CROSS-PLATFORM MOBILE APPLICATIONS DEVELOPMENT WITH CLOUD BACKEND

COSMIN TOMOZEI[1], IULIAN FURDU[2]
cosmin.tomozei@ub.ro
[1] *Université «Vasile Alecsandri» de Bacău, Calea Mărăşeşti 156, Bacău, 600115, Roumanie*
[2] *Université «Vasile Alecsandri» de Bacău, Calea Mărăşeşti 156, Bacău, 600115, Roumanie*

Abstract: The aim of this paper is to present the Android mobile application development process, by using Microsoft technologies, such as Xamarin platform and the corresponding Cloud infrastructure. Certain teste are to be realised tests will be built in the C# programming language and AXML mark-up. We will then evaluate the most effective way to write code for mobile devices and continue building a reliable infrastructure by the integration of mobile and Cloud technologies and the use of multimedia.

Keywords: XAMARIN, Android, C#, AXML, multimedia, sensors.

1. INTRODUCTION

Suite à l'évolution logicielle, le processus de développement des applications informatiques est orienté vers la croissance de la mobilité et de l'interopérabilité. Le fait que les applications soient conçues pour fonctionner sur plusieurs plates-formes et systèmes d'exploitation entraîne un niveau accru de productivité et de réutilisation du code. Dans le même temps, le processus de développement de logiciel est devenu plus complexe en raison des divers types de périphériques dotés de formes, de caractéristiques de construction et de capacités matérielles multiples.

Un autre aspect du développement d'applications mobiles est représenté par le traitement des données fournies par la technologie multimédia, par les capteurs (*sensors*) et par la connectivité avec les bases des données. Nous pensons que l'adoption du *back-end* en Cloud pourrait considérablement soutenir la gestion des données et améliorer la connectivité et la communication.

Dans [1], le multimédia est considéré comme un excellent outil pour attirer les utilisateurs vers les applications mobiles et les jeux interactifs, le son et l'animation jouant un rôle important dans ce contexte. La mise en œuvre de fonctionnalités photo, son et vidéo au moyen de caméras et microphones peut également être considérée comme une amélioration de l'interactivité et de l'expérience de l'utilisateur.

Les applications de caméra sont activées via des intentions spécifiques, envoyées en tant que paramètres dans la méthode *onActivityResult()*. Les images sont ensuite enregistrées dans la mémoire persistante, tandis que le nom de fichier est fourni par l'application [2]. Il est préférable que les images soient stockées dans la mémoire externe, mais la sauvegarde sur le cloud en garantit l'intégrité et les possibilités de partage ultérieur avec d'autres utilisateurs. Un moyen d'ajouter des fonctionnalités de cloud aux applications mobiles est présenté dans [2], où sont discutées les améliorations de la sécurité et la réduction des paquets d'installation (APK).

2. APPLICATIONS XAMARIN ANDROID MULTIMEDIA

Dans cette section, nous présentons une séquence logicielle développée au cours du laboratoire d'applications mobiles et implémentant des fonctions multimédias, telles que la capture d'images, de mouvements et de son, ainsi que le stockage sur mémoire externe. Les tests ont été effectués sur les émulateurs et les périphériques physiques. Les classes et APIs qui ont été utilisés pour la construction de l'application de laboratoire sont présentés dans [4]. Les tests ont été réalisés sur Android Nougat7.1API niveau 25 Nougat 7.0 Api niveau 24 API Marshmallow 6.0 niveau 23 [5]. Les packages utilisés pour travailler avec les images et le stockage externe sont

Android.Hardware, Android.Media, Android.Content et Java.IO. Aucune différence majeure n'a été constatée pendant les tests.

La séquence suivante montre les attributs de la classe *MainActivity*. Ces champs sont pratiquement des objets correspondant au multimédia. Les activités Android sont constituées de classes qui permettent aux utilisateurs d'ajouter des contrôles et des événements. Les activités apparaissent sous forme de formulaires ou d'interfaces utilisateur.

Tableau 1. Atributs et marquage pour l'application Android Multimedia

Atributs de la classe MainActivity	**Le marquage AXML**

namespace ADM_TEST_7_2019
{
[Activity(Label = "ADM_TEST_7_2019",
MainLauncher = true)]
public class MainActivity : Activity
{
public static MediaRecorder recorder = null;
public static MediaPlayer player = null;
public static File director;
public static string StockageDeFichier;
VideoView videoView1;
private Android.Hardware.Camera camera;
	<VideoView
	android:layout_width="fill_parent"
	android:layout_height="340dp"
	android:id="@+id/videoView1" />
	<Button
	android:text="Record"
	android:layout_width="match_parent"
	android:layout_height="wrap_content"
	android:id="@+id/buttonRecord" />
	<Button
	android:text="Stop"
	android:layout_width="match_parent"
	android:layout_height="wrap_content"
	android:id="@+id/buttonStop" />

Dans le tableau 1, nous pouvons voir le marquage AXML pour la description de l'interface utilisateur de l'application mobile. Nous avons une commande vidéo qui montre la capture d'image de la caméra et deux boutons, un pour démarrer le processus d'enregistrement et un pour arrêter l'enregistrement. La largeur de la vidéo est héritée du contrôle parent, par ex. la taille de l'écran. La hauteur est exprimée en dp, plus précisément 340dp, signifiant pixels indépendants de la densité. Par conséquent, le système met automatiquement à l'échelle les unités dp, en fonction de la densité réelle de l'écran utilisé.

Tableau 2. Variables et méthode multimédia Xamarin Android

Variables pour les contrôles	var buttonRecord = FindViewById<Button>(Resource.Id.buttonRecord);
	var buttonStop = FindViewById<Button>(Resource.Id.buttonStop);
	videoView1 = FindViewById<VideoView>(Resource.Id.videoView1);
	buttonRecord.Click += buttonRecord_Click;
	buttonStop.Click += buttonStop_Click;

Méthode de capture de média en C#	private void buttonRecord_Click(object sender, System.EventArgs e)
	{ videoView1.StopPlayback();
	CreateDirectoryForVideos();
	camera = GetCameraInstance();
	camera.SetDisplayOrientation(90);
	camera.Unlock();
	recorder = new MediaRecorder();
	recorder.SetCamera(camera);
	recorder.SetVideoSource(VideoSource.Camera);
	recorder.SetAudioSource(AudioSource.Mic);
	recorder.SetOutputFormat(OutputFormat.Mpeg4);
	recorder.SetVideoEncoder(VideoEncoder.H264);
	recorder.SetAudioEncoder(AudioEncoder.Default);
	recorder.SetOrientationHint(90);
	StockageDeFichier = director.AbsolutePath + "/Film" +
	Guid.NewGuid() + ".mp4";
	recorder.SetOutputFile(StockageDeFichier);
	recorder.SetPreviewDisplay(videoView1.Holder.Surface);
	recorder.Prepare();
	try
	{
	recorder.Start();
	}
	catch (Exception ex)
	{Toast.MakeText(this.ApplicationContext, ex.Message + " " +
	ex.Source, ToastLength.Long).Show();

	} }

Le tableau 2 présente les aspects suivants concernant les éléments de programmation des applications multimédia:

- Les variables définies par le type "var" associées au marquage déclaratif AXML. Pour chaque contrôle identifié dans le balisage AXML avec un id, une variable de programmation est associée. Ceci est réalisé en utilisant la méthode *FindViewById<contrôle_type>*, dans l'activité principale. De plus, les gestionnaires de clic sur les boutons sont associés aux contrôles et aux événements via l'opérateur + = et les délégués de méthodes.
- La méthode `buttonRecord_Click(object sender, System.EventArgs e).` La méthode *buttonRecord_Click* permettant d'identifier l'objet caméra, l'objet enregistreur multimédia, les sources vidéo et audio, le format de sortie - MP4 et le standard de codage vidéo H.264 [6]. Cette norme fournit une bonne qualité d'images de vidéo compressée.

Le démarrage de la capture d'image est géré dans un bloc de gestion des exceptions *try-catch*. L'arrangement de la capture d'image est géré dans un bloc de gestion des exceptions *try-catch* dans lequel les actions entreprises par les événements caméra non présente ou accès refusé au camera sont traités [7] avec élégance dans l'application.

3. CONCLUSIONS

Cet article a analysé le processus de développement d'applications multimédias pour les appareils mobiles. La plate-forme Xamarin pour Android a été utilisée pour la création de l'application. Nous avons présenté l'importance de l'image, de la vidéo et du son pour l'interaction et l'expérience de l'utilisateur. Des aspects pratiques ont également été décrits à la fois du côté déclaratif et du côté programmation. Les objets et les actions les plus importants tels que les éléments vidéo et sonores, les objets caméra et le traitement des exceptions ont également été abordés. Nous pensons que cet article pourrait être utile pour les spécialistes en génie logiciel intéressés par le développement d'applications multimédia mobiles en Xamarin et en C #.

RÉFÉRENCES

[1] Leibowitz M., Xamarin MobileDevelopment for Android Cookbook, Packt Publishing, 2015
[2] Smith D, Hellman E. A Problem-Solution Approach Fifth Edition, Apress, 2016
[3] https://blog.xamarin.com/xamarin-plus-azure-blob-cloud-storage/
[4] https://developer.xamarin.com/samples/android/Media/
[5] https://source.android.com/setup/start/build-numbers
[6] https://developer.android.com/training/camera/photobasics
[7] https://stackoverflow.com/questions/36552921/how-to-properly-error-catch-android-camera-activity-when-using-the-action-image