POLITECNICO DI TORINO

Corso di Laurea Magistrale
In Ingegneria Informatica

Tesi di Laurea Magistrale
Incognito Mode and Rebtel News Feed: Extending Calling App to improve the User Experience

Relatore
Giovanni Malnati

Candidato
Marco Caloiaro

Anno Accademico 2018/2019

ABSTRACT

Building a successful mobile app in a world which welcomes more than 1000 new apps everyday on Google Play Store and Apple Store is not an easy task.

Tons of applications are duplicated, useless and sometimes, even if they are performing, the world might not be ready for them yet.

However, by offering international calls and various services including Mobile Top Up and early access Remittance at a very competitive price, Rebtel Networks has risen in the last years generating a high revenue and customer satisfaction.

Calling has been a profitable business for years, but it might not be the same in the near future, due to the rise in popularity of new free messaging/video calling services such as WhatsApp, Messenger and Skype.

This is the main reason why establishing a good relationship with the customers and improving the user experience on the app are two fundamental aspects.

My internship at Rebtel has been focusing on implementing two new features for the company's existing calling app: **Incognito Mode** and **Rebtel News Feed**.

The first one mentioned has become very popular in the last years. Browsers like Google Chrome offer their customers local privacy through Incognito Mode, and Rebtel is putting the same efforts in order to satisfy one of its users most asked requests, "Hidden CLI". While a temporary version of the feature was already in place since earlier releases, with the help of the Design Team and Product Managers I redesigned the whole concept.

Rebtel was founded on solid values. Family is definitely the most important. By analysing the main reasons why people call their relatives the most, during my internship I have explored another interesting idea which might be introduced on the application in the future: Rebtel News Feed.

Being always up to date with what is going on in your own country is very difficult, especially if you live abroad and work many hours per day. This new feature would help the customers to get access to top headlines from a specific country of their interest, resulting into an increase in number of calls and in the average time a user spends daily on the app.

My internship at Rebtel has been really productive and the amount of learnings, both from a business and technical prospective, has been huge. While Incognito Mode is already in production and shipped for the next release of the app, Rebtel News Feed needs to be studied carefully by the commercial team, in order to verify whether it can represent a real opportunity for business or not.

A mio padre e ai suoi 7 anni da solo a Turbigo, a mia madre e alle sue pratiche lavorative fino a tarda notte. Grazie per tutto ciò che avete fatto per garantirmi un futuro migliore.

A mia zia Consiglia e a mio zio Vito, a mia cugina Danila e mio cugino Paolo che mi hanno sempre trattato come un figlio/fratello.

A mio nonno Donato e a tutti i suoi sacrifici per strapparmi un sorriso.

Ai miei zii, i miei nonni e i miei cugini che mi hanno sempre supportato nei momenti più difficili senza mai chiedere nulla in cambio.

E per finire a tutti i miei amici che non mi hanno mai abbandonato, anche se a volte sono stato poco presente, scontroso o nervoso.

A tutti voi,

Marco

TABLE OF CONTENTS

CHAPTER 1

1. Introduction

This thesis will focus on the projects I have taken part in during my stage at Rebtel Networks AB in Stockholm since the 4th March until the 28th June 2019.

All the activities carried out during the internship will be described in the next chapters, according to the following subdivision:

- Introduction, which will briefly describe the company history and the development platform.
- Development, which will focus directly on the implementation of the features.
- Conclusions, which will analyse results and next steps to improve the features introduced.

1.1 Aim of the internship

My internship at Rebtel had 2 main aims: exploring the existing Calling Android App and implementing new features that should improve the user experience and customer satisfaction.

The first part of my experience, which lasted for one month, could be considered as playground. I analyzed carefully Rebtel's java code and the main patterns adopted by the developers through the years, fixed some simple bugs and got a deeper understanding of the company's working flow with different tasks.

The second period, instead, regarded the research and the implementation of a good feature from the company's backlog[1], respecting its roadmap and production deadlines.

[1] Backlog: collection of tasks to be implemented and completed.

At the end of a series of cross-functional team meetings, the final feature chosen was Incognito Mode. Beside this project, required from the company, I was given the opportunity to explore a personal idea. My choice fell on Rebtel News Feed.

The next sections of this introductive will briefly provide a presentation of Rebtel company and will focus on the development platform adopted to implement the features: Android.

The next chapters will contain instead more in detail implementation choices, results expected and results obtained.

1.2 Introduction to Rebtel

Rebtel is a global communication platform serving internationals founded in 2006 by Hjalmar Winbladh & JonasLindroth.

From 2006 to 2014, under their management, the company went through different stages of growth and released in 2012 an SDK, allowing developers to integrate voice calling into their apps[1]. The same year, the company reported revenues of $80 million and 20 million users.

In 2015, after a drop of nearly 20% YoY revenue registered in the second half of 2014, the company had a new management take over. Magnus Larsson, the current CEO, has launched a new strategy with more "product verticals like banking, remittance and independent work for migrants and international nomads"[1]. Under the new management, Rebtel has reported a +20% YoY revenue growth.

More in the specific, the company serves one of the fastest growing group on Earth: the internationals. This large group, composed by migrants and travelers, contributes to a 15% yearly communication growth in a large industry already worth more than 100$Bln.

While nowadays immigrants are often portraited negatively, Rebtel sees them as an inspiration for a global aspirational brand.

This is probably the main reason why it is one of Stockholm's most diverse company, with 100+ employees representing more than 40 countries. Taking in consideration all the different backgrounds, Rebte pays attention really carefully to migrants special needs, such as money transfer/banking, international calling and Mobile Top Up.

By offering a very competitive price, Rebtel combines traditional telecom networks with App+Voip technology. It provides a simple user experience, allowing its customers unlimited calls in the world for a fixed price, always with the best connection based on quality and

price.

The company projects are far from done. Rebtel Networks AB development team is currently building a new banking service mobile app called Majority, which will offer unlimited calling, based on Rebtel existing framework, no fee money remittance and a debit card to all its users.

1.3 Description of operating system and Development Platform: Android

During the internship I have implemented the two new features for the existing Rebtel Android app.

Android is a mobile operating system developed by Google. Being based on a modified version of the Linux kernel and other open source softwares, it is designed primarily for touchscreen mobile devices such as smartphones and tablets. [2]

1.3.1 Brief History of Android

Android Inc. was founded in Palo Alto, California, in October 2003 by Andy Rubin, Rich Miner, Nick Sears, and Chris White.

Rubin described the Android project as "tremendous potential in developing smarter mobile devices that are more aware of its owner's location and preferences". [3]

While the early intentions of the company were to develop an advanced operating system for digital cameras[4], the founders then figured out that “Android goals were larger than the market for cameras, and by five months later it had pivoted its efforts and was pitching Android as a handset operating system that would rival Symbian and Microsoft Windows Mobile”.[4][5]

“In July 2005, Android Inc. was acquired by Google in a deal which is rumored to be at least $50 million”[6]. “Its key employees, including Rubin, Miner and White, joined Google as part of the acquisition”.[3]

“At Google, the team led by Rubin developed a mobile device platform powered by the Linux kernel. Google marketed the platform to handset makers and carriers on the promise of providing a flexible, upgradeable system”.[7].

“The first commercially available smartphone running Android was the HTC Dream, also known as T-Mobile G1, announced on September 23, 2008”[8][9].

Since 2008, numerous updates have incrementally improved the Android operating system, by adding new features and fixing bugs from previous releases. Each major release is named in alphabetical order after a dessert or sugary treat, with the first few Android versions being called "Cupcake", "Donut", "Eclair", and "Froyo".

1.3.2 Why Android?

There are many reasons why implementing an app in Android might be more beneficial than any other operating system.

First of all, the market share. With a share of 85.11% in 2018, Android dominates the mobile app development market. Analysing future estimates, this number will slightly increase, reaching the 87.1% of overall shipment volume in 2023. [10]

Even from the profitability point of view, Android can still achieve significant results. In fact, even if Apple Store has generated more than double the revenue of Google’s Play Store with fewer than half as many downloads, Android users spent more than 11.8$Bln, only considering the first half of 2018. [11]

Choosing Android as operating system for development might be also due to other reasons, not directly related to business, but still relevant.

Android development has a really low entry level barrier, which means that, in order to start developing for Android devices, users do not have to use a specific operating system as for iOS development. Besides, subscribing to Google Play as a developer is way cheaper than other platforms. [12]

Last but not less meaningful reason is the Google Play Store. While Apple Store’s review process is tedious and might take many days for a developer app review, Google Play Store’s guidelines are less strict and waiting time is slightly smaller than its main competitor. However, Google has announced policy changes to improve the quality of apps in Play Store, in wake of the recent scandals involving popular apps as Facebook, which in more than one occasion has exposed accidentally its users sensitive data. [12]

1.3.3 Android Environment

“Android apps are written using the Android software development kit (SDK) and, until 2017, the Java programming language, combined with C/C++”.[13]

However, In May 2017, Google announced support for Android app development in the Kotlin programming language[14], which is slowly replacing Java as the main programming language for Android.

“The SDK includes a comprehensive set of development tools, including a debugger, software libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials”. [15]

“Initially, until December 2014, Google's supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) plugin; today, Android Studio, based on IntelliJ IDEA, is the primary IDE for Android application development”. [16][17]

Some key features include:

- “Gradle-based build system
- Live-layout Editor with real time app layout rendering
- Option to preview a layout on multiple screen configurations while editing
- Build variants and multiple apk file generation
- Supports developing Android Wear, TV and Auto apps
- Enables app integration with Google Cloud Platform (App Engine and Google Cloud Messaging)”

[12]

The Android version currently adopted in Rebtel is Android PIE 9, with API version 28.

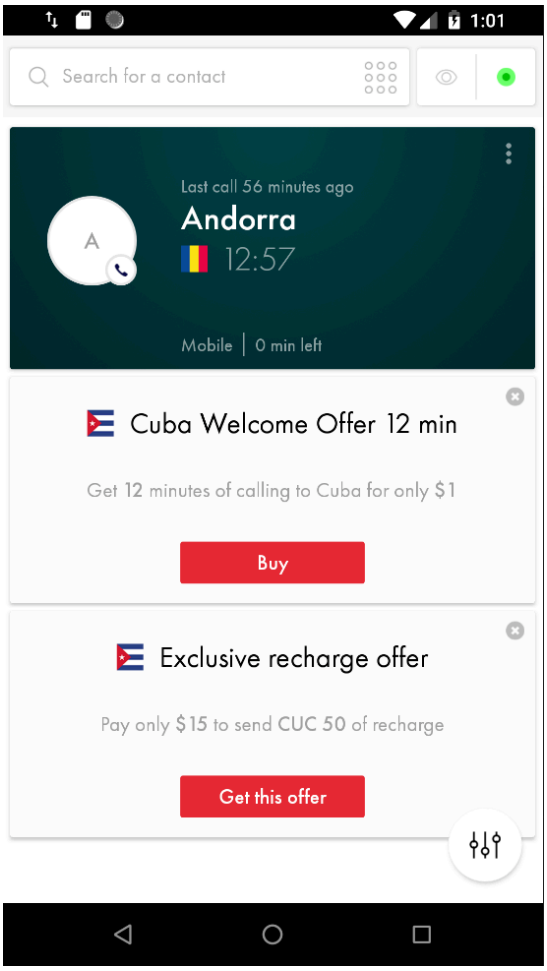
CHAPTER 2.

2. Incognito Mode.

2.1 Presentation of Rebtel's App

Rebtel's mobile app went through a series of different changes in the last years, and probably it will be significantly different from today in a short period of time in order to host the implementation of new features and the redesign of current ones.

However, during my entire internship the principal screens that characterized the app were 2: Living Room and Account View.



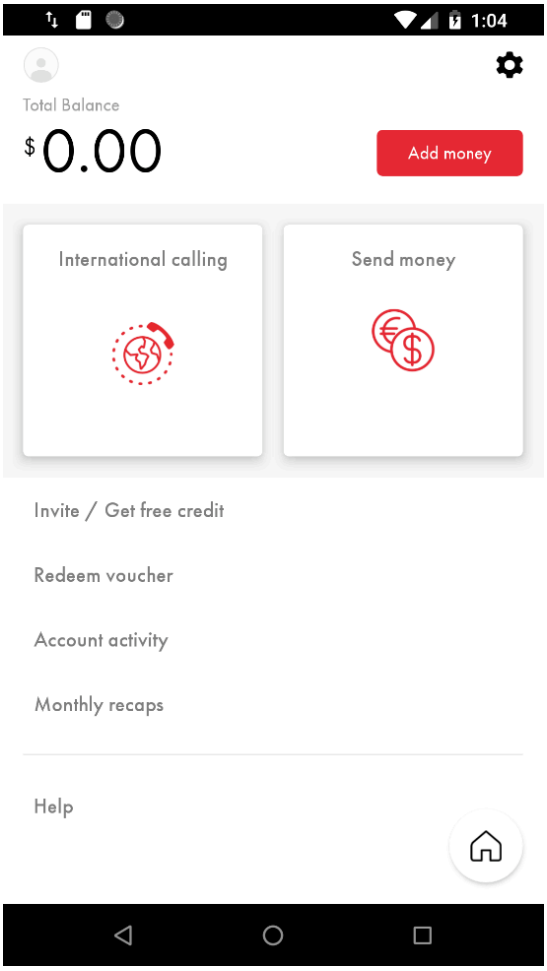
Living Room Design

The “Living Room” is the default page the user is shown after signing up on the Rebtel App and whenever he/she opens the application. It represents an on-boarding page from which the customer can have access to the following services:

- Latest calls sent and received;
- Latest transactions sent and received;
- Latest products and offers available for the user’s Top Country[2].

- [2] Top Cuntry: country for which the user has the most contacts in his address book or the Top country the customer has chosen during Sign Up.

Through an intuitive interaction with the various widgets shown on the screen, the user can start the calling process after tapping the search box, or move to the account view screen by tapping the floating action button at the right bottom of the Living Room page.



Account View Design

The “Account View” plays several roles inside the Application. As suggested by the names of the different fields, it hosts:

- **Wallet Balance** fragment on top of the page, from which the user can easily edit his settings;
- **Market Place** section, which provides recharges and products purchase;
- **Account Activity** stats and **Monthly Recap**;
- **FAQ** section
- **Redeem Voucher/TAF** activities.

The navigation inside the account view page is delegated to a different activity[3], called RebtelActionBarActivity.

"Living Room" and "Account View" are both fragments[4] of the Main Activity, called PagedActivity.

2.2 Incognito Mode. Why?

Incognito Mode is a feature available today in most of the browsers.

It enables users to browse the internet anonymously on a local device. "However, it does not hide internet activity from a service provider or other eventual observers". [18]

This means that, while ISP can be aware of all the search history of their customers, Incognito Mode protects a person's privacy from other people who may have access to the same device.

There are plenty of reasons for using it, but let´s focus on the most common scenarios:

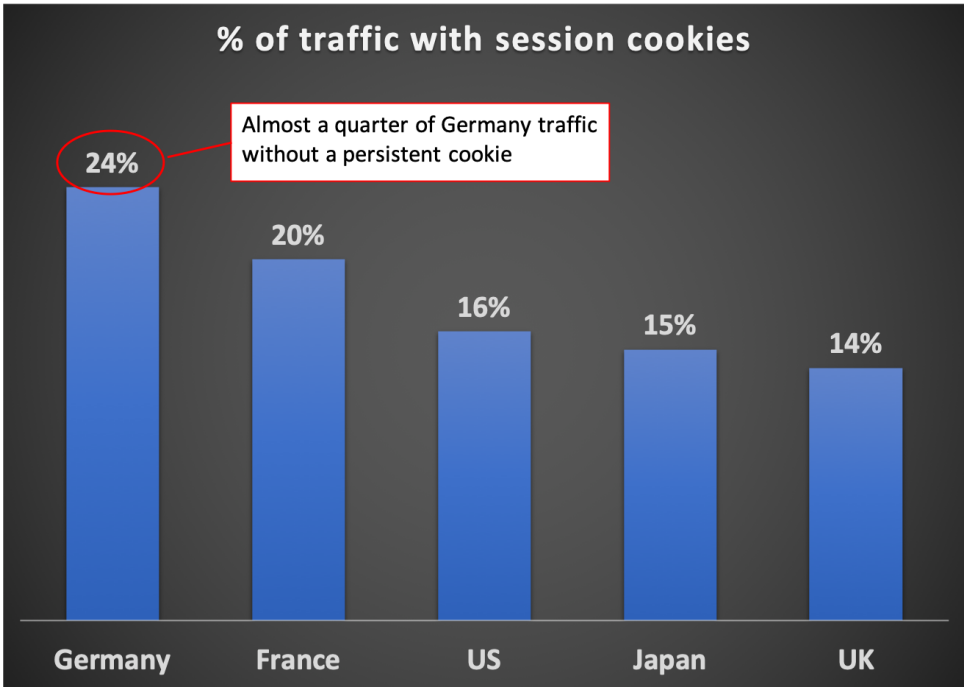
- **Buying gifts**: brainstorming for gifts idea is quite common, and it is easy to disclose informations with a regular web search. Incognito Mode prevents this from happening;
- **Signing in with Multiple Email Accounts**: it is possible to stay signed in to one account while simultaneously checking another one by opening an Incognito Session;
- **Related search and results**: Incognito Mode prevents Youtube and other services from spamming recommended videos based on their suggestion algorithm;
- **Public devices**: Incognito mode deletes everything associated with a specific session. This brings more safety logging in to personal accounts while using public devices;
- **Questionable Searches**: it could happen sometimes to look for unusual things on the web. For instance, if a user is a creative writer and wants to write a new story with suggestive particulars, he might look for questionable stuff on the web. A lot of scenarios regarding this matter are possible. With Incognito Mode privacy is ensured.

[12]

[3] Activity: window where UI is placed in Android.

[4] Fragment: portion of user interface contained inside an activity

Incognito Mode is a powerful feature which has gained much popularity over the years. Studies have suggested that almost 20% of web traffic comes from private browsing, reaching a peak of 24% in Germany[19].



2.3 Incognito Mode: Rebtel's Case

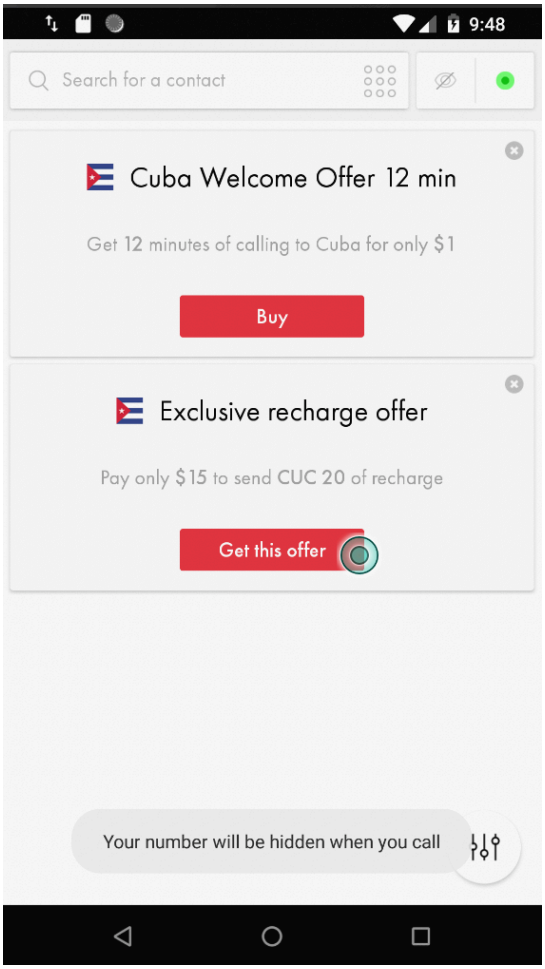
Rebtel is a calling app. The reasons listed above do not perfectly explain any meaningful reason for the tech company to implement Incognito Model, yet the customer support desk has collected in the last years many requests regarding a new feature to allow users to hide their numbers when they call.

In fact, some users may want to hide their numbers in order to conduct business using Rebtel, others would like to call their relatives using a hidden number to prevent them from calling back and incur in accidental charges. Other purposes for wanting an Incognito Mode are less common but in general a lot of users needed this feature inside the app and they made it clear.

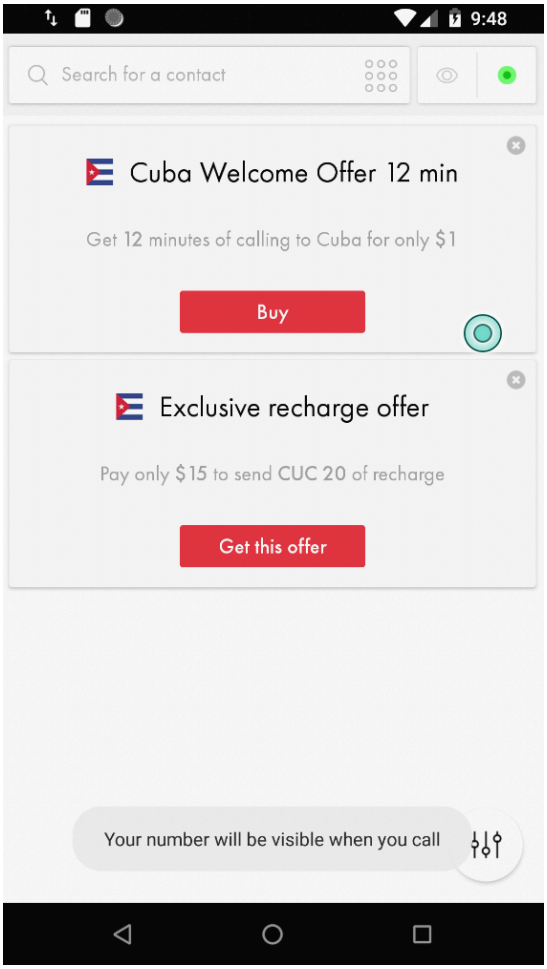
2.3.1 Initial Stage of the Feature

Rebtel satisfied its customers' needs, well at least kind of.

Before my implementation, on the app it was now possible to turn on the hidden number calling feature. However, when turning on, the user was presented just with a dialog at the bottom of the screen warning him/her that his/her number was now hidden:



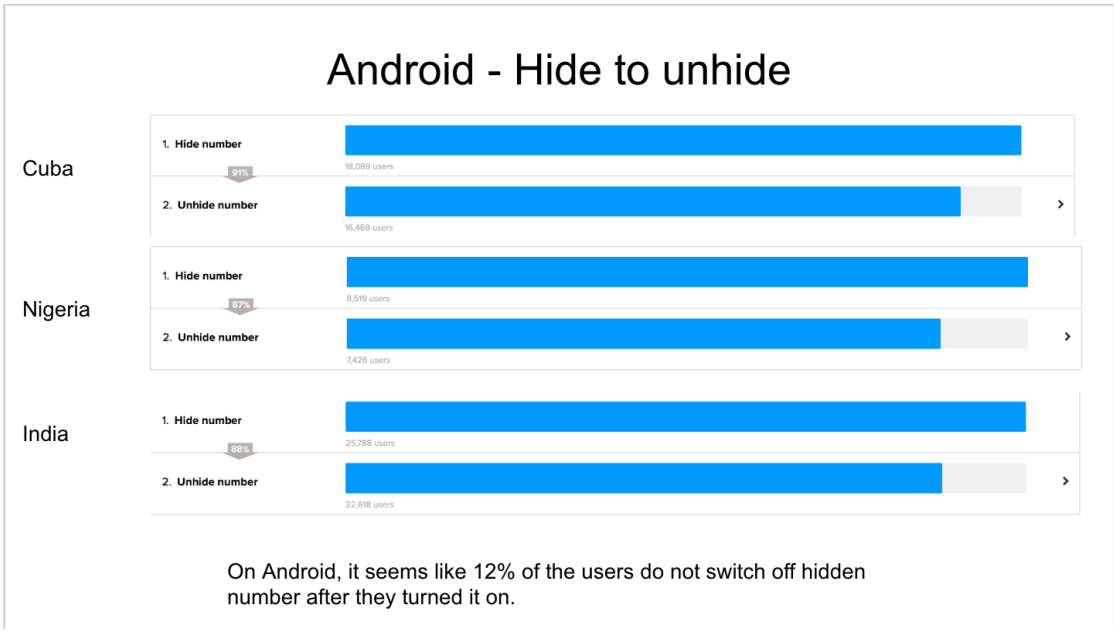
Enabling Hidden CLI

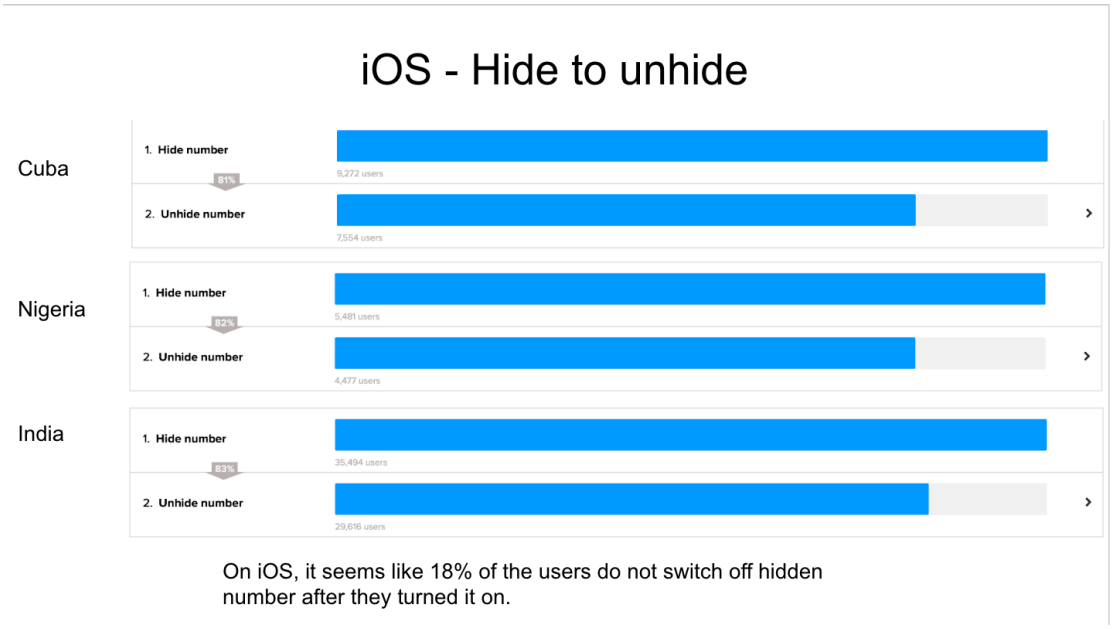


Disabling Hidden CLI

After this feature was launched, Rebtel analysts could notice a steady increase on calls made with hidden CLI in Nigeria, Cuba and India. These were the countries from which the users expressed the most the need for an Anonymous Calling Mode.

Nevertheless, according to the graph below, it seems like the design logic might have been confusing, as a remarkable percentage of people were not aware they were using the feature when calling:



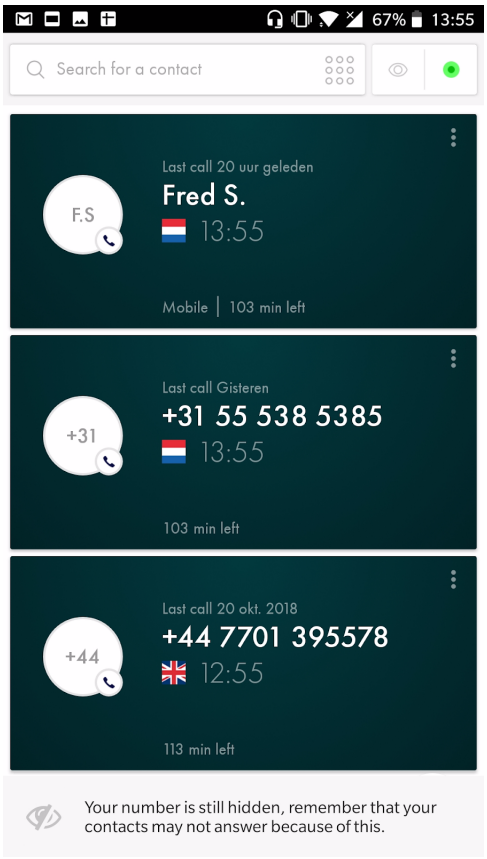


These results testified there was room for many significant improvements.

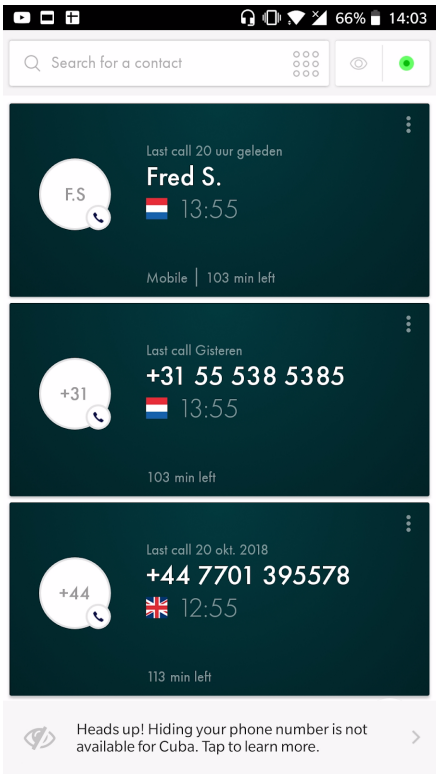
2.3.2 Initial brainstorming and Incognito Idea

Rebtel's design and product team have conducted many studies on how to optimize the feature for the customers.

One idea could have been reminding the user anytime he/she accessed the Living Room that he/she did not disable the hidden CLI feature, as shown below:



Through traffic analysis the teams also noticed ETECSA, Cuba's telecom monopoly operator, blocked calls with Hidden CLI. That is why many customers could not get calls connected to Cuba with the option enabled. One way to prevent this from happening might have been displaying a message with a link for requesting more informations:



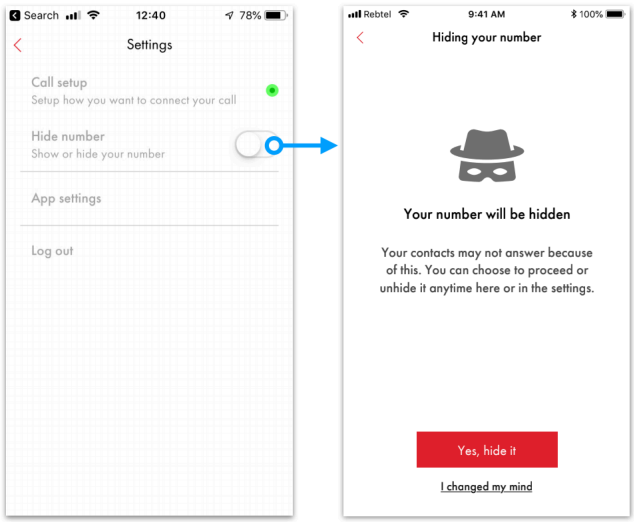
Other solutions proposed were about emphasizing the concept of making the user aware he had enabled a different mode. Ahead of all, a Pre call screen after first call with the feature showing the message: “Warning, this might cause your calls not get connected” and a post call screen: “Your next call will not be hidden” could have been a valuable solution.

All the ideas explained are valid and might overcome most of the issues, but the most interesting one was the exploration of a Dark Mode.

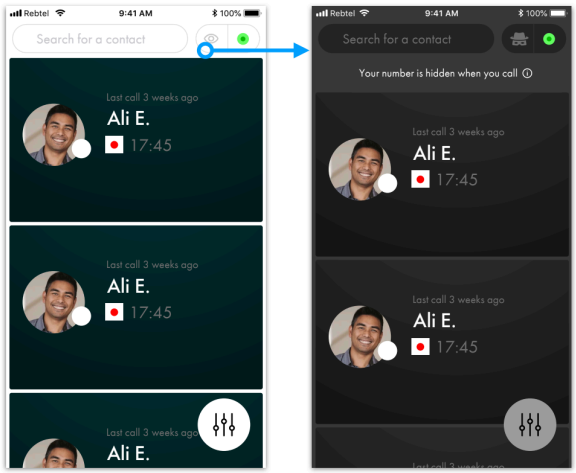
The dark mode, or Incognito Mode, would have consisted in changing the style of the Living Room and the main calling screens, leaving a lot of possibilities for future products development.

The first design choice suggested by the Design Team of the new feature is the one shown below:

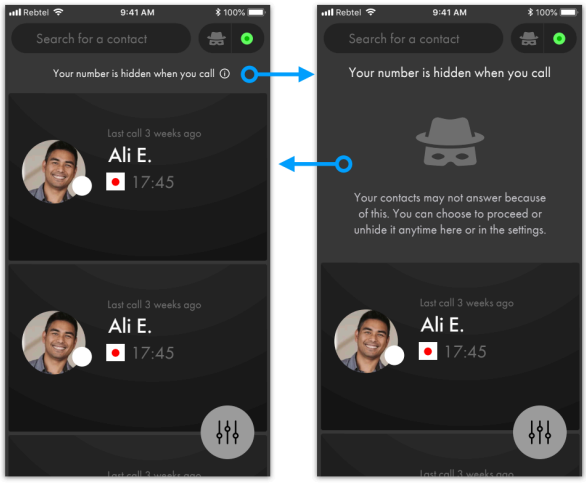
First time turning on the Anonymous Mode also from Settings



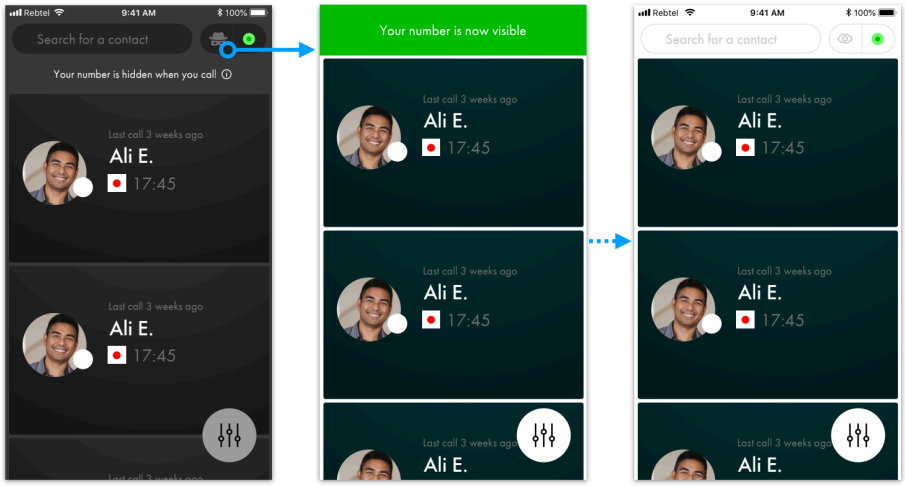
Next time turning on the Anonymous Mode



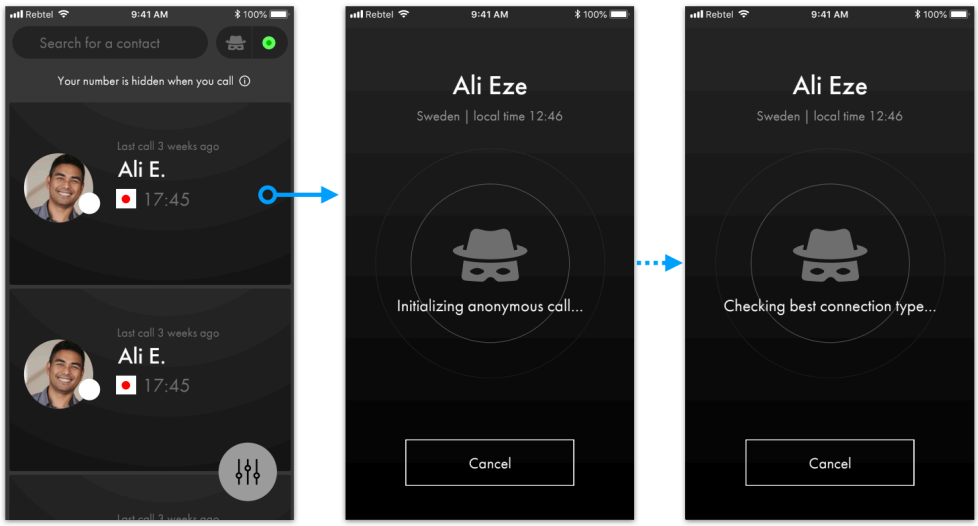
Showing user more information



Turning back to visible mode

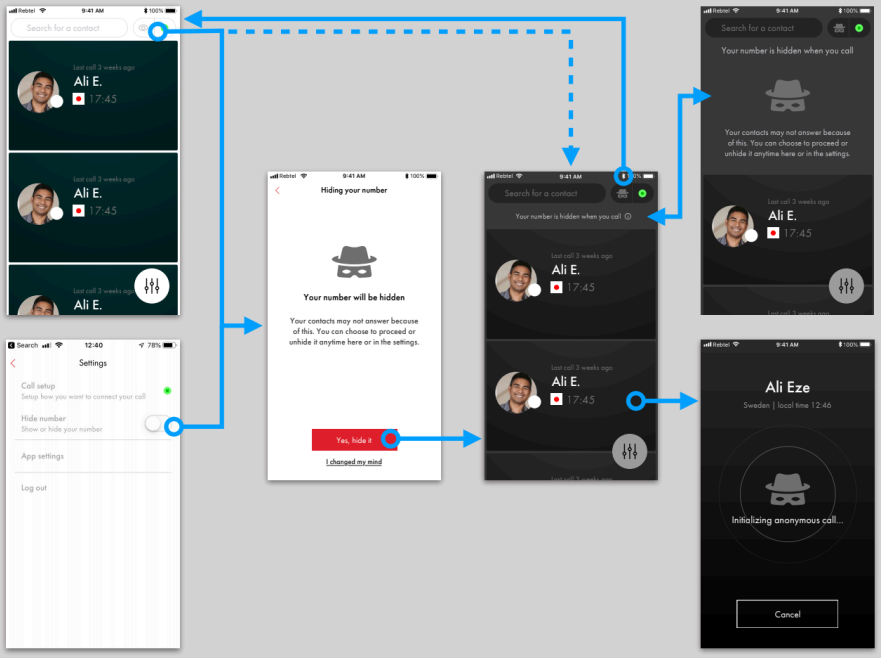


Making an anonymous call



The whole flow is shown below:

Flow overview



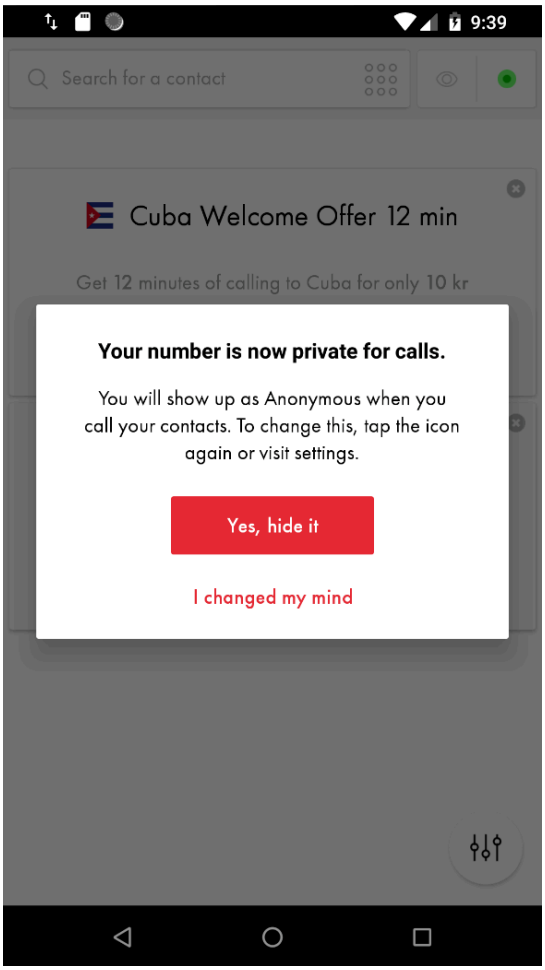
2.3.3 Implementation of the feature

Before moving forward with the implementation, one last sync meeting with the teams involved took place. It revealed some consistency issues with the design previously proposed. In fact, changing the theme of the Living Room would have implied to apply changes to Living Room cards (calling and products ones) and to decide whether or not the Account View page should have been affected. In brief, more design resources would have been requested for the project.

The design team finally suggested to keep the theme as it is. However, it would have been necessary to warn the user as much as possible using simple screens or dialogs that do not affect directly the graphical interface of the app screens.

In order not to bring more complexity to the app, we decided unanimously to take this road.

First of all, when the user enables the Incognito Mode for the first time, he will be presented with the warning screen shown below:



The user will then have to take the decision of keeping the regular mode or eventually switching to Incognito Mode. If he/she follows the second choice, its user's preference gets updated calling a method from the ClientPreference[5] class, updateIncognitoFirstTime().

This method updates the Boolean value "isFirstTimeHiddenCLI" in the SharedPreferences of the user, setting it to false. This value will be checked every time the user enables again the Incognito Mode in order to prevent the Warning dialog from displaying again.

```
public static void setFirstTimeHiddenCLI(Context context, boolean hiddenCLIEnabled) {
  getPrefEditor(context).putBoolean(PREF_FIRST_TIME_HIDDEN_CLI, hiddenCLIEnabled).apply();
}
```

```
public static boolean isFirstTimeHiddenCLI(Context context) {
  return getPref(context).getBoolean(PREF_FIRST_TIME_HIDDEN_CLI, true);
}
```

[5] Shared preference: Android api useful to store and retrieve app preferences.

```
@OnClick(R.id.hideNumberToggle)
void onHideNumberToggleClicked() {

  if (isFirstTimeHiddenCLI()) {
    handleFirstTimeHiddenCLI();

  } else {
    handleHiddenCLI();
  }
}


private boolean isFirstTimeHiddenCLI() {
  return ClientPreferences.isFirstTimeHiddenCLI(getContext());
}
```

The Living Room with Incognito Mode enabled presents a new field below the Search Box: the indicator. It shows a text that alerts the customer about his/her current state. The Incognito icon to the right of the Search Box gets updated accordingly to improve the awareness.

After tapping the information icon on the right of the Indicator text, the user will get access to more informations regarding the different activities available in Incognito Mode.

Beside these small edits in the Living Room page, the Call Screen Activity idea presented by the Design team found was preserved and implemented as shown in the previous design.

Regarding the feature block applied in Cuba and Nigeria, users warnings have been covered through Braze.

Braze is the leading customer engagement platform. It helps brands to build real long lasting relationships between brands and customers.

By using Braze it is possible to follow the lifecycle of customers by tracking exactly what they are doing on an application and when. Then it becomes easier to craft personalized messages based on their behaviour and automate the delivery using an adeguate channel.

In Rebtel, Braze is filled with data from a Tracking tool called mParticle and Rebtel's BackEnd[6] team. It then notifies correctly users who have the feature enabled to let them know Incognito Mode might be limited to some of the countries they are likely to call the most.

[6] Rebtel's Backend team stores all the products and the users personal informations.

CHAPTER 3

3. Rebtel News Feed, Introduction

After the implementation of the Incognito Mode, a feature for which the business case was already built and that is ready to be introduced in the main App, the company product managers gave me the opportunity to explore one of my personal ideas which might contribute to improve the daily experience for Rebtel's users in the future.

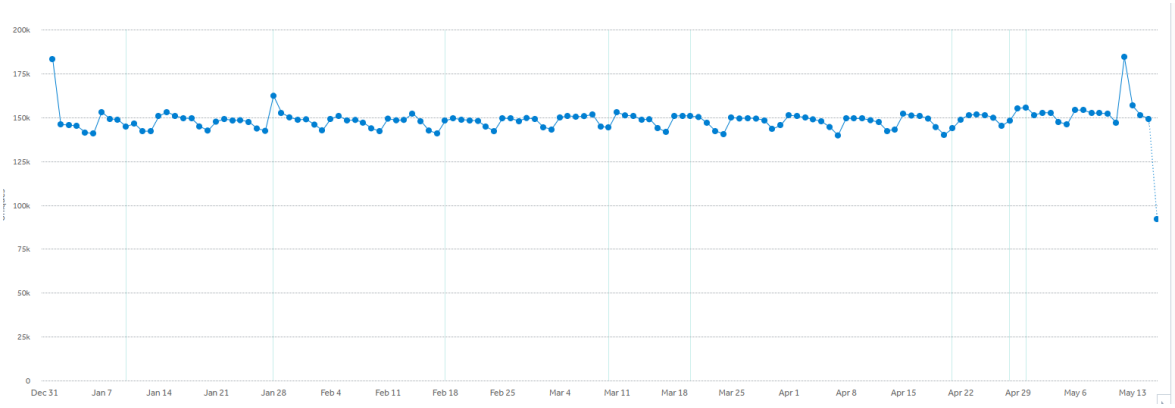
That is how the Rebtel News Feed concept, which will be the main focus of these two following chapters, was born.

3.1 Why?

When I was given the chance to think about a new feature for the Rebtel app, I knew that coming with an original idea was not an easy task. The calling structure is already solid and most of the features regarding the transactional world (remittance, recharges, credit) will be taken care of in their new application Majority, which will hit the digital stores in few months.

That is why me and my tutor started brainstorming and, through the help of the documentation provided to us by traffic analysts, we came to the conclusion that customers call using Rebtel mostly when Breaking news events happen (natural disasters or more general news related to politics/sports/economy).

For instance, the graph below shows the analysis of call attempts. The 28th january registered a peak, and it is related to a Tornado occurred in Cuba the 27th january. The second peak, instead, occurred on Mother`s day.



Graph showing calling peaks for specific days

These results made me think for the first time about a News Feed which could host top headlines from the Top Address Book of the customer, or alternatively from the Top Country chosen during the Sign Up process.

3.2 Expectations for the feature and business value analysis

Rebtel is a calling app. Why would it be worth it to introduce a News Feed?

This is a legit question. While the feature is not yet implemented inside the app and a real business case was not built around the idea, there are some interesting points to take in consideration.

First of all, according to the stats provided in the previous section, it is clear that Rebtel customers are following the news from the country they call the most.

Nowadays there are several online solutions that are completely devoted to displaying articles for a particular topic, such as Feedly or the more popular Google News.

However, none of the mentioned services would guarantee the users one single experience inside the app.

In fact, if a specific headline requires an immediate call (to make sure your relative is safe) or more in general if it seems interesting to have a discussion about it, Rebtel News Feed would allow users to make a call instantly inside the application. This would prevent a more common and general behaviour of postponing the call or totally forgetting about it while scrolling articles of the feed.

From a user experience point of view this makes a lot of sense, since the 2 behaviours are strictly related.

Beside this main deduction, which I believe to be the most relevant, there are other reasons why this feature might be very profitable for Rebtel in the future.

In fact, hosting a News Feed might not only produce an increase in the average daily time the user spends on the app, but could also boost the sales of the products the tech company has to offer.

As already mentioned in the presentation of the design of the app, Rebtel has launched for its customers a large number of services for different countries, and most of the company advertisement takes place today on the Living Room page. Obviously, having another region inside the app where to serve users with the range of products offered by the company, displayed in a smart and not invading way, would probably have a meaningful impact on the sales.

Last but not less important reason for hosting a Rebtel News is directly related to the company history. Its main purpose is to serve internationals, helping them to be in contact with their families and friends, and one fundamental aspect for the whole team is guaranteeing the user a good experience and maximum satisfaction.

Offering a news service for free, without asking anything in return, might retain the existing customers, intensifying their loyalty, and eventually attract new ones.

These main reasons brought me to explore the idea of a News Feed inside the app, which I honestly think could result in a successful outcome for Rebtel.

The next sections will focus on development choices and the design/building patterns I adopted.

3.3 Requirements Analysis

Before describing the various stages of the implementation, let`s quickly analyse in brief which are the exact requirements needed to carry out the task.

3.3.1 The News Feed Page

The News Feed Page should display the latest headlines for the user’s Top Country.

In order to accomplish the mentioned application behaviour, a specific network request to fetch the top headlines from a web source must be executed.

While fetching new items every few seconds is fundamental for a social network or other kinds of services, news do not need to be fetched every time the user refreshes the news feed, because it might be not essential and very costly in terms of resources.

This is why the latest news fetched from the system should be stored in a local database within the device, and refreshed every 15/20 minutes.

Besides, it is fundamental to allow the user to call a contact directly from the page, so a specific button should be placed within the fragment.

3.3.2 The News Feed Favorites Page

Some articles are so interesting that a user just might want to save them.

Or maybe the user cannot read an article at the moment, but would like to save it for later.

These are 2 common scenarios that happen every day to all of us.

This is why, like any other service, a mechanism to add a story to a list of favorites and display it in a different screen inside the app should be in place.

Since the news feed will be regenerated every 20 minutes, it will be necessary to flag articles which need to “survive” the deletion when new ones are fetched.

3.3.3 Handle communication with other pages

Rebtel News Feed should be accessible from the Living Room page, but the existing button only takes care of switching between the Living Room and the Account View.

So, editing the logic of the buttons or introducing a new one is required to correctly handle the communication between the fragments.

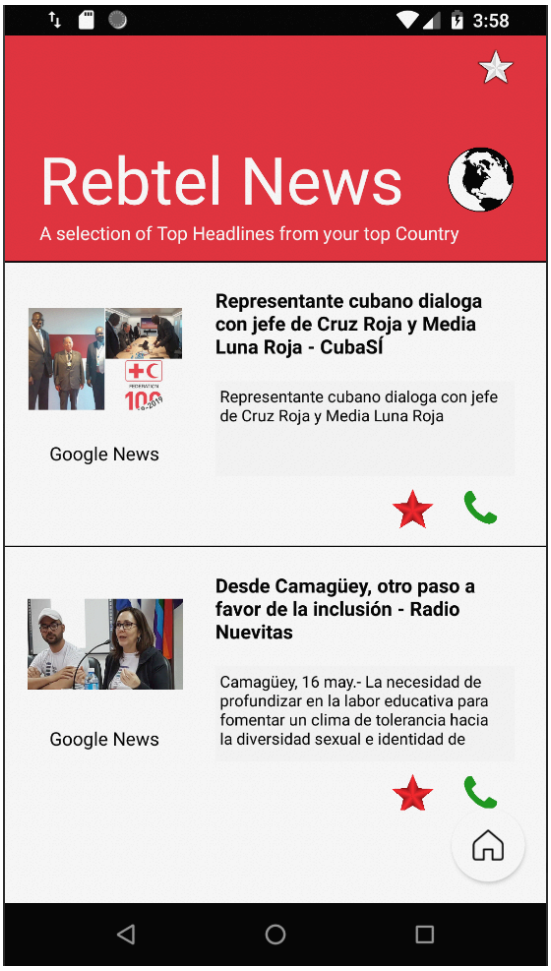
CHAPTER 4

4. News Feed Implementation

Having in mind the requirements described in the previous chapter, let's proceed analysing the main choices and patterns adopted for the implementation of the News feature.

4.1 Design Choices

Rebtel company has a whole team which worked on the design of the whole app. Considering as reference the app guidelines, with no assistance from the Design team due to their strict deadlines for other features already in production stage, I followed my taste and decided to have a simple design view for the Rebtel News Main page and the Rebtel News Favorite Articles Page, as shown in the picture below:



Rebtel News Feed Page Design

Rebtel News Feed Favorites Page Design

The 2 pages design consists in one Constraint Layout, which internally hosts an inner Constraint Layout and a RecyclerView inside a SwipeRefreshLayout[7] (for the News Feed Page only).

The inner Constraint Layout handles the title, the logo, the star/back icons and is clearly noticeable due to its red background.

The Swipe Refresh Layout allows the user to fetch again the articles currently stored in the Database.

As suggested by the two design pictures, the phone icon allows the user to call a contact, while the star icon has the power to flag the user's favourite articles.

[7] Layout used for refreshing contents of a page.

4.2 Implementation Choices

In this section we will focus on the main choices that influenced the implementation of the News Feed.

4.2.1 Architecture

During the first part of the internship I studied some of the main design patterns adopted for Rebtel app.

While in all my previous experiences with coding I adopted the MVC (Model-View-Controller), the tech company made large use of MVP (Model-View-Presenter) and MVVM (Model-View-View Model) patterns.

The MVC has 3 main components:

- **Model:** it is responsible for managing the data of the application and receiving user input from the controller.
- **View**: it is responsible for presenting the model in a specific format.
- **Controller**: it is responsible for receiving the user input, optionally validating it and passing it to the model.

[20]

"The Model–view–presenter (**MVP**), on the other hand, represents a derivation of the MVC architectural pattern and it is widely used for building *user interfaces*. In MVP, the *presenter* functions as "middle-man" and all presentation logic is pushed to the presenter. MVP advocates separating business and persistence logic out of the Activity and Fragment". [21]

As its parent MVC, MVP has 3 main components:

- **Model**: it is responsible for providing the data that will be displayed in the view.
- **View**: it contains a reference to the presenter. Every time an interface action occurs, the view calls a method from the presenter.
- **Presenter**: it is responsible for retrieving data from the Model and for returning it formatted to the view. It also decides what should be done when the user interacts with the View.

[21]

The main differences defined above are explained better in the flow picture below:

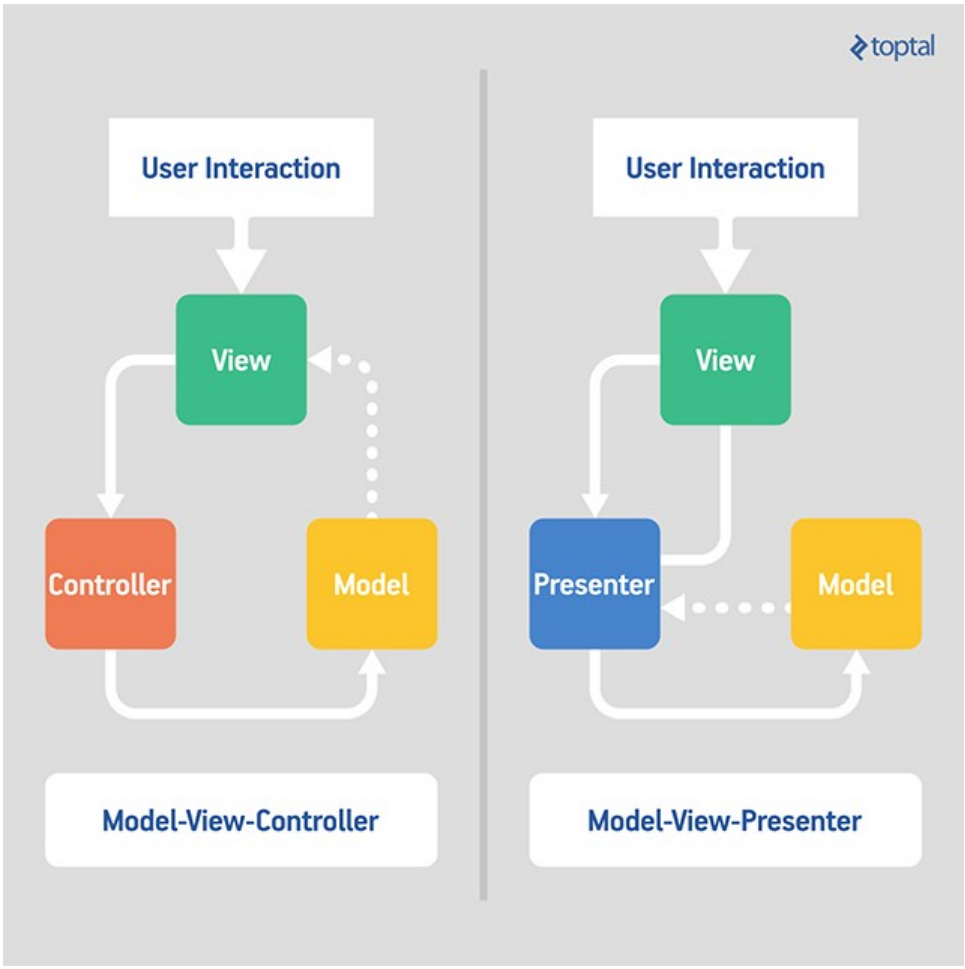


Image via Toptal

MVP is a mature pattern and it has gained a lot of popularity in the last years. However, due to some concerns about its testability and its tight coupling (1 view = 1 presenter), Google introduced Android Architecture Components which include a ViewModel rather than Presenter. [22]

MVVM, as its ancestor, has still three main components:

- Model: it refers to a domain model, which represents real state content, or to the data access layer, which represents content.
- View: it is a fragment, an activity, a layout. It represents anything the users sees on the screen.
- View model: "The *view model* is an abstraction of the view exposing public properties and commands. The main difference between the view model and the Presenter in the MVP pattern, is that the presenter has a reference to a view while the view model does not. In these case, a view binds directly to the view model to send and receive updates."

[23]

In the specific case of the feature, I decided to adopt the MVVM Pattern, extending it with a Repository, a Network Data Source, Live Data and Room Database. These components will be explained in the next section and any meaningful code needed to understand the implementation will be attached.

4.2.2 News Article Class + Builder Pattern

Let`s start from the Java Classes.

News Article class is a simple class with the following attributes:

- **ImageUri;**
- **Title;**
- **Content – unused for Developer Plan;**
- **Source;**
- **Author;**
- **Description;**

- **Favorite (flag to specify article has been favorited by the user);**
- **LastHour (flag to specify article belongs to latest fetch);**
- **Url (necessary for Developer Plan, explanation in next sections);**

This class contains 9 attributes, and it is predictable that remembering the order of the parameters for the constructor will be a challenge at variable initialization time.

This is the main reason why I have chosen to adopt the Builder Pattern for the News Article Class.

"The Builder is a design pattern that provides a flexible solution to various object creation problems in object-oriented programming. The intent of the Builder is to separate the construction of a complex object from its representation". [24]

An implementation of the Builder is provided below:

```
public static class Builder {

  private String imageUri;
  private String title;
  private String content;
  private String source;
  private String author;
  private String description;
  private int favorite;
  private int lastHour;
  private String url;

  public Builder() {
  }

  public Builder title (String title) {
    this.title = title;

    return this;
  }

  public Builder fromSource(String source) {
    this.source = source;

    return this;
  }

  public Builder byAuthor(String author) {
```

```
  this.author = author;

  return this;
}

public Builder withContent(String content) {
  this.content = content;

  return this;
}

public Builder withFavoriteFlag(int favorite) {
  this.favorite = favorite;

  return this;
}

public Builder withImage(String imageUri) {
  this.imageUri = imageUri;

  return this;
}

public Builder lastFetchFlag(int lastHour) {
  this.lastHour = lastHour;

  return this;
}

public Builder articleUrl(String url) {
  this.url = url;

  return this;
}

public Builder withDescription(String description) {
  this.description = description;

  return this;
}

public NewsArticle build() {

  NewsArticle newsArticle = new NewsArticle();
  newsArticle.title = this.title;
  newsArticle.lastHour = this.lastHour;
  newsArticle.favorite = this.favorite;
  newsArticle.content = this.content;
  newsArticle.author = this.author;
  newsArticle.description = this.description;
  newsArticle.imageUri = this.imageUri;
  newsArticle.source = this.source;
  newsArticle.url = this.url;
```

```
    return newsArticle;
}
```

Here is how a class object is built inside the view:

```
NewsArticle.Builder builder = new NewsArticle.Builder()
    .title(title)
    .fromSource(source)
    .articleUrl(articleUrl)
    .byAuthor(author)
    .lastFetchFlag(1)
    .withContent(content)
    .withImage(imageUri)
    .withDescription(description)
    .withFavoriteFlag(0);

NewsArticle article = builder.build();
```

Its main advantages are:

- "It supports to change the internal representation of objects.
- Encapsulates code for construction and representation.
- Provides more control over steps of construction process."

[25]

Compared to the default constructor, the Builder makes nearly impossible to make any mistake with parameters.

4.2.3 Articles List

As mentioned in the Design choices section, I opted for Recycler View as object to display the latest stories and the list of favorite articles.

The RecyclerView widget is a more advanced and flexible version of ListView.

“In the RecyclerView model, several different components work together to display data. The overall container for the user interface is a RecyclerView object that is added to the layout”. [26]

The RecyclerView is auto filled with views provided by a *layout manager* [26]. For the news feed case the choice fell on a LinearLayoutManager.

The items in the list are represented by *view holder* objects. These objects are instances of the RecyclerView.ViewHolder class. Each view holder is in charge of displaying a single item with a view. The RecyclerView creates as many view holders as are needed to display the various items, plus some extra holders. [26]

The adapter class manages the view holder objects. It also binds the view holders to their data. It achieves this behaviour by assigning each view holder to a position, and calling the adapter's onBindViewHolder() method. This method uses the view holder's position to determine what the contents should be, based on its list position. [26]

For the News Feed, I have designed only one type of item using a Constraint Layout.

ConstraintLayout is the most popular choice to create large and complex layouts with a flat view hierarchy. Similarly to RelativeLayout, views are connected by expressing relationships between sibling views and the parent layout, but it presents the advantage of being more flexible and easier to use with Android Studio's Layout Editor. [27]

All the power of ConstraintLayout is available directly from the Layout Editor's visual tools, because the layout API and the Layout Editor were specially built for each other. So, it is actually possible to build a layout with ConstraintLayout entirely by drag-and-dropping instead of editing the XML. [27]

“ConstraintLayout is available in an API library that is compatible with Android 2.3 (API level 9) and higher”. [27]

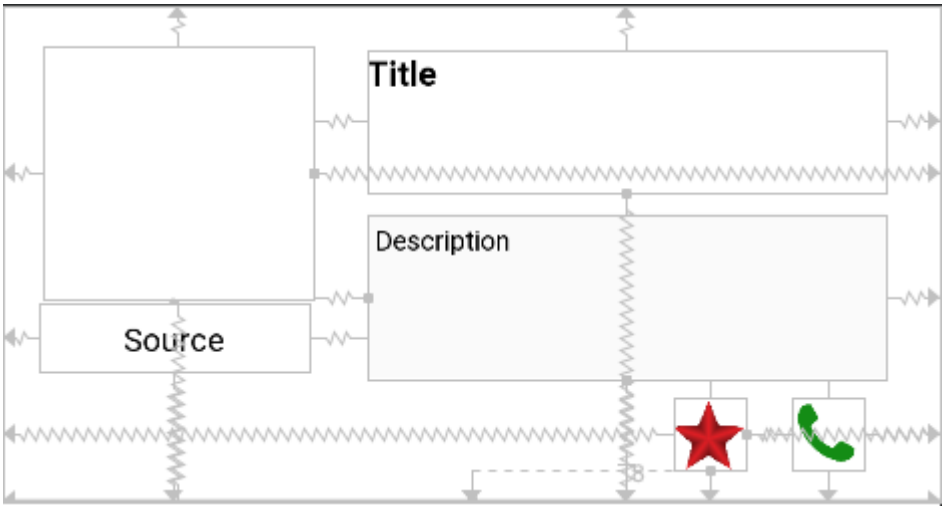
In order to define a view's position in ConstraintLayout, it is required to add at least one horizontal and one vertical constrain. A constraint connects a view to another view, the parent layout, or an invisible guideline. Each view must have a minimum of one constraint for each axis, but often more are necessary. [27]

The article item is very simple:

- Image View to host the story thumbnail
- TextView for Title
- TextView for Description

- TextView for Source
- Image View for Star Icon
- Image View for Phone Call

The design is shown below:



Article item

However, the Star Icon on the Rebtel Favorites Articles Page activity appears different, as its behaviour will be the opposite of the one in News Feed page (removing a favourite).

In fact, while there is only one item View Type, I have created two different kinds of ViewHolder. The first one, called ArticlesViewHolder, displays exactly the item from the picture above. The second one, FavoriteArticlesViewHolder, displays the same item, but the star icon is changed programmatically.

The adapter class receives the Context, the articles list and an additional parameter that plays the role of a flag. Based on this Tag attribute, whose String value represents the fragment from which the adapter is set, it will be easier for the adapter to recognize which type of View Holder it must create and display.

The code for achieving this setting is available below:

SETTING THE ADAPTER FROM REBTEL NEWS FRAGMENT

```
NewsAdapter adapter = new NewsAdapter(getContext(),newsArticles, newsDatabase, appExecutors, "newsFeed");
```

ADAPTER CONSTRUCTOR

```
public NewsAdapter (Context context, List<NewsArticle> articles, NewsDatabase database, AppExecutors appExecutors,
String extraOrigination) {
  this.mContext = context;
  this.articlesList = articles;
  this.newsDatabase = database;
  this.mAppExecutors = appExecutors;
  this.extraOrigination = extraOrigination;
}
```

ONBINDVIEWHOLDER

```
@Override
public void onBindViewHolder(@NonNull RecyclerView.ViewHolder viewHolder, int i) {

  if (viewHolder instanceof ArticleViewHolder){
    handleArticleViewHolder((ArticleViewHolder) viewHolder, i);
  } else if (viewHolder instanceof FavoriteArticlesViewHolder) {
    handleFavoriteArticleViewHolder((FavoriteArticlesViewHolder)viewHolder,i);
  }

}
```

4.2.4 News API

News fetching has been handled by using an online News API.

After checking several online options, my first choice fell on NEWSAPI.org.

"News API is a simple HTTP REST API for searching and retrieving live articles from all over the web." [28]

With this API it is possible to search for stories using a combination of these criteria:

- Keyword or phrase.
- Date published.
- Source name.
- Source domain name.

- Language.

[28]

The results can be further sorted by:

- Date published.
- Relevancy to search keyword.
- Popularity of the source.

[28]

Authentication for the API is handled with a simple API Key, which is available for free for all non-commercial projects (including open-source) and in-development commercial projects. [28]

However, the developer plan presents some limits: the content of the news is truncated to 260 characters, only 500 network requests can be sent per day, and new articles are available with a 15 minutes delay.

News API has 2 main entry points:

- "Top headlines: returns breaking news headlines for a country and category.
- Everything: returns every recent blog article published by over 30,000 different sources large and small. This endpoint is better suited for news analysis and article discovery, but can be used to retrieve news for display as well."

[28]

Due to the purpose of the Rebtel News Feed, I have adopted the Top headlines entry point for fetching articles.

Although client libraries for several languages like Node.js, Ruby and Python are available, the SDK for Java is still not implemented. The next sections will explain more in details the various steps for implementing the fetching of the articles.

4.2.5 HTTP Request and JSON

Since Java Client library is not yet available for NewsApi.org, it has been necessary to establish a HttpUrl connection to fetch the articles.

This can be easily achieved through the HttpURLConnection class, which represents a URLConnection with support for HTTP-specific features.

The steps required for a correct use of the class are summarized below:

1. Obtain a new HttpURLConnection by calling the method URL.openConnection(). The result will be casted to HttpURLConnection.
2. Prepare the request. The primary property of a request is its URI.
3. Read the response. Response headers typically include metadata such as the response body's content type and length, modified dates and session cookies. The response body can be read from the stream returned by URLConnection.getInputStream(). If the response has no body, the method returns an empty stream.
4. Disconnect. Once the response body has been read, the HttpURLConnection should be closed by calling disconnect(). All the resources held by connection will be released so they may be reused or finally closed.

[29]

The response returned from the News API is in the form a JSON file.

“JSON (**JavaScript Object Notation)** is a lightweight format which stores and transmits data objects, often used for communication between a server and a web page”.

JSON data is written as name/value pairs.

JSON's basic data types are: Number, String, Boolean, Array, Object and null.

“JSON arrays are written inside square brackets, while objects are written inside curly braces”. [30]

Below a simple example of a JSON file returned once a request is sent to News API:

```
{
  "status": "ok",
  "totalResults": 38,
  -
  "articles": [
    -
    {
      -
      "source": {
        "id": "cnn",
        "name": "CNN"
      },
      "author": "Analysis by Stephen Collinson, CNN",
      "title": "Trump offers deals where only he can win - CNN",
      "description": "President Donald Trump's ambitious plans to fix the immigration system, forge Middle East peace and coax Iran to the table might work -- but only if his negotiating partners agree to utter capitulation.",
      "url": "https://www.cnn.com/2019/05/17/politics/donald-trump-immigration-middle-east-iran/index.html",
      "urlToImage": "https://cdn.cnn.com/cnnnext/dam/assets/151222135452-donald-trump-flag-december-16-2015-super-tease.jpg",
      "publishedAt": "2019-05-17T11:06:00Z",
      "content": "Washington (CNN)President Donald Trump's ambitious plans to fix the immigration system, forge Middle East peace and coax Iran to the table might work -- but only if his negotiating partners agree to utter capitulation.\r\nThe President's new immigration bluepri… [+6882 chars]"
    },
    -
    {
      -
      "source": {
        "id": "fox-news",
        "name": "Fox News"
      },
      "author": "Paulina Dedaj",
      "title": "Body of West Virginia teen, 15, found in mountain area; mother's boyfriend arrested - Fox News",
      "description": "A West Virginia man was arrested in connection to the death of 15-year-old Riley Crossman after police discovered her “decomposed body” in a rural mountain area Thursday morning.",
      "url": "https://www.foxnews.com/us/west-virginia-man-arrested-riley-crossman",
      "urlToImage": "https://static.foxnews.com/foxnews.com/content/uploads/2019/05/Riley-Crossman.jpg",
      "publishedAt": "2019-05-17T11:00:49Z",
      "content": "A West Virginia man was arrested in connection to the death of 15-year-old Riley Crossman after police discovered her “decomposed body” in a rural mountain area
```

Thursday morning.\r\nAndy J. McCauley Jr., 46, was arrested shortly after search teams found the rem… [+1302 chars]"

}

]

}

*Sample from NEWS API Documentation.

If the application attempts to make a Network Request from its main thread, an "On Main Thread Exception" will occur. In fact, requesting network access from the `UI thread` would disrupt the user experience due to often expensive operations.

An Async Task or an Intent Service will fix the issue. The reasons which brought to a particular choice between the two components will be described in the next section.

The Http request could be also handled by adopting the Retrofit client, but I decided not to adopt external libraries for this project.

4.2.6 Room, MVVM Pattern, Live Data, Repository and Network Source.

The News Feed has been implemented following the MVVM Pattern logic.

RebtelNewsFragment and FavoriteArticlesFragment share the same ViewModel and associated LiveData.

The NewsRepository class handles communications between the SQLite database and a Network Data Source, which requests data from the server making use of an Intent Service.

For years Async Task has been one of the most popular and adopted Android Components for short operations to run asynchronously in the background.

While it is a very powerful class, it presents some remarkable issues regarding Activity Lifecycle. However, the choice of using an Intent Service instead of an Async Task is more related to the fact that Intent Services can be scheduled.

Usually, SQLite databases require the use of APIs like SQLiteOpenHelper, SQLiteDatabase and SQLiteQueryBuilder.

"These, although largely adopted and popular, reveal a lot of development challenges, which include a hard way to validate SQLite statements at compile time". [31]

So, for the Rebtel News feed, the database that stores the latest fetched articles has been built by using the new SQLite object mapping library, Room.

Room includes a lot of benefits:

- It can map database objects to Java objects. This result in ContentValues or Cursors being not necessary anymore and in less boilerplate code compared to the built-in APIs;
- It catches incorrect SQL statements at compile time, not at runtime;
- It allows for data observation via LiveData and RxJava (which will be explained later).

[31]

It has three main components:

- @Entity: it defines the schema of the database table. An Entity is created by converting a Model Object.
- @DAO: it represents a class or interface as a Data Access Object (DAO). DAOs define the methods useful to access a database and provide an API for reading and writing database data.
- "@Database: it represents the database holder. This class usually defines a list of entities for the database and the data access objects (DAOs) for the database".

[31]

Room makes use annotations to define the table structure. [31]

Entities must be compliant with the following requirements:

- They must be correctly associated with the annotation @Entity.
- One of the fields, the primary key, must be labeled with the @PrimaryKey annotation.
- Room needs access to all of the fields. In order to achieve this, fields should be public or getters and setters should be provided.
- Only one constructor should be exposed to Room: Room cannot compile an entity with two constructors because it doesn't know which one to use. In order to hide a constructor from Room, @Ignore annotation can be used.

- If some of the fields will not be stored in the database, @Ignore annotation can be used as well to hide them from Room.

[31]

An example of entity is shown below:

```
@Entity(tableName = "articlesDatabase")
public class NewsArticle {

  @PrimaryKey(autoGenerate = true)
  private int id;

  private String imageUri;
  private String title;
  private String content;
  private String source;
  private String author;
  private String description;
  private int favorite;
  private int lastHour;
  private String url;



  public NewsArticle(int id, String imageUri, String title, String content, String source,
          String author, String description, int favorite, int lastHour, String url) {
    this.id = id;
    this.imageUri = imageUri;
    this.title = title;
    this.content = content;
    this.source = source;
    this.author = author;
    this.description = description;
    this.favorite = favorite;
    this.lastHour = lastHour;
    this.url = url;
  }



  @Ignore
  public NewsArticle() {

  }
```

In the example code, which was the one used in the specific News Feed case, the Database table is called “ArticleDatabase”. The “id” key, which is the primary key, is autogenerated.

The “@Ignore” annotation tells Room to ignore the constructor which is used to instantiate the Builder class.

“Next component in line is DAO (Database Access Object). DAOs are either abstract classes or interfaces that define the read and write actions for the database data”. [31]

“The only thing a DAO needs is the **@Dao** annotation. Methods are annotated with **@Insert, @Delete, @Update** and **@Query**. @Insert, @Delete and @Update are **convenience annotations** that create methods which do as their name implies”. [31]

@Query instead allows the developer to write SQLite to create custom read/write database operations.

Below are listed the methods for the ArticlesDAO:

```
@Dao
public interface ArticlesDao {

  @Query("SELECT * FROM articlesDatabase WHERE lastHour = 1")
  LiveData<List<NewsArticle>> getCurrentArticles();

  @Insert(onConflict = OnConflictStrategy.REPLACE)
  void bulkInsert(List<NewsArticle> articles);

  @Query("DELETE FROM articlesDatabase WHERE lastHour = 1")
  void deleteOldArticles();

  @Query("UPDATE articlesDatabase SET lastHour = 0 WHERE favorite = 1")
  void updateFavoritesHourFlag();

  @Query("UPDATE articlesDatabase SET favorite = 1 WHERE id = :userId")
  void updatePost(int userId);

  @Query("SELECT * FROM articlesDatabase WHERE favorite = 1")
  LiveData<List<NewsArticle>> getFavoriteArticles();

  @Query("UPDATE articlesDatabase SET favorite = 0 WHERE id = :userId")
  void removeFavorite(int userId);

}
```

The first query returns the list of the last fetched articles that are currently stored in the database.

The method “bulkInsert” inserts any number of ListArticle objects. As the app receives lists of news article entries from the server, it uses bulkInsert to put them into the database. For “bulkInsert”, an additional annotation “OnConflictStrategy.REPLACE" can be used for replacing old articles fetched with new ones when a new network requests is sent to the server.

UpdateFavoritesHourFlag() sets the “lastHour” flag of all the previous fetched articles which were not flagged as favorites by the user to 0 before sending a new network request, so that the DeleteOldArticles() method can delete them.

Other methods listed allow the user to set an article as favorite or to remove a story from the favorites list.

With Entity and DAO in place, the last component of the Room Database is the Database itself, which returns a DAO object.

The simple code for implementing the Database class is shown below:

```
@Database(entities = {NewsArticle.class}, version = 1 )
public abstract class NewsDatabase extends RoomDatabase {

  private static final String DATABASE_NAME = "articlesDatabase";

  private static final Object LOCK = new Object();
  private static NewsDatabase sInstance;

  public static NewsDatabase getInstance(Context context) {
    if (sInstance == null) {
      synchronized (LOCK) {
        sInstance = Room.databaseBuilder(context.getApplicationContext(),
            NewsDatabase.class, NewsDatabase.DATABASE_NAME).build();
      }
    }
    return sInstance;
  }

  public abstract ArticlesDao ArticlesDao();
}
```

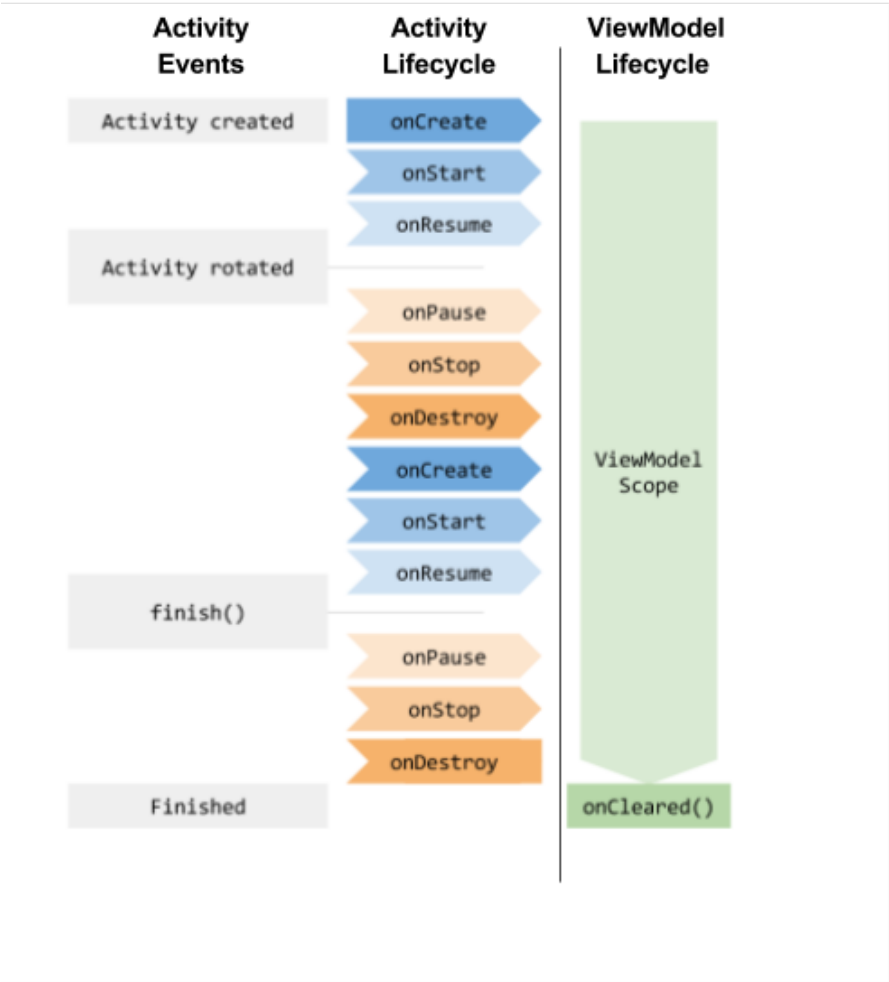
View Model is the main component of the MVVM Pattern.

The ViewModel class is designed to hold and manage UI-related data in a life-cycle conscious way. In this way data will survive configuration changes such as screen rotations. By adopting this class, a separation of responsibilities take place: ViewModel deals with providing, manipulating and storing UI state while UI Controller handles the display of the state. [31]

ViewModel is always associated with components with a LifecycleOwner (fragments or activities). "By providing a LifecycleOwner, a connection between the ViewModel and the LifecycleOwner is established."

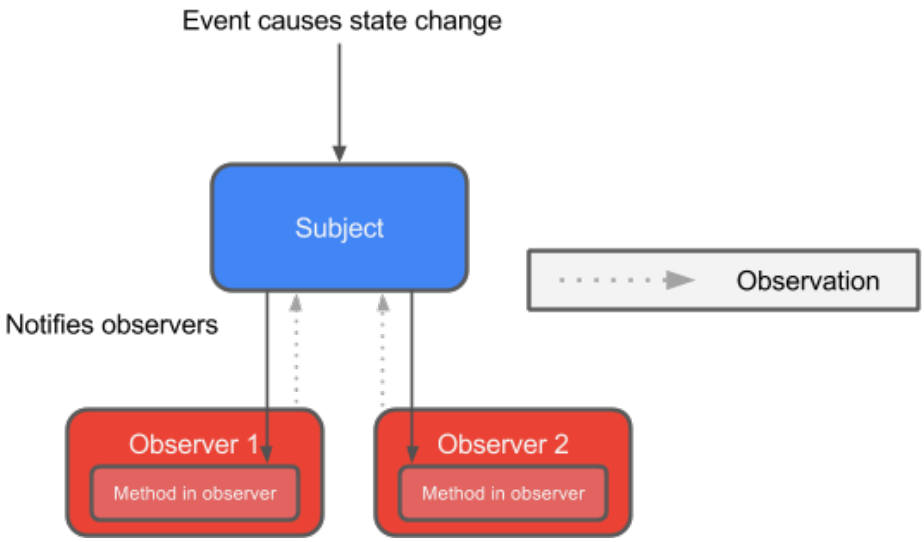
ViewModels do not share the same lifecycle behaviour with their associated UI Controllers. While the latter are destroyed and recreated on configuration changes, ViewModels are not. [31]

Below is a diagram showing how different is the **lifecycle of a ViewModel** compared to the **lifecycle of the activity**, when the activity is created, rotated and finished. [31]



ViewModels are often associated to the concept of LiveData objects.

"LiveData is a data holder class that is lifecycle aware. It keeps a value and allows this value to be observed by a list of associated objects, called observers. This is what is called observer pattern. The object observed, called subject, notifies all his observers whenever its state changes, usually by calling one of their methods." [31]



In the case of LiveData, the subject is characterized by the LiveData itself and the observers are objects which represent subclasses of the Observer class. Every time the methods postValue() or setValue() are called the subject's state changes, and the active observers will be triggered. [31]

"LiveData keeps a list of associated observers and LifecycleOwners. In general Observers are only considered **active** when their associated LifecycleOwner is on screen. This means it is in the STARTED or RESUMED state. The fact that LiveData keeps track of LifecycleOwners is why LiveData is called **lifecycle aware"**. [31]

Below an example of how the LiveData was used in the specific Rebtel News code:

RebtelNewsFragment

```
mViewModel.getArticles().observe(getViewLifecycleOwner(), newsArticles -> {
  NewsAdapter adapter = new NewsAdapter(getContext(),newsArticles, newsDatabase, appExecutors, "newsFeed");
  articlesRecycler.setAdapter(adapter);
});
```

View Model

```
public class ArticlesViewModel extends ViewModel {

  private final LiveData<List<NewsArticle>> newsArticlesList;
  private final LiveData<List<NewsArticle>> favoriteArticles;

  // Date for the weather forecast
  private final NewsRepository mRepository;

  public ArticlesViewModel(NewsRepository repository) {
    mRepository = repository;
    newsArticlesList = mRepository.getCurrentArticles();
    favoriteArticles = mRepository.getFavoriteArticles();
  }

  public LiveData<List<NewsArticle>> getArticles() {
    return newsArticlesList;
  }

  public LiveData<List<NewsArticle>> getFavoriteArticles() { return favoriteArticles; }
}
```

The activity or fragment observes a MutableLiveData<> object, which holds a NewsArticle. When the PostValue() is called and the object is updated, the observers are notified.

As shown in the code above, the ArticlesViewModel needs a reference to a NewsRepository. However, the default constructor that is automatically called by ViewModelProvider when instantiating a new ViewModel takes no argument. This small issue can be easily overcome by providing a V**iew Model Provider Factory,** as suggested below:

```
public class ArticlesViewModelFactory extends ViewModelProvider.NewInstanceFactory {

  private final NewsRepository mRepository;

  public ArticlesViewModelFactory(NewsRepository repository) {
    this.mRepository = repository;
  }

  @Override
  public <T extends ViewModel> T create(Class<T> modelClass) {
    return (T) new ArticlesViewModel(mRepository);
  }
}
```

The Repository plays the role of exposing the network and database data to UI.

In this specific case, NewsRepository delegates all the operations to the ArticlesDAO and NetworkDataSource. In fact, it observes LiveData from the NetworkDataSource in order to perform a database update as soon as it finishes fetching new data.

The NetworkDataSource handles all network operations and provides the most recently downloaded network data. This is achieved through a MutableLiveData object that is updated anytime a new network request is sent to the server.

The NetworkDataSource is able to perform network requests using an IntentService.

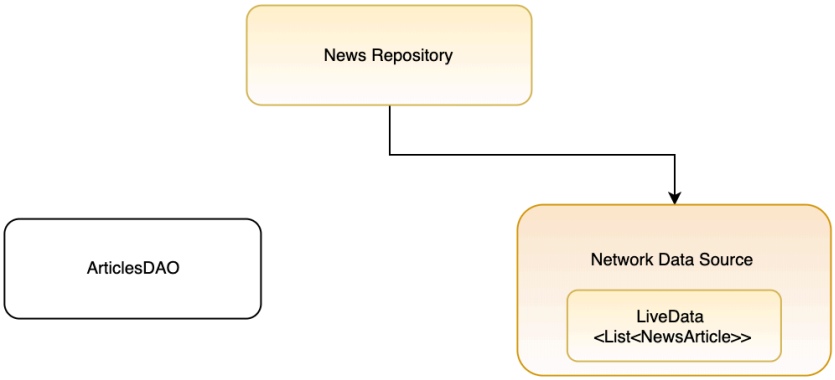
“IntentService is a base class for Services that handles asynchronous requests (expressed as intent calls). The service is started as needed, handles each Intent in turn using a worker thread, and stops itself when its work is done”. [32]

Intent Services use the so called "work queue processor" pattern in order to offload tasks from an application's main thread. [32] This purpose can be achieved by extending IntentService and implementing the onHandleIntent() method. IntentService will receive the Intents, launch a worker thread, and stop the service as appropriate.

All requests are handled on a single worker thread -- they may take as long as necessary (and will not block the application's main loop), but only one request will be processed at a time.

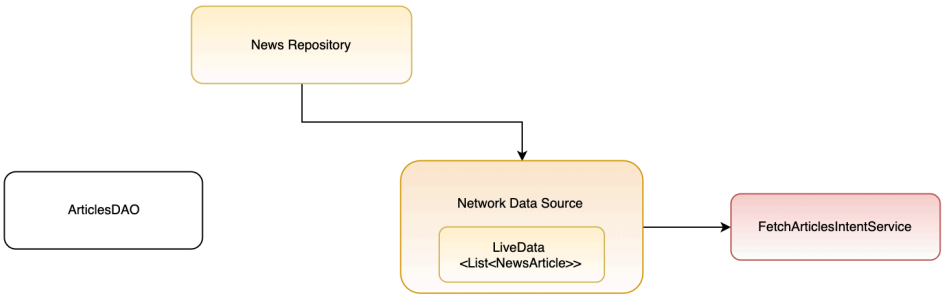
The fetching and storing flow for RebtelNewsFeed is shown below:

1. NewsRepository observes LiveData from NetworkDataSource.



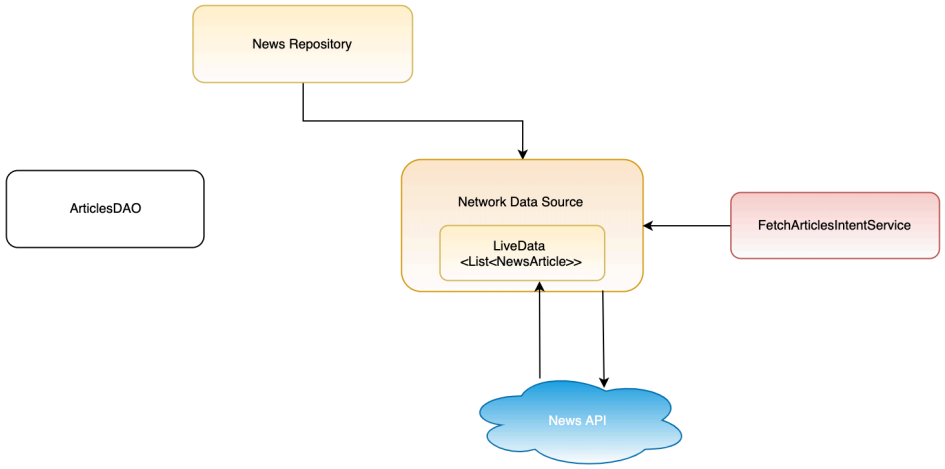

```
private NewsRepository(Context context, ArticlesDao mArticlesDao, NetworkDataSource mNetworkDataSource,
AppExecutors executors) {
  mContext = context;
  articlesDao = mArticlesDao;
  networkDataSource = mNetworkDataSource;
  mExecutors = executors;
  initializeArticlesData();
  LiveData<List<NewsArticle>> networkData = networkDataSource.getArticles();
  networkData.observeForever(newForecastsFromNetwork -> {

}
```

2. NetworkDataSource creates and immediately starts NewsArticleIntentService.



```
    public void startFetchArticlesService() {
      Intent intentToFetch = new Intent(context, NewsArticlesIntentService.class);
      context.startService(intentToFetch);
    }
```

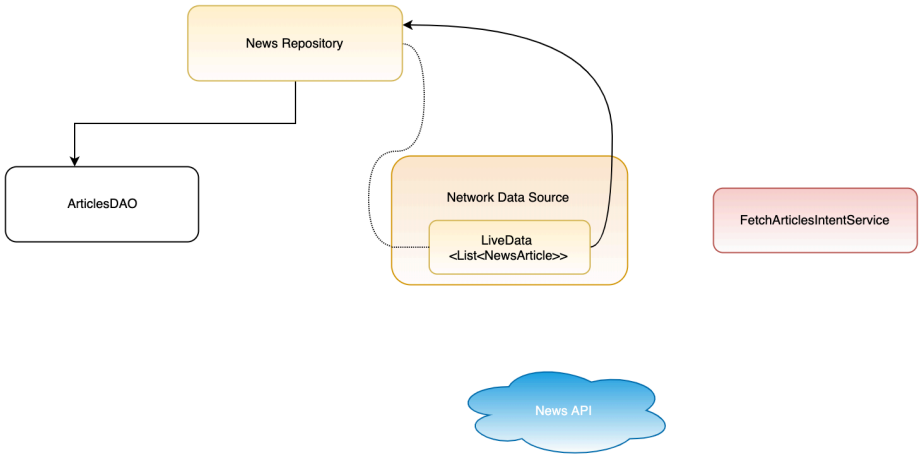
3. IntentService first retrieves an instance of NetworkDataSource, which performs the actual fetch. Once finished, it updates the MutableLiveData objects delegated to store the most recently downloaded data.




```
public class NewsArticlesIntentService extends IntentService {

  public NewsArticlesIntentService() {
    super("NewsIntentService");
  }

  @Override
  protected void onHandleIntent(Intent intent) {
    NetworkDataSource networkDataSource = InjectorUtils.provideNetworkDataSource(this.getApplicationContext());
    networkDataSource.fetchArticles();

  }
}


void fetchArticles() {
  executors.networkIO().execute(() -> {
    try {

      URL articlesRequestUrl = NetworkNewsUtils.buildURL(ClientPreferences.getTopCountryCode(context),
ContactAPI.getInstance(context).getTopCountriesCodes(2));
```

```
        // Use the URL to retrieve the JSON
        String jsonArticleResponse = NetworkNewsUtils.getResponseFromHttpUrl(articlesRequestUrl);


        NewsAPIResponse response = new OpenArticlesJSonParser().parse(jsonArticleResponse);

        if (response != null && response.getNewsArticles().size() != 0) {

          // When you are off of the main thread and want to update LiveData, use postValue.
          // It posts the update to the main thread.
          mDownloadedNewsArticles.postValue(response.getNewsArticles());

          // If the code reaches this point, we have successfully performed our sync
        }
        } catch (Exception e) {
        // Server probably invalid
        Log.d(TAG, "Exception occurred. ");
      }
    });
}
```

4. NewsRepository, triggered by the PostValue(), will update the database and return the LiveData object to the ViewModel.




```
        networkData.observeForever(newForecastsFromNetwork -> {
            mExecutors.diskIO().execute(() -> {

                updateFavoritesFlag();
```

```
        deleteOldArticles();

        articlesDao.bulkInsert(newForecastsFromNetwork);
      });
    });


}

private void updateFavoritesFlag() {
  articlesDao.updateFavoritesHourFlag();
}

public synchronized static NewsRepository getInstance(Context context, ArticlesDao articlesDao,
NetworkDataSource networkDataSource,
                            AppExecutors executors) {
  if (sInstance == null) {
    synchronized (LOCK) {
      sInstance = new NewsRepository(context, articlesDao, networkDataSource,
          executors);
    }
  }
  return sInstance;
}

private void deleteOldArticles() {
  articlesDao.deleteOldArticles();
}

/**
 * Creates periodic sync tasks and checks to see if an immediate sync is required. If an
 * immediate sync is required, this method will take care of making surCe that sync occurs.
 */

/**
 * Database related operations
 **/

public LiveData<List<NewsArticle>> getCurrentArticles() {
  return articlesDao.getCurrentArticles();
}
```

5. Finally, the ViewModel returns the articles List to the view, which will update the UI accordingly.

```
public ArticlesViewModel(NewsRepository repository) {
  mRepository = repository;
  newsArticlesList = mRepository.getCurrentArticles();
  favoriteArticles = mRepository.getFavoriteArticles();
}
```

```
public LiveData<List<NewsArticle>> getArticles() {
  return newsArticlesList;
}

public LiveData<List<NewsArticle>> getFavoriteArticles() { return favoriteArticles; }
```

Rebtel News Fragment

```
public void onViewCreated(View view, Bundle savedInstanceState) {
  super.onViewCreated(view, savedInstanceState);
  newsDatabase = InjectorUtils.provideNewsDatabase(getActivity());
  ArticlesViewModelFactory factory = InjectorUtils.provideArticlesViewModelFactory(getActivity());
  mViewModel = ViewModelProviders.of(this, factory).get(ArticlesViewModel.class);
  AppExecutors appExecutors = InjectorUtils.provideAppExecutors(getActivity());

  articlesRecycler.setLayoutManager(new LinearLayoutManager(getContext()));

  mViewModel.getArticles().observe(getViewLifecycleOwner(), newsArticles -> {
    NewsAdapter adapter = new NewsAdapter(getContext(),newsArticles, newsDatabase, appExecutors,
"newsFeed");
    articlesRecycler.setAdapter(adapter);
  });

  swipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {
    @Override
    public void onRefresh() {
      mViewModel.getArticles().observe(getViewLifecycleOwner(),newsArticles -> {
        NewsAdapter adapter = new NewsAdapter(getContext(),newsArticles, newsDatabase, appExecutors,
"newsFeed");
        articlesRecycler.setAdapter(adapter);
      });

      if (swipeRefreshLayout.isRefreshing()) {
        swipeRefreshLayout.setRefreshing(false);
      }
    }
  });
}
```

Since Repository is not associated with any Lifecycle owner, it calls the observeForever() method on the NetworkDataSource, which is very similar to observe() but it is considered always active. [31]

Repository, Database, NetworkSource are Singleton classes: this means only one Instance running for these classes is needed.

Creating a Singleton requires a static variable of the class and a Lock object to ensure thread safety. Then the getInstance() method returns an instance of the class if it exists or creates a new one, as shown below:

```
private static final Object LOCK = new Object();
private static volatile NewsDatabase sInstance;

public static NewsDatabase getInstance(Context context) {
  if (sInstance == null) {
    synchronized (LOCK) {
      sInstance = Room.databaseBuilder(context.getApplicationContext(),
          NewsDatabase.class, NewsDatabase.DATABASE_NAME).build();
    }
  }
  return sInstance;
}
```

Network fetching could happen every time the user updates or refreshes the News main page, but it might be highly under performing and useless, considering that News API updates the headlines each 15 minutes (at least for the Developer plan).

This is why for this purpose I have decided to schedule the fetching service through a Job Scheduler.

"Job Scheduler is an API for scheduling various types of jobs against the framework that will be executed in the application's own process". [33]

"A JobInfo object is passed to the JobScheduler using the schedule() method. When the criteria declared are met, the system will execute this job on the application's JobService. The service component that implements the logic for the job is identified when constructing the JobInfo using JobInfo.Builder.JobInfo.Builder(int, android.content.ComponentName)". [33]

"The framework will choose the appropriate time to execute jobs, and attempt to batch and delay them as much as possible. It can be run at any moment depending on the current state of the JobScheduler's internal queue". [33]

"While a job is running, a wakelock[8] is hold by the system on behalf of the app. For this reason, no action is required to guarantee that the device stays awake for the duration of the job". [33]

The scheduling follows this simple flow:

1. If it is a fresh install and the first time the user wants to access the news feed, the articles are fetched and stored inside the database. A flag is updated accordingly.
2. After the first time, the repository will schedule a job, which will be executed every 20 minutes and it will update the database.

[8] Wakelock: A wake lock is a mechanism to indicate that an application needs to have the device stay on.

3. After the second time, since the job has been already scheduled, the repository will always return by default the list of articles currently available in the database, and fetching and updating will happen periodically.

```
private void initializeArticlesData() {

  if (isFirstTimeFetching()) {

    startFetchArticles();

    ClientPreferences.setTimeStampLastNewsRequest(mContext, 1);

  } else {

    if (JobSchedulerUtils.isJobScheduled(mContext,JobId.DOWNLOAD_ARTICLES_JOB_ID)) return;

    networkDataSource.scheduleArticlesFetching();


  }

}

private Boolean isFirstTimeFetching() {

  long lastRequest = ClientPreferences.getTimeStampLastNewsRequest(mContext);

  return lastRequest == -1;

}

/**
 * Deletes old weather data because we don't need to keep multiple days' data
 */

/**
 * Checks if there are enough days of future weather for the app to display all the needed data.
 *
 * @return a fetch is needed
 */

/**
 * Network related operation
 */

private void startFetchArticles() {
  networkDataSource.startFetchArticlesService();
}
```

In the specific case, it is necessary to create a persistent and periodic JobInfo object, which requires any type of Network.

Below the code for scheduling the job:

```
public void scheduleArticlesFetching() {

  JobScheduler jobScheduler = (JobScheduler) context.getSystemService(Context.JOB_SCHEDULER_SERVICE);
  ComponentName componentName = new ComponentName(context, FetchingJobService.class);
  JobInfo jobInfo;
  if (Build.VERSION.SDK_INT < Build.VERSION_CODES.N) {
    jobInfo = new JobInfo.Builder(JobId.DOWNLOAD_ARTICLES_JOB_ID, componentName)
        .setMinimumLatency(3600000)
        .setPersisted(true)
        .setRequiredNetworkType(JobInfo.NETWORK_TYPE_ANY)
        .build();
  } else {
    jobInfo = new JobInfo.Builder(JobId.DOWNLOAD_ARTICLES_JOB_ID, componentName)
        .setPeriodic(3600000)
        .setPersisted(true)
        .setRequiredNetworkType(JobInfo.NETWORK_TYPE_ANY)
          .build();
  }
  jobScheduler.schedule(jobInfo);

}
```

4.2.7 News Feed Calls

One of the main reasons for introducing the news feed was to allow users to call their relatives or friends after reading an interesting story from their top countries. In order to achieve this I have connected the Rebtel News Feed to the ContactBookSearch Fragment that already implemented the calling logic in the App by setting an OnClickListener for the Phone Icon of each article item.

ContactBookSearch activity loads all contacts from the user's address book, but in the specific case only the contacts from the designed country should be displayed.

This is why the following method has been introduced:

```
public static void startContactBookSearchFromNewsFeed(Context context, int tabPosition, String origination) {
  Intent intent = new Intent(context, ContactBookSearchActivity.class);
  intent.putExtra(EXTRA_TAB_POSITION, tabPosition);
  intent.putExtra(ORIGINATION,origination);
  context.startActivity(intent);
}
```

The extra with tag “ORIGINATION” is only present in the intent bundle object when ContatBookSearch activity is started from the News Feed fragment.

Inside the ContactSearchFragment, a method checks if the intent Bundle contains key “ORIGINATION”. Then the user’s top country is returned by executing the code below:

```
private String getUserTopCountry() {
  String topCountryCode = ClientPreferences.getTopCountryCode(getContext());
  if ("".equals(topCountryCode)) {
    String topAddressCountryCode = ContactAPI.getInstance(getContext()).getTopCountriesCodes(2).get(0);
    return topAddressCountryCode;
  } else{
    return topCountryCode;
  }
}
```

At this point, by using Rebtel’s ContactAPI it is possible to fetch all the contacts for a specific country and display them on the screen.

4.2.8 Managing Pages Communications

The last part of the implementation focuses on the communication between the different pages (Living Room, Account View, Rebtel News Feed).

Currently, Rebtel app manages the switching of Living Room and Account View through a floating action button at the right bottom of the screen. It also handles enter and exit animations of the pages.

Rebtel News has been made accessible only from the Living Room page, by using a Floating Action Button at the left bottom of the screen, at the same height of the one used for opening the Account View.

Some changes in the logic for the animations were implemented as well, in order to prevent unnatural behaviours when switching from a page to another.

```
LIVING_ROOM(LivingRoomFragment.class, R.anim.enter_from_left, R.anim.exit_to_right, R.drawable.ic_settings),
ACCOUNT(AccountViewFragment.class, R.anim.enter_from_right, R.anim.exit_to_left, R.drawable.ic_home),
REBTEL_NEWS (RebtelNewsFragment.class,R.anim.enter_from_left,R.anim.exit_to_right,R.drawable.ic_home);
```

```
public void setEnterAnim() {
  this.enterAnim=R.anim.enter_from_right;
  this.exitAnim = R.anim.exit_to_left;
```

```
}

public void unsetEnterAnim() {
  this.enterAnim = R.anim.enter_from_left;
  this.exitAnim = R.anim.exit_to_right;
}


@OnClick(R.id.fab)
void switchPages() {
  if (currentPage == Page.LIVING_ROOM) {
    setPage(Page.ACCOUNT,true);
    fabNews.setVisibility(View.GONE);
    TrackManager.getInstance().getGeneralFlowTracker().trackTapOneViewSwitch();
  } else if (currentPage == Page.ACCOUNT) {
    Page.LIVING_ROOM.unsetEnterAnim();
    setPage(Page.LIVING_ROOM,true);
    fabNews.setVisibility(View.VISIBLE);
    TrackManager.getInstance().getGeneralFlowTracker().trackTapOneViewSwitch();
  } else if (currentPage ==  Page.REBTEL_NEWS) {
    Page.LIVING_ROOM.setEnterAnim();
    setPage(Page.LIVING_ROOM, true);
    fabNews.setVisibility(View.VISIBLE);
  }

}
```

While the default behaviour for Living Room Page was to enter from the left and exit to right, if the current page is Rebtel News (which similarly to Living Room enters from left and goes out to the right), the Living Room will now enter from right and exit from the left. The current setting is reset once the page is switched to Account View page.

CHAPTER 5

5. CONCLUSIONS

5.1 Rebtel News Feed: Future Improvements

The Rebtel News Feed might be implemented in the next releases of Rebtel or not, but there is for sure room for improvements.

First of all, since this was an exploration I made without any support from the Design Team, the graphical interface could be totally redesigned to improve the user experience.

News API, beside top headlines articles, offers a lot of other options. One idea to improve it might be to take track of the user behaviour. Storing the keywords contained in the stories with which users interact the most might be very useful in the future. In fact, Rebtel could build a suggestion algorithm to provide the customer recommended news and topics of his/her interest.

Allowing users to select some topics and display the latest news about them might be another thing to take in consideration as well.

At the current stage of the feature, the News Feed only shows headlines from one country. One improvement might be displaying stories from different countries by using a ViewPager and a TabLayout; in this way, switching from one country to another would be easy and intuitive.

Beside the News Feed, one idea that came to my mind was a Weather forecast activity, which would have shown the weather predictions for the user's top country, and maybe alert him/her if anything relevant (storms, hurricanes etc.) was currently happening.

Although the messaging feature has been removed by Rebtel in the latest releases, maybe sharing News Feed articles inside the app through a chat could have been really powerful.

5.2 Incognito Mode: Future Improvements

At the current stage, the Incognito Feature has solved the most urgent issues related to the first roll out.

However, before reaching a final idea for the implementation, several meetings with the designers and product managers brought to different ideas, which have been currently discarded for lack of business case built on them but might have a shot in the near future.

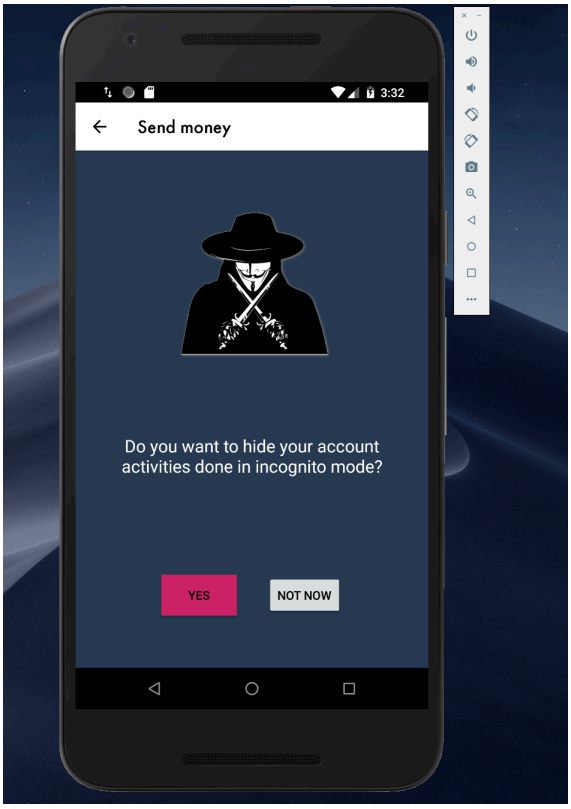
The most outstanding idea for Incognito Mode was, beside changing the theme and efficiently alerting users about their current calling mode, to make account activities and monthly recaps private.

In order to understand the reason, let´s have a look at which informations the Monthly Recap shows:

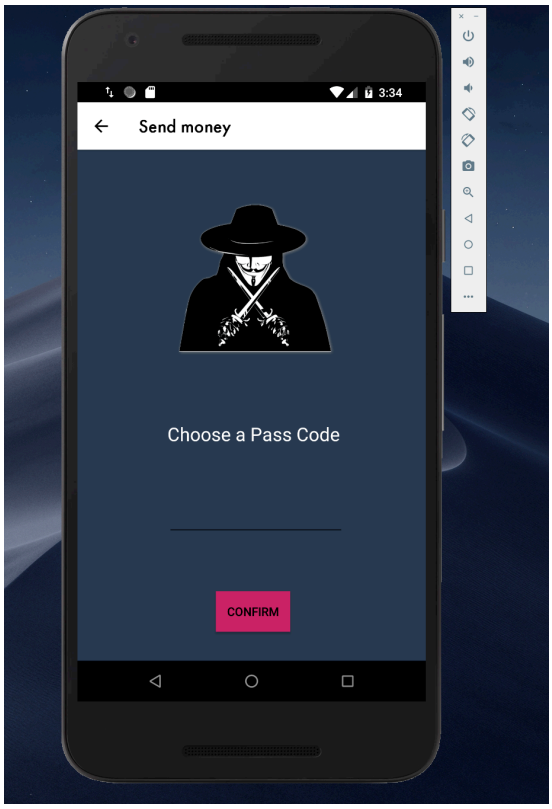
Some users may want to keep some of these infos to themselves; my initial idea was to ask the user whether he wanted or not to choose a simple Passcode the first time he/she enabled the Incognito Mode. If the user opted for a Passcode, access to Monthly Recap and

Account Activity with Incognito Mode set would have required to enter the password picked.

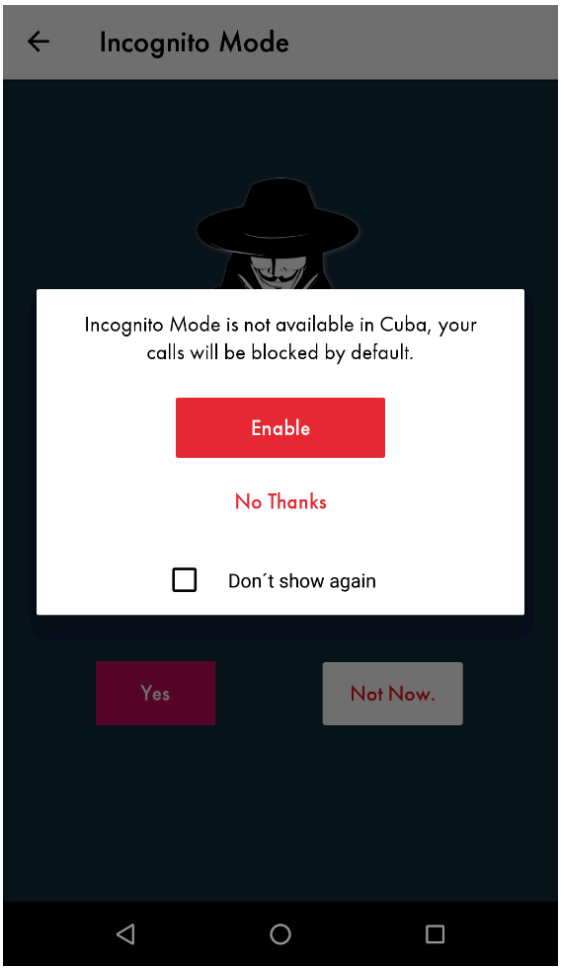
This idea was initially well welcomed by product managers and designers, and I started developing it.



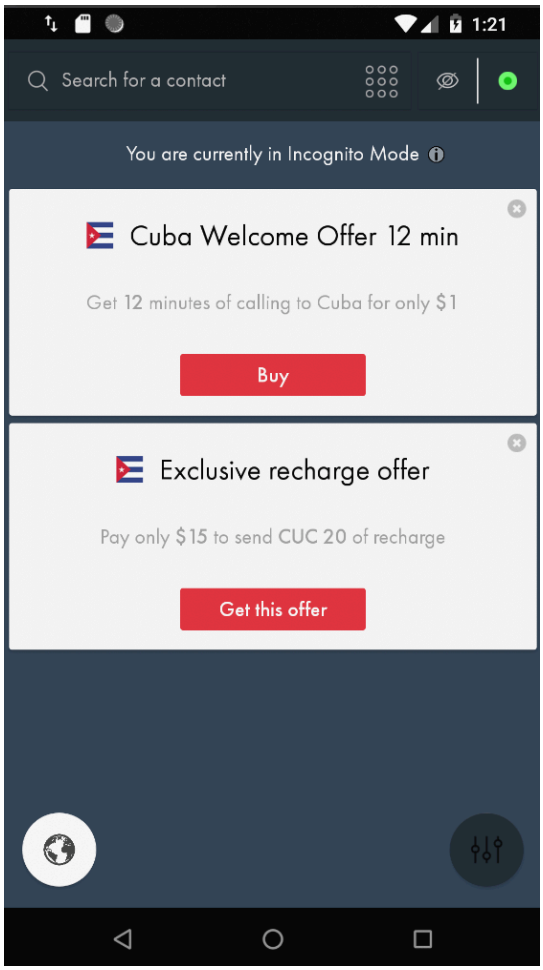
First time Enabling Incognito Mode



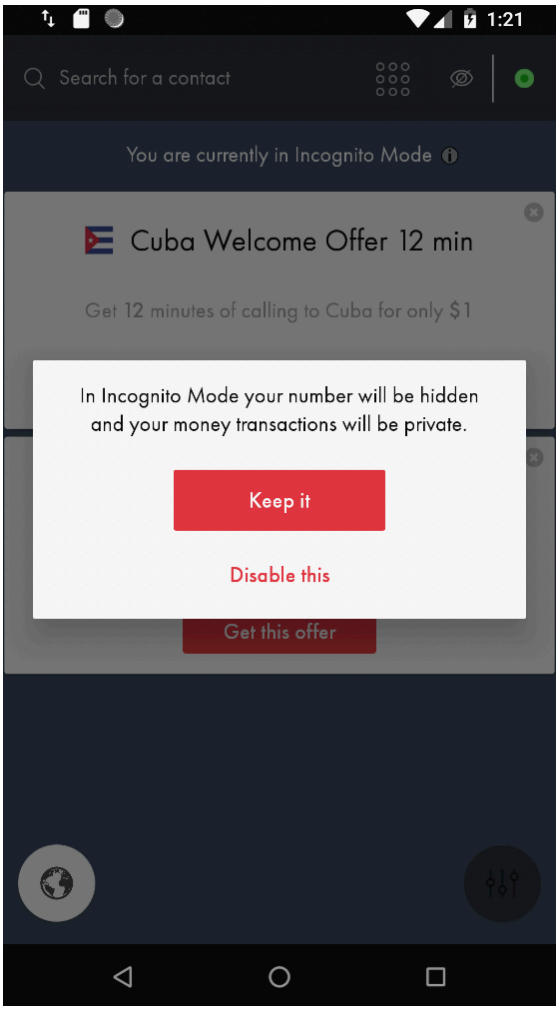
User decides to enable Passcode Verification



User is warned Incognito Mode is not available in Cuba, if his/her top country is Cuba.



Incognito Mode Indicator in Living Room



Incognito Mode Dialog with informations

The main difference between Regular Mode and Incognito mode would have been that the Account Activity in regular mode would have displayed only calls and transaction made in regular mode while, when accessing Account Activity in Incognito Mode, the user would have seen all his/her activities with no restrictions (This task would have required Rebtel BackEnd Team support). Monthly Recap instead would have been the same, but by enabling Incognito Mode access to the feature would have required the chosen code.

Unfortunately, while a Business case was already built for the final idea implemented, there was not enough data about the business value these 2 sub features could eventually bring to the product.

The Backend team and the design team were both really busy with the new app Rebtel is launching and could not provide enough support for implementing the feature with the explained settings.

Besides, if we take in consideration the values promoted by Rebtel (being connected to your family and relatives), this extension of the feature might have been a contradiction with the ethic of the app. This is why my idea did not see the light of the day.

However, I think privacy is something really important nowadays, and it would be really interesting to explore these solutions in the future. One improvement to my implementation might be the fingerprint authentication instead of a simple code. This could be made easily achievable using a Biometric Prompt dialog in Android Pie, but for older version the deprecated version of FingerPrintManager allows to obtain performing results as well.

5.3 CONCLUSIONS: MY EXPERIENCE

Before starting this internship abroad I had a lot of expectations. I wanted to move for some months to a different country to improve my English communication skills, to come in touch with a different culture and to understand better how companies in other countries organize their work timeline.

Beside my thesis project, my main goals were basically 2:

- To become a better Android developer;
- To learn how to work with a real development team.

Honestly, I can say Rebtel gave me everything I was looking for.

First of all, I had the huge opportunity to work in a very international environment with people from more than 40 countries and this has been an amazing experience.

The communication between different departments happens on a daily basis and everything is handled properly by respecting deadlines very carefully, although the company also reserves some time for fun activities inside the office, aiming to create a better connection between the employees.

My company tutor, Rafael, has been really helpful since the first days and gave me tons of material to study from. He suggested me how to use different design patterns, like MVVM, MVP, Builder Pattern, Singleton and more. Today, I feel like I have learnt a lot of new stuff and new programming tricks in a very short time.

Thanks to Andy, the head of mobile and the person who gave me this huge opportunity, I observed how to manage a team and how to make everything work, even if there are issues inside a group of developers.

I am highly satisfied by the Rebtel News feature I implemented. For this one, I finally got rid of the “Spaghetti Code” style I was used to adopt whenever I had to do something on my own. I understood that coding for small and especially large teams comes with the responsibility of making your work comprehensible to other developers and other people involved in a project.

I am also happy for the other feature implementation I took part in, the Incognito Mode. Although the final result obtained is pretty simple and basic, I could directly experience the whole flow companies follow in order launch new features and I have to say that I found it more difficult than one may think, because it is fundamental to take in consideration so many aspects that I took for granted in the past.

I just wish I could have done more for the company. To be honest, I think the Internship was really useful for me and my future experiences, the amount of learnings has been noticeable so I really wanted to work on something more meaningful than a single feature.

But I also came to understand that most of the features in a successful company require a business case built already, and during my specific internship there was none available beside Incognito Mode.

6. Bibliography

[1] Sven Carlsson. “Migranter i USA sätter ny fart på Rebtel”. URL: https://digital.di.se/artikel/migranter-i-usa-satter-ny-fart-pa-rebtel

[2] Anirudh Bishnoi. “Android,Your Co-Partner”.

[3] Elgin, Ben. “Google Buys Android for Its Mobile Arsenal”. Bloomber Businessweek.

[4] Alabaster, Jay (April 16, 2013). "Android founder: We aimed to make a camera OS". PC World. International Data Group

[5] Welch, Chris (April 16, 2013). "Before it took over smartphones, Android was originally destined for cameras". The Verge. Vox Media.

[6] Manjoo, Farhad (May 27, 2015). "A Murky Road Ahead for Android, Despite Market Dominance"

[7] Block, Ryan (August 28, 2007). "Google is working on a mobile OS, and it's due out shortly". Engadget. AOL.

[8] Aamoth, Doug (September 23, 2008). "T-Mobile officially announces the G1 Android phone". TechCrunch. AOL.

[9] Gao, Richard (September 23, 2016). "Android and its first purchasable product, the T-Mobile G1, celebrate their 8th birthdays today". Android Police.

[10] Reith, Ryan. Chau, Melissa. “Smartphone Market Share”. URL:

https://www.idc.com/promo/smartphone-market-share/os

[11] Perez, Sarah. “Apple’s App Store revenue nearly double that of Google Play in first half of 2018 “ URL: https://techcrunch.com/2018/07/16/apples-app-store-revenue-nearly-double-that-of-google-play-in-first-half-of-2018/

[12] Ogbo, Obaro. “7 reasons why you should develop apps for Android rather than iOS”. URL: https://www.androidauthority.com/develop-apps-for-android-rather-than-ios-607219/

[13] Android Developers Application Fundamentals. URL: https://developer.android.com/guide/components/fundamentals.html

[14] Lardinois, Frederic. “Kotlin is now Google’s preferred language for Android app development”. URL: https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/

[15] “Tools Overview”. Android Developers

[16] “Android Studio’s Website”

[17] Ducrohet, Xavier; Norbye, Tor; Chou, Katherine. "Android Studio: An IDE built for Android"

[18] Lynch, Ryan. “5 reasons why you should use Incognito Mode for Browsing”. URL: https://www.maketecheasier.com/reasons-for-browsers-incognito-mode/

[19] Epstein, Yoni. “Where did my cookies go?”. URL: https://engineering.taboola.com/where-did-my-cookies-go/

[20] Vyas, Anshul. “MVC Pattern”. Medium. URL: https://medium.com/@anshul.vyas380/mvc-pattern-3b5366e60ce4

[21] Pandey, Bipin. “MVP in Android with a simple demo project”. URL: https://medium.com/cr8resume/make-you-hand-dirty-with-mvp-model-view-presenter-eab5b5c16e42

[22] Sharma, Ankit**.** “Why to choose MVVM over MVP — Android Architecture

“. URL: https://android.jlelse.eu/why-to-choose-mvvm-over-mvp-android-architecture-33c0f2de5516

[23] “The MVVM Pattern”. Microsoft. URL: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/hh848246(v=pandp.10)

[24] “Builder Design Pattern”. SourceMaking. URL: https://sourcemaking.com/design_patterns/builder

[25] "Index of /archive/2010/winter/51023-1/presentations" (PDF). www.classes.cs.uchicago.edu. Retrieved 2016-03-03.

[26] Android Developers. “Create a list with RecyclerView”. URL: https://developer.android.com/guide/topics/ui/layout/recyclerview

[27] Android Developers. “Build a Responsive UI with Constraint Layout”. URL: https://developer.android.com/training/constraint-layout

[28] News Api Documentation. URL: https://newsapi.org/docs

[29] Android Developers. “HttpURL Connection”. URL: https://developer.android.com/reference/java/net/HttpURLConnection

[30] W3School. “What is JSON”. URL: https://www.w3schools.com/whatis/whatis_json.asp

[31] Codelabs Developers. “Build an App with Architecture Components”. DEPRECATED. URL: https://codelabs.developers.google.com/codelabs/build-app-with-arch-components/index.html#0

[32] Android Developers. “Intent Service”. URL: https://developer.android.com/reference/android/app/IntentService

[33] Android Developers. “Job Scheduler”. URL: https://developer.android.com/reference/android/app/job/JobScheduler