

**Escola Universitària d’Enginyeria Tècnica**

**de Telecomunicació La Salle**

Treball Final de Grau

Grau en Enginyeria Informàtica

Angel Farre Echaburua Daniel Amo Filvà

|  |
| --- |
| **ACTA DE L'EXAMEN**  **DEL TREBALL FI DE CARRERA** |

Reunit el Tribunal qualificador en el dia de la data, l'alumne

D.

va exposar el seu Treball de Fi de Carrera, el qual va tractar sobre el tema següent:

Offline transmission of forms in a local and proximity-based environment, oriented for didactic use.

Acabada l'exposició i contestades per part de l'alumne les objeccions formulades pels Srs. membres del tribunal, aquest valorà l'esmentat Treball amb la qualificació de

|  |
| --- |
|  |

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENT DEL TRIBUNAL

Acta

**Abstract**

In an educational environment, the staff in charge of lecturing students often makes use of external services which both provide a desired feature, but also pose a security problem regarding user privacy and data security.

In this work, I will present an application designed to offer the capability of sending and receiving forms in a secure, offline, and proximity-based manner, thus increasing security measures.

This application has been implemented with Google’s Nearby connections API and written in dart via the flutter framework, with the aim to offer a cross-platform solution. Other technologies have been studied too, which will also be explained in detail, defining what features and shortcomings they offer.

The resulting flutter application allows for android users to make full use of the features defined, although iOS users are, as of now, not able to make use of Nearby’s API, thus limiting its cross-platform capabilities.

**Acknowledgments**

I would like to express my gratitude to a series of persons and communities which contributed to my well-being and to my knowledge growth for the duration of this project.

Dr. Daniel Amo Filvà for his round the clock support and mentoring service. His encouragements and advices kept this project running while steering it as to not crash on a wall.

Mr. Alex Tarragó Pulvé for his disposal to attend my questions and for shedding light in some fields where I was in the dark.

Mss. Marta Zapatero i Sentís for cheering me up and providing emotional support, single-handedly pulling me out of some desperation moments.

Last, but not least, the communities of stack overflow and the official FlutterDev discord, for their educational interaction and discussions in helping to solve some problems.

**Content**

[1 Introduction 1](#_Toc60220408)

[1.1 Motivation 1](#_Toc60220409)

[1.2 Theoretical framework 1](#_Toc60220410)

[1.3 Problem 2](#_Toc60220411)

[1.4 Objective 2](#_Toc60220412)

[1.5 Methodology 2](#_Toc60220413)

[1.6 Documentation layout 2](#_Toc60220414)

[2 Research 3](#_Toc60220415)

[2.1 Technologies considered 3](#_Toc60220416)

[2.1.1 Classic Bluetooth 3](#_Toc60220417)

[2.1.2 BLE 3](#_Toc60220418)

[2.1.3 Wi-Fi 3](#_Toc60220419)

[2.1.4 Others 5](#_Toc60220420)

[2.1.5 Google Nearby 7](#_Toc60220421)

[2.1.5.1 Nearby share 7](#_Toc60220422)

[2.1.5.2 Nearby notifications 8](#_Toc60220423)

[2.1.5.3 Nearby messages 8](#_Toc60220424)

[2.1.5.4 Nearby connections 10](#_Toc60220425)

[2.2 Technology conclusion 12](#_Toc60220426)

[2.3 Plugins used 12](#_Toc60220427)

[3 Development 14](#_Toc60220428)

[3.1 Teacher module 14](#_Toc60220429)

[3.1.1 Teacher view 14](#_Toc60220430)

[3.1.2 Create forms 14](#_Toc60220431)

[3.1.3 Endpoint list 14](#_Toc60220432)

[3.1.4 My forms 14](#_Toc60220433)

[3.1.5 Edit forms 14](#_Toc60220434)

[3.1.6 Student management 14](#_Toc60220435)

[3.1.6.1 List of answered users 14](#_Toc60220436)

[3.1.6.2 List of answered forms 14](#_Toc60220437)

[3.1.6.3 Building an answered form 14](#_Toc60220438)

[3.2 Student module 17](#_Toc60220439)

[3.2.1 Student view 17](#_Toc60220440)

[3.2.2 Building forms 17](#_Toc60220441)

[3.3 Settings module 18](#_Toc60220442)

[3.4 Common 19](#_Toc60220443)

[3.4.1 File management 19](#_Toc60220444)

[3.4.2 Shared preferences 19](#_Toc60220445)

[3.5 Model 20](#_Toc60220446)

[3.5.1 Form 20](#_Toc60220447)

[3.5.2 Endpoint data 20](#_Toc60220448)

[4 Conclusions 21](#_Toc60220449)

[5 Limitations 23](#_Toc60220450)

[6 Future lines of work 24](#_Toc60220451)

[7 References 25](#_Toc60220452)

**Acronyms**

AP: Access Point.

API: Application Programming Interface.

BLE: Bluetooth Low Emission.

BR: Basic Rate.

BSSID: Basic Service Set Identifier.

EDR: Enhanced Data Rate.

GATT: Generic Attribute Profile.

HS: High Speed.

IP: Internet Protocol.

IR: Infrared.

ISM: radio spectrum reserved for Industrial, Scientific, and Medical purposes.

LAN: Local Area Network.

MAC: Media Access Control.

OS: Operating System.

QR Code: Quick Response Code.

SIG: Special Interest Group.

SSID: Service Set Identifier.

TFG: *Trabajo Final de Grado*.

UUID: Universally Unique Identifier.

WebRTC: Web Real-Time Communication.

WPA: Wi-Fi Protected Access.

En este capítulo se listan los acrónimos que aparecen en el documento, por orden alfabético. Ya no se desplegarán de nuevo en el resto del documento. Si los acrónimos provienen de palabras escritas en un idioma diferente al del documento, se escribirá en cursiva.

# Introduction

Technology is present in every aspect of our lives. Its flexibility allows for its use in most fields while also offering an improvement on the already existing experience.

One field where technology es slowly taking over is in education. Taking online lessons, surveys, chats, and meetings are a reality today, and they have only been strengthened by the COVID-19 pandemic. Therefore, approaching technology from a simplistic use-on-feature view should no longer be applied.

Educational technology should not only be there to do something for its users, but to perform its duties whilst remaining save. Security and privacy can no longer be overlooked, but at times it is a characteristic that is left out.

For these reasons, an application has been developed which allows for a teacher to share a form, and to receive his student’s answers, all while staying off the net and using local communication. An already existing feature brought by websites like Socrative[[1]](#footnote-2) or Google forms[[2]](#footnote-3) but with a coating of security brought both by Nearby’s API and by the aforementioned offline characteristic. This solution also has the bonus that it does not require the user to log in or create an account, easing the user experience.

The resulting application is capable of sharing forms in approximately a 100m radius and retrieve the results. This, however, is only available to Android users. Nearby’s connections API is currently not available for iOS users. Other options that allowed for iOS compatibility where considered, such as Bluetooth or WiFi Direct, but they had a series of shortcomings that made Nearby connections to be the final technology to be used.

## Motivation

The widespread use of some of the websites that allows to create and share forms and being most of its public oblivious to privacy concerns, quickly lead to the realisation that this project presented an uncovered need, providing the motivation needed to work in this project. Besides the challenge set by the application development itself, this project also intends to raise some degree of awareness about the lack of measures taken in schools and colleges regarding user privacy.

## Theoretical framework

This concept was born out of solutions that QR codes where able to solve, but quickly found its own limits. As explained by smekens[[3]](#footnote-4), QR codes is a great technology to share data on a 1 to 1 basis or a 1 to N unidirectional manner. Examples of this include having QR codes at the back of a book for more information or a fellow student review, having a QR code added to your foreign language homework that lets you listen correct pronunciations, or having QR codes that host a video of the latest class for those students who missed it.

QR codes are great in this regard, but as VIDEO DEL DANI demonstrates, sharing information to several users is fine, but retrieving it from them can be quite a hassle. Having seen the QR code limitation, this project expands on this area by bringing forth the next technological step, wireless offline local communication between users.

## Problem

The need for this application is born purely for two reasons:

* The lack of data security and user privacy when using external services providers in classrooms.
* The inability of the current local offline technology, QR codes, to send data to multiple users and to efficiently retrieve it. It is unidirectional in 1 to N communications.

## Objective

To demonstrate that local data sharing is feasible in a local environment, by developing an application that allows teacher to share a form with his students in an offline environment. Students’ answers should be received and stored in the teacher’s device to be accessible later.

As a personal objective, in addition to the form sharing feature, to make it available to all users, regardless of what mobile OS they use, hence the development in the flutter framework.

## Methodology

The constraints set for this project, other than what it should be capable of doing, where low to non-existent.

//TODO

## Documentation layout

This document is distributed in two great blocks, research, and development.

In the research block, an explanation is given regarding all the decisions taken that affected the latter development of the application. What technologies where looked up, their strengths and weaknesses, and why Nearby connections was the chosen one.

In the development block, an overview of the developed system will be given, followed by a detailed look to its four main modules. Inside each module, the .dart files used will be explored and detailed as to how they fit into the initial grand scheme of things.

# Research

In this chapter all the research done will be detailed. This includes //TODO

## Technologies considered

### Classic Bluetooth

A technology developed in 1994, Bluetooth, or classic Bluetooth, refers to all Bluetooth versions up to 3.0. This includes Bluetooth 1.0 BR, Bluetooth 2.0 EDR and Bluetooth 3.0 HS. It has a speed of up to 24Mbps (in HS mode) and its compatibility is of no concern as all modern cell phones carry a Bluetooth antenna.

The major drawback of classic Bluetooth is that it can hold a maximum of 6-8 connections (depends on factors such as your hardware or amount of RAM). When the maximum is reached and a new device is connected, the oldest device is disconnected.

### BLE

BLE refers to Bluetooth 4.0, released in 2010. It is “Low Emission” names comes from its distinctive feature that allows it to have the transmitter in an awaiting status most of the time, only activating when a connection is initiated. The connection and data transmission lasts only a few milliseconds, compared to the ~100ms it takes for classic Bluetooth (it has a longer pairing process) and it transmits de data at about 1Mbps.

BLE is mainly used for wearable devices, smart IoT devices, fitness monitoring equipment and battery-powered accessories.

Although faster and 100 times more energy efficient, BLE was found not suitable mainly because the transmission between devices is limited to 20 theoretical devices (usually less when implemented).

Also, permission management in Android and iOS differ substantially, mostly because iOS generates its own UUID for each connected Bluetooth peripheral, whilst android uses MAC address and some characteristics and data types that may allow the device’s identification[1].

Also, the advertisement data can be up to 31 bytes long, being composed of three flag bytes, a byte containing how long it is, one byte announcing the service data packet and 16 service UUID bytes, leaving 10 custom bytes which may not be enough.[2]

### Wi-Fi

An alternative to Bluetooth is Wi-Fi. With Wi-Fi direct, devices can establish a direct connection between themselves and multiple devices at once. This feature is widely used when sharing internet from a device that has connection, to one that does not and is well documented[3]. But besides acting as a net provider, Wi-Fi direct can be used for data transfer.

While transferring files is not natively supported, there are several third-party software that manage it, such as sharedrop[[4]](#footnote-5) or Send anywhere[[5]](#footnote-6), so the goal for this project would be creating a similar application.

As opposed to Bluetooth, theoretical maximum user count is 254. Presuming the IP 192.168.0.0, the owner would be assigned the IP 192.168.0.1 and users would range from 192.168.0.2 to 192.168.0.254. Connectivity between devices should not be a problem as all devices have a Wi-Fi antenna, regardless of OS.

Although Wi-Fi direct looks like a promising candidate, it has several problems upon implementation which do not reflect the on-paper specifications mentioned earlier. While it is stated that it has a high maximum user count and compatibility, the reality shows that there are several connectivity issues between devices, specially if they belong to different manufacturers[4].

As a testament to the previous statement, a testing performed by Cirrent concluded that 20% of users fail when using Soft AP. They also state the fact that 40% of negative reviews in connected products are related to connectivity problems[5]. Reliability is highly regarded amongst users, so these figures demonstrate that this technology is not fit for this project’s specifications.

A slight variation of Wi-Fi direct could be interconnecting devices within the same network, using only the local LAN.

In this case scenario, one device acts as the central AP through the local network, while the remaining devices establish a connection via the given 192.168.x.x IP. The IP assigned is random and it directs the user to 192.168.x.x:5000/index.html where shared files can be retrieved.

This functionality is preestablished is some devices and can be tested without the need to download any additional software. This method offers a theoretical maximum user count of 254, including the device acting as central AP (presuming the 192.168.0.0 IP, then users can range from 192.168.0.1 to 192.168.0.254).

While this solution is feasible, it is not entirely offline. Devices need to be connected to the same network for communication to be established and whilst the connection is not being sent to offsite servers, it still uses local resources which might be prone to attacks or already be compromised before its use.

Also, managing a html website hosted in the local network, via an application, can be quite a challenge as a filesystem should be implemented for the teacher to classify all his students/forms and users’ permission should be tightened so that student cannot access another students’ folder, avoid deleting submitted forms, editing submitted forms and so on.

Finally, all communications over Wi-Fi use WPA security, a technology widely used but quite insecure, as it is vulnerable to brute force attacks.

### Others

As seen, Bluetooth has nice features which are hampered by the peculiarities of their versions but being that all Bluetooth signals work at 2.4Ghz ISM it is not out of the question to attempt and **hybrid solution** despite their differences[6].

Luckily, all mobile devices run on Bluetooth smart ready, a naming convention given by SIG to clarify that the Bluetooth being used ranges from 1.0 to 4.0, and that is both compatible with Bluetooth smart, comprised of BLE, and Bluetooth (or Bluetooth classic) comprised of Bluetooth 1.0, 2.0 and 3.0[7].

To communicate devices, both technologies mostly share the same profiles specifications. Profiles are a set of rules that define what task will this Bluetooth device perform[8]. To communicate both a BLE device and a classic Bluetooth one, they must be using the same profile. Among other profiles, GATT is an example or a compatible one, which allows data exchange.

Although and hybrid solution can be taken to term, its implementation provides a challenge for a developer with basic Bluetooth knowledge. Also, it may improve overall performance, but it will not overcome the shortcomings of both classic Bluetooth and BLE, therefore it is deemed an unfit approach.

Another approach could be using **IR communication**, but this technology was quickly found out to fall short of all minimum requirements for this project. Namely, it does not have a very long reach (~30m), it can be obstructed by objects such as walls, smoke, modern devices have dropped the use of IR blasters and most importantly, with IR you can only connect to one device at the time[9].

**Ultrasound** opens another window of opportunities. The device’s speakers can be used to emit a set of signals, inaudible for adults (~17+kHz), that transmit data to other devices which receive it through their microphone[10].

While this is a valid data transferring method, its shortcomings come quite obviously, as in an environment with a lot of ambient sound, this technology is unusable, not to talk about the fact that its transmission speed is around 7kbps and that it has no pairing system, meaning any device within range can listen and read the data transmitted.

**Mesh network**. Many projects have observed similar problems when choosing a technology for local data sharing, and the solution for them was to create a mesh with all the devices that wished to connect. This way, mixing previously discussed technologies such as Bluetooth and Wi-Fi direct, all devices can be interconnected. A clear example of this would be **Firechat**, which although is currently discontinued, it reportedly was used in several protest such as Honk Knong protests (2014), Ecuadorian protests (2015) and Catalan protests (2015).

Also, a Bluetooth mesh could be developed, such as that of **Bridgefy**[[6]](#footnote-7), where devices share messages until each one reaches its destination. In our use case, roles would have to be carefully determined as the role of “teacher” would be thinly separated from the role of “student”.

Luckily enough, Bridgefy does have a Github repository[[7]](#footnote-8) with android samples, however, this is not the case for flutter, where there is a complete lack of documentation and plugins.

Another mesh example, the **Serval**[[8]](#footnote-9) project also offers a mesh built up using a combination of Wi-Fi and Bluetooth. Serval does offer a free license to any user wishing to use its software, however, in a similar fashion to Bridgefy, its deployment in flutter is non-existent.

As a final example, **p2pkit[[9]](#footnote-10)** also offers connectivity between two nearby devices. Much like some of the previously detailed examples, it uses a mix of Bluetooth and Wi-Fi to achieve local communication. While it has an API, it is not available for flutter and it comes at a cost (free version available too, but quite limited).

Besides each individual project API, the problem with these approaches lies in the fact that programming a mesh requires a substantial effort by an experienced team of developers, so it is out of scope for this project to develop a mesh for flutter using any of the previously mentioned services.

### Google Nearby

An API presented in 2017, google nearby offers users an API which makes usage of Wi-Fi, Bluetooth, BLE, IP and audio. The aim of combining these technologies is to ease the communication between proximity devices[11].

#### Nearby share

Presented in 2019 under the name “Fast share” and released in august 2020 under the name “Nearby share”, it intends to be the android equivalent to the popular AirDrop, available for iOS only.

Nearby share will make use of Bluetooth, BLE, WebRTC and peer-to-peer Wi-Fi to both be fully offline and to make use of the most appropriate and faster technology at that given time. Also, nearby share is capable of sharing images, files, links, and other contents and allows the user to share data anonymously or not. It also offers a visibility setting that makes you visible to contacts only, to stay hidden, or visible to only some of the contacts[12][13].

Available for devices running Android 6+, this means it is available for 84.9% of the devices, according to Android Studio, as shown in Figure 1.

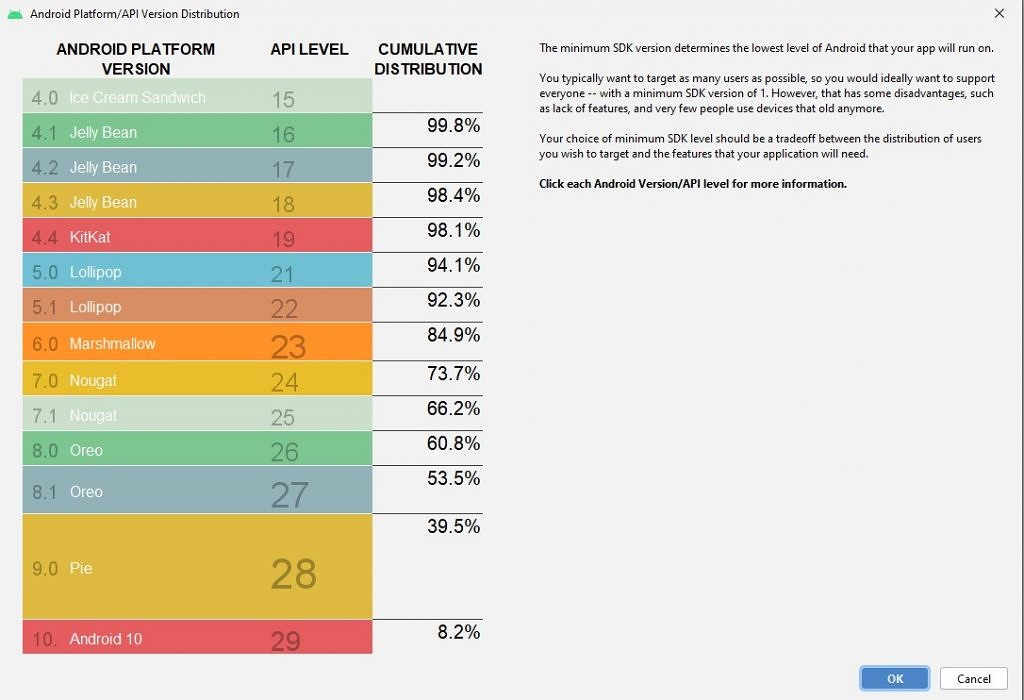


Figure 1 Table showing the distribution of Android OS versions.

While it may seem a good candidate, there are some nuisances we must point out as to why this option was discarded.

As a starter, this is a service provided by google, not an API. That means there are no docs, no codes samples, all there is, is an update to your phone and this feature appearing on it afterwards. At the most, Microsoft offers an API that manages shared connections, mostly to integrate them into the Windows 10 OS and share data between devices[14] as a part of the Project Rome (Microsoft's cross-device experiences platform for apps).

While Microsoft has its own project for cross-device experiences, other services might not, and google share is, as of now, not supported for platforms such as iOS and Macs. They “plan to try to expand the feature to additional platforms in the future”. Not surprising as it is rather new. Also, it must be said that testing revealed that sharing data to more than one device at the time proved impossible.

Finally, regarding flutter packages, since there is no API available, there is also no packages available to code this service into a flutter application.

#### Nearby notifications

Developed in 2015, and part of the API proper of Google’s nearby technology, nearby notifications allows Android users to see a message associated to an app or website which is location-specific based. As stated in the official documentation, amongst other examples, it can “launch conversations or chatbots inside messaging apps”[15].

On top of that, it can open an application to display its contents, or if the user does not have it, direct him to the corresponding download page. This could greatly improve user experience and ease use for young pupils, elder pupils, or students who are not tech savvy.

As a downside to notifications is that it transports URL data through BLE, with all that BLE implies. In this case, presuming you make use of Eddystone beacon, from the 17 bytes available, you are left with 11 bytes for payload. On top of that, the payload is a URL, so it needs internet access to retrieve its contents. Also, the channels used are unidirectional making a student answer unviable[16].

Finally, and most importantly, google deprecated this API in 2018, just three years after its release. This decision was taken because the use of nearby notifications led to an increase of spam for the users. This spam could be filtered or tuned but nonetheless, google stated that this solution, and therefore this technology, did not live up to their high-quality standards, this removing support. The beacons are still available through the Proximity Beacons API[[10]](#footnote-11), which in turn was recently deprecated too on December 7th, having its support formerly shut down on April 1st, 2021[17].

#### Nearby messages

Next in the nearby family is messages. This technology uses a mixture of Bluetooth, BLE, Wi-Fi and near ultra-sonic audio to communicate near devices. Having this many technologies running under the hood, means that nearby messages can leverage the strengths from each one, to supplement the remaining weaknesses[18].

Additionally, the inclusion of ultra-sonic audio allows for the developer to adjust the range of the local communication. By default, it is set at about 30 meters, but when in ultra-sonic mode, it can be reduced to 1.5 meters[19].

Also, this API is not deprecated and does not even require a Google Account. It is not heavily payload limited and is available both for Android and iOS.

As an example of its usage in a local environment is card against humanity test application, called Manatee, developed by Shepherd, Meredith C. for his thesis. While she demonstrates that a local application through this technology is viable, she also gives some insights regarding flutter development:

“Having no previous Android development experience, I had planned to use Flutter, Google’s newly introduced cross-platform app development framework, in the hopes that it would provide an easier road to incorporating Nearby Messages. However, despite the fact that both projects were produced by the same company, no Flutter plugin for Nearby existed at the time, nor did one became available until March of 2019. (Cross, 2019) In the meantime, my advisor suggested Xamarin.Forms, a more established cross-platform framework with a far more robust mechanism for incorporating platform-specific code. With Flutter’s Nearby Messages support lagging behind Xamarin’s, and with Flutter still in Beta at the time (Sneath, 2018), there was no good reason not to switch to a more mature product.”[20]

While the previous statement is only a year old, its to be believed that it still stands as of today and defeats the personal objective imposed for this project, to use the Flutter framework.

Also, nearby messages makes use of a many to many communication model (Figure 2), meaning that much like Wi-Fi direct, permissions as to who the “host” of the session is must be established, ensuring to student can peek at another student’s answers.

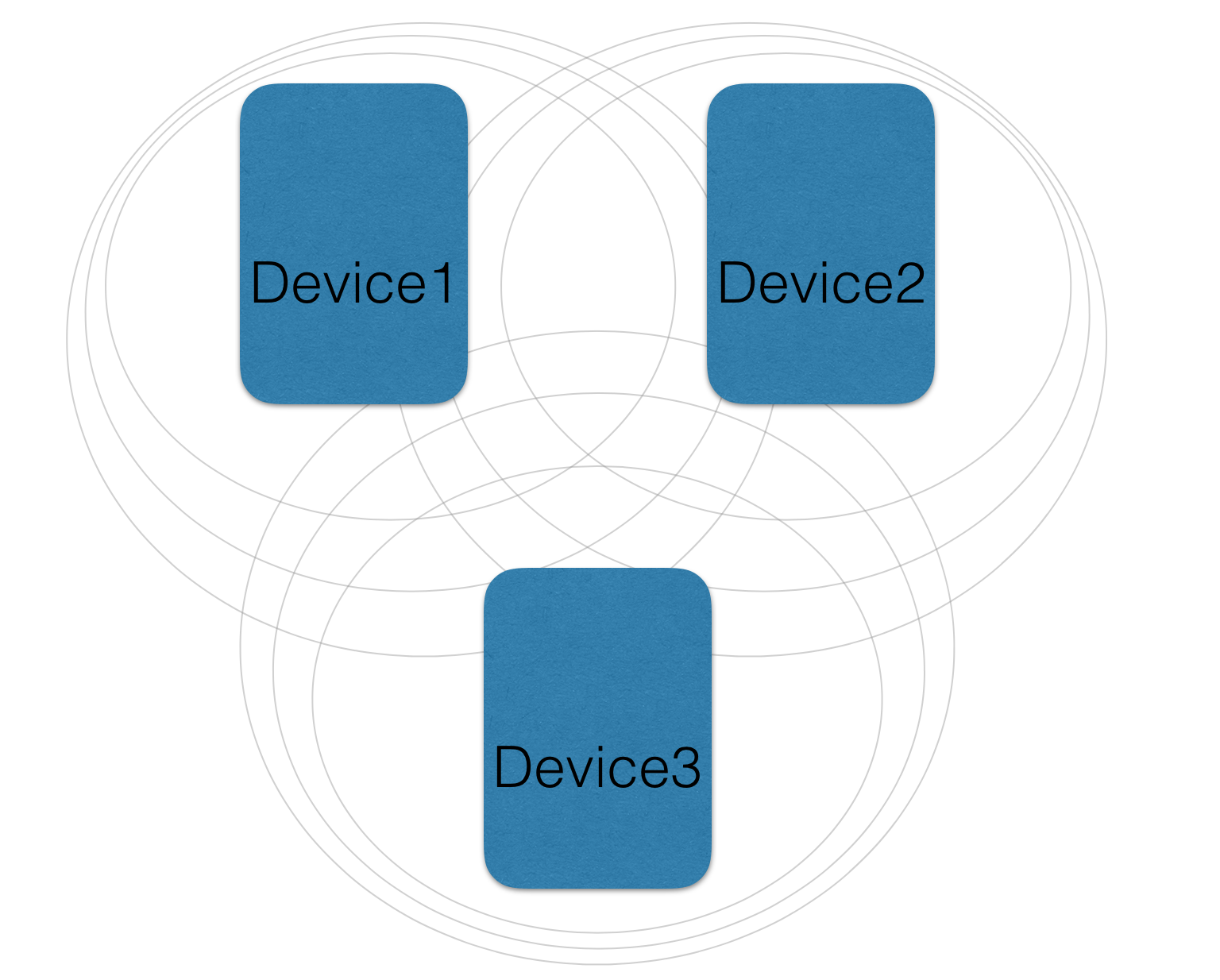


Figure 2 Many to many communication model for nearby messages.

Besides development hardships, nearby messages states that it is a nearby based communication technology, which is true, but it also requires internet connection (albeit, not all devices must be in the same network). This requirement becomes an apparent issue as nearby messages only share locally a token, used to retrieve a message stored in Google’s servers. While only used that have been locally shared this token may access the message, its necessity to connect to an external server defeats the purpose of this project, thus rendering this technology unfit for use.

#### Nearby connections

As the last member of the nearby family, nearby connections offers similar features to that of nearby messages, but with some key differences such as disregard for network connectivity and as of nearby connections 2.0 (release in 2017), fully offline capabilities[21][22].

In more detail, nearby connection uses BLE, Bluetooth and Wi-Fi hotspots to advertise, discover and connect to peer devices in a fully offline manner. On top of that, connections are encrypted, low latency and high bandwidth since much like nearby messages, it leverages the strengths of each technology to supplement its weaknesses. Notice that, unlike nearby messages, nearby connections does not make use of ultra-sounds, thus its unable to adjust its the default range of 100 meters and also nearby messages uses Wi-Fi and nearby connections uses Wi-Fi hotspots[23].

Nearby connections also includes three connections topologies, star 1:N, cluster (much like the one defined in Figure 2 for nearby messages) N:M and point-to-point or peer-to-peer. This fits perfectly into our project as the star topology, see Figure 3, can be utilised by the teacher as the central node, thus eliminating the need to control unauthorised access amongst students.

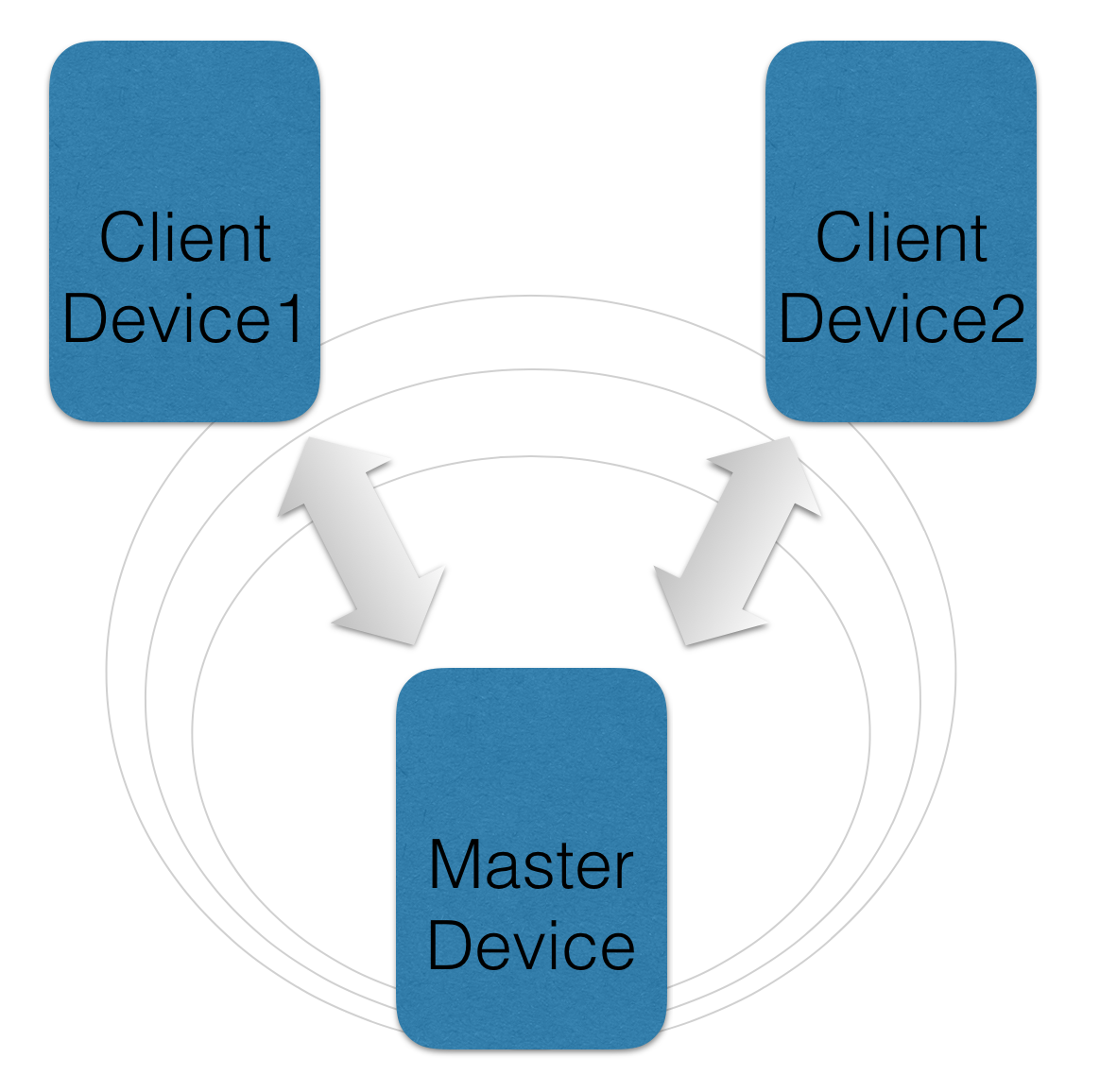


Figure 3 Nearby connections star topology.

On top of the topology, there are three means through which the encrypted data can be transferred. Bytes, file, and stream.

* Bytes: Byte arrays with a maximum length of 32k.
* File: Files of any size.
* Stream: Data generated on the fly, used mostly to transfer audio or video without knowing the end beforehand.

Much like other technologies discussed, it has some shortcomings too. Mainly, it is not compatible with iOS devices. It is a feature Google is planning to introduce in a future, but there is no estimation as to when they might implement iOS compatibility.

Also, both nearby connections and nearby messages are not allowed to work in the background, they must always be in the foreground[24]. This is explicitly designed by Google so that the final user is always aware when his device might be advertising/sharing/discovering. This might prevent the user from involuntarily perform nearby actions (much like when users forget Wi-Fi on Bluetooth is ON in their devices) and also its to avoid an excessive battery loss. Using Bluetooth, BLE and Wi-Fi means a lot of battery is required to perform actions, thus having the activity in the foreground forces to user to use nearby, and as soon as they have finished, to discard it and avoid wasting excessive battery.

## Technology conclusion

Having explored so many technologies for local data transport, one this becomes apparent. Whilst there are means through which you may operate in a local offline manner, there is always a shortcoming such as compatibility.

There are paid API that offer all the functionalities and features required for this project but being a proof of concept for local offline data transmission in an educational environment, making use of paid services is out of the question.

As such, the chosen technology for the project is **nearby connections**. While it may not be compatible with iOS devices, it offers features such as star topology and a choice in data transmission types. Additionally, it is compatible with a wide range of devices, it has several plugins available for flutter and Google plans to add iOS support in the future, meaning a development in flutter today can be the iOS application of tomorrow.

## Plugins used

For the development of this project, several plugins where used. This, and other plugins, may be found in the official pub.dev website. Some of the plugins used in this project are not used to its full extent, or are not used at all, for they are there as a next step in the project. For more information about next steps, see the chapter Future lines of work.

As of now, the plugins implemented to its full usage are:

* cupertino\_icons: It is the default package included in any new Flutter project. It contains all default icon assets.
* nearby\_connections: A package that offers a neat usage of Google’s nearby connections. Developed by Prerak Mann and in collaboration with Gourav Saini, it is available both from the packages’ website and in the form of a Github project. As an example of its usage, it was implemented in a Monopoly Money Handler[[11]](#footnote-12) project.
* shared\_preferences: This package allows for simple platform-specific data storage. The data in stored in a pair of key-value pair. This plugin will be further explained in the chapter Shared preferences. Its developers are the official flutter.dev group.
* path\_provider: A plugin used to navigate the filesystem of the device, such as the temp and the data directories (amongst others). Much like shared\_preferences, its developers are the official flutter.dev group and more information regarding its usage can be found in the chapter File management.
* device\_info: Also developed by the official flutter.dev group, this package offers information from the device such as model, name, UUID… This package is used throughout the project, although shortly, it is used to obtain a unique reference from each connecting device as connection id changes each session.
* flutter\_launcher\_icons: This last plugin, developed by the fluttercommunity.dev group, simply offers the means to update the application’s launch logo.

Regarding the unused, or partially implemented plugins:

* wifi\_iot: This plugin, developed by alternadom.com, offers means through which Wi-Fi connections can be handled. It checks the Wi-Fi status, it enables/disables Wi-Fi, gets information such as SSID, BSSID, frequency, signal strength…

This plugin is used in the settings window, detailed in the chapter 3.3 Settings module and it is partially implemented as some of its features have been overtaken by nearby connections API, as it offers the means to automatically turn on the Wi-Fi in case it is turned off, rendering this plugin obsolete. Despite this, it is kept in the project as it may be used for future features.

* image\_picker: This official plugin (flutter.dev), allows the developer to pick images from the existing library and take pictures with the camera. It was implemented into the project as the means to test the capabilities to send images with nearby connections.

As images are successfully sent and received, this plugin was kept in the project as a future feature might encompass the listing of connected students to the teacher by name and picture, making it easier for the teacher to recognize the connected users. As of now, this feature is of low priority hence it is not implemented.

# Development

For the development of this project, the workflow of nearby connections must be understood first. As nearby connections is the main feature of this project, most other features will need to adapt to its workflow, hence its importance.

Nearby connections woks in and advertise and discovery manner, meaning that for a connection to take place, at least one device must be advertising at the same time for another device to be discovering. On top of that, we stated previously that Google enforced the use of nearby connections while the application is in the foreground, and also recommends notifying the user of the current advertise or discovery mode via an icon or a special screen.

Taking this into account, the application will have a workflow time utilised only be the establishment of the connection amongst devices. As specified in Figure 4, this encompasses the discovery/advertising phase, the connection request by the discoverer, and the acceptance of the connection on both ends. From this point onwards, the connection is established, and data can be sent, as well as other actions can be performed.

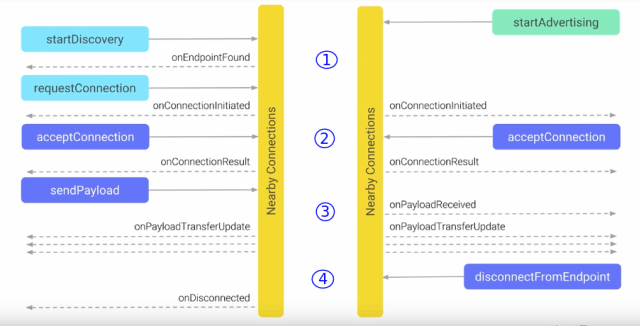


Figure 4 Nearby connections workflow

Other actions, which may also be used before the connection is established include tinkering with the settings, and specific to the teacher, create forms, edit forms, and consulting previously sent forms by the students.

What follows is a detailed explanation for each module used in this project.

## Teacher module

This module implements all the functionalities available to the teacher. As an educational application, the teacher is the one in control of most features, therefore this module contains most of the features implemented for this project and thus is the one with most importance.

The features encompass:

* Creating forms.
* Editing/sharing/deleting forms.
* Consulting connected devices.
* Advertising/halt the advertisement of the device.
* Disconnecting all connected devices.
* Consulting the student’s answers.

These features are distributed amongst several .dart files, grouped according to their nature and stage in the nearby connections’ workflow. The individual .dart file are detailed in the following chapters, although a general diagram for the teacher module can be found in the following page (Figure 5).

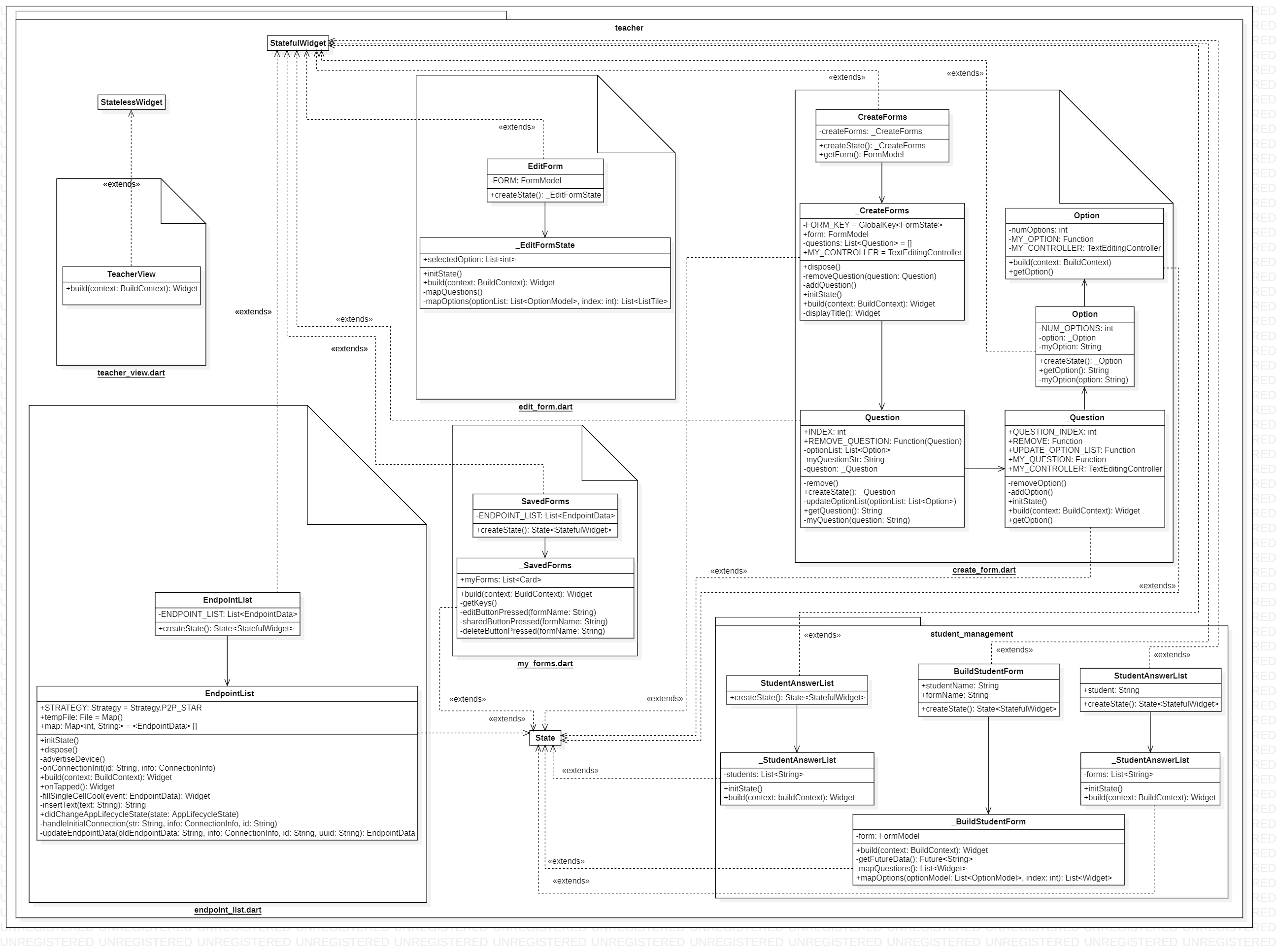


Figure 5 Teacher module UML diagram

### Teacher view

This simple view serves as a main menu for the teacher. Easily available, he may tap in any of the four buttons to navigate to their corresponding window. This view extends from StatelessWidget, meaning its state does not change on runtime. It is the only class that has this characteristic.

### Create forms

This view, designed to dynamically create forms and map them to its corresponding model, might be one of the most complex of the project.

When creating a form, a user might add as many questions as desired, and each question may contain as many options as desired too. The user also has the option to remove any of the inserted questions and options, although when it comes to the options, the user can only remove the last option added. This decision was taken as a measure to keep a clean view. Implementing an option to delete any option is feasible, but it would require a button, or similar, for each option, on top of the already existing marker that each question has, making the view too crowded.

Therefore, within the create forms feature, we find the Option class and the Question class. That brings to total of classes used to three, and each of the three classes (CreateForms, Question and Option) extend from StatefulWidget, with each having their own corresponding class that extends their parent’s state (\_CreateForms, \_Option and \_Question).

As such, upon user interaction, questions and answer are created without limits. Once the user is satisfied with the form created, and has given it a title, he can save it into the device. To do such a thing, each question with its own corresponding options is iterated through to create a Json file reflecting the form. It is this form the one it is stored in the device using shared preferences (refer to chapter 3.4.2 Shared preferences for more information on shared preferences).

### Endpoint list

To develop this view, a close look has been taken at Google’s guidelines where they state that the user should always be fully aware of when the device is in discovery/advertising mode and that the user should always know what information is being shared.

While information sharing will come at a later stage, this window serves the purpose of advertising the device to the students. When the “Advertise device” option is selected, the device enters advertise mode and displays in a list all the connected devices (empty if none are connected). As long as the teacher remains in this view, his device is being advertised and all connections requests are prompted. Loosing focus of the application or returning to the previous view (via return button or back arrow in the view’s title) will cause the device to stop advertising until the focus is returned to the advertise view.

It must be said that the official documentation states that connections requests can be received after stopping the advertise mode, as discovering users may see an advertising device (while it is advertising) and make a later request for a connection once the advertising has ended. In other words, devices may recognise each other during advertise/discover phase, but it is the user who requests a connection, which can be done after a device has been recognised.

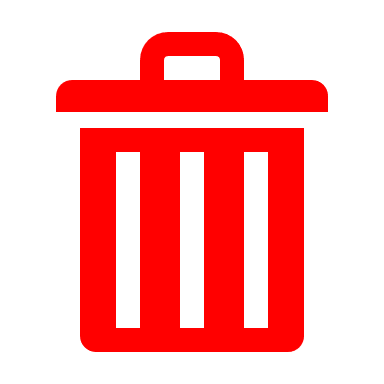
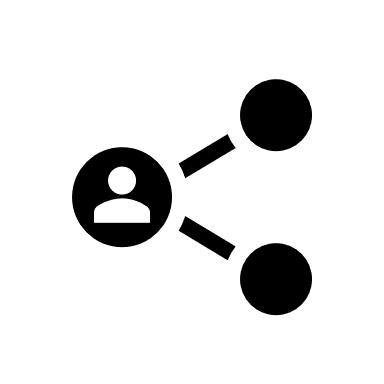
Upon accepting connections, the list fills up with the connected devices, which display the users’ name, the connection ID, and the verification token for that connection. Also, a space for an image has been added. This image is intended as a future feature, which is detailed in chapter 6 Future lines of work.

The user’s UUID is not displayed. Instead, when first meeting a new user, it is stored into shared preferences alongside the user’s name. Upon reconnection, the UUID is checked to see if it matches that of any previous user connected. If it does, and the user is under another name, then the user’s folder is renamed with the latest name. Besides checking if the user had previously connected to the device, it also checks whether the connecting user is already connected, thus avoiding duplicates in the list.

The usage of UUID (via the device\_info plugin), is done as a consequence of nearby connections not implementing the means to identify previously connected devices. It makes use of the ID and validation token, but those are valid only for the duration of the current connection. A device connected under the ID “sampleID” might connect later with the ID “anotherSampleID”, thus making ID not reliable.

### My forms

In order to consult your forms, this view lists them for you. Within each displayed form, three actions can be taken:

* Delete form : Deletes the form from memory. It does not, as of now, include a confirmation pop up.
* Edit form : Allows you to make changes to an existing form.
* Share form : It sends the form to all connected users. To be able to send the form, it converts it to Json (after retrieving it from memory) and sends it as a byte payload. Much like the delete form functionality, it lacks a confirmation pop up.

While basic, this view offers the users some flexibility in the management of the forms. Besides this, it is the only view in the app that allows for a share of information, and listing all available forms makes it easy for the user to decide what to send and when to send it, keeping in line with Google’s quality standards.

### Edit forms

When selecting the “Edit form” option mentioned in chapter 3.1.4 My forms this v

//TODO

### Student management

This sub directory is designed to enclose all actions the teacher can take with a received form from a student. These actions, so far, include only to display the answered form, although the steps previous to displaying a form include showing the filesystem where all student’s answers are stored (by student name) and within each student, a list of all his/her forms.

All modules contained within this directory make use of the functionalities associated with file management, which can be found in more detail in chapter 3.4.1 File management.

#### List of answered users

When receiving a connection, regardless of it is a new connection or one from an existing user, the filesystem is updated, or created, with a directory with the user’s name. It is on this filesystem that the student’s response will be stored.

When accessing this view, the teacher will be able to see a directory for each connected user (bear in mind that a connected user may not necessarily have sent any answer to the teacher).

#### List of answered forms

In a similar fashion to the previous point, within the list of users when accessing a user that has responded to the teacher, a list of all his responses is displayed. The forms answered by the student are saved using the same name under which they were created and sent to the student. Selecting any form will display it.

#### Building an answered form

As the last step, selecting a form from a student, will begin the process to display it to the teacher. This process accesses the saved form stored in a file in the system, parses the retrieved form in Json format and dynamically creates a view with uneditable radio buttons to indicate the user’s choices.

## Student module

### Student view

Texto.

### Building forms

Texto.

## Settings module

Diagrama

Descripción generada automáticamente

Figure Settings module UML diagram

## Common

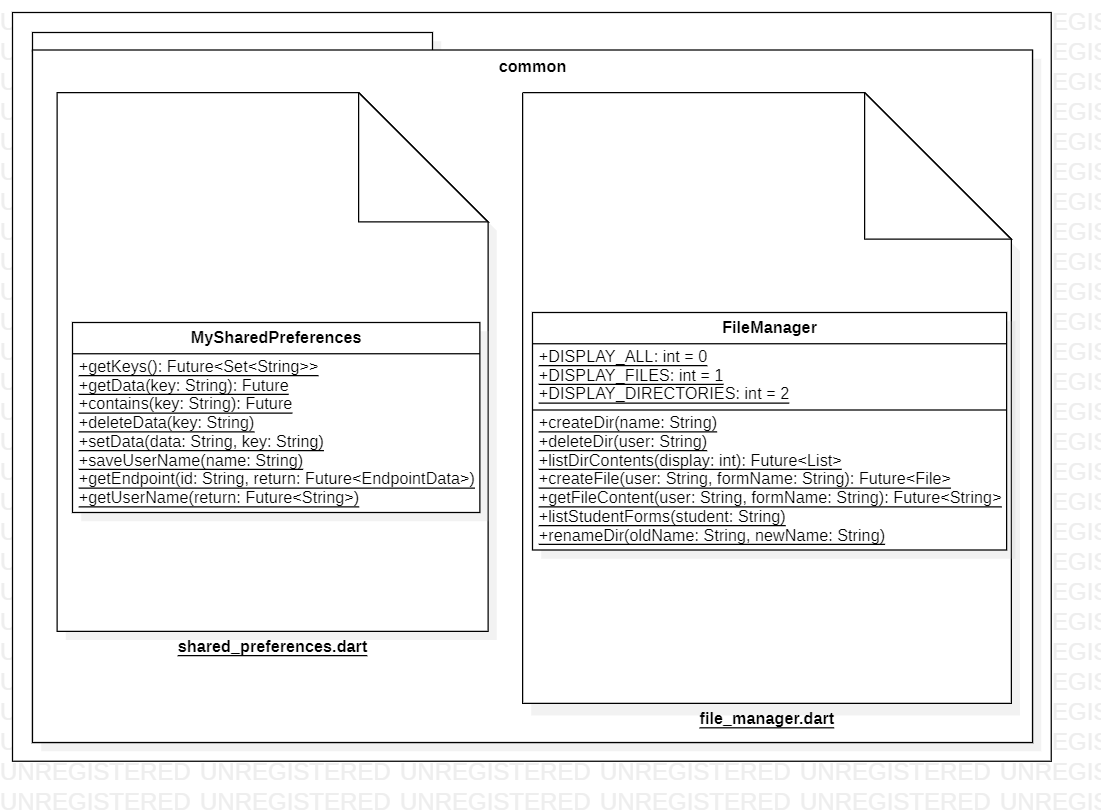


Figure 7 Common module UML diagram

### File management

Texto.

### Shared preferences

Texto.

## Model

Asd

Diagrama

Descripción generada automáticamente

Figure 8 Model module UML diagram

### Form

Texto.

### Endpoint data

Texto.

# Conclusions

Contenido:

* Conclusiones del trabajo realizado.
* Puntos fuertes y puntos débiles del trabajo realizado.
* Limitaciones del trabajo realizado.
* Líneas de continuación del trabajo.

# Limitations

# Future lines of work

# References

[1] D. Seweryn, ‘A Curious Relationship: Android BLE and Location | Polidea’, 2019. https://www.polidea.com/blog/a-curious-relationship-android-ble-and-location/ (accessed Oct. 21, 2020).

[2] M. Cunche *et al.*, ‘On using Bluetooth-Low-Energy for contact tracing On using Bluetooth-Low-Energy for contact tracing. [Research Report] Inria Grenoble Rhône-Alpes; INSA de Lyon On using Bluetooth-Low-Energy for contact tracing’, 2020. Accessed: Dec. 22, 2020. [Online]. Available: https://hal.inria.fr/hal-02878346v3.

[3] Brendan, ‘List of Android Wi-Fi Direct Documentation and Resources’, 2016. https://groups.google.com/g/wi-fi-direct/c/uWpuOzHY6y0 (accessed Nov. 11, 2020).

[4] Thali project, ‘Android Wireless Issues’. http://thaliproject.org/androidWirelessIssues/ (accessed Dec. 22, 2020).

[5] B. Nelson, ‘The Challenges of Soft AP: What Goes Wrong and Why’, 2017. https://blog.cirrent.com/challenges-soft-ap (accessed Nov. 09, 2020).

[6] B. Ray, ‘Bluetooth Vs. Bluetooth Low Energy (BLE): What’s The Difference?’ https://www.link-labs.com/blog/bluetooth-vs-bluetooth-low-energy (accessed Oct. 17, 2020).

[7] R. Castro, ‘Bluetooth Classic Vs. Low Energy Vs. Smart Vs. Smart Ready ¿Cuál es cuál?’, 2020. https://www.wikiversus.com/informatica/bluetooth-classic-vs-bluetooth-low-energy/ (accessed Dec. 22, 2020).

[8] ‘Traditional Profile Specifications | Bluetooth® Technology Website’. https://www.bluetooth.com/specifications/profiles-overview/ (accessed Dec. 22, 2020).

[9] ‘Advantages and Disadvantages of Infrared sensor’. https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-Infrared-Sensor.html (accessed Dec. 22, 2020).

[10] B. Armstrong, ‘quiet/org.quietmodem.Quiet: Quiet for Android - TCP over sound’, *2020*. https://github.com/quiet/org.quietmodem.Quiet (accessed Dec. 22, 2020).

[11] ‘Nearby  |  Google Developers’. https://developers.google.com/nearby (accessed Dec. 26, 2020).

[12] ‘Google’s Nearby Share, its file-sharing AirDrop clone for Android, rolls out’. https://www.xda-developers.com/google-nearby-share-file-sharing-airdrop-clone-android-rolling-out/ (accessed Oct. 25, 2020).

[13] ‘Android’s “Nearby Share” file sharing feature is finally launching - The Verge’. https://www.theverge.com/2020/8/4/21353020/android-nearby-share-file-sharing-feature-launch-airdrop (accessed Oct. 25, 2020).

[14] ‘Android Nearby sharing API reference - Project Rome | Microsoft Docs’. https://docs.microsoft.com/en-us/windows/project-rome/nearby-sharing/api-reference-for-android (accessed Dec. 26, 2020).

[15] Google Developers, ‘Nearby Notifications Overview | Google Developers’, 2015. https://developers.google.com/nearby/notifications/overview (accessed Dec. 28, 2020).

[16] J. Adkins, B. Ghena, and P. Dutta, ‘Freeloader’s Guide Through the Google Galaxy’, 2019, doi: 10.1145/3301293.3302376.

[17] R. Nayak M, ‘Android Developers Blog: Discontinuing support for Android Nearby Notifications’, *2018*. https://android-developers.googleblog.com/2018/10/discontinuing-support-for-android.html (accessed Dec. 28, 2020).

[18] ‘Overview  |  Nearby Messages API  |  Google Developers’. https://developers.google.com/nearby/messages/overview (accessed Dec. 28, 2020).

[19] Google Developers, ‘Nearby Messages (100 days of Google Dev) - YouTube’, Jul. 17, 2015. https://www.youtube.com/watch?v=hultDpBS22s&feature=youtu.be (accessed Dec. 28, 2020).

[20] M. C. Shepherd, ‘Manatee: Using Google Nearby Messages to Build a Cross-Platform, Proximity-Based Mobile Client for Cards Against Humanity-Style Party Games’, 2019. https://nrs.harvard.edu/URN-3:HUL.INSTREPOS:37364571 (accessed Nov. 06, 2020).

[21] Google Developers, ‘Android Developers Blog: Announcing Nearby Connections 2.0: fully offline, high bandwidth peer to peer device communication’, 2017. https://android-developers.googleblog.com/2017/07/announcing-nearby-connections-20-fully.html (accessed Nov. 08, 2020).

[22] ‘Overview  |  Nearby Connections API  |  Google Developers’. https://developers.google.com/nearby/connections/overview?authuser=1 (accessed Nov. 21, 2020).

[23] S. Bansal, ‘Nearby Connection API. Nearby connections is a peer-to-peer… | ProAndroidDev’. https://proandroiddev.com/nearby-connection-api-b235529e6643 (accessed Nov. 12, 2020).

[24] E. Proj, ‘Android Nearby, Connections vs. Messages| Mobile App Development Publication | Medium’, 2016. https://medium.com/mobile-app-development-publication/android-nearby-connections-vs-messages-a2bdf6a59ff3 (accessed Dec. 28, 2020).

Plantilla:

Por otro lado, en el caso de elaborar listados dentro del texto, los estilos permitidos son los siguientes. Para listados consistentes en elementos breves, menores de 4 líneas aproximadamente, se utiliza el estilo Listado 1.

* Elemento 1.
  + Subelemento.
* Elemento 2.
* Elemento 3.
* Elemento 4.

Para listados consistentes en una sucesión de elementos con tamaño considerable, se utilizará el estilo Listado 2, que aumenta el espaciado entre párrafos facilitando así la lectura:

Elemento 1.

* + Subelemento.

Elemento 2.

Elemento 3.

Elemento 4.

Queda a criterio del alumno la decisión última de utilizar un estilo de listado u otro.

A grandes rasgos, el alumno debe tener en cuenta en la elaboración del documento que sólo puede utilizar los estilos creados en este documento, a saber:

* Listado 1.
* Listado 2.
* Normal.
* Título 1.
* Título 2.
* Título 3.
* Título 4.

Queda a criterio del alumno el uso de **negrita**.

La letra en cursiva se utilizará en aquellas palabras escritas en un idioma diferente al que se ha escrito el documento. Por ejemplo:

* Para realizar un mezclado digital se utilizará un *down-converter*.
* En este caso se utiliza un esquema de *routing* híbrido.
* En este caso se utiliza un esquema de encaminado híbrido.

El escrito del documento debe realizarse en modo impersonal. En el caso de querer enfatizar un alto nivel de implicación en alguna tarea o punto crítico, se podrá utilizar en última instancia la primera persona del plural.

Es muy común que en documentos de TFG-TFM o similares se utilicen elementos de apoyo como figuras, tablas o fórmulas. Cada vez que aparezca uno de estos tres elementos se debe asegurar:

* La asignación de un título para dicho elemento.
  + Referencias 🡪 Insertar título.
* Una vez creado el título, éste debe ser llamado o referenciado desde el texto mediante una referencia cruzada.
  + Referencias 🡪 Referencia cruzada.

Por ejemplo: Tal y como se puede observar en la Figura 1, en la Ecuación 1 y de manera resumida en la Tabla 1.



Figura 1. Sensor de piezoeléctrico de Pro-Wave.

Ecuación 1. *Fast Fourier Transform*.

Los elementos ecuación deben escribirse con el editor de ecuaciones.

Tabla 1. Control de cambios.

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Autor | Fecha | Cambios respecto la versión anterior |
| 00 | RA | 29/09/11 | No aplica. |
| 01 | RA | 04/11/11 | * Definición tipología de receptores. * Inclusión aspectos normativos. * Particularización arquitectura de interfaz radio. * Se desestima posicionamiento. |

Se debe notar que los títulos asociados a figuras y ecuaciones se sitúan en la parte inferior de dichos elementos mientras que los títulos asociados a tablas se sitúan en la parte superior de éstas. La forma de la tabla queda a criterio del alumno siempre que se mantenga un estilo limpio y que no dificulte su lectura y comprensión.

Finalmente se aborda el tema de las referencias a otros documentos. Es muy usual y no puede ser de otra manera que un documento como este se apoye en otros documentos (libros, artículos de congreso, artículos de revista, etc.) para poder conseguir un discurso científico y/o técnico coherente. La estructura de este documento ya dispone de un capítulo de referencias (último capítulo). La manera de elaboración de referencias es la siguiente. Cuando se necesite hacer referencia a un documento de apoyo, en primer lugar se acudirá al capítulo de referencias y se irán añadiendo referencias (estilo Referencia) a los documentos que el alumno considere oportunos. Suponiendo que se desea hacer una referencia a la última referencia añadida se debe ir a Referencias à Referencia cruzada à Elemento numerado y Número de párrafo. Aquí la referencia cruzada al último elemento [3] .

1. https://www.socrative.com/ [↑](#footnote-ref-2)
2. https://www.google.com/intl/en\_en/forms/about/ [↑](#footnote-ref-3)
3. https://www.smekenseducation.com/communicate-with-qr-codes-in-th/ [↑](#footnote-ref-4)
4. https://www.sharedrop.io/ [↑](#footnote-ref-5)
5. https://play.google.com/store/apps/details?id=com.estmob.android.sendanywhere&hl=es [↑](#footnote-ref-6)
6. https://bridgefy.me/ [↑](#footnote-ref-7)
7. https://github.com/bridgefy/bridgefy-android-samples [↑](#footnote-ref-8)
8. http://www.servalproject.org/ [↑](#footnote-ref-9)
9. http://p2pkit.io/ [↑](#footnote-ref-10)
10. https://developers.google.com/beacons/proximity/guides [↑](#footnote-ref-11)
11. https://github.com/mannprerak2/monopoly\_money\_game [↑](#footnote-ref-12)