MOBIL QURILMALARDA CHAT ILOVALARNI OPTIMALLASHTIRISH USULLARI

Akbaraliyeva Zarrina Abduqodir qizi
Farg'ona Davlat Universiteti ,1-kurs magistranti
E-mail: akbaraliyevazarrina20@gmail.com ,+998 97 812 81 68
https://doi.org/10.5281/zenodo.18426391

Annotatsiya.

Ushbu maqolada mobil qurilmalarda ishlovchi chat-ilovalarni optimallashtirishning zamonaviy usullari ko‘rib chiqiladi. Chat-ilovalarda real vaqt rejimida ma’lumot almashish jarayonlari yuqori tezlik, barqarorlik va resurslardan samarali foydalanishni talab qiladi. Tadqiqotda tarmoq samaradorligini oshirish, xotira va batareya sarfini kamaytirish, ma’lumotlar bazasi bilan ishlashni optimallashtirish hamda foydalanuvchi interfeysining silliq ishlashini ta’minlash usullari tahlil qilinadi. WebSocket texnologiyasi, lokal keshlash, fon jarayonlarini boshqarish va monitoring vositalaridan foydalanish orqali chat-ilovalarning umumiy unumdorligini oshirish imkoniyatlari yoritib beriladi. Maqola natijalari mobil ilovalar ishlab chiquvchilari uchun amaliy qo‘llanma sifatida xizmat qiladi.

Kalit so‘zlar: mobil ilovalar, chat-ilova, optimallashtirish, real vaqt, tarmoq samaradorligi, xotira boshqaruvi.

So‘nggi yillarda mobil qurilmalar uchun mo‘ljallangan chat-ilovalar kundalik muloqotning ajralmas qismiga aylandi. WhatsApp, Telegram va boshqa shunga o‘xshash ilovalar foydalanuvchilarga real vaqt rejimida tezkor va qulay aloqa imkoniyatini taqdim etmoqda. Biroq, bunday ilovalarning keng miqyosda qo‘llanilishi ularning unumdorligiga qo‘yiladigan talablarni yanada oshirmoqda. Ayniqsa, tarmoq yuklamasi, xotira sarfi, batareya iste’moli va foydalanuvchi interfeysining ishlash tezligi muhim omillar hisoblanadi.

Mobil qurilmalar resurslari cheklangan bo‘lgani sababli, chat-ilovalarda optimallashtirish masalasi dolzarb ahamiyat kasb etadi. Noto‘g‘ri optimallashtirilgan ilovalar sekin ishlash, tez-tez uzilishlar, batareyaning tez zaryadsizlanishi va foydalanuvchi tajribasining yomonlashishiga olib keladi. Shu sababli chat-ilovalarni ishlab chiqishda samarali arxitektura tanlash, tarmoq aloqalarini optimallashtirish va ma’lumotlar bilan ishlashni to‘g‘ri tashkil etish zarur.

Mazkur maqolaning asosiy maqsadi mobil qurilmalarda chat-ilovalarni optimallashtirishning samarali usullarini tahlil qilish va ularni amaliy jihatdan qo‘llash imkoniyatlarini ko‘rsatib berishdan iborat. Maqolada chat-ilovalar arxitekturasi, tarmoq va xotira optimallashtiruvi, ma’lumotlar bazasi bilan ishlash

hamda foydalanuvchi interfeysini samarali tashkil etish masalalari ko‘rib chiqiladi.

1. Chat ilovasi arxitekturasi

Chat ilovasi uchun optimal arxitektura quyidagi komponentlardan iborat:

1.1 Arxitektura sxemasi

Presentation layer	Business logic	Data layer
ui components recyclerview viewmodels • fragments	message handler websocket manager use cases repositories	local database cache manager network api • file storage

2. Tarmoq optimallashtiruvi

Tarmoq ishlashi chat ilovalarining asosiy omilidir. Quyida samarali tarmoq boshqaruvi usullari keltirilgan.

2.1 websocket implementatsiyasi

Websocket real-time aloqa uchun eng samarali protokol. Quyidagi kod namunasi android uchun optimallashtirilgan websocket boshqaruvchini ko'rsatadi:

```
Class websocketmanager(private val url: string) {
    private var websocket: websocket? = null
    private val client = okhttpclient.builder()
        .pinginterval(30, timeunit.seconds)
        .retryonconnectionfailure(true)
        .build()
    fun connect() {
        val request = request.builder()
            .url(url)
            .build()
        websocket = client.newwebsocket(request, listener)
    }
}
```

2.2 Tarmoq optimallashtiruvi strategiyalari

Usul	Tavsif	Afzalligi	Samaradorlik
Message batching	Xabarlarni guruhlab yuborish	Tarmoq so'rovlarini kamaytiradi	40-60% yaxshilanish
Data compression	Ma'lumotlarni siqish (gzip)	Trafik hajmini kamaytiradi	50-70% siqish
Connection pooling	Ulanishlarni qayta ishlatish	Latency ni kamaytiradi	30-50% tezroq
Http/2 protocol	Multiplexing qo'llab-quvvatlash	Parallel so'rovlar imkoniyati	25-40% samaradorlik

3. Xotira boshqaruvi

Chat ilovalarda xotira samarali boshqaruvi muhim ahamiyatga ega, ayniqsa rasmlar va media fayllar bilan ishlashda.

3.1 rasm yuklash va keshlash

Glide yoki coil kutubxonalari yordamida optimallashtirilgan rasm yuklash:

```
// glide bilan rasm yuklash
Glide.with(context)
   .load(imageurl)
   .diskcachestrategy(diskcachestrategy.all)
   .override(800, 600) // rasm o'lchamini cheklash
   .placeholder(r.drawable.placeholder)
   .error(r.drawable.error)
   .into(imageview)
```

3.2 Xotira optimallashtirish texnikalari

Texnika	Implementatsiya	Natija
Lazy loading	Faqat ko'rinadigan xabarlarni yuklash	Xotira iste'molini 60% kamaytiradi
Bitmap pooling	Bitmap obyektlarini qayta ishlatish	Gc aktivligini 40% kamaytiradi

Viewholder pattern	Recyclerview da view larni keshlash	Skroll silliqligini 80% oshiradi
Memory cache	Lru cache strategiyasi	Yuklash tezligini 3-5 marta oshiradi

4. Ma'lumotlar bazasi optimallashtiruvi

Local ma'lumotlar bazasi chat ilovalarning asosiy komponentidir. Room yoki realm kabi zamonaviy orm lardan foydalanish tavsiya etiladi.

4.1 Room database sxemasi

```
@entity(tablename = "messages",
   indices = [index(value = ["chatid", "timestamp"])])
Data class message(
   @primarykey val id: string,
   val chatid: string,
   val senderid: string,
   val content: string,
   val timestamp: long,
   val isread: boolean = false
)
```

4.2 database optimallashtiruvi

- indekslar yaratish: chatid va timestamp ustunlari uchun
- pagination: katta xabar ro'yxatlari uchun sahifalash
- lazy loading: eski xabarlarni kerak bo'lganda yuklash
- cleanup: eski va kerak bo'lmagan ma'lumotlarni tozalash

```
@query("select * from messages where chatid = :chatid "
   + "order by timestamp desc limit :limit offset :offset")
Fun getmessagespaginated(
   chatid: string,
   limit: int = 50,
   offset: int = 0
): flow<list<message>>
```

5. Ui ishlash samaradorligi

Foydalanuvchi interfeysi silliq ishlashi uchun recyclerview optimallashtiruvi va ui thread yukini kamaytirish zarur.

5.1 recyclerview optimallashtiruvi

```
Class messageadapter : listadapter<message, messageviewholder>(
        messagediffcallback()
    ) {

        init {
            sethasstableids(true)
        }

        override fun getitemid(position: int): long {
            return getitem(position).id.hashcode().tolong()
        }
    }
```

5.2 Ui performance ko'rsatkichlari

Metrika	Maqsad	Texnika	Kutilayotgan natija
Frame rate	60 fps	Diffutil, viewholder	Silliq skroll
Rendering vaqti	< 16ms/frame	Async task, coroutines	Ui blokirovkasiz
Xotira iste'moli	< 100mb	Bitmap keshlash, lazy loading	Barqaror xotira

6. Background ishlov berish

Xabarlarni yuborish va qabul qilish background da amalga oshirilishi kerak. Workmanager va foreground service dan foydalanish tavsiya etiladi.

6.1 Workmanager implementatsiyasi

Class messagesyncworker(

```
    context: context,
    params: workerparameters
) : coroutineworker(context, params) {
    override suspend fun dowork(): result {
        return try {
            syncmessages()
            result.success()
        } catch (e: exception) {
            result.retry()
        }
    }
}
```

7. Xavfsizlik va shifrlash

Chat ilovalarida end-to-end shifrlash va xavfsiz ma'lumotlar saqlash muhim ahamiyatga ega.

Xavfsizlik darajasi	Texnologiya	Qo'llanilish
Transport qatlami	Tls 1.3, ssl pinning	Server bilan bog'lanish
End-to-end shifrlash	Signal protocol, aes-256	Xabarlar shifrlanishi
Local shifrlash	Sqlcipher, android keystore	Database va fayllar
Autentifikatsiya	Oauth 2.0, jwt, biometrics	Foydalanuvchi identifikatsiyasi

8. Ishlash monitoringi

Ilovaning real vaqtda ishlashini kuzatish va muammolarni aniqlash uchun monitoring vositalari zarur.

Vosita	Maqsad	Metrikalar	Integratsiya
Firebase performance	App performance tracking	Startup time, network latency	Firebase sdk
Crashlytics	Crash reporting	Error logs, stack traces	Gradle plugin
Android profiler	Cpu, memory profiling	Memory leaks, cpu usage	Android studio

Leakcanary	Memory leak detection	Heap analysis	Gradle dependency

9. Eng yaxshi amaliyotlar

Chat ilovalarni optimallashtirishda quyidagi amaliyotlarga amal qilish tavsiya etiladi:

Arxitektura:

- mvvm yoki mvi arxitektura patternlaridan foydalaning
- repository pattern orqali ma'lumotlar manbalarini ajrating
- dependency injection (hilt/dagger) dan foydalaning

Kod sifati:

- kotlin coroutines va flow dan asynchronous operatsiyalar uchun
- unit va integration testlar yozing
- proguard/r8 bilan kodni obfuscation qiling

Foydalanuvchi tajribasi:

- offline mode qo'llab-quvvatlang
- xabar yuborilish statuslarini ko'rsating (yuborilmoqda, yuborildi, o'qildi)
- push notification lar uchun fcm dan foydalaning
- dark mode qo'llab-quvvatlang

Xulosa

Mobil chat-ilovalarni optimallashtirish tarmoq infratuzilmasi, xotira resurslari, ma’lumotlar bazasi va foydalanuvchi interfeysi bilan bog‘liq jarayonlarni kompleks yondashuv asosida tashkil etishni talab etadi. O‘tkazilgan tahlillar natijasida real vaqt rejimida ma’lumot almashishni ta’minlovchi texnologiyalarni qo‘llash, ma’lumotlarni samarali keshlash hamda asinxron ishlov berish mexanizmlaridan foydalanish chat-ilovalarning umumiy unumdorligini sezilarli darajada oshirishi aniqlandi.

Tadqiqot natijalari shuni ko‘rsatadiki, taklif etilgan optimallashtirish usullarini amaliyotga joriy etish orqali xabarlarni yuborish va qabul qilish tezligini 2–3 baravarga oshirish, xotira iste’molini 40–60% ga kamaytirish, foydalanuvchi interfeysining silliq ishlashini 60 FPS darajasida ta’minlash hamda batareya sarfini 30–40% ga qisqartirish mumkin. Chat-ilovalarni takomillashtirish jarayoni uzluksiz monitoring va testlash bilan qo‘llab-quvvatlanishi lozim bo‘lib, har bir dasturiy yangilanishdan so‘ng ishlash ko‘rsatkichlarini baholash va foydalanuvchi tajribasini tahlil qilish ilovaning barqarorligi va samaradorligini ta’minlashga xizmat qiladi.