NORTHWEST NAZARENE UNIVERSITY

Assisting Frog Identification in Costa Rica Using a Mobile App

THESIS
Submitted to the Department of Mathematics and Computer Science
in partial fulfillment of the requirements
for the degree of
BACHELOR OF ARTS

Justin Tyler Laplante
2021

THESIS
Submitted to the Department of Mathematics and Computer Science
in partial fulfillment of the requirements
for the degree of
BACHELOR OF ARTS

Justin Tyler Laplante
2021

Assisting Frog Identification in Costa Rica Using a Mobile App

Author: ______________________________
Justin Tyler Laplante

Approved: ______________________________
Dale Hamilton, Ph.D., Professor, Department of Mathematics and Computer Science, Faculty Advisor

Approved: ______________________________
John Cossel Jr., Ph.D., Professor, Chair, Department of Biology
Second Reader

Approved: ______________________________
Barry L. Myers, Ph.D., Chair, Department of Mathematics & Computer Science

ABSTRACT

Assisting Frog Identification in Costa Rica Using a Mobile App.
LAPLANTE, JUSTIN (Department of Mathematics and Computer Science).

Quickly identifying a single frog species from over a hundred other possible species can be a challenge for research while in the Costa Rican jungle. Though researchers can use field guides to assist in, these still mean you may have look through all currently identified frog species to find the frog being looked at. This project was created to help researchers narrow the list of possible frog species ~~down~~ quickly based on Geolocation. Using Xamarin.Forms an app was developed that worked offline, used an ArcGIS API and was cross platform. However, to ensure performs and accuracy so certain design choice were made for designing the ArcGIS map that was used within the app. The used geospatial data for the frog species and generalized it into a hexagonal pattern. For the app to handle the frog and app data using xml and the MVVM Architectural pattern.

Acknowledgement

I would like to thank my parents for giving me the wonderful opportunity of going to college. I would like to thank Dr. Myers, Dr. Hamilton, and Prof. McCarty for being excellent professors in all the computer science classes and teaching me more than I know what to do with. I would like to thank Dr. Cossel for giving me a senior project that I really enjoyed working on and showed me how much I have yet to learn. Finally, I want to thank my first three years of excellent roommates in college, I never knew how lucky I was to have each one of them until senior year when I didn’t have them.

Table of Contents

Table of Figures:

Overview:

Project Overview:

This project used Xamarin forms, a .NET framework extension, to build a cross-platform friendly mobile app that can be used on android and IOS to assist researchers in Costa Rica in identifying frogs quicker using an interactive ArcGIS map within the application.

Application Requirements:

When designing and coming up with the applications, it needed to accomplish four main things:

1. The application needed to narrow down a list of frogs using the application device's built-in geolocation against an ArcGIS map.
2. The user needed to be able to look at and view each frog both in the narrower list and in general to enable the user to visually see what the selected frog species looks like and receive written information on the frog.
3. It needed to work on android and IOS with usability on tablet and phone screen size in mind.
4. The application also needs to be able to be used offline and in the field.

Design Choices:

Designing Application:

At the beginning of the project, a straightforward design was ideal. The design would allow the user to see the ArcGIS map displayed over Costa Rica. Then let the user click a button to show them all the frogs within the area. Ending with them clicking on the frog, they believe it is to see more information on that frog. To storyboard the application and run the design past Dr. Cossel, Adobe XD was used. XD allowed for quick prototyping of the application without needing to code anything. Giving a clear design path to build towards for the application.

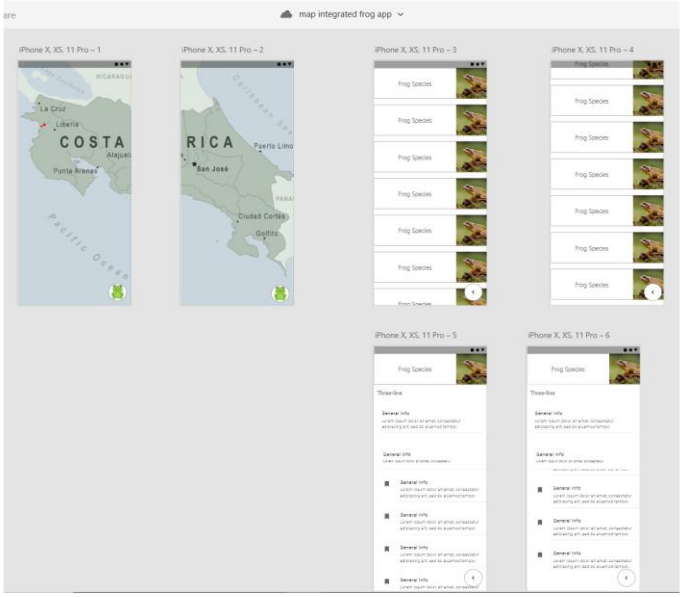


Figure 1. Adobe XD storyboard

Application Development Framework:

When deciding what framework to develop the application in, there were two desired features. First, that it would allow for cross-platform development. Second, that it could effectively use and interact with ArcGIS maps. When choosing between the different cross-platform frameworks, most are based on JavaScript, such as React Native, NativeScript or PhoneGap. JavaScript is problematic when working with ArcGIS because the ArcGIS SDK for JavaScript cannot take ArcGIS maps offline in Mobile Map Packages (MMPK), which is the only way to use them offline. So Xamarin ended up being the framework used for development because it’s built on the .NET framework. On top of developing cross-platform applications, it also allowed the use of ArcGIS SDK for .NET, which enables applications to use MMPK.

Application Data Design Pattern:

With the framework chosen, it was time to develop the application. A major decision that had to be made for this step was how to handle data movement behind the scenes between pages. There were a few different patterns that were considered, Model-View-Controller (MVC), Model-View-Presenter (MVP), Model-View-ViewModel (MVVM). All of these have three key similarities:

1. The “Model” in each of these primarily refers to where data for the application is held.
2. The “View” in each refers broadly to the UI where the user will see data held in the model.
3. Though the name changes the C, P, and VM, all represent how the view and model interact in one way or another. But This interaction is where these models differ.

For MVC, when an event occurs, it goes through the control, and if needed, the control will update the data within the model. The model then tells the view when its data changes, and the view gets that new data and updates the UI. However, this means the business logic and the UI intermingle, so to test the application, all features need to be working before debugging can occur. So, for work to be done on UI in future work, the developer would need to have a good grasp on how all the behind-the-scenes logic works, making it harder for future people to work on it and update the application (Hanmer, 2013; Maxwell, 2017).

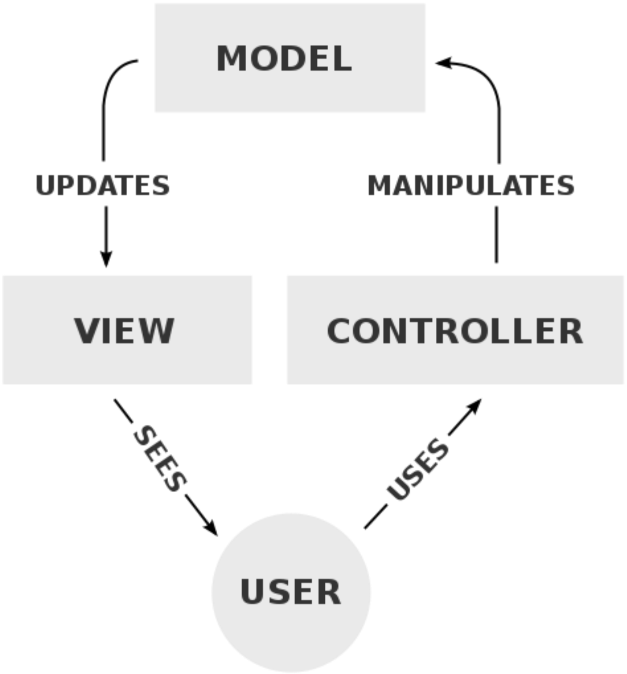


Figure 2. MVC Process (Frey, 2010)

For MVP, the view never interacts directly with the model. Instead, the view lets the presenter know when the user acts, then the presenter either tells the view how to change or tells the model what data needs changing, and then the model informs the up the chain what data was changed and the UI updates. The most important point about this model is that the view and the presenter are a one-to-one relationship meaning they rely on and interact with each other. This one-to-one relationship is a problem because it still means that the UI is tied to the business logic, and any work done on it would require knowledge outside of designing the UI (Maxwell, 2017; Mishra, 2020).

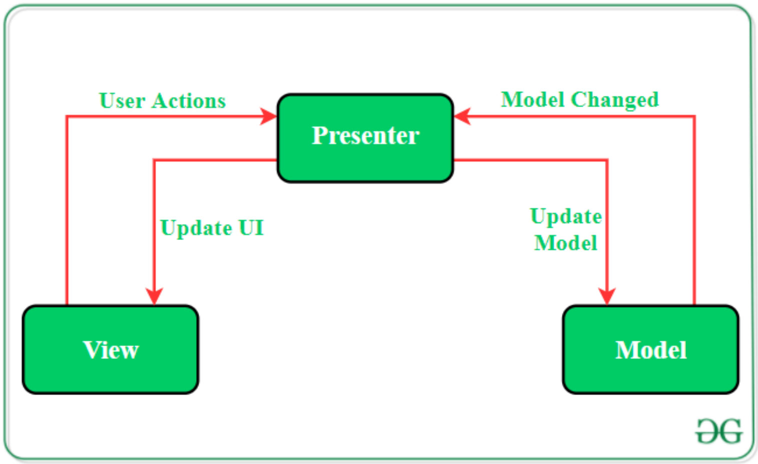


Figure 3. MVP Process (Mishra, 2020)

Finally, for MVVM, the view only contains logic related to the UI, such as animation. The view only gets data from the model through the ViewModel, similar to the presenter in MVP. However, the difference lies in that all business logic and data from the model stay in the ViewModel connected to the UI through data binding, essentially just referencing the data and events and linking it to the UI. (picture of code using data binding). This data binding is important because it enables the view to be disconnected from the ViewModel and model and manipulated and tweaked. So long as the name used in the binding is a public variable within the ViewModel, they can functionally separate. This separation means that the ViewModel can run entirely by itself and be tested without the need for a view (Britch et al., 2017; Maxwell, 2017).

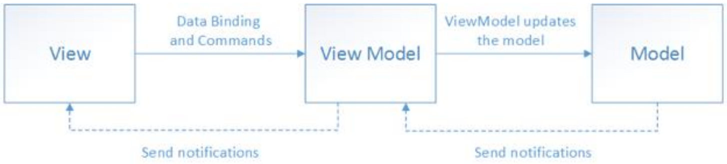


Figure 4. MVVM Process (Britch et al., 2017)

Since the MVVM pattern gave the application the ability to be worked on with both the logic and the UI separately for future work, the application was designed with the MVVM pattern in mind.

Designing a custom ArcGIS Map:

For this application, the user needed to be able to click on an ArcGIS map and see what frogs were in that area or use their GPS Coordinates to see frogs within their area. To do this and simplify the amount of loading and processing the device with the application would need to do, a custom map was designed for the application. To accomplish this, a hexagon overlay that used 10km hexagons, each one containing all frog species within that area, over all of Costa Rica. Using a hexagon pattern instead of the true regions meant the device would not need to handle millions of points that make up the edge of those shapes. The 10km size optimized the accuracy of possible frogs.

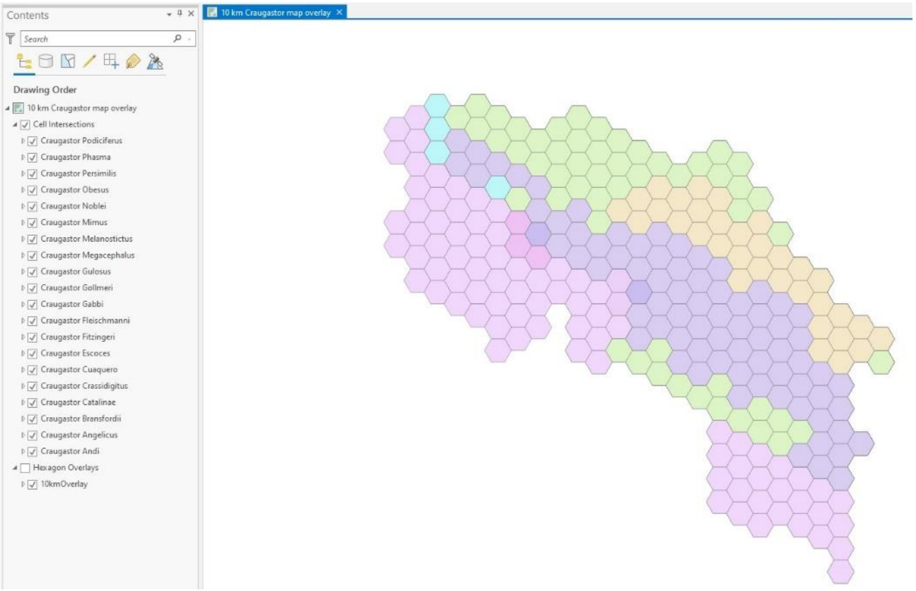


Figure 5. Custom ArcGIS Map

Development:

Application Views:

The application consists of 4 main views: MMPK selection, Map View, List of Frogs, Frog Details. The MMPK selection view welcomes the user to the app and has a button to select the MMPK that they will use to identify frogs. The idea behind this view is so that the app could potentially be used for either more precise MMPK's or in other regions, allowing the app to have expansive possibilities.

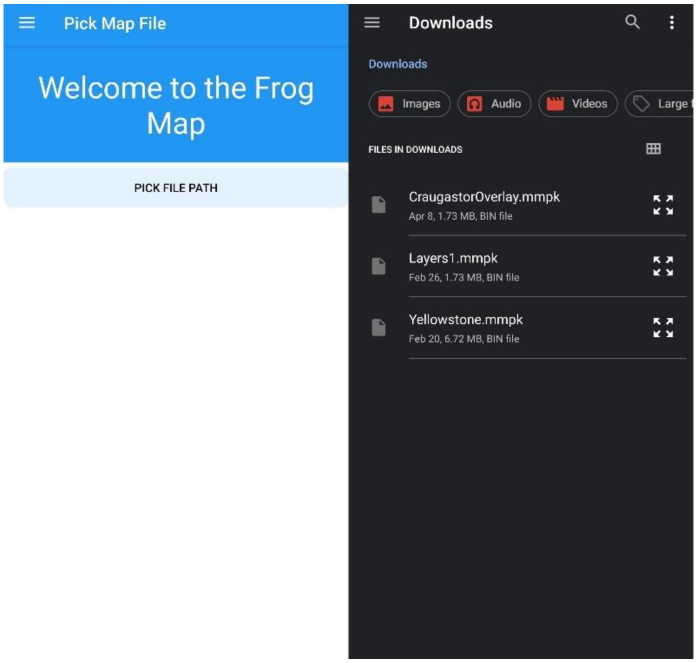
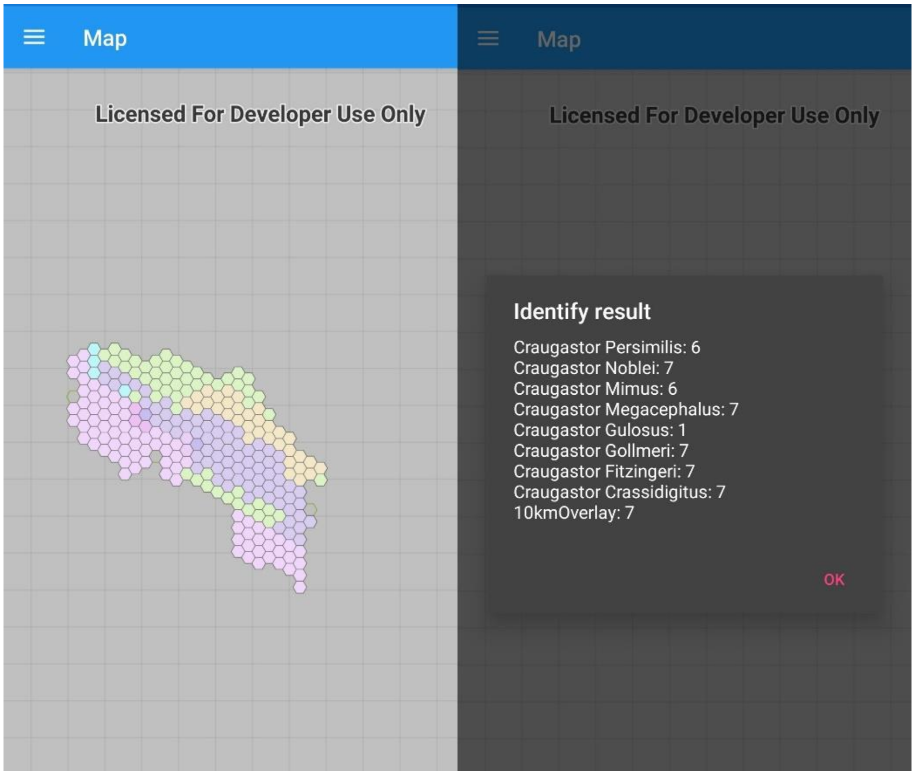


Figure 6. MMPK Selection View

Whichever MMPK is loaded is displayed within the map view. This view displays the ArcGIS layers that are visible within the MMPK and gives the user the ability to click on the map. Supposing the MMPK the user loaded is the custom MMPK created for this application when the user taps on a hexagon. Then they are greeted with a message that lets them quickly know the frog species in the area they clicked, the number of hexagons they tapped on that had that frog species, and how many hexagons they clicked on.

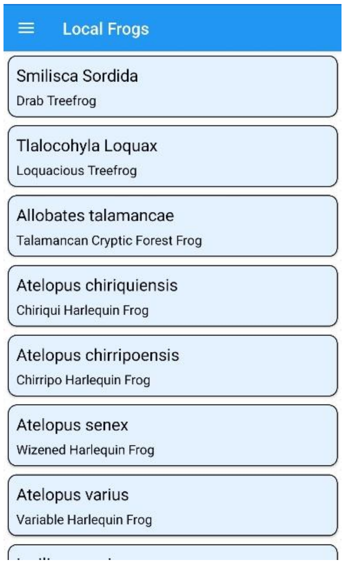

Map
Licensed For Developer Use Only
Map
Licensed For Developer Use Only
OK

Identify result

Craugastor Persimilis: 6
Craugastor Noblei: 7
Craugastor Mimus: 6
Craugastor Megacephalus: 7
Craugastor Gulosus: 1
Craugastor Gollmeri: 7
Craugastor Fitzingeri: 7
Craugastor Crassidigitus: 7
10kmOverlay: 7

Figure 7. Map View

If the user wants to view this lister closer, they can hit ok and are taken to the list of local frogs that were present on the Identified Results message. On this list, you see the species Latin name with their common name just underneath it.



Smilisca Sordida
Drab Treefrog

Tlalocohyla Loquax
Loquacious Treefrog

Allobates talamancae
Talamancan Cryptic Forest Frog

Atelopus chiriquiensis
Chiriqui Harlequin Frog

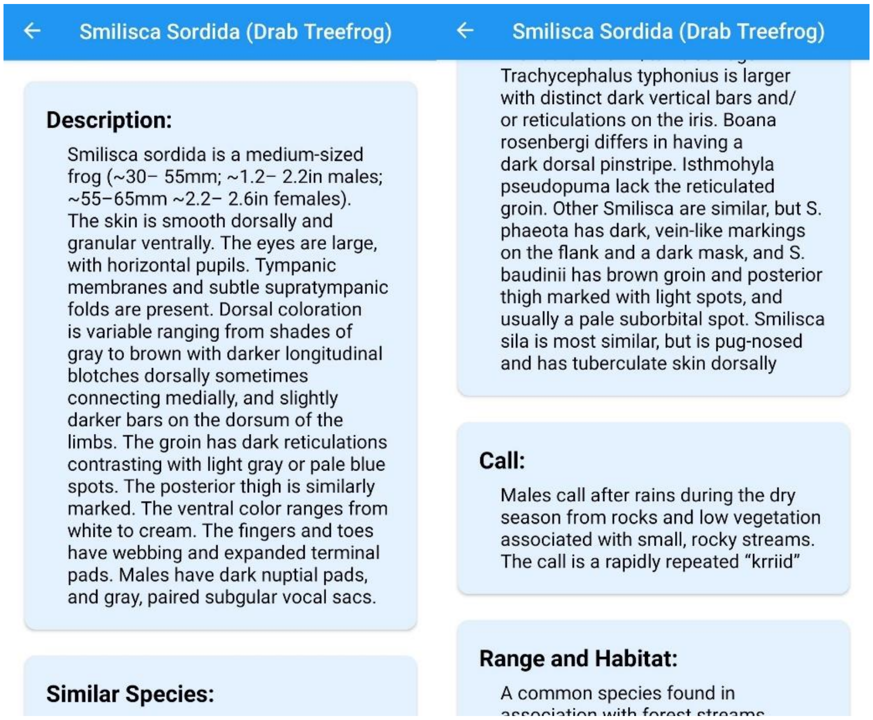
Atelopus chirripoensis
Chirripo Harlequin Frog

Atelopus senex
Wizened Harlequin Frog

Atelopus varius
Variable Harlequin Frog

Figure 8. List of Frogs View

Each species is present like a button and can be clicked on to take the user to the frog's detail View. This view displays details on the frog, including their description, similar species, calls, and Range and Habitat.



←
Smilisca Sordida (Drab Treefrog)
←
Smilisca Sordida (Drab Treefrog)

Description:

Smilisca sordida is a medium-sized frog (~30– 55mm; ~1.2– 2.2in males; ~55–65mm ~2.2– 2.6in females). The skin is smooth dorsally and granular ventrally. The eyes are large, with horizontal pupils. Tympanic membranes and subtle supratympanic folds are present. Dorsal coloration is variable ranging from shades of gray to brown with darker longitudinal blotches dorsally sometimes connecting medially, and slightly darker bars on the dorsum of the limbs. The groin has dark reticulations contrasting with light gray or pale blue spots. The posterior thigh is similarly marked. The ventral color ranges from white to cream. The fingers and toes have webbing and expanded terminal pads. Males have dark nuptial pads, and gray, paired subgular vocal sacs.

Similar Species:

Trachycephalus typhonius is larger with distinct dark vertical bars and/or reticulations on the iris. Boana rosenbergi differs in having a dark dorsal pinstripe. Isthmohyla pseudopuma lack the reticulated groin. Other Smilisca are similar, but S. phaeota has dark, vein-like markings on the flank and a dark mask, and S. baudinii has brown groin and posterior thigh marked with light spots, and usually a pale suborbital spot. Smilisca sila is most similar, but is pug-nosed and has tuberculate skin dorsally

Call:

Males call after rains during the dry season from rocks and low vegetation associated with small, rocky streams. The call is a rapidly repeated "krriid"

Range and Habitat:

A common species found in association with forest streams

Figure 9. Frog Detail View

Final Remarks

Complications:

There were a few issues that were encountered during the development of this project. First, when allocating the ArcGIS map design to a group's final assignment, somewhere during the final stages of putting the map together, the Map's Coordinate system was corrupted. The corrupted coordinates meant that even though all the hexagons remained together when overlayed with a base layer map, they do not sit over Costa Rica. This is problematic because it means that GPS can not be used within the application to find your current location and tell you local frogs within your hexagon.

The other issue was that Xamarin is being deprecated in a few years by Microsoft in favor of their new .NET Multi-platform App UI they are developing. This depreciation means that though the app still could function for a while, it is already running a shorter life cycle. Thankfully, MAUI is being built using many Xamarin features and is more like a clean evolution of Xamarin, so it could be easily ported over once MAUI is released and operational.

Future Work:

The Application is close to being fully functional; however, there are three things that it needs to be done before this can be true:

1. The custom ArcGIS map needs to be complete both with all of the frog species in it as well as fixing the coordinate system that they are all projected onto so that it can be used on a base layer that actually gives the surrounding details of Costa Rica and flushes out the Map View.
2. Once the ArcGIS map is done, the Map View needs GPS coordinates to be implemented into the ViewModel so that the user can also use that to access what local frogs are in their area, making it usable within the field.
3. It needs pictures to be added to the local frogs view next to each frog species and on the frog description page to help researchers more quickly identify the frog species they may be looking at.

Conclusion:

This project has helped show me how similar goals on different development tools can change how the development goes dramatically. For an internship, I had to do a similar task with ArcGIS but for a website online. Switching it to .NET and offline meant that I had to start over from scratch in how I had to approach the problem, and it meant that I only had better knowledge of terms. This project also taught me the importance of having a healthy “home” life on the productivity of a large-scale project like this. Ultimately what this project taught me is that no matter how hard a task is and no matter how much life drags you down, the point is not whether you came first or last but that you finished

References

Britch, D., Schonning, N., Dunn, C., & Osborne, J. (2017, July 8). *The Model-View-ViewModel Pattern - Xamarin*. Xamarin | Microsoft Docs. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm.

Frey, R. (2010). *Diagram of interactions within the Mvc pattern*. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:MVC-Process.svg#filelinks.

Hanmer, R. (2013). Structuring Your Interactive Application with Model-View-Controller. In *Pattern-oriented software architecture for dummies* (pp. 189–208). essay, John Wiley.

Maxwell, E. (2017, January 26). *MVC vs. MVP vs. MVVM on Android*. Realm Academy - Expert content from the mobile experts. https://academy.realm.io/posts/eric-maxwell-mvc-mvp-and-mvvm-on-android/.

Mishra, R. (2020, November 11). *Difference Between MVP and MVVM Architecture Pattern in Android*. GeeksforGeeks. https://www.geeksforgeeks.org/difference-between-mvp-and-mvvm-architecture-pattern-in-android/.