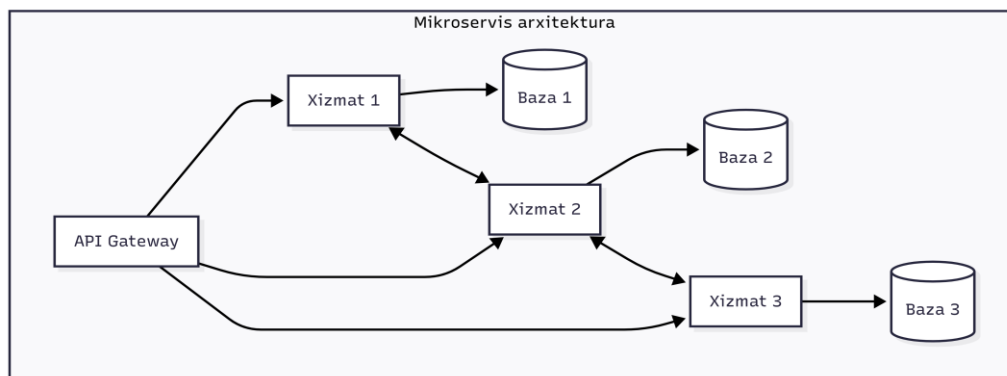
**Mikroservis arxitektura** tizimni kichik, mustaqil xizmatlarga ajratadi. Newman [6] mikroservisni "small, autonomous services that work together" deb ta'riflaydi. Har bir mikroservis o'z ma'lumotlar bazasiga ega (database per service pattern), mustaqil deploy qilinadi va boshqa xizmatlar bilan API orqali aloqa qiladi. Fowler va Lewis [7] mikroservis arxitekturasining asosiy

xususiyatlarini quyidagicha aniqlaganlar: xizmatlar atrofida tashkillash (organized around business capabilities), markazlashtirilmagan boshqaruv (decentralized governance), markazlashtirilmagan ma'lumotlar boshqaruvi (decentralized data management), infratuzilma avtomatlashtirish (infrastructure automation), nosozliklarga chidamlilik (design for failure).

Mikroservis arxitekturasining afzalliklariga mustaqil kengaytirish (faqat kerakli xizmatni ko'paytirish), texnologik erkinlik (har bir xizmat o'z tilida), mustaqil deploy (bir xizmat o'zgarganda boshqalari ta'sirlanmaydi), kichik jamoalar (Conway qonuniga mos) kirsas, kamchiliklari taqsimlangan tizim murakkabligi (tarmoq kechikishi, xatoliklar), ma'lumotlar izchilligi qiyinligi (distributed transactions), operatsion murakkablik (ko'p xizmatlarni

monitoring qilish), xizmatlar orasidagi aloqa qo'shimcha xarajatlardan (overhead) iborat.

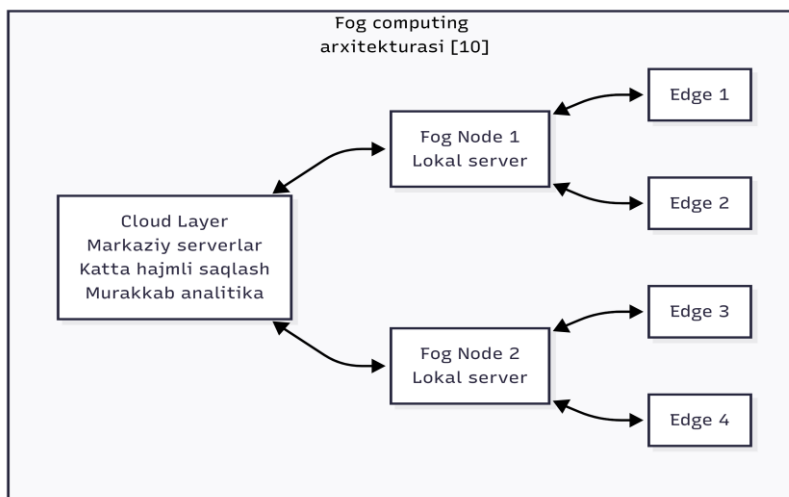
Amazon 2001 yilda "two-pizza teams" kontsepsiyasini, ya'ni har bir jamoa ikki pizza bilan to'yishi mumkin bo'lgan kichik guruhni joriy etdi. Bu kontsepsiya mikroservis arxitekturasiga asos bo'lib, unda har bir kichik jamoa o'z xizmatiga javobgar [8].



**Edge computing** hisoblashni ma'lumot manbasiga yaqinlashtiradi. Shi va boshqalar [9] edge computing ni "hisoblash ilovalari, ma'lumotlari va xizmatlari chegarasini markazlashtirilgan tugunlardan tarmoqning mantiqiy chegaralariga surish" deb ta'rifladilar. Edge computing IoT qurilmalarida keng qo'llaniladi, ya'ni sensorlar ma'lumotni to'g'ridan-to'g'ri cloudga yuborish o'rniga, yaqin joylashgan edge serverda qayta ishlaydi. Bu tarmoq kechikishini kamaytiradi va bandwidth tejaydi. Edge computing arxitekturasida qurilmalar uch turga bo'linadi [9]: (1) narsalar qatlami (things layer) – sensorlar, aktuatorlar, oxirgi qurilmalar; (2) qatlam (edge layer) – gatewaylar, edge serverlar, lokal hisoblash; (3)

bulutli qatlam (cloud layer) – markaziy serverlar, katta ma'lumotlar qayta ishlash. Chekka qatlam narsalar va bulutl o'rtasida oraliq qatlam vazifasini bajaradi.

**Fog computing** edge computing kontsepsiyasini kengaytirib uni Bonomi va boshqalar (Cisco) [10] 2012 yilda taklif qildilar. Fog computing edge va cloud o'rtasida ko'p qatlamli ierarxik struktura yaratadi. Fog tugunlari edge qurilmalarga nisbatan ko'proq resursga ega, ammo cloudga nisbatan kamroq. Fog computing geografik taqsimlanganlik, past kechikish, harakatchanlikni qo'llab-quvvatlash, real vaqtda o'zaro ta'sir qilish, geterogenlik (turli xillik) xususiyatlariga ega.



Yousefpour va boshqalar [11] fog computing va edge computing o'rtasidagi farqlarni tahlil qilganlar. Asosiy farq edge computing ikki qatlamli (edge-cloud), fog computing esa ko'p qatlamli (things-fog-cloud). Fog computing ierarxik tuzilmani qo'llab-quvvatlaydi, ya'ni bir nechta fog qatlamlari bo'lishi mumkin.

Biz tadqiq qilayotgan manzil-koloniya tizimi uchun sof monolit arxitektura mos emas. Chunki (1) geografik taqsimlanganlik (koloniya turli hududlarda); (2) offline ishlash talabi (tarmoq uzilganda ham ishlashi kerak); (3) mustaqil kengaytirish oson bo'lishi kerak. Sof mikroservis arxitektura ham to'liq mos emas, chunki edge qurilmalarda murakkab mikroservis infratuzilmasini ishlatish resurslarga zid.

Shu sababli gibril yondashuv tanlanib, u ushbu darajalardan iborat: (1) cloud darajada – mikroservis arxitektura (face-service, sync-service, report-service mustaqil); (2) fog darajada – yengil monolit (bitta fog server ilovasi); (3) edge darajada – embedded ilova (planshet/kamera dasturi). Mazkur yondashuv Cisco fog computing referens arxitekturasiga [10] asoslangan va IoT tizimlarida keng qo'llaniladi.

## 2.2. Ma'lumotlar bazasi loyihalash metodologiyasi

Ma'lumotlar bazasi tanlashda relatsion (SQL) va norelatsion (NoSQL) yondashuvlar tahlil qilindi.

**Relatsion ma'lumotlar bazalari** strukturalangan ma'lumotlarni jadvallar shaklida saqlaydi. Codd [12] 1970 yilda relatsion modelni taklif qilib, ma'lumotlar bazalari sohasida inqilob yasadi. Relatsion model ACID xususiyatlarini (Atomicity, Consistency, Isolation, Durability) ta'minlaydi, bu moliyaviy va boshqa muhim ma'lumotlar uchun zarur. SQL tili murakkab so'rovlarni (JOIN, aggregation, subquery) bajarish imkonini beradi.

**NoSQL ma'lumotlar bazalari** katta hajmli, strukturalanmagan ma'lumotlar uchun mo'ljallangan. CAP teoremasi (Brewer, 2000) [13] taqsimlangan tizimda bir vaqtda faqat ikkita xususiyatni ta'minlash mumkinligini ko'rsatadi: Consistency, Availability, Partition tolerance.

NoSQL bazalar odatda AP (availability + partition tolerance) ni tanlaydi, eventual consistency bilan.

Bizning tizim uchun quyidagi sabablarga ko'ra relatsion model (PostgreSQL) tanlandi: (1) ma'lumotlar strukturalangan – mahkumlar, embeddinglar, loglar oldindan belgilangan sxemaga ega; (2) murakkab so'rovlar – turli filtrlash, guruhlash, statistika talab etiladi; (3) ma'lumotlar yaxlitligi – ACID xususiyatlari muhim (masalan, mahkum o'chirilganda uning embeddingleri ham o'chirilishi kerak); (4) tranzaksion ishlov – bir nechta jadvalga yozish atomik bo'lishi kerak.

**Embedding vektorlarni saqlash** alohida muammo hisoblanadi. Birinchi maqolada MobileFaceNet 128 o'lchamli embedding hosil qilishi ko'rsatildi. Har bir embedding float32 formatda 512 bayt joy egallaydi. An'anaviy relatsion bazalar vektor operatsiyalarini (cosine similarity, L2 distance) qo'llab-quvvatlamaydi.

Dunyodagi eng mashhur relyatsion ma'lumotlar bazalaridan biri bo'lgan pgvector kengaytmasi [14] PostgreSQL uchun vektor ma'lumotlar turini va operatsiyalarini qo'shadi. Pgvector bizga VECTOR(n) ma'lumot turi (n – o'lcham), <=> operatori (cosine distance), <-> operatori (L2 distance), <#> operatori (inner product), HNSW va IVFFlat indekslari (tezkor ANN qidirish) imkoniyatlarini beradi Johnson va boshqalar (Facebook) [15] FAISS kutubxonasini yaratib, bu millionlab vektorlarda tezkor qidirish imkonini beradi. pgvector FAISS ga o'xshash funktsionallikni PostgreSQL ichida ta'minlaydi.

Bizning tizimda embedding qidirish quyidagicha ishlaydi. Birinchi maqolada taklif etilgan HNSW algoritmi pgvector da ham qo'llaniladi:

```
-- HNSW indeks yaratish
CREATE INDEX ON embeddings
USING hnsw (vector vector_cosine_ops)
WITH (m = 16, ef_construction = 64);
-- 1:N qidirish (eng yaqin 5 ta)
SELECT prisoner_id, 1 - (vector <=> '[0.1, 0.2, ...]')
AS similarity
FROM embeddings
WHERE colony_id = 'uuid'
ORDER BY vector <=> '[0.1, 0.2, ...]'
LIMIT 5;
```



Bu so'rov birinchi maqoladagi HNSW\_Search funksiyasiga ekvivalent, ammo SQL da ifodalangan.

**Normalizatsiya va denormalizatsiya.** Ma'lumotlar bazasi loyihalashda Kod normalizatsiya qoidalariga [12] rioya qilindi. Asosiy jadvallar uchinchi normal shaklga (3NF) keltirildi – bu ma'lumotlar takrorlanishini kamaytiradi va anomalialarni oldini oladi. Biroq ba'zi holatlarda denormalizatsiya qo'llandi. Masalan, sanoq loglari jadvalida (attendance\_logs) mahkum ismi va koloniya nomi takrorlanadi:

```
CREATE TABLE attendance_logs (  
    id UUID PRIMARY KEY,  
    prisoner_id UUID REFERENCES  
prisoners(id),  
    prisoner_name VARCHAR(255), --  
denormalizatsiya  
    colony_id UUID REFERENCES  
colonies(id),  
    colony_name VARCHAR(100), --  
denormalizatsiya  
    ...  
);
```

Bu denormalizatsiya ikkita sababga asoslangan, ya'ni so'rovlar tezligi (loglarni ko'rsatishda JOIN kerak emas) va tarixiy yaxlitlik (mahkum ismi yoki koloniya nomi o'zgarsa ham, eski loglar to'g'ri ko'rinadi).

### 2.3. Xavfsizlik metodologiyasi

Xavfsizlik arxitekturasida "defense in depth" (chuqur himoya) prinsipi qo'llandi. NIST SP 800-53 [16] standartida ta'riflanganidek, bu prinsip bir nechta xavfsizlik qatlamlarini yaratishni nazarda tutadi (bitta qatlam buzilsa ham, boshqalari himoyani davom ettiradi).

**Autentifikatsiya** uchun JWT (JSON Web Token) standarti tanlandi. Jones va boshqalar [17] JWT ni RFC 7519 da standartlashtirdilar. JWT uch qismdan iborat: header (algoritm va token turi), payload (claims – foydalanuvchi ma'lumotlari) va signature (imzo). JWT stateless server sessiya saqlamaydi, token o'zida barcha kerakli ma'lumotni saqlaydi. Bu gorizontal kengaytirishni osonlashtiradi – har qanday server tokenni tekshirishi mumkin. JWT ning muqobillari (session-based auth, OAuth tokens) tahlil qilindi.

Session-based autentifikatsiya serverda sessiya saqlashni talab qiladi – bu taqsimlangan tizimda murakkablik qo'shadi. OAuth 2.0 tashqi identity provider bilan ishlash uchun mo'ljallangan – bizning ichki tizim uchun ortiqcha murakkablik. Shu sababli JWT tanlandi.

**Avtorizatsiya** uchun RBAC (Role-Based Access Control) modeli qo'llandi. Sandhu va boshqalar [18] RBAC ni 1996 yilda formallashtirib, huquqlarni boshqarishning samarali usulini taklif qildilar. RBAC da foydalanuvchilarga to'g'ridan-to'g'ri huquqlar emas, balki rollar biriktiriladi. Har bir rol huquqlar to'plamiga ega. Bu boshqarishni soddalashtiradi – yangi foydalanuvchiga kerakli rolni biriktirish kifoya. RBAC ning muqobillari – ABAC (Attribute-Based Access Control) va ACL (Access Control Lists) – tahlil qilindi. ABAC moslashuvchan, ammo murakkab va sekin. ACL soddaga, ammo ko'p foydalanuvchilarda boshqarish qiyin. RBAC o'rtacha murakkablik va yetarli moslashuvchanlik ta'minlaydi.

**Biometrik ma'lumotlar himoyasi.** O'zbekiston Respublikasining "Shaxsga doir ma'lumotlar to'g'risida"gi qonuni [3] biometrik ma'lumotlarni maxsus kategoriyaga kiritadi. Qonunning 17-moddasiga ko'ra, shaxsga doir ma'lumotlarni qayta ishlashda ularning xavfsizligini ta'minlash tadbirlari ko'rilishi shart. Embedding vektorlar shifrlangan holda saqlanadi. AES-256-GCM algoritmi tanlandi – bu NIST tomonidan tasdiqlangan [16] va keng qo'llaniladigan standart. GCM (Galois/Counter Mode) shifrlash bilan birga autentifikatsiyani ham ta'minlaydi – ma'lumotlar o'zgartirilganligini aniqlash mumkin.

### 2.4. Ishlab chiqish vositalari va texnologiyalar

**Backend.** Python 3.11 dasturlash tili va FastAPI freymvorki tanlandi. Ramirez [19] FastAPI ni "high performance, easy to learn, fast to code, ready for production" deb ta'riflaydi. FastAPI ASGI (Asynchronous Server Gateway Interface) asosida ishlaydi va asinxron operatsiyalarni qo'llab-quvvatlaydi. OpenAPI spesifikatsiyasi avtomatik generatsiya qilinadi – bu API dokumentatsiyasi va client SDK yaratishni osonlashtiradi. FastAPI ning muqobillari – Django, Flask, Tornado – tahlil qilindi. Django

"batteries included" yondashuvi bilan to'liq veb-freymvork, ammo API uchun ortiqcha. Flask minimalist va moslashuvchan, ammo asinxron qo'llab-quvvatlash cheklangan. FastAPI zamonaviy Python (type hints, async/await) imkoniyatlaridan to'liq foydalanadi va eng yuqori unumdorlikni ta'minlaydi.

**Ma'lumotlar bazasi.** PostgreSQL 15 tanlandi. PostgreSQL ochiq kodli, kengaytiriluvchan va korporativ darajadagi ishonchlilikka ega. pgvector kengaytmasi vektor operatsiyalarini qo'shadi. PostGIS kengaytmasi geografik ma'lumotlar (GPS koordinatalar) bilan ishlash imkonini beradi.

**Edge qurilmalar.** Android platformasi tanlandi – bu manzil-koloniyalarda ishlatiladigan planshetlarning asosiy platformasi. Kotlin dasturlash tili Java ga nisbatan xavfsizroq va ifodali. Jetpack Compose zamonaviy deklarativ UI freymvorki. ONNX Runtime [20] turli platformalarda (Android, iOS, Windows, Linux) bir xil modelni ishlatish imkonini beradi – birinchi maqoladagi MobileFaceNet modeli ONNX formatga eksport qilinadi va mobil ilovada ishlatiladi.

**Konteynerlashtirish.** Docker konteynerlari dasturiy ta'minotni izolyatsiyalangan muhitda ishlatish imkonini beradi. Merkel [21] Docker ni "lightweight virtualization" deb ta'riflaydi – virtual mashinalarga nisbatan kam resurs sarflaydi va tez ishga tushadi. Docker Compose bir nechta konteynerlarni birgalikda boshqarish uchun ishlatiladi.

### 3. NATIJALAR

#### 3.1. Tizim arxitekturas

Ishlab chiqilgan tizim uch darajali arxitekturaga ega. Har bir daraja o'z vazifasiga ega va boshqa darajalar bilan belgilangan interfeyslar orqali aloqa qiladi. Bu arxitektura Bonomi va boshqalar [10] tomonidan taklif etilgan fog computing modeliga asoslangan, ammo biometrik identifikatsiya tizimining o'ziga xos talablariga moslashtirilgan.

**Edge daraja (Things Layer)** oxirgi qurilmalardan iborat – Face-ID planshetlar va kuzatuv kameralari. Bu daraja birinchi maqolada

[1] tavsiflanagan gibrid yuzni tanib olish usulini amalga oshiradi. Planshetlar nazoratchilar tomonidan mahkumlarni qo'lda qayd etish uchun ishlatiladi – nazoratchi mahkumga planshet kamerasini yo'naltiradi, tizim yuzni aniqlaydi va identifikatsiya qiladi. Kameralar esa avtomatik monitoring uchun mo'ljallangan – ular doimiy video oqimdan yuzlarni aniqlaydi va identifikatsiya qiladi.

Edge qurilmada birinchi maqoladagi barcha algoritmlar lokal ishga tushiriladi. SCRFD detektor (0.5MB) yuzni aniqlaydi va 5 ta landmark nuqtasini qaytaradi. Yuzni tekislash moduli affine transformatsiya orqali yuzni 112×112 formatga keltiradi. Preprocessing pipeline (YCbCr konversiya, CLAHE, adaptive gamma) yoritish muammolarini hal qiladi. MobileFaceNet modeli (1MB, ONNX formatda) 128 o'lchamli embedding hosil qiladi. Jami inference vaqti 58ms – bu birinchi maqoladagi eksperimental natijaga mos.

Edge qurilmada lokal ma'lumotlar bazasi mavjud – SQLite formatida. Bu baza markaziy bazaning qisman nusxasini saqlaydi – faqat shu koloniyaga tegishli mahkumlar ro'yxati va ularning embeddinglari. Lokal bazada HNSW indeksi yaratilgan – bu birinchi maqoladagi qidirish algoritmining SQLite tatbiqi. Lokal baza hajmi odatda 50-100MB ni tashkil etadi (500 mahkum, har biri uchun 2 ta embedding, har biri 512 bayt = 512KB embeddinglar + metadata).

Offline rejimda edge qurilma lokal bazadan foydalanib identifikatsiya qiladi. Natijalar (sanoq loglari) lokal saqlanadi va tarmoq tiklangach fog serverga yuboriladi. Bu "store-and-forward" mexanizmi tarmoq uzilishlariga bardoshlilikni ta'minlaydi.

**Fog daraja (Fog Layer)** koloniya lokal serveridan iborat. Har bir koloniyada bitta fog server o'rnatiladi. Fog server edge va cloud o'rtasida oraliq qatlam vazifasini bajaradi – Bonomi va boshqalar [10] ta'riflaganidek, "intermediate computing layer between edge and cloud".

Fog server quyidagi vazifalarni bajaradi. Birinchidan, edge qurilmalardan ma'lumotlarni qabul qilish va validatsiya qilish – noto'g'ri formatdagi yoki shubhali ma'lumotlar rad etiladi.

Ikkinchidan, ma'lumotlarni vaqtinchalik saqlash – PostgreSQL bazasida (cloud bilan sinxronlashgunicha). Uchinchidan, cloud serverga sinxronlash – yangi sanoq loglari, yangi embeddinglar, o'zgargan ma'lumotlar. To'rtinchidan, cloud dan yangilanishlarni olish – yangi mahkumlar, yangilangan embeddinglar, o'chirilgan yozuvlar. Beshinchidan, edge qurilmalarni yangilash – lokal bazalarni sinxronlash.

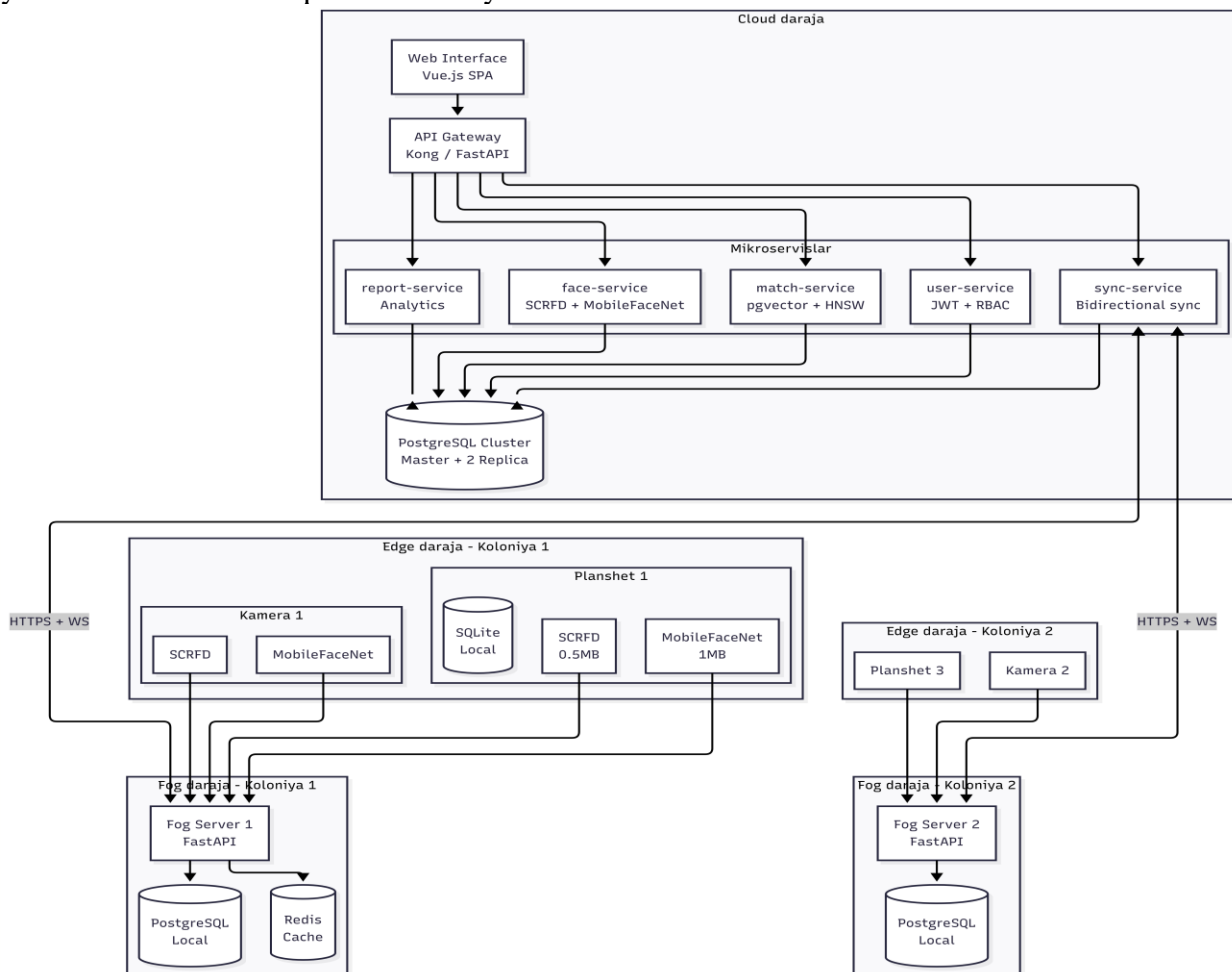
Fog server nisbatan oddiy apparat talablariga ega – Intel Core i5 yoki undan yuqori protsessor, 8GB RAM, 256GB SSD. Operatsion tizim sifatida Ubuntu Server 22.04 LTS ishlatiladi. Fog serverdagi dasturiy ta'minot Docker konteynerlarida ishlaydi – bu deploy va yangilashni osonlashtiradi.

Fog daraja muhim ahamiyatga ega, chunki u tarmoq uzilishlariga bardoshlilikni ta'minlaydi. Agar cloud bilan aloqa uzilsa, fog server bufer vazifasini bajaradi – edge qurilmalardan kelgan ma'lumotlarni yig'adi va aloqa tiklangach cloudga yuboradi. Bu vaqtda koloniya ichida

identifikatsiya to'liq ishlaydi – faqat markaziy hisobotlar va boshqa koloniyalar ma'lumotlariga kirish cheklanadi.

**Cloud daraja (Cloud Layer)** markaziy serverdan iborat. Cloud server Jazoni ijro etish departamentining ma'lumotlar markazida joylashgan. Cloud server quyidagi vazifalarni bajaradi: barcha koloniyalardan ma'lumotlarni qabul qilish va birlashtirish, markaziy ma'lumotlar bazasini yuritish, yagona "source of truth" sifatida, analitika va hisobotlar generatsiya qilish, foydalanuvchilarni boshqarish va web interfeys xizmati.

Cloud server yuqori mavjudlik (high availability) uchun klasterda ishlaydi. PostgreSQL ma'lumotlar bazasi master-replica konfiguratsiyasida o'rnatilgan – master serverga yozish, replica serverlardan o'qish. Replikatsiya sinxron – ma'lumot yo'qolish xavfi minimal. Tizim 99.9% uptime ta'minlashga mo'ljallangan – bu yiliga maksimum 8.76 soat ishlamay qolish degani.



### Darajalar o'rtasidagi aloqa protokollari.

Edge va fog o'rtasida HTTP/REST protokoli ishlatiladi. So'rovlar JSON formatda, TLS 1.3 bilan shifrlangan. Autentifikatsiya device token orqali – har bir qurilma ro'yxatdan o'tganda unikal token oladi. Fog va cloud o'rtasida ikki turdagi aloqa mavjud. REST API – standart CRUD operatsiyalari uchun (yangi mahkum qo'shish,

ma'lumotlarni yangilash). WebSocket – real-time yangilanishlar uchun (yangi sanoq logi kelganda darhol cloudga xabar berish). WebSocket aloqasi doimiy ochiq turadi va fog serverdan cloudga push notification imkonini beradi.

**Tarmoq uzilishi holatida ishlash algoritmi.** Quyidagi algoritm tarmoq uzilishiga bardoshlilikni ta'minlaydi:

EDGE QURILMA
<ul style="list-style-type: none"> <li>1. Identifikatsiya so'rovi keldi</li> <li>2. Lokal bazada HNSW qidirish</li> <li>3. Natijani foydalanuvchiga ko'rsatish</li> <li>4. Sanoq logini lokal saqlash (SQLite)</li> <li>5. Fog serverga yuborishga urinish</li> <li>6. AGAR muvaffaqiyatsiz: <ul style="list-style-type: none"> <li>- Logni "pending" deb belgilash</li> <li>- Keyingi urinish uchun navbatga qo'shish</li> </ul> </li> <li>7. Tarmoq tiklanganda: <ul style="list-style-type: none"> <li>- Barcha "pending" loglarni yuborish</li> <li>- Fog serverdan yangilanishlarni olish</li> </ul> </li> </ul>

FOG SERVER
<ul style="list-style-type: none"> <li>1. Edge dan log qabul qilindi</li> <li>2. Lokal bazaga yozish</li> <li>3. Cloud ga yuborishga urinish</li> <li>4. AGAR muvaffaqiyatsiz: <ul style="list-style-type: none"> <li>- Logni "pending_cloud" deb belgilash</li> </ul> </li> <li>5. Tarmoq tiklanganda: <ul style="list-style-type: none"> <li>- Barcha "pending_cloud" loglarni yuborish</li> <li>- Cloud dan yangilanishlarni olish</li> <li>- Edge qurilmalarni yangilash</li> </ul> </li> </ul>

Bu algoritm "eventual consistency" modelini amalga oshiradi – ma'lumotlar darhol sinxronlanmasa ham, tarmoq tiklanganda oxir-oqibat barcha darajalarda bir xil bo'ladi.

### 3.2. Ma'lumotlar bazasi modeli

Ma'lumotlar bazasi 12 ta asosiy jadvaldan iborat. Jadvallar mantiqiy guruhlariga bo'lingan: shaxslar (mahkumlar, xodimlar), tashkiliy birliklar (koloniyalar, zonalar), operatsion ma'lumotlar (sanoq, tadbirlar) va tizim ma'lumotlari (foydalanuvchilar, loglar).

**Mahkumlar jadvali (prisoners)** tizimning asosiy jadvali hisoblanadi. Har bir mahkum uchun quyidagi ma'lumotlar saqlanadi: noyob identifikator (UUID formatda), ism-sharif, tug'ilgan sana, jinsi, JShShIR (Jismoniy shaxsning shaxsiy identifikatsiya raqami), jazo muddati boshlanish va tugash sanalari, koloniya identifikatori (tashqi kalit) va holat (faol, ozod qilingan, ko'chirilgan). Indekslar koloniya identifikatori va holat ustunlarida yaratilgan, chunki aksariyat so'rovlar shu maydonlar bo'yicha filtrlaydi.

**Embeddinglar jadvali (embeddings)** yuz embeddinglarini saqlaydi. Har bir mahkumning bir yoki bir nechta embeddingi bo'lishi mumkin (turli vaqtda, turli sharoitda olingan). Jadvalda quyidagi maydonlar mavjud: noyob identifikator, mahkum identifikatori (tashqi kalit), embedding vektori (512 bayt, BYTEA turi), yaratilgan sana, manba (ro'yxatdan o'tish, yangilash, avtomatik) va sifat balli (0-100). pgvector kengaytmasi yordamida embedding ustunida HNSW indeksi yaratilgan – bu 1:N qidiruvni tezlashtiradi.

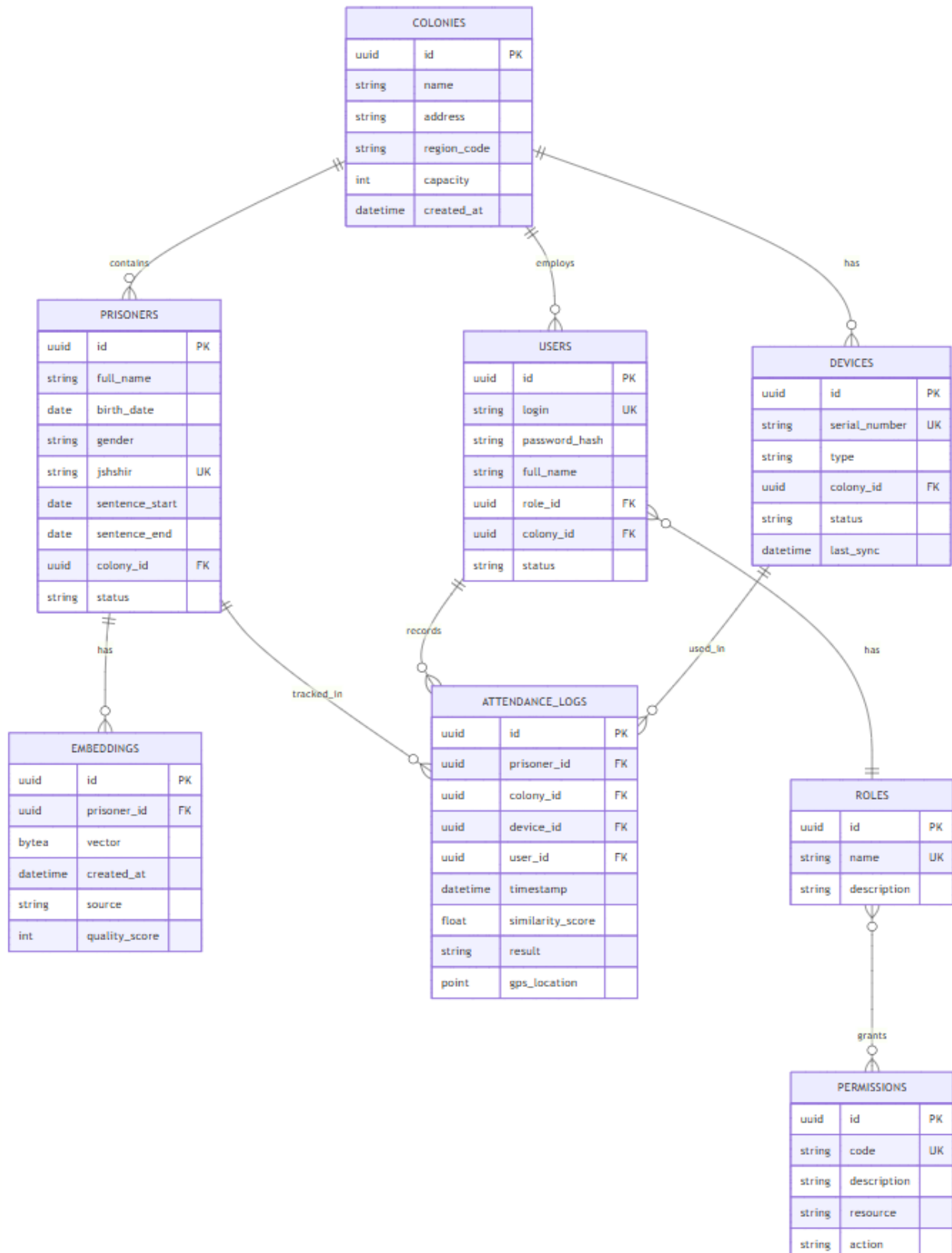
**Koloniyalar jadvali (colonies)** tashkiliy birliklarni saqlaydi. Har bir koloniya uchun: noyob identifikator, nomi, manzili, viloyat kodi, hudud identifikatori (ierarxiya uchun), sig'imi (maksimal mahkumlar soni) va aloqa ma'lumotlari saqlanadi.

**Sanoq loglari jadvali (attendance\_logs)** operatsion ma'lumotlarni saqlaydi. Bu jadval eng tez o'sadigan jadval – har bir identifikatsiya natijasi bu yerga yoziladi. Maydonlar: noyob identifikator, mahkum identifikatori, koloniya identifikatori, qurilma identifikatori, sana va vaqt, o'xshashlik balli, natija (muvaffaqiyatli, rad etilgan, noma'lum), GPS koordinatalari va tasvir havolasi (ixtiyoriy). Jadval oy bo'yicha partitsiyalangan – bu eski ma'lumotlarni arxivlash va so'rovlar tezligini oshirish imkonini beradi.



**Foydalanuvchilar jadvali (users)** tizim foydalanuvchilarini saqlaydi. Maydonlar: noyob identifikator, login, parol xeshi (bcrypt), ism-sharif, rol identifikatori, koloniya identifikatori (agar xodim ma'lum koloniyaga biriktirilgan bo'lsa), holat va oxirgi kirish vaqti.

**Rollar jadvali (roles)** va **Huquqlar jadvali (permissions)** RBAC tizimini amalga oshiradi. Rollar va huquqlar ko'p-ko'p munosabatda – bitta rol ko'p huquqlarga, bitta huquq ko'p rollarga ega bo'lishi mumkin. Oraliq jadval (role\_permissions) bu munosabatni saqlaydi.



**Indeksalar va optimizatsiya.** Tez-tez ishlatiladigan so'rovlar uchun kompozit indeksalar yaratilgan. Masalan, (colony\_id, timestamp) indeksi "ma'lum koloniyada ma'lum vaqt oralig'idagi sanoqlar" so'rovi uchun. EXPLAIN ANALYZE yordamida so'rovlar tahlil qilindi va sekin so'rovlar optimizatsiya qilindi.

Ma'lumotlar bazasi hajmi prognozi: 50 koloniya, har birida 500 mahkum (jami 25,000), har bir mahkumda 2 embedding, kuniga 5 sanoq – yiliga taxminan 45 million sanoq logi. Sanoq loglari jadvali yiliga taxminan 10GB o'sadi. 5 yillik ma'lumotlar uchun 50GB joy ajratilgan.

### 3.3. Dasturiy modullar

Tizim dasturiy ta'minoti bir nechta mustaqil modullardan iborat. Har bir modul o'z vazifasiga ega va boshqa modullar bilan API orqali aloqa qiladi.

**face-service** – yuzni aniqlash va embedding hisoblash xizmati. Bu xizmat SCRFD va MobileFaceNet modellarini o'z ichiga oladi. Kirish: tasvir (base64 yoki URL). Chiqish: yuz koordinatalari, landmark nuqtalar, embedding vektor va sifat bali. Xizmat stateless – har bir so'rov mustaqil qayta ishlanadi. GPU mavjud bo'lsa, undan foydalanadi, aks holda CPU da ishlaydi. Bir soniyada 10-15 ta tasvirni qayta ishlash imkoniyatiga ega (CPU rejimda).

**match-service** – embedding moslik xizmati. Kirish: probe embedding va qidirish parametrlari (koloniya filtri, chegara qiymati). Chiqish: eng

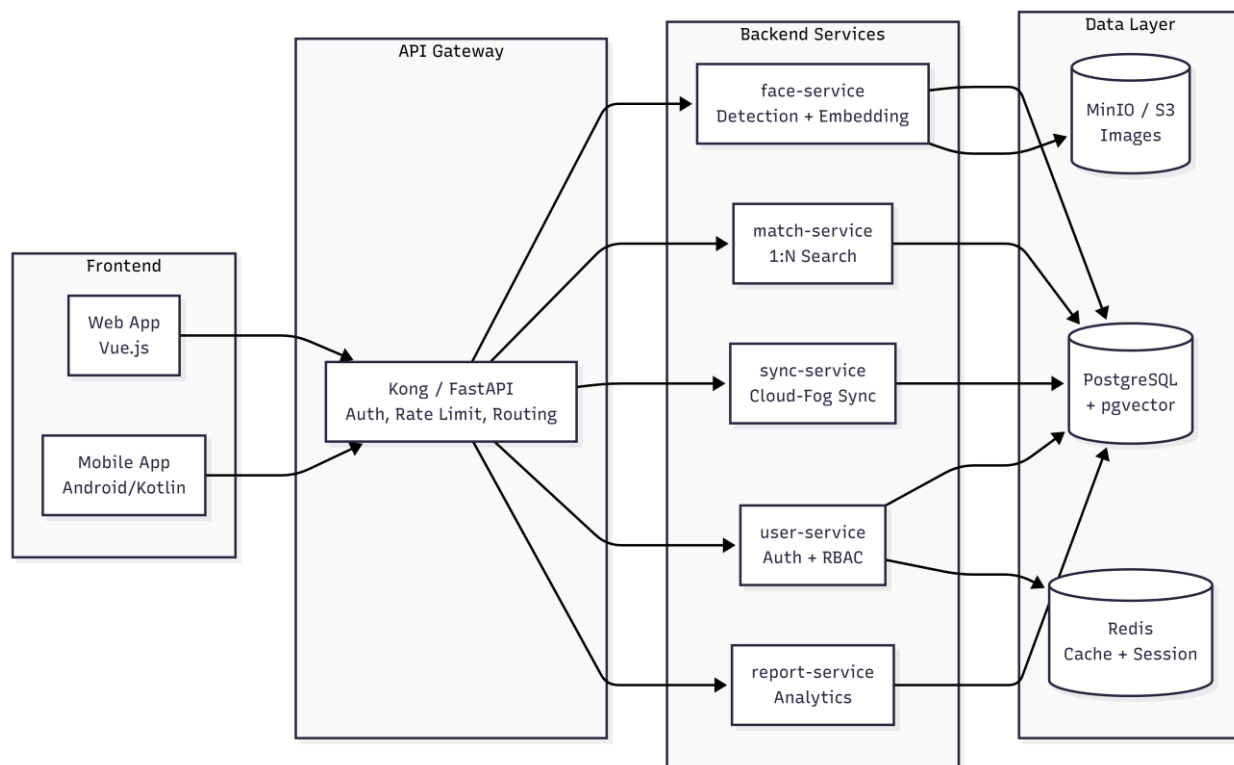
yaqin N ta moslik, har biri uchun mahkum identifikatori va o'xshashlik balli. Xizmat pgvector orqali ma'lumotlar bazasida qidiradi. Katta bazalar uchun (N>10,000) HNSW indeksi ishlatiladi.

**sync-service** – sinxronizatsiya xizmati. Bu xizmat fog va cloud o'rtasidagi ma'lumot almashuvini boshqaradi. Ikki yo'nalishli sinxronizatsiya: cloud dan fog ga (yangi mahkumlar, yangilangan embeddinglar), fog dan cloud ga (sanoq loglari, yangi embeddinglar). Konfliktlarni hal qilish "last-write-wins" strategiyasi bilan, ammo muhim maydonlar (masalan, mahkum holati) uchun qo'lda tasdiqlash talab etiladi.

**api-gateway** – API gateway xizmati. Barcha tashqi so'rovlar shu xizmat orqali o'tadi. Vazifalar: autentifikatsiya (JWT tekshirish), avtorizatsiya (huquqlarni tekshirish), rate limiting (DDoS himoya), so'rovlarni tegishli xizmatlarga yo'naltirish va loglar yozish. Kong yoki custom FastAPI ilovasi sifatida amalga oshirilgan.

**web-app** – web interfeys. Vue.js 3 asosida single-page application (SPA). Komponentlar: login sahifasi, dashboard (statistika va grafiklar), mahkumlar ro'yxati, sanoq loglari, hisobotlar generatori va admin panel. Responsive dizayn – planshet va kompyuterda ishlaydi.

**mobile-app** – Android ilovasi. Kotlin + Jetpack Compose. Vazifalar: kameradan tasvir olish, yuzni aniqlash va embedding hisoblash (lokal), fog serverga natijalarni yuborish, offline rejimda ishlash va lokal bazani sinxronlash. Ilova 50MB dan kam joy egallaydi.



**API spesifikasiyasi.** REST API OpenAPI 3.0 spesifikasiyasiga mos. Asosiy endpointlar:

- POST /api/v1/identify – tasvirdan shaxsni aniqlash. Kirish: tasvir (base64), koloniya identifikatori. Chiqish: mahkum ma'lumotlari yoki "noma'lum".
- GET /api/v1/prisoners – mahkumlar ro'yxati. Query parametrlari: colony\_id, status, page, limit. Chiqish: mahkumlar massivi, pagination ma'lumotlari.
- POST /api/v1/prisoners/{id}/embeddings – yangi embedding qo'shish. Kirish: tasvir (base64). Chiqish: embedding identifikatori, sifat balli.
- GET /api/v1/attendance – sanoq loglari. Query parametrlari: colony\_id, start\_date, end\_date, prisoner\_id. Chiqish: loglar massivi.
- POST /api/v1/sync/push – fog dan cloud ga ma'lumot yuborish. Kirish: yangi loglar, yangi embeddinglar. Chiqish: qabul qilingan yozuvlar soni.
- GET /api/v1/sync/pull – cloud dan fog ga ma'lumot olish. Query parametrlari: colony\_id, last\_sync\_time. Chiqish: yangi o'zgargan mahkumlar va embeddinglar.

### 3.4. Foydalanuvchilar va rollar

Tizimda to'rt asosiy rol mavjud: Administrator, Nazoratchi, Koloniya rahbari va

Departament xodimi. Har bir rol o'ziga xos huquqlar to'plamiga ega.

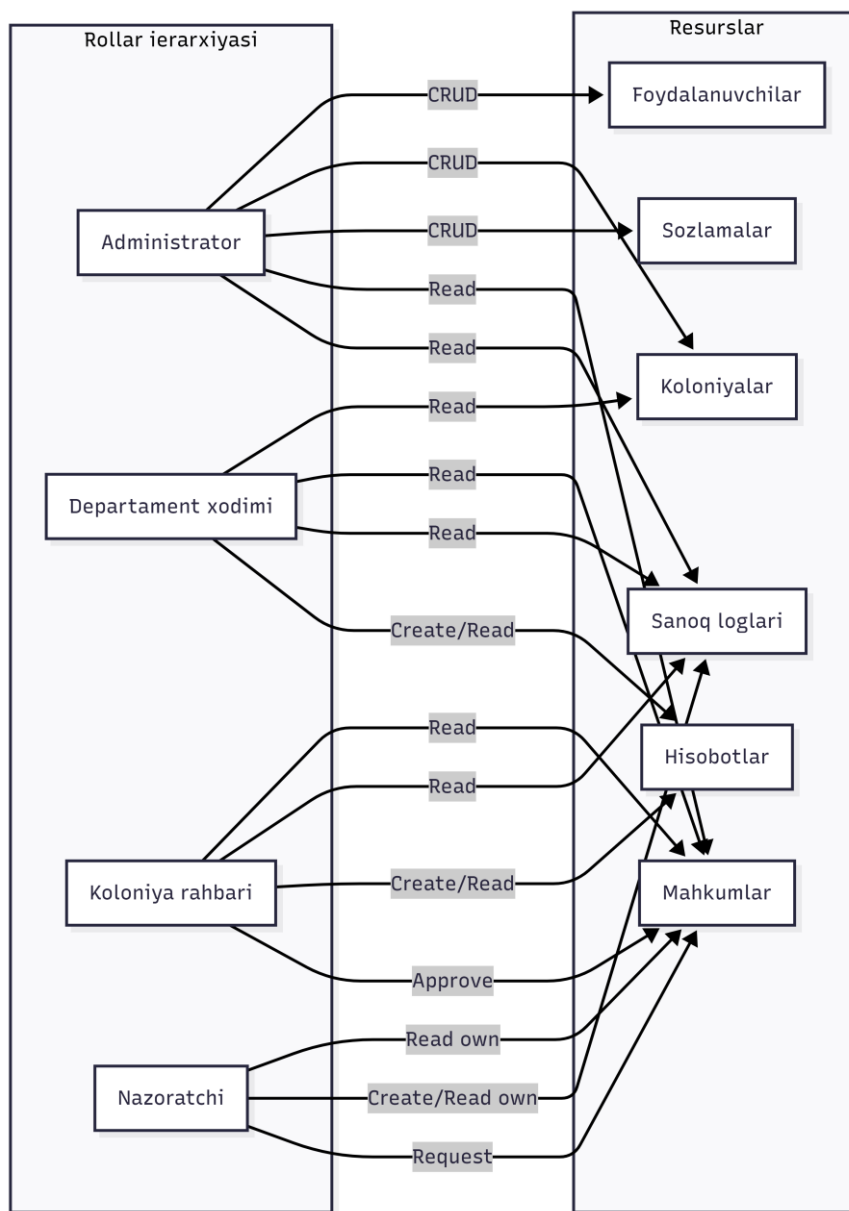
**Administrator** tizimni boshqaradi. Huquqlari: foydalanuvchilarni yaratish, tahrirlash va o'chirish; rollar va huquqlarni boshqarish; koloniyalarni ro'yxatga olish; qurilmalarni ro'yxatga olish; tizim sozlamalarini o'zgartirish va audit loglarini ko'rish. Administrator soni cheklangan – butun tizim uchun 2-3 nafar.

**Nazoratchi** mahkumlarni kundalik qayd etadi. Huquqlari: mahkumlarni identifikatsiya qilish (planshet orqali); o'z koloniyasidagi mahkumlar ro'yxatini ko'rish; sanoq loglarini ko'rish (faqat o'zi yozganlar); mahkum uchun yangi embedding qo'shish (rahbar tasdig'i bilan). Nazoratchi faqat biriktirilgan koloniyada ishlaydi – boshqa koloniya ma'lumotlariga kirish huquqi yo'q.

**Koloniya rahbari** koloniyani boshqaradi. Huquqlari: o'z koloniyasidagi barcha ma'lumotlarni ko'rish; nazoratchilarning ishini nazorat qilish; hisobotlar generatsiya qilish; yangi embedding qo'shishni tasdiqlash; koloniya darajasidagi statistikani ko'rish. Rahbar mahkumlarni tahrirlash huquqiga ega emas – bu faqat markazdan amalga oshiriladi.

**Departament xodimi** markaziy monitoring qiladi. Huquqlari: barcha koloniyalar bo'yicha statistikani ko'rish; umumlashtiruvchi hisobotlar generatsiya qilish; koloniyalararo taqqoslash;

ogohlantirishlarni ko'rish. Departament xodimi operatsion amallarni (identifikatsiya, tahrirlash) bajarmaydi – faqat monitoring va hisobot.



**Huquqlar matritsasi.** Har bir resurs uchun CRUD (Create, Read, Update, Delete) huquqlari alohida belgilanadi. Quyidagi jadval huquqlar

taqsimotini ko'rsatadi (✓ – to'liq huquq, ○ – qisman huquq, faqat o'z koloniyasi, – huquq yo'q):

Resurs	Administrator	Departament	Rahbar	Nazoratchi
Foydalanuvchilar (CRUD)	✓	–	–	–
Koloniyalar (CRUD)	✓	R	R○	R○
Mahkumlar (CRUD)	CRU	R	R○	R○
Embeddinglar (CRUD)	CRUD	R	CR○	CR○
Sanoq loglari (CRUD)	R	R	R○	CR○
Hisobotlar (CRUD)	CRUD	CR	CR○	–
Sozlamalar (CRUD)	CRUD	–	–	–



### 3.5. Xavfsizlik va autentifikatsiya

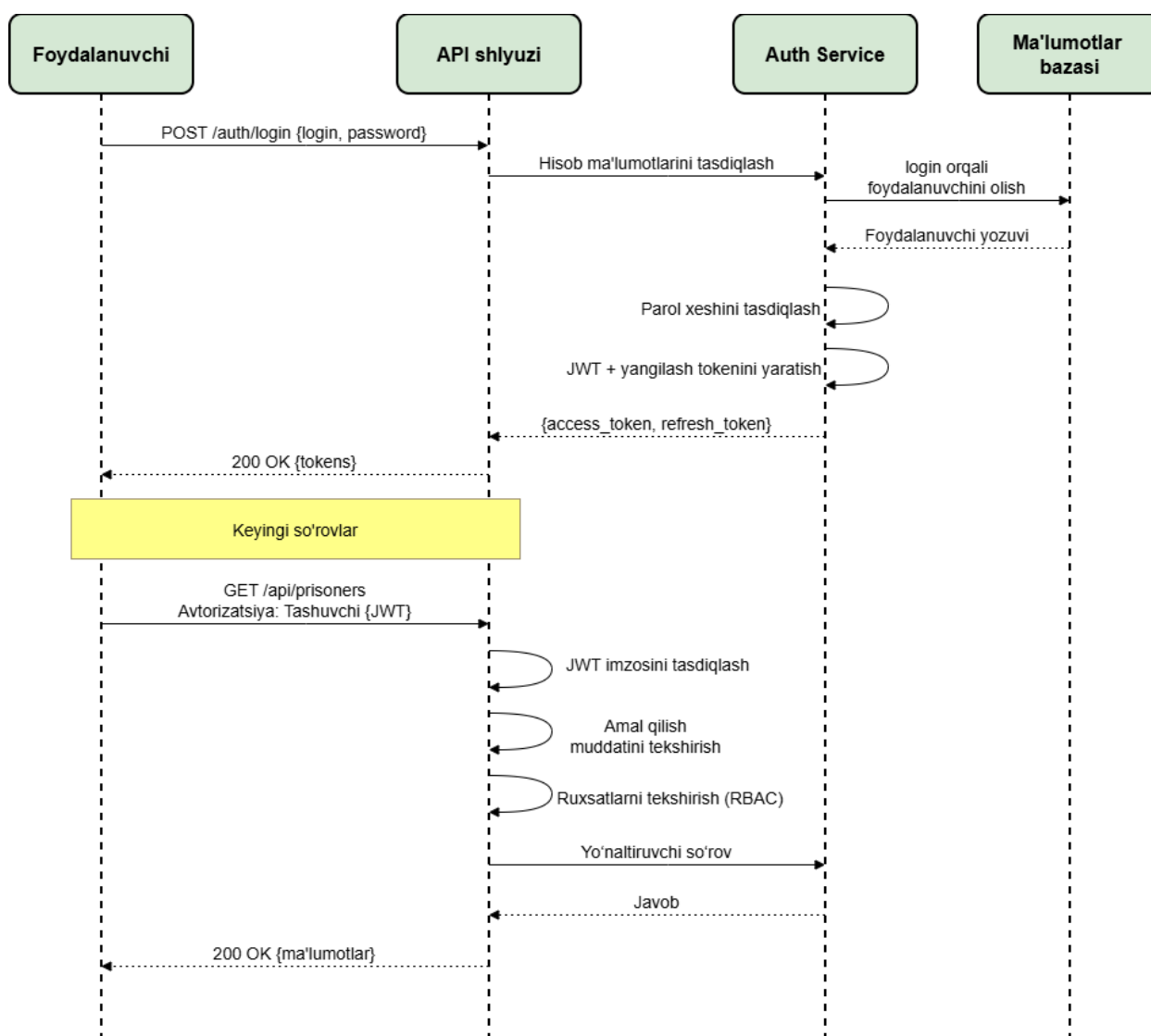
Tizim xavfsizligi bir nechta qatlamda ta'minlanadi: tarmoq darajasi, ilova darajasi va ma'lumotlar darajasi.

**Tarmoq darajasi.** Barcha aloqalar HTTPS (TLS 1.3) orqali shifrlangan. Fog va cloud o'rtasida VPN tunnel (WireGuard) o'rnatilgan. Firewall qoidalarini faqat zarur portlarni ochiq qoldiradi. DDoS himoyasi uchun rate limiting va IP bloklash qo'llaniladi.

### Ilova darajasi – autentifikatsiya.

Foydalanuvchilar login va parol bilan autentifikatsiya qiladi. Parollar bcrypt algoritmi bilan xeshlanadi (cost factor 12). Muvaffaqiyatli autentifikatsiyadan so'ng JWT token beriladi. Token 24 soat amal qiladi, refresh token esa 7 kun.

JWT token strukturasi: header (algoritm HS256), payload (foydalanuvchi ID, rol ID, koloniya ID, exp vaqti) va signature. Token har bir API so'rovida Authorization headerda yuboriladi. Server tokenni tekshiradi va payload dan foydalanuvchi ma'lumotlarini oladi.



**Ilova darajasi – avtorizatsiya.** Har bir API endpoint uchun talab qilinadigan huquq belgilangan. API Gateway so'rovni qabul qilganda, JWT dan rol ID ni oladi, keyin rol huquqlarini tekshiradi. Agar foydalanuvchida kerakli huquq bo'lmasa, 403 Forbidden qaytariladi.

Koloniya cheklovi ham qo'llaniladi. Nazoratchi va rahbar faqat o'z koloniyasi ma'lumotlariga kirishi mumkin. So'rovda colony\_id parametri foydalanuvchining colony\_id si bilan solishtiriladi. Mos kelmasa, 403 qaytariladi.

**Ma'lumotlar darajasi.** Biometrik ma'lumotlar (embedding vektorlar) AES-256-GCM algoritmi bilan shifrlangan. Shifrlash kaliti environment variable da saqlanadi va faqat xizmatlar kirishi mumkin. Ma'lumotlar bazasi darajasida ham shifrlash qo'llaniladi (PostgreSQL TDE).

**Audit logging.** Barcha muhim amallar (login, logout, ma'lumotlarni o'qish/yozish, sozlamalarni o'zgartirish) audit logga yoziladi. Log yozuvi: vaqt, foydalanuvchi ID, IP manzil, amal turi, resurs, eski qiymat (update/delete uchun) va yangi qiymat. Audit loglar alohida jadvalda saqlanadi va o'chirilishi mumkin emas (faqat arxivlash).

### 3.6. Tashqi tizimlar bilan integratsiya

Tizim bir nechta tashqi tizimlar bilan integratsiyalangan.

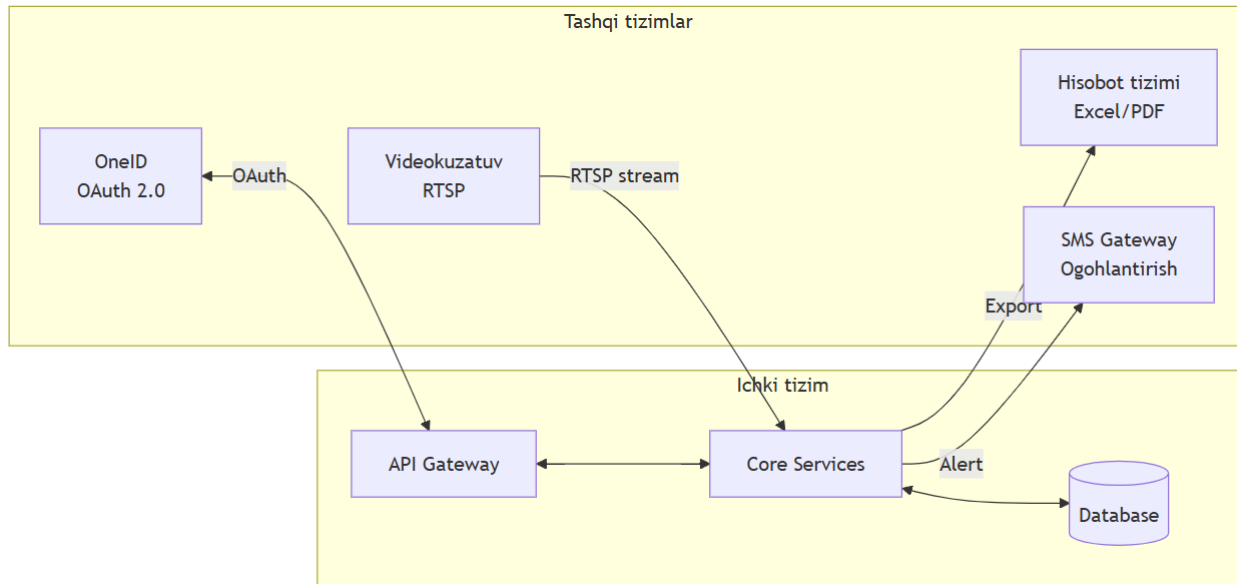
**OneID integratsiyasi.** O'zbekiston Respublikasining Yagona identifikatsiya tizimi (id.egov.uz) foydalanuvchilarni autentifikatsiya

qilish uchun ishlatilishi mumkin. Integratsiya OAuth 2.0 Authorization Code Flow asosida amalga oshiriladi. Foydalanuvchi OneID orqali tizimga kirganda, uning JShShIR si olinadi va ichki foydalanuvchi bazasi bilan solishtiriladi. Bu xodimlarni alohida ro'yxatdan o'tkazish zaruratini yo'qotadi.

### Videokuzatuv integratsiyasi.

Koloniyalardagi mavjud videokuzatuv tizimlari (CCTV) bilan integratsiya qo'llab-quvvatlanadi. RTSP protokoli orqali video oqim olinadi, kadrlar ajratiladi va face-service ga yuboriladi. Avtomatik identifikatsiya natijalari sanoq loglariga yoziladi. Bu passiv monitoring imkonini beradi – mahkum maxsus qurilma oldiga kelmasdan ham identifikatsiya qilinishi mumkin.

**Hisobot tizimi integratsiyasi.** Tizim Jazoni ijro etish departamentining mavjud hisobot tizimi bilan integratsiyalangan. Kundalik, haftalik va oylik hisobotlar avtomatik generatsiya qilinadi va belgilangan formatda (Excel, PDF) eksport qilinadi. Hisobotlar elektron pochta orqali avtomatik yuborilishi mumkin.



## 4. MUHOKAMA

Ishlab chiqilgan arxitektura bir qator afzalliklarga ega. Birinchidan, uch darajali tuzilma tarmoq uzilishlariga bardoshli – edge qurilma fog serverga ulanmasa ham lokal bazadan foydalanib ishlaydi, fog server cloud ga ulanmasa ham ma'lumotlarni buferlaydi. Ikkinchidan, mikroservis arxitekturasi har bir xizmatni mustaqil

rivojlantirish va deploy qilish imkonini beradi – face-service yangilanganda boshqa xizmatlar ta'sirlanmaydi. Uchinchidan, RBAC modeli moslashuvchan huquqlar tizimini ta'minlaydi – yangi rollar va huquqlarni qo'shish oson.

Biroq ba'zi cheklovlar ham mavjud. Birinchidan, sinxronizatsiya murakkabligi – ikki yo'nalishli sinxronizatsiya konfliktlarga olib kelishi mumkin va ularni hal qilish qo'shimcha

logika talab qiladi. Ikkinchidan, edge qurilmalardagi model yangilanishi – MobileFaceNet modeli yangilanganda barcha qurilmalarga tarqatish kerak, bu vaqt talab qiladi. Uchinchidan, biometrik ma'lumotlar maxfiyligi – shifrlash va xavfsizlik choralari qo'llansa ham, markaziy bazada ko'plab shaxslarning biometrik ma'lumotlari saqlanishi potentsial xavf hisoblanadi.

Kelgusida quyidagi yo'nalishlarda tadqiqotlar davom ettiriladi. Birinchidan, federated learning yondashuvi – embeddinglarni markazda saqlamasdan, faqat model parametrlarini sinxronlash. Ikkinchidan, blockchain asosidagi audit log – o'zgartirilmas va tekshirilishi mumkin bo'lgan audit yozuvlari. Uchinchidan, anti-spoofing mexanizmlarini qo'shish – foto yoki video bilan aldashdan himoya.

## 5. XULOSA

Ushbu maqolada manzil-koloniyalarda mahkumlarni masofaviy identifikatsiya qilish tizimining to'liq dasturiy-texnik arxitekturasini taqdim etildi. Asosiy natijalar quyidagilardan iborat.

Birinchidan, uch darajali (edge-fog-cloud) arxitektura ishlab chiqildi. Bu arxitektura tarmoq uzilishlariga bardoshli, kengaytiriluvchan va markazlashtirilgan boshqaruv imkonini beradi. Edge qurilmalar offline rejimda ishlashi mumkin, fog serverlar ma'lumotlarni buferlaydi, cloud server barcha ma'lumotlarni birlashtiradi.

Ikkinchidan, ma'lumotlar bazasi modeli ishlab chiqildi. 12 ta asosiy jadval mahkumlar, embeddinglar, sanoq loglari va boshqa ma'lumotlarni saqlaydi. pgvector kengaytmasi embedding qidiruvini optimallashtiradi. Partitsiyalash va indekslar so'rovlar tezligini oshiradi.

Uchinchidan, dasturiy modullar arxitekturasini tavsiflandi. Mikroservislar (face-service, match-service, sync-service) mustaqil rivojlanish imkonini beradi. REST API OpenAPI spesifikasiyasiga mos. Mobile ilova offline rejimni qo'llab-quvvatlaydi.

To'rtinchidan, foydalanuvchi rollari va huquqlar tizimi ishlab chiqildi. RBAC modeli moslashuvchan avtorizatsiya imkonini beradi.

To'rt asosiy rol (Administrator, Departament, Rahbar, Nazoratchi) turli huquqlar to'plamiga ega.

Beshinchidan, xavfsizlik arxitekturasini tavsiflandi. JWT autentifikatsiya, AES-256 shifrlash, audit logging va boshqa mexanizmlar ma'lumotlar xavfsizligini ta'minlaydi.

Taklif etilgan arxitektura manzil-koloniyalarda amalda qo'llanilishi uchun tayyor. Kelgusida federated learning, blockchain audit va anti-spoofing mexanizmlarini qo'shish rejalashtirilgan.

## ADABIYOTLAR

1. O'zgaruvchan muhitda shaxs yuzini tanib olishning gibrid usuli. *Jurnal nomi*, 2024. (Ushbu maqolaning 1-qismi – preprocessing va MobileFaceNet asosidagi gibrid usul)
2. O'zbekiston Respublikasi Ichki ishlar vazirligi. Jazoni ijro etish bosh boshqarmasi statistikasi. <https://iiv.uz> (2024 yil holati)
3. O'zbekiston Respublikasi. Shaxsga doir ma'lumotlar to'g'risidagi Qonun. O'RQ-547-son, 2019 yil 2 iyul.
4. Bass, L., Clements, P., & Kazman, R. (2021). *Software Architecture in Practice* (4th ed.). Addison-Wesley. (Monolit va taqsimlangan arxitekturalar asoslari)
5. Cockcroft, A., & Sheahan, D. (2011). Benchmarking Cassandra Scalability on AWS. *Netflix Tech Blog*. <https://netflixtechblog.com> (Netflix mikroservis o'tish tajribasi)
6. Newman, S. (2021). *Building Microservices: Designing Fine-Grained Systems* (2nd ed.). O'Reilly Media. (Mikroservis arxitekturasini to'liq qo'llanma)
7. Fowler, M., & Lewis, J. (2014). Microservices: A Definition of This New Architectural Term. *martinfowler.com*. <https://martinfowler.com/articles/microservices.html>
8. Vogels, W. (2006). A Word on Scalability. *All Things Distributed*. <https://www.allthingsdistributed.com> (Amazon two-pizza teams konsepsiyasi)
9. Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, 3(5), 637-646. doi:10.1109/JIOT.2016.2579198

10. Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012). Fog Computing and Its Role in the Internet of Things. *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, 13-16. ACM. doi:10.1145/2342509.2342513 (Fog computing asosiy maqola – Cisco tomonidan)
11. Yousefpour, A., Fung, C., Nguyen, T., Kadber, K., Kreml, P.N., et al. (2019). All One Needs to Know about Fog Computing and Related Edge Computing Paradigms: A Complete Survey. *Journal of Systems Architecture*, 98, 289-330. doi:10.1016/j.sysarc.2019.02.009
12. Codd, E.F. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(6), 377-387. (Relatsion model asosiy maqola)
13. Brewer, E. (2000). Towards Robust Distributed Systems. *ACM Symposium on Principles of Distributed Computing (PODC)*. (CAP teoremasi) pgvector contributors. (2023). pgvector: Open-source vector similarity search for Postgres. *GitHub repository*. <https://github.com/pgvector/pgvector>
14. Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3), 535-547. (FAISS – Facebook vektor qidirish kutubxonasi)
15. NIST. (2020). Security and Privacy Controls for Information Systems and Organizations. *NIST Special Publication 800-53*, Revision 5. (Xavfsizlik nazorati standartlari)
16. Jones, M., Bradley, J., & Sakimura, N. (2015). JSON Web Token (JWT). *RFC 7519*, Internet Engineering Task Force. <https://tools.ietf.org/html/rfc7519>
17. Sandhu, R., Coyne, E.J., Feinstein, H.L., & Youman, C.E. (1996). Role-Based Access Control Models. *IEEE Computer*, 29(2), 38-47. doi:10.1109/2.485845 (RBAC asosiy maqola)
18. Ramírez, S. (2023). FastAPI Documentation. <https://fastapi.tiangolo.com/> (FastAPI rasmiy dokumentatsiyasi)
19. ONNX Runtime contributors. (2023). ONNX Runtime: Cross-platform, high performance ML inferencing and training. <https://onnxruntime.ai/>
20. Merkel, D. (2014). Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux Journal*, 2014(239), Article 2.