UDK. 004.42

Buriyev J.N.
“Axborot texnologiyalari va aniq fanlar” kafedrasi o’qituvchisi
Termiz iqtisodiyot va servis universiteti
Ilmiy rahbar: Toshev Sh.E.
“Elektr ta’minoti va qayta tiklanuvchi energiya manbalari” kafedrasi dotsenti,
“TIQXMMI” milliy tadqiqot universiteti

SHAMOL QURILAMASI MOBIL ILOVASINI XAMARIN FREYMVORKI VA ANDROID STUDIO (KOTLIN) MUHITIDA YARATISH USULLARINING QIYOSIY TAHLILI

Annotatsiya: Ushbu maqolada Android platformasi uchun mobil ilova yaratishning ikki asosiy usuli — Microsoft kompaniyasining Xamarin freymvorki (C#) va Android Studio muhitida Kotlin tilidagi native dasturlash — real kod misollari asosida qiyoslanadi. Har bir usulning arxitekturasi, afzalliklari va kamchiliklari, kodni qayta ishlatish, interfeys yaratish va apparat vositalari, hamda IoT qo’llangan loyihalar uchun qaysi usul afzalligi haqida xulosa berilgan.

Kalit so‘zlar: Xamarin, .NET MAUI, Kotlin, Android Studio, mobil ilova, kross-platforma dasturlash, native dasturlash, Jetpack Compose, C#, XAML, foydalanuvchi interfeysi, unumdorlik, REST API, IoT monitoring, Retrofit, coroutine.

Buriyev J.N.
Information technology and exact sciences department
Termiz university of economics and service
scientific supervisor: Toshev Sh.E.
Power Supply and Renewable Energy Sources department, Tashkent institute of irrigation and agricultural mechanization engineers” national research university

COMPARATIVE ANALYSIS OF METHODS FOR CREATING A WIND DEVICE MOBILE APPLICATION IN THE XAMARIN FRAMEWORK AND ANDROID STUDIO (KOTLIN) ENVIRONMENT

Annotation: This article compares two principal approaches to building mobile applications for the Android platform — Microsoft's Xamarin framework (C#) and native development in Kotlin within Android Studio — using real code samples. The architecture, advantages and drawbacks of each approach are analysed in terms of performance, code reuse, user-interface construction and hardware access, and a conclusion is drawn on which method is preferable for IoT-oriented projects.

Key words: Xamarin, .NET MAUI, Kotlin, Android Studio, mobile application, cross-platform development, native development, Jetpack Compose, C#, XAML, user interface, performance, REST API, IoT monitoring, Retrofit, coroutine.

Kirish.

Mobil ilovalar bugungi raqamli iqtisodiyotning ajralmas qismiga aylangan. Android platformasi uchun ilova ishlab chiqishda dasturchi oldida ikki asosiy yondashuv turadi: native (mahalliy) dasturlash — Android Studio muhitida Kotlin tilida, hamda kross-platforma yondashuvi — Microsoftning Xamarin freymvorki orqali C# tilida. Ushbu maqolaning maqsadi ikkala usulni bir xil funksiyani bajaruvchi real kod misollarida solishtirish, ularning afzallik va kamchiliklarini ochib berish va qaysi usul qaysi vaziyatda afzalligini asoslashdan iborat. Tahlil, jumladan, muallifning past tezlikli shamol elektr qurilmalarini sun'iy intellekt va IoT yordamida real vaqtda monitoring qiluvchi mobil ilova yaratish tajribasiga tayanadi.

Xamarin freymvorki. Xamarin — Microsoft tomonidan qo'llab-quvvatlangan ochiq kodli kross-platforma freymvork bo'lib, u Mono ish vaqti muhiti (runtime) ustiga qurilgan. Dasturchi yagona C# kod bazasini yozadi va u Android, iOS hamda Windows platformalari uchun kompilyatsiya qilinadi. Interfeys XAML

belgilash tilida (Xamarin.Forms) tavsiflanadi, biznes mantiq esa platformalararo qayta ishlatiladi. Quyida eng oddiy “hoisoblagich” ilovasining interfeys va mantiq qatlamlari keltirilgan.

Xamarin: XAML interfeysi (MainPage.xaml)

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             x:Class="WindApp.MainPage">
    <StackLayout Padding="20">
        <Label x:Name="lblCount" Text="0" FontSize="40"
               HorizontalOptions="Center" />
        <Button Text="Hisoblash" Clicked="OnButtonClicked" />
    </StackLayout>
</ContentPage>
```

Xamarin: C# mantiq qatlami (MainPage.xaml.cs)

```
public partial class MainPage : ContentPage
{
    int count = 0;
    public MainPage() => InitializeComponent();
    void OnButtonClicked(object sender, EventArgs e)
    {
        count++;
        lblCount.Text = count.ToString();
    }
}
```

Muhim jihat: 2024-yil 1-maydan boshlab Microsoft Xamarinni qo’llab-quvvatlashni rasman to’xtatdi va uni o’zining vorisi — .NET MAUI (Multi-

platform App UI) bilan almashtirdi[1]. Shu sababli yangi loyihalar uchun bugungi kunda toza Xamarin emas, balki .NET MAUI tanlanishi tavsiya etiladi.

Android Studio va Kotlin muhiti. Android Studio — Google tomonidan taqdim etilgan rasmiy integratsiyalashgan ishlab chiqish muhiti bo'lib, native Android ilovalari uchun asosiy vositadir. Kotlin tili 2017-yildan rasman qo'llab-quvvatlanadi va 2019-yildan Google tomonidan birinchi darajali til ("Kotlin-first") deb e'lon qilingan. Kotlin kodi Android Runtime (ART) uchun bevosita kompilyatsiya qilinadi, bu esa eng yuqori unumdorlik va apparat resurslariga to'g'ridan-to'g'ri kirishni ta'minlaydi. Zamonaviy interfeyslar deklarativ Jetpack Compose vositasida quriladi. Quyidagi 3-listingda yuqoridagi bilan bir xil funksiyani bajaruvchi Kotlin kodi keltirilgan.

Kotlin: Jetpack Compose interfeysi

```
@Composable
fun CounterScreen() {
    var count by remember { mutableStateOf(0) }
    Column(
        modifier = Modifier.padding(20.dp),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(text = "$count", fontSize = 40.sp)
        Button(onClick = { count++ }) { Text("Hisoblash") }
    }
}
```

E'tibor bering: Compose yondashuvida holat (count) "remember" va "mutableStateOf" orqali boshqariladi va o'zgarganda interfeys avtomatik qayta

[1]Microsoft. Support for Xamarin ended May 1, 2024 — Upgrade from Xamarin to .NET MAUI. https://learn.microsoft.com/dotnet/maui/migration/

chiziladi (rekompozitsiya). Bu Xamarindagi “kod orqasida” (code-behind) qo’lda yangilashga nisbatan ixchamroq va xatolikka past moyil.

Real vaqt ma’lumotlari bilan ishlash (IoT misoli). Shamol turbinasi monitoringi ilovasida eng muhim vazifa — sensor ma’lumotlarini (podshipnik harorati, vibratsiya, kuchlanish) bulutli serverdan (masalan, Firebase) real vaqtda olib, ekranda ko’rsatishdir. Quyida ikkala muhitda REST so’rovini bajarish va JSON javobini ob’ektga aylantirish ko’rsatilgan.

Xamarin: sensor ma’lumotini olish (C#, HttpClient)

```
public async Task<SensorData> GetSensorAsync()
{
    using var client = new HttpClient();
    string url = "https://wind-iot.firebaseio.com/data.json";
    string json = await client.GetStringAsync(url);
    return JsonConvert.DeserializeObject<SensorData>(json);
}
```

Kotlin: sensor ma’lumotini olish (Retrofit + coroutine)

```
interface ApiService {
    @GET("data.json")
    suspend fun getSensor(): SensorData
}
 lifecycleScope.launch {
    val data = retrofit.create(ApiService::class.java).getSensor()
    binding.tvTemp.text = "${data.bearingTemp} °C"
}
```

Kotlindagi “suspend” funksiyalar va coroutine’lar asinxron oqimni tilning o’zida tabiiy boshqaradi, bu esa ko’p sonli sensorlardan kelayotgan uzluksiz ma’lumotlarni past kechikish bilan qayta ishlashga juda qulay. Xamarinda esa

async/await yordamida shunga o’xshash natijaga erishiladi, biroq Mono qatlami sabab sovuq ishga tushish (cold start) biroz sekinroq bo’ladi.

Afzallik va kamchiliklar.

Xamarin (.NET) afzalliklari:

– yagona C# kod bazasidan Android, iOS va Windows uchun bir vaqtda ilova ishlab chiqish;
– boy .NET ekotizimi, NuGet kutubxonalari va Visual Studio integratsiyasi;
– mavjud .NET jamoalari va korxona (enterprise) loyihalari uchun biznes mantiqni 60–90% qayta ishlatish imkoni.

Xamarin kamchiliklari:

– 2024-yil 1-maydan qo’llab-quvvatlash to’xtatilgan — xavfsizlik yangilanishlari kelmaydi;
– Mono runtime sabab ilova hajmi katta, sovuq ishga tushish sekinroq;
– yangi Android imkoniyatlari bog’lovchilar orqali kechikib yetadi, murakkab native UI va sensorlar bilan ishlashda qo’shimcha kod talab etiladi.

Kotlin / Android Studio afzalliklari:

– Google tomonidan rasman tavsiya etilgan, eng yuqori unumdorlik;
– Android SDK va apparatga (sensorlar, BLE, kamera) to’g’ridan-to’g’ri kirish;
– Jetpack Compose orqali zamonaviy deklarativ interfeys hamda coroutine’lar bilan samarali asinxron va real vaqt ma’lumotlarini boshqarish.

Kotlin / Android Studio kamchiliklari:

– faqat Android — iOS uchun alohida kod (Swift) yoki Kotlin Multiplatform talab etiladi;
– .NET dasturchilari uchun boshlang’ich o’rganish bosqichi ko’proq vaqt oladi;

– kross-platforma loyihalarda kodni qayta ishlatish darajasi past.

Xulosa

O’tkazilgan qiyosiy tahlil shuni ko’rsatadiki, har bir usulning o’z qo’llanilish sohasi mavjud. Agar loyiha bir nechta platforma (Android, iOS, Windows) uchun mo’ljallangan bo’lsa va jamoa C#/.NET bilan ishlasa — kross-platforma yondashuvi (Xamarin, aniqrog’i uning vorisi .NET MAUI[2]) maqsadga muvofiqdir. Aksincha, maqsad faqat Android uchun yuqori unumdorlikka ega, apparat va sensorlar bilan chuqur ishlaydigan ilova bo’lsa, native yondashuv — Kotlin + Android Studio afzal hisoblanadi.

Shuni alohida ta’kidlash kerakki, toza Xamarin endi qo’llab-quvvatlanmaydi, shu bois yangi loyihalarda undan emas, balki .NET MAUI yoki native Kotlindan foydalanish lozim. Mazkur tadqiqotda ko’rib chiqilgan past tezlikli shamol turbinasini real vaqtda monitoring qiluvchi, ko’p sonli sensor va apparat bilan ishlovchi ilova uchun Kotlin + Android Studio yondashuvi afzal deb topildi: u eng so’nggi Android API’laridan to’g’ridan-to’g’ri foydalanish, sensor ma’lumotlarini past kechikish bilan qayta ishlash va resurslarni tejash imkonini beradi.

Foydalanilgan adabiyotlar.

1. JetBrains. *Kotlin Programming Language Documentation*. https://kotlinlang.org/docs/
2. Google Developers. *Android Developers Guide & Jetpack Compose*. https://developer.android.com/
3. Microsoft. *Mobile development with Xamarin; Upgrade from Xamarin to .NET MAUI*. https://learn.microsoft.com/dotnet/maui/
4. Hermes, D. (2015). *Xamarin Mobile Application Development*. Apress.

[2]Buriyev J. N. (2024). Windows Forms .NET ilovalarning dizaynida UI/UX freymvorklarining afzalliklari. Ekonomika va Sotsium jurnali.

5. Petzold, C. (2016). *Creating Mobile Apps with Xamarin.Forms*. Microsoft Press.
6. Leiva, A. (2019). *Kotlin for Android Developers*. Leanpub.
7. Späth, P. (2018). *Pro Android with Kotlin*. Apress.
8. Phillips, B., Stewart, C., & Marsicano, K. (2019). *Android Programming: The Big Nerd Ranch Guide* (4th ed.). Big Nerd Ranch.
9. Buriyev J. N. (2024). Windows Forms .NET ilovalarning dizaynida UI/UX freymvorklarining afzalliklari. *Ekonomika va Sotsium jurnali*.
10. Kholyorov E., Turayev D., Buriyev J. (2024). Numerical solution of boundary inverse problem for fluid relaxation filtration in porous media. *AIP Conference Proceedings*. https://doi.org/10.1063/5.0241626