Технічні науки

Сергеєв Єгор Ігорович
студент,
Національний технічний університет України
«Київський політехнічний інститут»
Сергеев Егор Игоревич
студент,
Национальный технический университет Украины
«Киевский политехнический институт»
Serheiev Y.
student,
National Technical University of Ukraine
«Kyiv Polytechnic Institute»

РОЗРОБКА КРОС-ПЛАТФОРМНИХ ДОДАТКІВ З ВИКОРИСТАННЯМ ФРЕЙМВОРКА XAMARIN
РАЗРАБОТКА КРОССПЛАТФОРМНЫХ ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ ФРЕЙМВОРКА XAMARIN
DEVELOPING CROSS-PLATFORM SOFTWARE USING XAMARIN FRAMEWORK

Анотація: Досліджено особливості та підходи розробки крос-платформних додатків з використанням фреймворка Xamarin.

Ключові слова: крос-платформність, мобільний, додаток, iOS, Android.

Аннотация: Исследованы особенности и подходы разработки кроссплатформенных приложений с использованием фреймворка Xamarin.

Ключевые слова: кроссплатформенность, мобильный, приложение, iOS, Android.

Summary: Were investigated features and approaches of developing cross-platform mobile applications.

Keywords: cross-platform, mobile, application, iOS, Android.

Вступ

Коли постає питання, яким чином будувати iOS чи Android додатки, багато людей, в першу чергу, згадують такі мови програмування як Swift, Java, або Objective-C. Проте в цій роботі буде розглянуто зовсім іншу технологію.

Xamarin – це унікальний фреймворк який дозволяє будувати такі додатки з використанням однієї мови програмування – C#. З використанням данного фреймворка ми компілюємо (не інтерпретуємо, як, наприклад, при використанні аналогічних JavaScript фреймворків) так званий “нативний” код для цих операційних систем(ОС).

Кожна з цих ОС має свій власний набір можливостей і кожна різниться в здатності писати нативні додатки котрі компілюються в машинний код. Наприклад, ОС Android дозволяє писати код на Java, Windows Phone (WP) – з використанням JavaScript та C#, iOS – Objective-C та Swift. Звичайно є й інші фреймворки, які дозволяють писати додатки на не нативних для платформи мовах програмування, наприклад Corona SDK, PhoneGap та інші, проте при використанні таких фреймворків швидкодія роботи додатку значно знижується, ніж в тих додатках, що використовують нативні засоби розробки.

Основні підходи розробки з використанням Xamarin

З найпершого релізу фреймворка Xamarin в ньому було доступно такі два блоки – Xamarin.iOS та Xamarin.Android – ці частини фреймворка дозволяють розробляти додатки під відповідні платформи з використанням єдиної мови програмування, що, звичайно сильно полегшує процес розробки програмного продукту, та дозволяє використовувати спільну бізнес логіку додатку та, наприклад, логіку доступу до SQLite бази даних, що встановлено

на мобільному пристрої чи планшеті. Також використання єдиної логіки додатку веде за собою скорочення часу на розробку, що є гарною перевагою в порівнянні з використанням різних мов програмування, проте є й недоліки, основним з яких є складність для розробника. Розробнику потрібно гарно знати платформи Android та iOS щоб писати дійсно складні та функціональні додатки, тобто потрібно витратити дуже багато часу на те, щоб почати створювати конкурентоспроможні програмні продукти. Xamarin.Android та Xamarin.iOS є частиною підходу розробки, що називається Xamarin Native, тобто написання чистого нативного додатку з використанням C#.

Однак, з плином часу та вдосконаленням інфраструктури Xamarin Native розробниками з компанії Xamarin було вирішено створити ще одну частину свого фреймворка – Xamarin.Forms. Даний підхід є набагато простішим для розробника, оскільки він полегшує роботу з різними операційними системами. Звісно, розробнику і досі потрібно писати певний платформо-залежний код, проте з використанням даного підходу даного коду стане набагато менше, оскільки в Xamarin.Forms використовується XAML (Extensible application markup language) для побудови графічного інтерфейсу, що дозволяє винести увесь код пов’язаний з графічним дизайном у спільну бібліотеку, тобто розробник тепер не повинен знати специфіку інтерфейсу для кожної платформи, достатньо лише знати XAML та пов’язані з ним класи, даний підхід, тобто отримуємо зв’язку C# та XAML, що схожа на JavaScript та CSS.

Xamarin.Forms повністю “покриває” усі три платформи та дозволяє розробнику зосередитися на бізнес логіці додатку та його структурі і економити час розробки. Однак, не усе так гарно як могло б здаватися, звичайно, такий додаток буде крос-платформним, однак даний додаток матиме багато обмежень з точки зору функціональної частини додатку, оскільки неможливо об’єднати усі можливості платформ і використовувати як одну спільну, адже вони реалізовані по-різному та й деякі з них не наявні у

інших платформах. Оскільки тема Xamarin.Forms досить обширна, розглянемо основні недоліки та переваги цієї технології:

1. Дуже схожа на стандартні Windows технології;
2. Спільна логіка для всіх додатків;
3. Не потрібно вглиблюватись в можливості конкретної платформи
4. Нижча швидкодія, порівняно з Xamarin.iOS, Xamarin.Android;
5. Неповна реалізація усіх можливостей платформ;
6. Компромісні рішення для реалізації функціоналу, що відрізняється на різних платформах;
7. Можлива різна поведінка додатку для платформ.

Тобто, точних рекомендацій для розробника, який з даних підходів використовувати – Xamarin Native чи Xamarin.Forms немає, усе залежить від бажання та можливостей розробника, масштабів проекту. Є лише декілька критеріїв селекції одного з цих підходів – час розробки та можливості додатку. Якщо програмний продукт повинен мати широкий функціонал та використовувати усі можливості програмного інтерфейсу (api) платформи, то гарним вибором, звичайно буде Native підхід, якщо ж основним критерієм є час розробки, то єдиним логічним вибором є Xamarin.Forms.

Технологія Xamarin.Android

Xamarin.Android додатки виконуються в СВ Mono. Mono працює пліч-о-пліч з віртуальною машиною Android Runtime (ART). Обидва СВ працюють над ядром Linux і надають різні інтерфейси API для коду програміста, що дозволяє розробникам отримати доступ до даної системи.

Розробникам надано простори імен System, System.IO, System.Net та інші бібліотеки класів .NET, щоб отримати доступ до основних можливостей операційної системи Linux [2].

На Android, більшість системних можливостей, таких як аудіо, графіки, OpenGL і телефонії не доступні безпосередньо до нативних додатків, вони доступні тільки через Android Java Runtime API, що були реалізовані в просторі імен Java. * або Android. *[2].

В самому центрі роботи Xamarin.Android розташоване ядро Linux, над яким знаходяться два середовища виконання – Mono та ART, що взаємодіють між собою за допомогою Android Callable Wrappers(ACW). Вже над даними середовищами виконання розташовані .NET API та спеціальні прив'язки до Android.* та Java.* можливостей, що взаємодіють з використанням Managed Callable Wrappers(MCW) [2].

Android Callable Wrappers – спеціальний Java-Native-Interface(JNI) "міст", що використовується ОС Android для виконання коду.

Manager Callable Wrappers – спеціальні обгортки типу JNI для того, щоб виконувати Android код, що був створений з використанням ACW. Кожна така обгортка зберігає у собі глобальне Java посилання(reference) на об'єкт. Кількість таких об'єктів є лімітованою. Загалом доступно 52.000 таких посилань під час роботи додатку.

Так як Xamarin додатки компілюються в нативні додатки, то програмістам доступні усі контроли (controls), що використовуються при написанні додатків з використанням Java – Popup Menus, Views, Date Pickers і т.д.

Також такі додатки можуть використовувати усі можливості API Android – Android Beam, камера, аутентифікація з використанням відбитку пальця, карти, локація, Android.Speech та інші.

Кожен додаток Android працює з використанням спеціальних об'єктів, названих activity. Наприклад, кожна сторінка додатку на Android і є activity, а кожне таке activity має свій цикл існування(lifecycle) [1]. Xamarin.Android також підтримує цю концепцію, дозволяючи описувати Activity з використанням C#. Усі activity в Xamarin.Android обрамляються спеціальними атрибутами, що дозволяють додавати дані activity до спеціального Android маніфеста, де описуються та додаються основні компоненти додатку. Також Xamarin.Android додатки надають такі можливості:

1. Створення спеціальних провайдерів контенту та резолверів контенту, що надають можливість отримувати та надавати дані, а також шукати дані з для інших додатків у системі;
2. Створення сервісів, що мають свій життєвий цикл та дозволяють виконання деяких складних задач в іншому потоці, наприклад, скачати статтю з певного сайту. Дані сервіси бувають трьох типів, а також можуть використовуватися іншими додатками у системі, створювати повідомлення для користувача та інше;
3. Використання фрагментів замість декількох activity для створення більш швидкого додатку, оскільки не потрібно підгружати різні activity у відповідь користувачу;
4. Xamarin.Android підтримує усі API рівні ОС Android, отже розробникам завжди доступні усі можливості даної платформи;
5. Як і в звичайному додатку під Android, Xamarin.Android надає змогу створювати ресурси – зображення, елементи інтерфейсу, локалізація, аудіо та відео, файлова система та інше.

Отже, Xamarin.Android наділена багатим функціоналом для розробки додатків під ОС Android, що не гірші від Java аналогів, дозволяє використовувати усе, що надає платформа, а додатки по швидкодії не програють аналогам.

Технологія Xamarin.iOS

Технологія Xamarin.iOS відрізняється від Xamarin.Android, в першу чергу це пов’язано з кардинальними відмінностями даних платформ. Навідміну від Android, iOS не дозволяє використовувати середовища виконання, тому використовувати Mono не вийде. Xamarin.iOS не використовує Mono для додатку. Для iOS використовуються спеціальні прив’язки, для нативних компонентів та можливостей даної ОС. Однак, неможливість використання середовища виконання – не проблема, оскільки навідміну від використання JIT(Just-In-Time) компіляції, додаток, що

написаний на C# під iOS компілюється одразу в байт код і для даного додатку уже не потрібно використання будь-якого середовища виконання.

Як і для Xamarin.Android, Xamarin.iOS дозволяє використання усіх нативних контролів та можливостей платформи та пристрою, на якому виконується програма.

Основними робочими блоками в Xamarin.iOS є event, delegate та protocol. Event викликається під час взаємодії користувача з пристроєм, такі event можна зв'язати із кнопкою чи іншим елементом управління (UIKit) для створення певної поведінки додатку та реагування на дії користувача, ви можете прив'язувати до елементів багато подій (events) та присвоювати їм різну поведінку. Protocol – щось на зразок інтерфейсу (interface) в C#, що дозволяє "дізнатися" ОС який метод викликати, не знаючи його поведінку, а delegate використовуються як відповідь (callback) на якусь дію.

Технологія Xamarin.Forms

Xamarin.Forms – це зовсім несхожа технологія на дві інші, що були описані раніше. Дана технологія більше схожа на JavaScript фреймворки, які було описано в першому розділі. Різниця в тому, що Xamarin.Forms дозволяє писати нативні додатки з більшою кількістю спільного коду. Наприклад, інтерфейс програми буде спільний для усіх додатків(по структурі та по написанню, проте матиме дещо різний вигляд на платформах) оскільки для побудови інтерфейсу користувача використовується спеціальна мова розмітки – XAML. Тобто розробнику не потрібно знати особливості кожної платформи, розробники із Xamarin все зробили за вас [3].

Xamarin.Forms надає розробнику абстракції графічного інтерфейсу, що використовуються для кожної платформи. Дана технологія використовує нативні компоненти під час роботи додатку, тобто код графічного інтерфейсу на різній платформі має різне підґрунтя, а саме його буде перетворено на нативний код окремої конкретної платформи. Код Xamarin.Forms має можливість взаємодіяти з операційною системою, використовувати її програмний інтерфейс.

Основними типами компонентами, що використовуються в розробці з використанням Xamarin.Forms є:

1. Представлення

- Представлення – це основний блок графічного інтерфейсу, бо саме з цих елементів його побудовано. Прикладом представлення є кнопка, поле вводу і т.д.

2. Макет

- Макети визначають, як представлення будуть розташовані на екрані, їх розмір при різних розмірах екрану, поведінку представлень в певних ситуаціях;

3. Сторінка

- Сторінка має дві основні функції. Перша - сторінка представляє собою контейнер для представлень та макетів, тобто сторінка і є екраном телефону, також сторінка має і іншу функцію – навігаційну. Саме сторінка дозволяє перехід від однієї сторінки до іншої.

Основними блоками, що дозволяють писати логіку додатку є клас Device, центр обміну повідомленнями(message center) та сервіс залежностей (dependency service). Клас Device надає методи для використання платформо-специфічних можливостей додатку та побудови поведінки додатку з точки зору ОС. Центр обміну повідомленнями надає можливість використовувати методи publish/subscribe – тобто підписуватися на якусь подію, що сталася в ОС, чи, наприклад, зіставити метод, що буде спрацьовувати при натисканні кнопки [3].

Висновки

Було досліджено можливості фреймворка Xamarin, розглянуто основні підходи розробки додатків з використанням Xamarin. Можна зробити висновок, що Xamarin – перспективний фреймворк, використання якого дає багато переваг в циклі створення мобільних додатків, скорочуючи розробку та зменшуючи кількість коду за рахунок спільної бізнес логіки та методів

доступу до даних, а з використанням Xamarin.Forms дає й можливість розробляти спільний інтерфейс користувача для додатків. Хоча підхід, що використовується в Xamarin має й свої недоліки, основним з яких є те, що програмісту потрібно мати гарні знання одразу трьох різних ОС, що мають мало спільного, тобто розробник повинен мати високу кваліфікацію, хоча цей недолік зникає з роками досвіду.

Література

1. Jonathan Peppers. Xamarin Cross-platform Application Development Second Edition / Peppers J. – Birmingham : Packt, 2015. – 459 с.
2. Nilanchala Panigrahy. Xamarin Mobile Application Development for Android Second Edition / Panigrahy N. – Birmingham : Packt, 2015. – 296 с.
3. Charles Petzold. Creating Mobile Apps with Xamarin.Forms First Edition / Petzold C. – Redmond : Microsoft Press, 2016. – 1187 с.