

Vitucci, Bruno Nicolás

Diseño, desarrollo e implementación de la plataforma móvil para la gestión de turnos en salud en la Municipalidad de Berazategui

2018

Instituto de Ingeniería y Agronomía
Ingeniería en Informática



Esta obra está bajo una Licencia Creative Commons Internacional.
Atribución - No Comercial 4.0
<https://creativecommons.org/licenses/by-nc/4.0/>

Documento descargado de RID - UNAJ Repositorio Institucional Digital de la Universidad Nacional Arturo Jauretche

Cita recomendada:

Vitucci, B. N. (2018). *Diseño, desarrollo e implementación de la plataforma móvil para la gestión de turnos en salud en la Municipalidad de Berazategui. [Informe final]. Universidad Nacional Arturo Jauretche. Disponible en RID - UNAJ Repositorio Institucional Digital UNAJ <https://biblioteca.unaj.edu.ar/rid-unaj-repositorio-institucional-digital-unaj>*

Practica Profesional Supervisada

Informe final

29/10/2018

Bruno Nicolás Vitucci
Universidad Nacional Arturo Jauretche
Instituto de Ingeniería y Agronomía
Ingeniería Informática



**PRÁCTICA PROFESIONAL SUPERVISADA (PPS)
Informe Final**

DATOS DEL ESTUDIANTE

Apellido y Nombres: Vitucci, Bruno Nicolás

Documento: DNI 32.792.726

Nº de Legajo: 3777

Correo electrónico: vituccibruno@gmail.com

Cantidad de materias aprobadas al comienzo de la PPS: 44

PPS enmarcada en artículo 7, inciso "b" de la Resolución (CS) 103/16

Periodo en que se realizó la PPS: agosto – octubre de 2018

PROFESOR TUTOR DE LA UNAJ

Apellido y Nombres: Morales, Martín

Correo electrónico: martin.morales@unaj.edu.ar

**PROFESOR TUTOR DEL TALLER DE APOYO A LA PRODUCCIÓN DE TEXTOS ACADÉMICOS
DE LA UNAJ**

Apellido y Nombres: Lavigna, Lía

Correo electrónico: lialavigna@gmail.com

DATOS DE LA ORGANIZACIÓN DONDE SE REALIZA LA PPS

Nombre o Razón Social: Municipalidad de Berazategui

Dirección: Av. 14 (Rigolleau) entre 131 y 131A, Berazategui, Buenos Aires.

Teléfono: (+54) 4356-9200 (interno 1200).

Sector: Secretaría de Modernización del Estado, Dirección de Sistemas.

TUTOR DE LA EMPRESA/INSTITUCIÓN

Apellido y Nombres: Musicco Andrés


Correo electrónico: amusicco@berazategui.gov.ar

FIRMA DEL COORDINADOR DE LA CARRERA

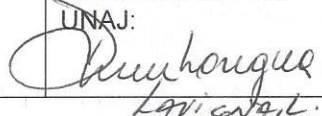
Firma Estudiante:


BRUNO
VITUCCI

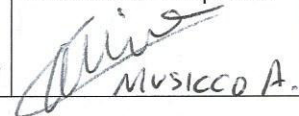
Firma tutor UNAJ:


Dr. Ing. MARTIN MORALES
Vicedirector
Instituto de Ingeniería y Agronomía
Universidad Nacional Arturo Jauretche

Firma tutor TAPTA

UNAJ:

LAVIGNA LIA

Firma tutor Empresa:


MUSICCO A.



Introducción

El presente trabajo exhibirá el diseño, desarrollo e implementación de una aplicación móvil nativa para dispositivos Android, orientada a maximizar la eficiencia en la gestión de turnos para ser atendidos en Centros de Atención Primaria de Salud del partido de Berazategui (CAPS).

El mismo consiste en crear una aplicación de reserva de turnos online para usuarios/pacientes, a quienes se les ofrecerá la comodidad y evitará la demora o indisponibilidad que se genera en la actualidad al efectuar las solicitudes de manera telefónica o personal en cualquier centro de salud. Su función será la de consultar horarios, especialidades, disponibilidades de turnos y otras prestaciones. Esta nueva modalidad también agilizará el proceso de gestión de turnos a nivel administrativo y directivo, para una correcta toma de decisiones y una adecuada asignación de recursos.

Como en la actualidad ya se dispone de un rol usuario administrador vía web, esta aplicación móvil hará foco en el rol usuario paciente, para que de una manera sencilla pueda gestionar sus turnos y a la vez descomprimir la labor administrativa de las entidades que lo gestionan.

El desarrollo se realizó mediante el IDE oficial Android Studio, bajo el lenguaje de programación orientado a objetos Java, con librerías propias del IDE, Google y otras de Java. Para las vistas se utilizaron *layouts* en XML. El almacenamiento se realizó previamente en una base de datos MySQL mediante una API que comunica un *backend* desarrollado en .NET, por lo que para este desarrollo fue necesario el estudio y análisis de dicha base de datos y API para la petición de los recursos o servicios.

Beneficiarios del sistema:

- Ciudadanos de Berazategui
- Personal administrativo de los Centros de Salud
- Personal del área de Salud que accede a la información

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------



Metodología, Tecnologías y Herramientas utilizadas

A continuación se enlistará la metodología, las herramientas y tecnologías que se utilizaron para llevar a cabo el diseño, desarrollo e implementación de la aplicación.

- Metodología de trabajo
- Control de Versiones
- Almacenamiento
- Principio de Programación Orientada a Objetos (POO)
- Java
- Android SO
- Android Studio (IDE)
- Patrón de diseño Adapter
- XML
- Servicio web RESTful
- Postman

Metodología de trabajo

Se estableció la metodología ágil Kanban para llevar a cabo la gestión del proyecto, como conjunto de buenas prácticas y excelentes procesos para obtener el mejor resultado posible, haciendo énfasis en optimizar tareas. Dicha metodología se basa en el desarrollo incremental, que divide las tareas en partes, y se utiliza para controlar el avance del trabajo en un contexto de una línea de producción. Se destaca que no es una técnica específica del desarrollo de software, su objetivo básicamente es gestionar cómo se van completando tareas de la manera más óptima posible.

Kanban es una palabra japonesa y significa "tarjetas visuales", técnica surgida a partir de una metodología llamada Lean, creada por Toyota para la mejora de producción usando técnicas "just-in-time". Está dentro de la estrategia del método Kaisen, que se destaca por utilizar técnicas para el mejoramiento continuo y la gestión de la calidad.

En este proyecto se organizó un tablero utilizando el software Trello, que es una plataforma basada en Kanban con interfaz web y mobile para organizar y administrar proyectos.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------

Principales reglas de Kanban

- Visualizar el trabajo y las fases del ciclo de producción o flujo de trabajo:

Se utilizan técnicas visuales para ver la situación de cada tarea, esto se observa representado en pizarras o tableros compuestos por columnas que constituyen los estados de las tareas y por tarjetas denominadas "post-it", donde se anotan las operaciones a desarrollar con la descripción de la misma y una duración estimada. Si bien está diseñado para el trabajo en grupo, en este caso es individual. A continuación, en la imagen 01 se muestra un tablero con un flujo de trabajo de cinco fases, las mismas utilizadas en el proyecto, pero con tareas enumeradas de ejemplo:

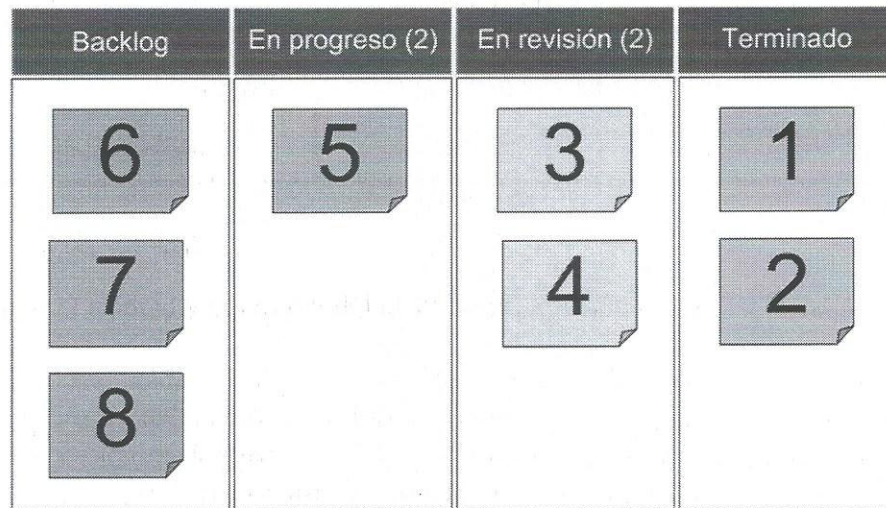


Imagen 01. Tablero Kanban.

Fuente: Elaboración propia, basada en la práctica.

Si bien son comprensibles, se detallan cada una de las fases del flujo de trabajo:

Backlog: es el trabajo acumulado y solicitado por el cliente, las tareas que están listas para ser trasladadas a la fase "en progreso" para que se pueda comenzar a trabajar en ellas.

En progreso: es el trabajo en actual desarrollo, se trata de las tareas de las que se está ocupando el programador de software actualmente.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



En revisión: corresponde a las tareas que contienen algún módulo que necesita pasar por una revisión, pruebas funcionales o unitarias por tener un mal rendimiento, funcionalidad incompleta, con algún error o bug específico.

Terminado: representa las tareas finalizadas y sin errores, concluyendo el flujo de dichas tareas.

- Determinar el límite del Work In Progress (WIP)

Se debe definir cuántas tareas como máximo puede realizarse en cada fase del ciclo de trabajo, con la idea de antes de iniciar una nueva tarea se deba terminar una previa. El número que determina el límite del WIP se visualiza entre paréntesis en el título de la fase que se requiera limitar, como se observa en la imagen 01, donde hay un límite de dos tareas a realizar en simultáneo tanto para "en progreso" como para "en revisión". Dicho WIP se puede ajustar dependiendo del tipo de proyecto, como también del número de los integrantes del equipo de trabajo.

- Estimar o medir el tiempo en completar una tarea

Hay dos métricas para estimar tiempos en Kanban, en este proyecto se utilizó "cycle time" para medir el rendimiento del proceso. Este mide con exactitud el trabajo que conlleva desde que el desarrollador comienza a trabajar en una tarea hasta que termina (desde "en progreso" hasta "terminado"); en cambio, "lead time" mide a partir que la tarea se encuentra disponible en "backlog", hasta que finaliza, siguiendo una traza orientada al tiempo del cliente.

Control de Versiones

GitLab

Es el servicio web de control de versiones y desarrollo de software colaborativo basado en Git seleccionado como repositorio del proyecto en desarrollo, donde se almacenan, organizan, mantienen y sincronizan los archivos del mismo. Lo publicado está bajo una licencia de código abierto, fue programado en Ruby y es propiedad de GitLab Inc., siendo utilizado por organizaciones como la NASA, IBM, el CERN y Sony, entre otros. La conexión a este repositorio se logra mediante el protocolo HTTPS en la rama

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



master, utilizando la consola de git o git Bash. El proyecto está alojado específicamente en el repositorio privado <https://gitlab.com/vituccibruno/turnosSalud.git>

Almacenamiento

El almacenamiento de los datos se encuentra en un servidor alojado en la sala de servidores principal o *datacenter* de la red de fibra óptica de la Municipalidad de Berazategui, Buenos Aires, Argentina. Este servidor cuenta con un *backend* desarrollado con Microsoft .NET y una base de datos relacional MySQL, por medio de los cuales se implementó una API para así poder consumir sus recursos vía HTTP y utilizarlos en cualquier otro entorno de desarrollo, como es el caso del actual proyecto con Android.

La aplicación contará solamente con el rol de “usuario”, quien generará sus turnos en caso de haber disponibilidad, por otro lado, el rol de usuario “administrador” se encuentra gestionando dichos turnos desde cada centro de salud por sus responsables.

Principios de Programación Orientada a Objetos (POO)

Si el programador viene de un ambiente de programación estructurada, la proposición de valores de POO puede que no le sea clara todavía. Después de todo, los atributos de una persona y cualquier lógica para recuperar (y convertir) sus valores pueden escribirse en C o COBOL. Es necesario, entonces, hacer claros los beneficios del paradigma POO al explicar sus principios distintivos: encapsulamiento, herencia y polimorfismo.

Encapsulamiento

Un objeto es diferenciado o independiente, éste es el principio de *encapsulamiento* en funcionamiento. *Ocultación* es otro término que, a veces, se utiliza para expresar la naturaleza independiente y protegida de los objetos; sin tener en cuenta la terminología, lo que es importante es que el objeto mantiene un límite entre su estado y comportamiento y el mundo exterior. Como los objetos del mundo real, los objetos usados en la programación de computadora tienen diversos tipos de relaciones con diferentes categorías de objetos en las aplicaciones que los utilizan.

En la plataforma Java, pueden usarse *especificadores de acceso* para variar la naturaleza de las relaciones de los objetos desde lo *público* a lo *privado*. El acceso

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



público tiene una gran apertura, mientras que el acceso privado significa que los atributos del objeto son accesibles solo dentro del objeto mismo.

El límite entre lo público y lo privado hace cumplir el principio orientado a objetos de encapsulamiento. En la plataforma Java, el desarrollador puede variar la fortaleza de ese límite sobre una base de objeto a objeto, al depender de un sistema de confianza. El encapsulamiento es una potente función del lenguaje Java.

Herencia

En la programación estructurada, es común copiar una estructura, darle un nombre nuevo y agregar o modificar los atributos que hacen que la nueva identidad (por ejemplo, un registro de cuenta) sea diferente de su fuente original. Con el tiempo, este abordaje genera una gran cantidad de códigos duplicados, que pueden crear problemas de mantenimiento.

POO presenta el concepto de *herencia*, por el cual los objetos especializados, sin ningún código adicional, pueden "copiar" los atributos y el comportamiento de los objetos de origen en los que se especializan. Si alguno de esos atributos o comportamientos necesita modificarse, entonces simplemente hay que transformarlos temporalmente. Solo modificar lo que se necesite para crear objetos especializados. El objeto origen se denomina *padre* y la especialización nueva se denomina *hijo*.

Polimorfismo

El polimorfismo es un concepto más difícil de entender que el encapsulamiento o la herencia. Básicamente, significa que los objetos que pertenecen a la misma ramificación de una jerarquía, cuando se envía el mismo mensaje (es decir, cuando se le indica que realice lo mismo) pueden manifestar ese comportamiento de modo diferente.

El polimorfismo es uno de los conceptos más complejos de POO en la plataforma Java y no dentro del ámbito de un tutorial introductorio que ofrezca ejemplificaciones.

Java

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Su creador fue James Gosling en 1991.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



El lenguaje es propiedad de Oracle desde 2010 tras haber comprado a su creador original, Sun. La primera versión era bastante básica, porque se creó para un propósito muy concreto, pero conforme a esto el lenguaje ha ido evolucionando. Las versiones posteriores ofrecen un entorno seguro, al usarse en la web en forma de *applets* y posteriormente en otros entornos, como móviles (incluso antes de Android) o servidores.

Hay muchas aplicaciones y sitios web que no funcionarían a menos que tenga Java instalado. Es rápido, seguro y fiable, desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes.

Java es la base para prácticamente todos los tipos de aplicaciones de red, además del estándar global para desarrollar y distribuir aplicaciones móviles y embebidas (IoT), juegos, contenido basado en web y software de empresa. Permite desarrollar, implementar y utilizar de forma eficaz interesantes aplicaciones y servicios.

La tecnología Java se usa para desarrollar aplicaciones para un amplio alcance de entornos, desde dispositivos del consumidor hasta sistemas empresariales heterogéneos. Algunos datos importantes sobre Java:

- El 97% de los escritorios empresariales lo ejecutan
- El 89% de los escritorios (o computadoras) en Estados Unidos lo ejecutan
- 9 millones de desarrolladores en todo el mundo
- La primera opción para los desarrolladores
- La primera plataforma de desarrollo
- 3 mil millones de teléfonos móviles lo ejecutan
- El 100% de los reproductores de Blu-ray lo incluyen
- 5 mil millones de Java Cards en uso
- 125 millones de dispositivos de televisión lo ejecutan
- 5 de los 5 principales fabricantes de equipos originales utilizan Java ME

El lenguaje de programación Java

El lenguaje para la programación en Java, fue desarrollado por la compañía Sun Microsystems, con la idea original de usarlo para la creación de páginas web. Permite el desarrollo de aplicaciones bajo el esquema de Cliente-Servidor, como de aplicaciones distribuidas, lo que lo hace capaz de conectar dos o más computadoras ejecutando tareas simultáneamente, y de esta forma, lograr distribuir el trabajo a realizar.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------

Como cualquier lenguaje de programación, Java tiene su propia estructura, reglas de sintaxis y paradigma de programación. El paradigma de programación de este lenguaje se basa en el concepto de programación orientada a objetos (POO), que las funciones del lenguaje soportan.

Es un derivado del lenguaje C, por lo que sus reglas de sintaxis se parecen mucho a este: por ejemplo, los bloques de códigos se modularizan en métodos y se delimitan con llaves ({}) y las variables se declaran antes de que se usen.

Estructuralmente, el lenguaje Java comienza con *paquetes*. Un paquete es el mecanismo de espacio de nombres del lenguaje Java. Dentro de los paquetes se encuentran las clases y dentro de las clases se encuentran métodos, variables, constantes, entre otros.

Características del lenguaje

Java es un lenguaje orientado a objetos, eso implica que su concepción es muy próxima a la forma de pensar humana. También posee otras características importantes:

- Es un lenguaje de tipo compilado que genera ficheros de clases del mismo tipo, pero, en realidad, las mismas son interpretadas por la JVM (Java Virtual Machine), siendo ésta la que mantiene el control sobre las clases que se estén ejecutando.
- Es un lenguaje multiplataforma: el mismo código java, que funciona en un sistema operativo, funcionará en cualquier otro que tenga instalada la máquina virtual java.
- Es un lenguaje seguro: la máquina virtual al ejecutar el código java realiza comprobaciones de seguridad, además, el propio lenguaje carece de características inseguras como, por ejemplo, los punteros.
- Gracias al API de java podemos ampliar el lenguaje para que sea capaz de, por ejemplo, comunicarse con equipos mediante red, acceder a bases de datos, crear páginas HTML dinámicas, crear aplicaciones visuales al estilo "window",

Para poder trabajar con Java es necesario emplear un software que permita desarrollar en Java, para esto existen varias alternativas comerciales en el mercado: JBuilder, Visual Age, Visual Café, y un conjunto de herramientas *shareware* e incluso *freeware* que permiten trabajar con Java. Pero todas estas herramientas en realidad se basan en el uso de una herramienta proporcionada por Sun, el creador de Java, que es el Java Development Kit (JDK). A tal fin, se hará énfasis en el uso de dicha herramienta.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------

Teniendo en cuenta que existen diversas versiones del JDK, es posible obtener cualquiera de dichas versiones desde la propia página de Sun: <http://java.sun.com>.

Una vez obtenida la máquina virtual se procedió a realizar la instalación, proceso en el cual será solicitado el directorio en el que serán copiados los ficheros del JDK (directorio raíz JAVA_HOME). El ejemplo estándar de dicho directorio es "/usr/local/jdk". Tras ser instalada se generarán una serie de directorios dentro del directorio mencionado, entre ellos cabe destacar:

- bin: donde se encuentran todos los programas ejecutables del JDK
- lib: contiene las clases del API de java

Una vez efectuado el proceso de instalación fue conveniente realizar la configuración de dos variables de entorno: PATH y CLASSPATH:

- PATH es una variable de entorno que le dice al sistema operativo dónde puede encontrar los programas ejecutables del JDK.
- CLASSPATH es una variable de entorno que le dice a la JVM dónde puede encontrar las clases que se van a emplear o ejecutar.

Estas variables deben ser integrarlas a nivel usuario en el fichero "~/.bashrc" de la siguiente manera:

```
JAVA_HOME="/usr/local/jdk"  
CLASSPATH="/usr/local/lib"  
PATH="$PATH:/usr/local/jdk/bin"  
export JAVA_HOME  
export CLASSPATH  
export PATH
```

El proceso de creación de un programa Java se resume en pocos pasos:

1. Escribir el código fuente correspondiente a las clases a emplear. Los ficheros generados tendrán la extensión ".java"
2. Compilar el código fuente mediante la utilidad "javac.exe" (archivo compilador de Java), este paso generará las clases compiladas en ficheros con extensión ".class"
3. Ejecutar la clase principal, para ello se pasa el nombre de la clase a ejecutar a la aplicación "java.exe".

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



El compilador Java

Cuando se programa para la plataforma Java, se escribe el código de origen en archivos con extensión “.java” y luego los compila. El compilador verifica su código con las reglas de sintaxis del lenguaje, luego escribe los códigos byte en archivos con extensión “.class”. Los códigos byte son instrucciones estándar destinadas a ejecutarse en una JVM. Al agregar este nivel de abstracción, el compilador Java difiere de los otros compiladores de lenguaje, que escriben instrucciones apropiadas para el chipset de la CPU en el que el programa se ejecutará.

Java Virtual Machine (JVM)

Al momento de la ejecución, la JVM lee e interpreta archivos “.class” y ejecuta las instrucciones del programa en la plataforma de hardware nativo para la que se escribió la JVM. La JVM interpreta los códigos byte del mismo modo en que una CPU interpretaría las instrucciones del lenguaje del conjunto. La diferencia es que la JVM es un software escrito específicamente para una plataforma particular. La JVM es el corazón del principio "escrito una vez, ejecutado en cualquier lugar" del lenguaje Java. Su código puede ejecutarse en cualquier chipset para el cual una implementación apropiada de la JVM está disponible. Las JVM están disponibles para plataformas principales como Linux y Windows y se han implementado subconjuntos del lenguaje Java en las JVM para teléfonos móviles y aficionados de chips.

El recolector de basura

En lugar de forzarlo a mantenerse a la par con la asignación de memoria (o usar una biblioteca de terceros para hacer esto), la plataforma Java proporciona una gestión de memoria lista para usar. Cuando su aplicación Java crea una instancia de objeto al momento de ejecución, la JVM asigna automáticamente espacio de memoria para ese objeto desde el *almacenamiento dinámico*, que es una agrupación de memoria reservada para que utilice su programa. El *recolector de basura* Java se ejecuta en segundo plano y realiza un seguimiento de cuáles son los objetos que la aplicación ya no necesita y recupera la memoria que ellos ocupan. Este abordaje al manejo de la memoria se llama *gestión de la memoria implícita* porque no le exige que escriba un determinado código de manejo de la memoria. La recogida de basura es una de las funciones esenciales del rendimiento de la plataforma Java.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------



Java Development Kit (JDK)

Al descargar un kit de desarrollo de Java (JDK), se obtiene (además del compilador y otras herramientas) una librería de clase completa de programas de utilidad pre construidos que lo ayudan a cumplir cualquier tarea común al desarrollo de aplicaciones. El mejor modo para tener una idea del ámbito de los paquetes y bibliotecas JDK es verificar la documentación API JDK. El JDK incluye un conjunto de herramientas de línea de comandos para compilar y ejecutar su código Java, que incluye una copia completa del JRE. Aunque se puede usar estas herramientas para desarrollar aplicaciones, la mayoría de los desarrolladores valoran la funcionalidad adicional, la gestión de tareas y la interfaz visual de un IDE.

Java Runtime Environment (JRE)

El Entorno de Ejecución de Java incluye las bibliotecas de códigos de la JVM y los componentes que son necesarios para programas en ejecución escritos en el lenguaje Java. Está disponible para múltiples plataformas y, además, se puede redistribuir libremente el JRE con las aplicaciones del usuario, de acuerdo a los términos de la licencia del JRE, para darles a los usuarios de la aplicación una plataforma en la cual ejecutar su software. El JRE se incluye en el JDK.

Elección de java

Java ha sido probado, ajustado, ampliado y probado por toda una comunidad de desarrolladores, arquitectos de aplicaciones y entusiastas de Java. Está diseñado para permitir el desarrollo de aplicaciones portátiles de elevado rendimiento para el más amplio rango de plataformas informáticas posible. Al poner a disposición de todo el mundo aplicaciones en entornos heterogéneos, las empresas pueden proporcionar más servicios y mejorar la productividad, las comunicaciones y colaboración del usuario final y reducir drásticamente el costo de propiedad tanto para aplicaciones de usuario como de empresa. Java se ha convertido en un valor impagable para los desarrolladores, ya que nos permite:

- Escribir software en una plataforma y ejecutarla virtualmente en otra
- Crear programas que se puedan ejecutar en un explorador y acceder a servicios Web disponibles
- Desarrollar aplicaciones de servidor para foros en línea, almacenes, encuestas, procesamiento de formularios HTML y mucho más

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------

- Combinar aplicaciones o servicios que utilizan el lenguaje para crear aplicaciones o servicios con un gran nivel de personalización
- Escribir aplicaciones potentes y eficaces para teléfonos móviles, procesadores remotos, microcontroladores, módulos inalámbricos, sensores, gateways, productos de consumo y prácticamente cualquier otro dispositivo electrónico

Actualmente, Java es el lenguaje más popular en el mundo según el índice de la Comunidad de Programación de TIOBE, compañía de calidad de software cuyo índice se actualiza mensualmente y sus calificaciones se basan en un gran número de ingenieros calificados en todo el mundo, cursos y proveedores externos, como también se utilizaron varios motores de búsqueda para calcular dichas calificaciones.

Oct 2018	Oct 2017	Change	Programming Language	Ratings
1	1		Java	17.801%
2	2		C	15.376%
3	3		C++	7.593%
4	5	^	Python	7.156%
5	8	^	Visual Basic .NET	5.884%
6	4	v	C#	3.485%
7	7		PHP	2.794%
8	6	v	JavaScript	2.280%

Imagen 02. Índice de TIOBE 2018 (acotado).

Fuente: Recuperado de <https://www.tiobe.com/tiobe-index> (2018)

Desventajas de Java

La primera desventaja que se puede mencionar es que Java no siempre es recomendable. Por ejemplo, para empezar a programar y para aprender un primer lenguaje de programación y desarrollo, lo más aconsejable es optar por otras opciones que no transmitan malos hábitos de desarrollo y que aporten una experiencia completa, no sólo destinada al desarrollo de objetos, como es Java. No se trataría de una decisión errónea ni poco recomendable, sino que para empezar a desarrollar por primera vez hay otras opciones preferibles, según la experiencia de los mismos profesionales e informáticos.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



Por otro lado, la sintaxis que trae Java resulta difícil y compleja en comparación con otros lenguajes como lo son .NET o C, que además cuentan con una evolución favorable y más beneficiosa que la de Java. Además, este lenguaje y los programas y proyectos realizados con él no pueden ser ejecutados si no se dispone de la JVM que permite visualizar y disfrutar de toda la experiencia de usuario y todo el contenido sin problemas. Esto hace que, en caso de que esté desactivado o inactivo el permiso de Java, no se pueda ver ciertos contenidos o se vuelvan incompatibles para el dispositivo y para el equipo.

Como última desventaja y la más importante para el mercado y para las empresas de hardware y software, se debe mencionar que es un lenguaje interpretado en el que, por su tipología y sus características, cuenta con un rendimiento menor, que hace que los dispositivos y equipos requieran de una mayor potencia y una mayor autonomía para usos similares a los que se tendrían con programas y con software realizados con otros lenguajes de programación.

Ventajas de Java

La primera de las ventajas que se destaca es la facilidad de uso y aprendizaje que conlleva, ya que, al estar orientada a los objetos, no sólo supone una mejora para desarrollo y para la programación, sino que también mejora la forma de pensar de su software y el funcionamiento, aunque, como ya se ha mencionado, el rendimiento sigue siendo su inconveniente. También es una ventaja el hecho de que Java sea independiente de su multiplataforma y de todo lo que ésta engloba, ya que eso le permite emplear el software y ejecutarlo desde cualquier equipo o cualquier dispositivo, sin importar su sistema operativo, o incluso en dispositivos móviles más modernos y en tablets.

Java cuenta con una serie de librerías y opciones que les permiten a los usuarios llegar más lejos y contar con herramientas y utilidades para todo lo que deseen y para todo lo que requieran en cualquier momento. Esta es una de las principales características de Java que los usuarios distinguen y que hace que se destaque en el mercado y en el mundo de los lenguajes de programación y la informática. Su librería se llama Java API e incluye tres bloques básicos para todos los usuarios.

Otra ventaja es que existen ciertos editores (IDE's) de gran calidad y excelencia, que les permite a los programadores y desarrolladores hacer más por sus proyectos, a un ritmo más fluido y con un trabajo más cómodo. La última ventaja a destacar es la solución de errores y fallos a través del lenguaje de Java que ofrece, permitiendo revisar las

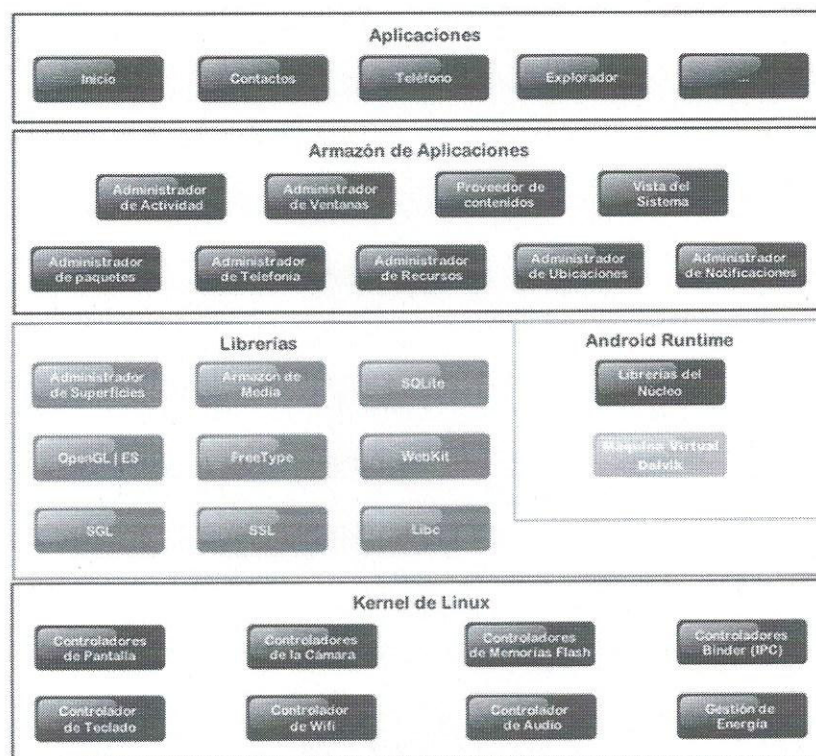
Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------

creaciones y corregirlas. En otros lenguajes como C o C++ no existe ese tipo de herramientas y soluciones, lo que hace que Java cuente con una ventaja competitiva evidente y destacable.

Android SO

Es un sistema operativo basado en el núcleo Linux, desarrollado por Android Inc. y respaldado económicamente por Google, cuando en 2005 realizó su compra. El propósito del sistema fue utilizarse para dispositivos móviles con pantalla táctil (Smartphones, tablets), aunque más tarde se expandió a Smart Watches, Smart TV's, automóviles y otros. Fue presentado en 2007 junto a la fundación Open Handset Alliance, que es un consorcio de compañías de hardware, software y telecomunicaciones, para la mejora de estándares de dispositivos móviles. Actualmente, Android es el sistema operativo móvil más utilizado en todo el mundo, con una cuota de mercado que supera el 80%, data al año 2017.

En la imagen 03 se detalla su estructura en capas:



Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



Imagen 03. Componentes principales de Android SO.

Fuente: Recuperado de <http://www.javahispano.org/android/2011/10/8/android-un-vistazo-rapido.html> (2018)

Android Studio

Es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones Android y se basa en IntelliJ IDEA, uno de los primeros IDE Java disponibles que fue desarrollado por JetBrains. Como dato interesante, en un informe de Infoworld (medio de comunicación online y negocio de eventos dedicado a Tecnologías de Información) en 2010 IntelliJ recibió la puntuación más alta entre las cuatro mejores herramientas de programación de Java (Eclipse, IntelliJ IDEA, NetBeans y Oracle JDeveloper). Android Studio es el IDE oficial para la plataforma Android, se anunció en 2013, su primera versión estable fue publicada en 2014 y está disponible para los sistemas operativos Microsoft Windows, macOS y GNU/Linux. Este entorno reemplazó a Eclipse como IDE oficial para el desarrollo de aplicaciones Android.

Funciones y Características

Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones y características que aumentan la productividad del desarrollador durante la compilación de apps para Android, como las siguientes:

- Un sistema soporte de compilación basado en Gradle flexible
- Un emulador rápido con varias funciones
- Refactorización específica de Android y soluciones rápidas
- Un entorno unificado en el que se puede realizar desarrollos para todos los dispositivos Android
- Instant Run para aplicar cambios mientras la app se ejecuta sin la necesidad de compilar un nuevo APK
- Dispositivo virtual Android (emulador) para ejecutar y depurar aplicaciones
- Asistentes para integración de plantillas de código para crear diseños y GitHub para ayudar a compilar funciones comunes de las apps e importar ejemplos de código
- Gran cantidad de herramientas y frameworks de prueba
- Soporte para la construcción de aplicaciones Android Wear

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



- Un editor de diseño que permite a los usuarios arrastrar y soltar componentes de UI, opción para previsualizar diseños en configuraciones de pantalla múltiple
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versión, etc.
- Compatibilidad con C++ y NDK
- Integración de ProGuard y capacidades de firma de aplicaciones
- Soporte incorporado para Google Cloud Platform, que permite la integración con Firebase Cloud Messaging (antes 'Google Cloud Messaging') y Google App Engine

Android Studio admite los mismos lenguajes de programación de IntelliJ y PyCharm, por ejemplo, Python y Kotlin; además Android Studio 3.0 es compatible con "las características del lenguaje Java 7 y un subconjunto de las características del lenguaje Java 8 que varían según la versión de la plataforma". Asimismo hay proyectos externos que respaldan algunas características de Java 9.

El núcleo de Android Studio se ha actualizado recientemente con mejoras de IntelliJ IDEA a través de la versión 2017.3.3. Las mejoras incluyen un mejor análisis de flujo de control para colecciones y cadenas, inferencia de nulabilidad mejorada, nuevas soluciones rápidas y mucho más.

Android Studio incluye las SDK Tools, pero si el desarrollador quiere utilizar otro IDE, puede descargar las SDK Tools del sistema operativo con el cual trabaja y utilizar el SDK manager provisto para descargar otros paquetes de SDK que necesite.

Requisitos

La desventaja principal de este IDE es que consume bastantes recursos del sistema y los requisitos para utilizarlo son muy altos, por lo que se requiere de un potente equipo donde instalarlo. A continuación, en la imagen 04 se muestran los requisitos:

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------

	Windows	OS X/macOS	Linux
OS version	Windows 10/8/7 (32- o 64-bit)	Mac OS X 10.10 (Yosemite) o superior, hasta 10.13 (macOS High Sierra)	GNOME o KDE desktop
RAM	3 GB RAM mínimo, 8 GB RAM recomendado más 1GB adicional para el emulador de Android		
Espacio en disco	2 GB de espacio en disco para Android Studio, 4GB recomendados (500MB para la IDE y al menos 1.5 GB para Android SDK, imágenes de sistema de emulador y cachés)		
Java version	Java Development Kit (JDK) 8		
Resolución de pantalla	1280x800 mínimo, 1440x900 recomendado		

Imagen 04. Requisitos para la utilización del IDE Android Studio.
Fuente: Recuperado de https://es.wikipedia.org/wiki/Android_Studio (2018)

Cambios a partir de Android N

Como una plataforma de código abierto, Android se basa en la colaboración de la comunidad open-source. A partir de Android N (Android 7.0 Nougat), lanzado en el año 2016, se sustituyen las librerías del lenguaje Java de Android por un enfoque basado en OpenJDK para evitarse problemas legales con Oracle, actual propietario de Java, creando una base de código común para que los desarrolladores creen aplicaciones y servicios. Google ha trabajado durante mucho tiempo con él, como también ha contribuido a la comunidad OpenJDK y, con este cambio de paradigma, se espera que se realicen aún más colaboraciones al proyecto, por lo que Google ya no usará más las API's Java de Oracle, en favor de OpenJDK, la versión de código abierto del Java Development Kit.

Este cambio provoca migrar código base de Android a OpenJDK y es un beneficio transparente a Java, que lo refuerza frente a otras soluciones. De esta manera, se puede trasladar de Java 7 no sólo hacia Java 8, sino a alcanzar Java 9 de forma equivalente a otras plataformas, como también implica obtener un mejor rendimiento de la máquina virtual de Android.

Gradle

Es un sistema de automatización de compilación de código abierto que se basa en los conceptos de Apache Ant y Apache Maven e introduce un lenguaje específico de dominio basado en Groovy (DSL) en lugar del formato XML utilizado por Apache Maven para declarar la configuración del proyecto. Gradle usa un grafo acíclico dirigido ("DAG") para determinar el orden en que se pueden ejecutar las tareas.

Se diseñó para compilaciones multi-proyecto que pueden llegar a ser amplias. Admite compilaciones incrementales al determinar inteligentemente qué partes

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



del árbol de compilación están actualizadas y cualquier tarea que dependa sólo de esas partes no necesita ser re-ejecutada.

Máquina Virtual DALVIK

Para ingresar en la temática de DALVIK (surgido de Apache Harmony, software modular de tiempo de ejecución de Java con bibliotecas de clases y herramientas asociadas), primero es importante comprender la relación que tiene Java con Android. En realidad, JAVA en sí mismo tiene poca relación con Android. El funcionamiento de las aplicaciones en Android, de cualquier tipo (fuera del propio sistema operativo), hace que éstas corran sobre una máquina virtual (MV) llamada DALVIK. Esta máquina es la responsable de que no sea necesario diseñar las aplicaciones específicamente para cada teléfono (aunque se preste atención a elementos como ser la pantalla, para que todo funcione de mejor manera y sea adaptable), sino que solamente haya que diseñarlos para Android, y que sea ésta quien se encargue de comunicarle al sistema qué dispositivos deben ser utilizados en cada momento.

Esta MV, aunque es compatible con JAVA, no es específicamente la misma plataforma, pero al momento de desarrollar es prácticamente similar para muchas cuestiones e incluso existen múltiples herramientas para realizar conversiones de JAVA a Android.

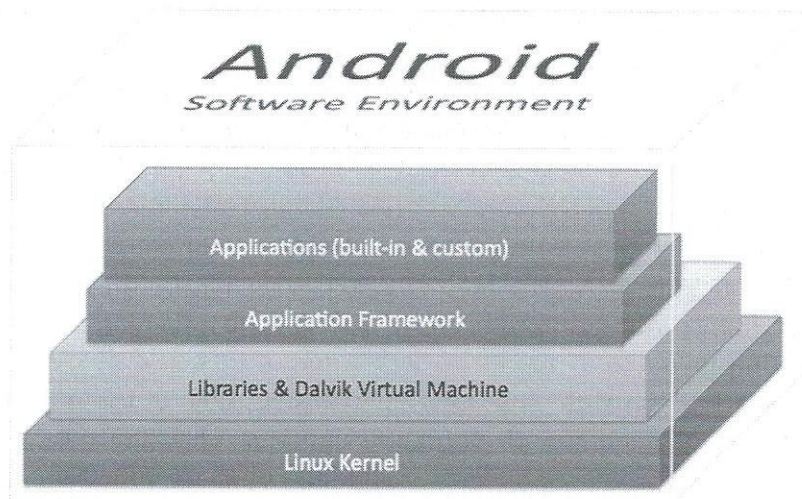


Imagen 05. Capas del sistema operativo Android, donde Dalvik que hace de intermediario con el núcleo y el entorno de aplicaciones.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------

Fuente: Recuperado de <https://www.slashgear.com/oracle-targets-dalvik-vm-as-google-case-continues-08226903> (2012)

La máquina virtual (MV) es un intermediario entre las aplicaciones y el sistema núcleo, como se observa en la imagen 03. Cuando aparece una nueva plataforma hardware, hay que adaptar DALVIK para que funcione en ella y automáticamente las aplicaciones que existían con anterioridad comiencen a funcionar en ésta también. Es decir, esta arquitectura es la diferencia entre tener que preparar aplicaciones para todas las plataformas de hardware que soporta Android a desarrollar específicamente para Android. Es lo que permite que sea una plataforma altamente compatible y que se pueda aprender con rapidez permitiendo, además, que las aplicaciones lleguen a una gran cantidad y variedad de dispositivos. Esa es la principal diferencia entre Android y otras plataformas y, al mismo tiempo, la principal razón de que tenga gran extensión.

Desventaja

El problema que tiene el usar DALVIK como MV, como se ha mencionado en el apartado anterior, es que funciona como intermediario. Como es usual en añadir intermediarios hace que las tareas que se pretendan resolver requieran de mayor cantidad de recursos para ello. En el caso de las MV (como JAVA o DALVIK), estas plataformas necesitan recursos hardware del sistema para funcionar, así como otros recursos hardware para poder ejecutar las aplicaciones. Sin embargo, las soluciones nativas (como por ejemplo las de iOS) solamente requieren los recursos del propio sistema, permitiendo con ello que se obtengan resultados esperables con un hardware menos potente, u óptimos resultados con el mismo.

Ventaja

Si bien es cierto que las MV restan eficiencia a la resolución final de cada tarea, tienen una notoria ventaja sobre las tecnologías nativas, pues permiten trabajar sobre muchas y variadas arquitecturas. Es posible que precedentemente se haya planteado un cambio de filosofía para un ecosistema como Android, pero en la actualidad y debido al éxito obtenido, no presenta alternativa. Además, las mejoras en el propio hardware unidas a los avances en la propia MV hacen que, si bien se sigue notando la MV, cada vez afecte menos a la experiencia de usuario, por lo que a menos que se desarrollen otras tecnologías con mayores ventajas, DALVIK seguirá aún en vigencia.

Alternativas

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------



Kotlin

Es un lenguaje de programación desarrollado por JetBrains, compañía responsable de IntelliJ, el IDE en el que se basa Android Studio. Su idea inicial era sustituir a Java en los proyectos desarrollados por la compañía, por lo que era fundamental basarse en la JVM para poder ser totalmente interoperable a Java y su ecosistema. Kotlin es un lenguaje de tipado estático que corre sobre la JVM y que también puede ser compilado a código fuente de JavaScript. No tiene una sintaxis compatible con Java, pero está diseñado para interoperar con código Java y es dependiente del mismo de su biblioteca de clases, tal como lo es el JCF (Java Collections Framework).

Además, es un lenguaje sencillo, potente y pragmático. La curva de aprendizaje es bastante más ligera que otros lenguajes como Scala, por ejemplo, lo que apoya su adopción. Se considera interoperable enteramente con Java, por lo que cualquier código escrito en Java se puede utilizar directamente desde Kotlin.

Fue anunciado en Google IO 2017 (Congreso de desarrolladores organizado anualmente por Google para presentar y discutir las aplicaciones de Google y las tecnologías abiertas de Internet) donde la empresa le daba el apoyo definitivo a Kotlin en Android, por lo que desde ese entonces se adoptó como lenguaje oficial en Android al mismo nivel que Java. Este acuerdo implica una colaboración más estrecha entre Google y Jetbrain para mejorar el ecosistema de Android con Kotlin, al igual que se venía haciendo con el IDE de desarrollo. De esta manera, se posibilita trabajar con un lenguaje de programación más moderno y robusto, lo que permitirá olvidar paulatinamente las limitaciones del uso obligatorio de Java 6 en Android, y con la ventaja de poder interoperar con Java y las librerías actuales sin inconvenientes y sin perder el soporte en Android.

Para que Android pudiera adoptarlo, el lenguaje debía ser ligero para poder ejecutarse en cualquier dispositivo sin sobrepasar el peso específico de las APK's en Android. Lo destacable de este lenguaje es que va más allá de Android, pudiendo desarrollar en JavaScript mediante Kotlin JS, como también programar aplicaciones para iOS con Kotlin/Native. No existe problema en las versiones mínimas de Android que podrán soportar esta mejora con Kotlin, ya que es compatible con JDK 6, como ya se mencionó, y puede ejecutarse en versiones antiguas de Android sin inconvenientes.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------

Kotlin puede ser el futuro, incluso puede dejar en el pasado a Java y la JVM con lo nuevo que desarrolla Google, como el desarrollado paralelamente a Android y basado en el mismo, Fuchsia SO, una especie de Android IoT (Internet of Things), el cual es también una oportunidad para Kotlin.

Integraciones de Kotlin para IDE's:

- Android Studio soporta este lenguaje desde su versión 3.0. (octubre de 2017).
- IntelliJ IDEA y NetBeans tienen un plugin de soporte para Kotlin.
- JetBrains también tiene un plugin de Kotlin para Eclipse.

Xamarin (C#)

Es una compañía de software adquirida por Microsoft. Con un código compartido del lenguaje de programación C#, los desarrolladores de software pueden usar Xamarin para escribir aplicaciones móviles nativas para Android, iOS y Windows, y compartir código a través de múltiples plataformas, incluyendo Windows y macOS. C# es uno de los lenguajes preferidos de Microsoft, lo que hace que sea bastante popular dentro de los programadores .NET.

Es necesario saber que existe una gran desventaja cuando se crean aplicaciones móviles. Cuando se crean aplicaciones iOS es preciso escribir código en Objective-C y si se quiere crear Apps Android se debe conocer Java. Lo que hace Xamarin es unificar estas diferencias con el IDE Xamarin Studio, ya que solo se necesita dominar C# a la hora de crear aplicaciones iOS, Android y Windows Phone.

Proyecto Xobot OS

Es un proyecto de investigación que propone re implementar el código de Android, de forma que todo lo que se ejecutaba sobre DALVIK (máquina virtual basada en Java) ahora lo hace sobre Mono (máquina virtual basada en C#) donde las pruebas realizadas han marcado grandes diferencias en velocidad y consumo de batería, los principales puntos fuertes de Android. El código del proyecto está publicado por desarrolladores de Xamarin en <https://github.com/xamarin/XobotOS> para ser probado de forma gratuita.

Xamarin Studio (XS):

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



Como se mencionó anteriormente, XS aporta el entorno de desarrollo perfecto para que los programadores que dominan C# tengan las herramientas disponibles para migrar sus aplicaciones a iOS, Android y Windows Phone. Es necesario descargar el IDE desde su página oficial, debiendo saber que su licencia no es gratis, pero está disponible para instalar la herramienta durante 29 días con un *trial* para PC o Mac.

Xamarin para Visual Studio:

En el caso de que se haya adquirido el IDE de Microsoft Visual Studio se puede descargar Xamarin para añadirlo a esta plataforma.

Desarrollos para Android/iOS/Windows Phone:

Es preciso saber que Xamarin Studio usa los recursos nativos de cada plataforma, por lo que si se requiere crear aplicaciones para iOS uno de sus requisitos previos es contar con un sistema Mac OS X, donde la Mac enlazará con Windows para compilar el proyecto que se esté creando en ese momento. Como novedad de Xamarin, desde un único lenguaje de programación (C#) se pueden hacer aplicaciones para Android, iOS y Windows Phone. Respecto a esto, hay que mencionar que en cada plataforma donde se quiera hacer un desarrollo es necesario obtener sus correspondientes herramientas.

Estructura de una aplicación desarrollada con Xamarin en comparación con el tradicional Android Studio:

- Proyecto móvil. El concepto de proyecto es el mismo, en ambos casos un proyecto se refiere a una sola aplicación.
- AndroidManifest.xml. Este archivo en el que se define gran parte de la configuración del proyecto, se mantiene idéntico en ambas opciones de desarrollo. Sin embargo, cuando se utiliza Xamarin se puede trasladar parte de esta configuración al código.
- Recursos. Los recursos se organizan de la misma manera (en carpetas dependiendo de su resolución/tamaño), solo que mientras que en Xamarin la carpeta se llama *Resources*, en el tradicional IDE con Java se llama *res*.
- Layouts. Tanto el archivo *Main.axml* como el *activity_main.xml* tienen prácticamente el mismo contenido y de hecho son compatibles entre ellos. La diferencia en las extensiones es para que Visual Studio lo reconozca como un

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------

archivo de interfaz gráfica para Android y pueda dar uso de Microsoft Intellisense (aplicación de autocompletado).

- Drawables por resolución. A diferencia de iOS, en donde se distingue un recurso de otro mediante los sufijos `@x</code>`, en Android estos deben ir en carpetas separadas totalmente, y en Xamarin no es la excepción.
- Drawables en xml. En este caso el archivo `empty_divider.xml` tiene el mismo contenido, para Xamarin el archivo contiene un divisor transparente para el `StackView`.
- Recursos en xml. Cualquier recurso como `strings`, `colors`, `values` es empleado de la misma manera.
- Biblioteca de lógica externa. En ambos casos la lógica de nuestra aplicación puede ser separada en un proyecto externo, en ambos casos, una librería. Cabe mencionar que usando C# se crea una biblioteca de clases portable y comparte este código con apps de iOS u otras plataformas.
- Código fuente en general. El código se guarda en clases.

Recursos:

Los recursos se manejan de manera similar, los muestra fuera de las carpetas y con una anotación indicando a qué resolución pertenecen. En cuanto a la clase `R.java` de Android Studio, en Xamarin.Android se llama `Resource` y su contenido es similar al que contiene en la tradicional, como se observa en la imagen 06:

El código:

<pre>using Android.Support.V7.App; using Android.OS; using Android.Widget; using Android.App; using Xevensthein.Core; namespace Xevensthein { [Activity(MainLauncher = true)] <small>0 references Antonio Feregino, 3 days ago 1 author, 3 changes</small> public class MainActivity : AppCompatActivity {</pre>	C#	<pre>package com.thatcsharpguy.xevensthein.java; import android.support.v7.app.*; import android.os.*; import android.widget.*; import android.view.View; import com.thatcsharpguy.xevenstheincore.LevenstheinDistance; public class MainActivity extends AppCompatActivity {</pre>	Java
--	----	--	------

Imagen 06. Comparación de `MainActivity` en C# y Java.

Fuente: Recuperado de <https://thatcsharpguy.com/post/xamarin-android-vs-traditional-java/> (2016)

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------

Desde aquí se revelan algunas de las diferencias entre el código de C# y Java, por ejemplo, se puede ver que *using* \approx *import*, o que el *namespace* es un equivalente de *package* y que las clases se llaman de la misma forma. Algo importante a destacar es el código [*Activity(MainLauncher = true)*] que cumple la función de indicarle al *manifest* cuál es el punto de inicio al ejecutarse la app.

OnCreate

En cuanto al método *OnCreate* y la declaración *Views*, las diferencias son mínimas, en C# para hacer referencia a la clase de la que se hereda se usa la palabra *base* en lugar de *super*. Por convención los métodos comienzan con mayúscula. Aquí es donde se ven las similitudes entre la clase *R* y *Resource* para acceder a los recursos.

FindViewById

Una vez creada la vista se puede acceder a los controles y referenciarlos desde la clase *Activity*, para esto se utiliza el método *FindViewById*, mientras que con Xamarin está disponible la opción de hacer el *cast* por nuestra cuenta (como en Java), también es posible usar el método genérico *FindViewById* que hará el *cast* por el desarrollador. La imagen 07 presenta ambas vistas:

Métodos y manejadores de evento

```

_compareButton.Click += (sender, e) => {
    var res = LevenshteinDistance.Compute(
        _firstWordEditText.Text,
        _secondWordEditText.Text);
    _resultTextView.Text = res.ToString();
};

```

C#

```

_compareButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        int res = LevenshteinDistance.Compute(
            _firstWordEditText.getText().toString(),
            _secondWordEditText.getText().toString());
        _resultTextView.setText(Integer.toString(res));
    }
});

```

Java

Imagen 07. Comparación de métodos y manejadores de eventos en C# y Java.

Fuente: Recuperado de <https://thatcsharpguy.com/post/xamarin-android-vs-traditional-java/> (2016)

En cuanto a los métodos, la declaración es casi idéntica (y más con Java 8) ya que mientras que en C# se usa una expresión *lambda* en la que se define el comportamiento del botón, en Java se utiliza una clase anónima para realizar esta acción.

Código fuente en general

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------

En cuanto al código, la traducción entre C# y Java no es muy complicada ya que son muy similares, una de las mayores diferencias es que en Java se tienen únicamente métodos para acceder a las propiedades de un objeto como es el caso de *length()*.

Interfaz gráfica

El editor de interfaces para Android que contiene Visual Studio o Xamarin Studio es idéntico al disponible en los editores de Android Studio o en Eclipse. Además de la vista “en vivo” del diseño, también ambos dan acceso al XML del archivo, según se muestra en la imagen 08:

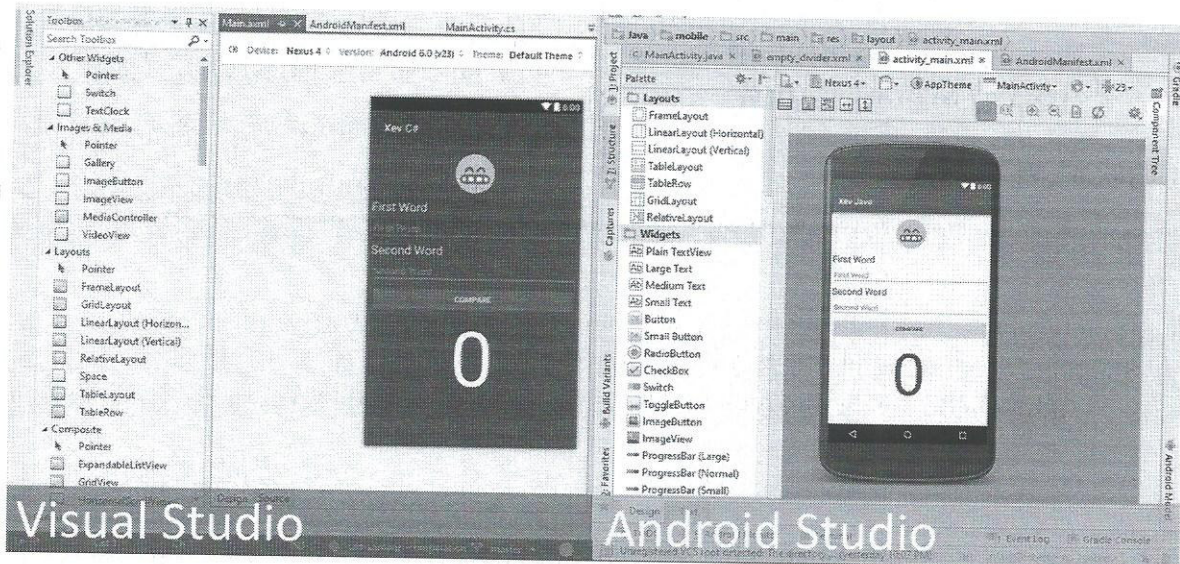


Imagen 08. Comparación de editor de interfaz gráfica en C# y Java.

Fuente: Recuperado de <https://thatcsharpguy.com/post/xamarin-android-vs-traditional-java/> (2016)

En conclusión, las diferencias entre C# y Java no son notorias y resulta conveniente comenzar a programar con Xamarin para así poder trasladar la app a iOS y Windows Phone sin mayores problemas.

Eclipse

Eclipse es de código abierto para el desarrollo Java y fue el IDE oficial para el desarrollo Android, y actualmente es una alternativa. Maneja las tareas básicas, tales

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



como la compilación y la configuración de un entorno de depuración, para que pueda centrarse en escribir y probar códigos. Además, se puede usar Eclipse para organizar archivos de origen en proyectos, compilar y probar esos proyectos y almacenar archivos de proyectos en cualquier cantidad de repositorios de origen. Se necesita tener instalado un JDK para usar Eclipse para el desarrollo Java y el SDK de Android para programar aplicaciones para ese sistema.

Patrón de Diseño Adapter

También llamado "wrapper" (envoltorio en inglés), este patrón de diseño estructural es muy utilizado en Java y, de hecho, hasta en su propia API se ha incluido para el manejo de eventos de interfaces gráficas. Los patrones estructurales se basan en la forma en la que un conjunto de clases se relaciona entre sí para proporcionar una funcionalidad compleja, proporcionando una estructura para conseguir lograr ese objetivo.

Adapter se puede utilizar en muchos contextos y es de especial utilidad cuando se utilizan códigos o librerías ajenos al que el desarrollador está utilizando y sobre el que no tiene control. Como su nombre lo indica, trata de adaptar una clase para que no implemente todos los métodos declarados en una interfaz, esto lo hace usando una clase que la interfaz en cuestión y la clase que utiliza el adaptador debe sólo implementar el método que necesite o requiera.

Establece una capa intermedia que permite comunicar dos clases que de otro modo no podrían hacerlo, realizando una adaptación de la interfaz de la clase que proporciona el servicio a la que la solicita. Para ello, un objeto adaptador reenvía al otro objeto los datos que recibe (a través de los métodos que implementa, definidos en una clase abstracta o interface) tras manipularlos en caso necesario como se observa en la imagen 09.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------

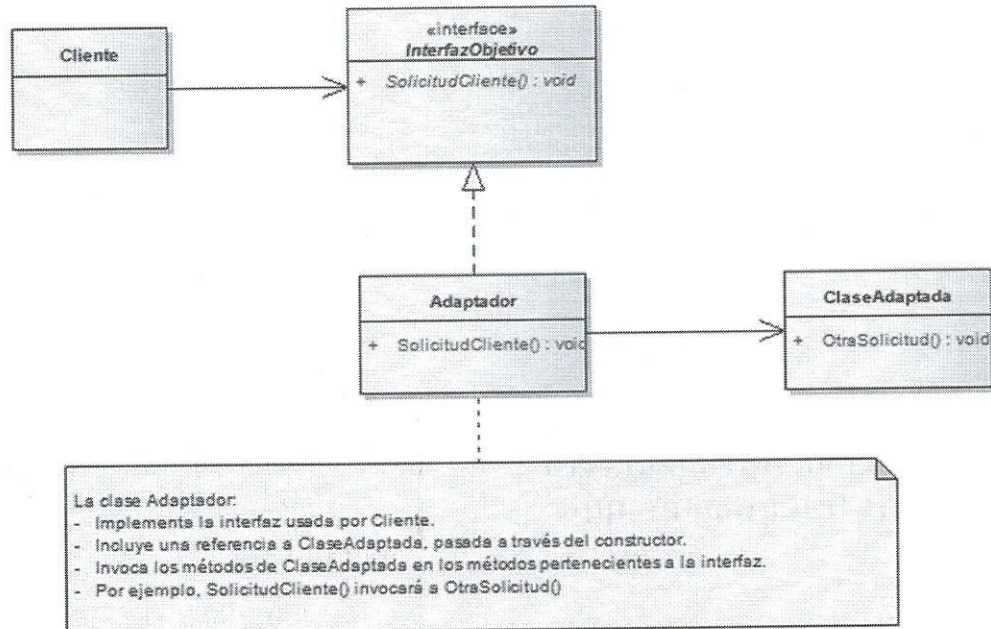


Imagen 09. Diagrama del Patrón Adapter.

Fuente: Recuperado de <https://danielggarcia.wordpress.com/2014/02/28/patrones-estructurales-i-patron-adapter-wrapper/> (2014).

Como se mencionó, se utiliza para transformar una interfaz en otra, de tal modo que una clase que no pueda utilizar la primera haga uso de ella a través de la segunda. Convierte la interfaz de una clase en otra interfaz que el cliente espera. Este patrón es muy eficaz como también sencillo, básicamente permite que trabajen juntas clases con interfaces incompatibles.

El escenario de este patrón es también bastante claro. Se utilizará en los casos en los que sea necesario usar una clase cuya interfaz no se adapte a los requerimientos de otra clase cliente. Se puede ver algunos ejemplos reales cuando se realizan transformaciones entre enumeraciones e iteradores, arrays y listas, etc.

El lenguaje Java, por ejemplo, ofrece los siguientes usos del patrón *Adapter*:

- `java.util.Arrays#asList()`
- `java.io.InputStreamReader(InputStream)` (devuelve un Reader)
- `java.io.OutputStreamWriter(OutputStream)` (devuelve un Writer)

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



- `javax.xml.bind.annotation.adapters.XmlAdapter#marshal()`
- `javax.xml.bind.annotation.adapters.XmlAdapter#unmarshal()`

XML (eXtensible Markup Language)

Es un lenguaje que se utiliza para decir algo acerca de otro, por lo que se denomina “metalenguaje” extensible de etiquetas. Fue desarrollado por el W3C (World Wide Web Consortium) para su uso en la WWW. Este lenguaje es una adaptación del Standard Generalized Markup Language (SGML). Básicamente, es un estándar que estructura y representa datos mediante el intercambio de información entre diferentes plataformas y aplicaciones, y tiene muchos campos de aplicación.

Facilita la organización de los recursos y la configuración de un programa, por ejemplo, cuando se desarrolla un programa con interfaz gráfica es necesario organizar todas las imágenes de manera que se vayan cargando y renderizando a medida que se necesiten, y en este aspecto XML es una muy buena opción. Permite agrupar, etiquetar, y especificar la ubicación de las imágenes y relacionarlas con otros datos, según las necesidades del diseñador.

Su uso en SDK de Android

Siendo uno de los estándares más utilizados para codificar información, XML es ampliamente utilizado en internet, servicios web, como también puede ser utilizado para múltiples usos en el SDK de Android. Es empleado para definir layouts, animaciones, spinners selectores, AndroidManifest.xml, entre otros.

Una gran fortaleza de la plataforma Android es que aprovecha el lenguaje Java y sus librerías, este dispone de una cantidad considerable de API's para trabajar con XML, sin embargo, no todas están disponibles para Android.

Librerías disponibles:

- Java's Simple API for XML (SAX) (paquetes `org.xml.sax.*`)
- Document Object Model (DOM) (paquetes `org.w3c.dom.*`)

Librerías no disponibles:

- Streaming API for XML (StAX). Aunque se dispone de otra librería con funcionalidad equivalente (paquete `org.xmlpull.v1.XmlPullParser`).

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



- Java Architecture for XML Binding (JAXB). Resultaría demasiada pesada para Android.

Diseño UI en Android

Un diseño define la estructura visual para una interfaz de usuario UI. En Android se puede declarar un diseño de dos maneras:

- Declarar elementos de la IU en XML. Android proporciona un vocabulario XML simple que coincide con las clases y subclases de vistas, como las que se usan para widgets y diseños.
- Crear una instancia de elementos del diseño en tiempo de ejecución. La aplicación que se desarrolle puede crear objetos *View* y *ViewGroup* (y manipular sus propiedades) programáticamente.

El framework de Android ofrece la flexibilidad de usar uno de los métodos mencionados anteriormente, o ambos para declarar y administrar la IU de la aplicación en desarrollo. Por ejemplo, se puede declarar los diseños predeterminados de la aplicación en XML, incluidos los elementos de pantalla que aparecerán en ellos y sus propiedades. Luego se puede agregar código en la aplicación para modificar el estado de los objetos de la pantalla, incluidos los declarados en XML, en tiempo de ejecución.

La ventaja de declarar la IU en XML es que permite separar mejor la presentación de la aplicación del código que controla su comportamiento. Las descripciones de la IU son externas al código de la aplicación, lo que significa que se puede modificar o adaptar sin tener que modificar el código fuente y volver a compilar. Por ejemplo, se puede crear diseños XML para diferentes orientaciones de pantalla, diferentes tamaños de pantalla de dispositivos y diferentes idiomas. Además, declarar el diseño en XML facilita la visualización de la estructura de la IU, de modo que sea más fácil depurar problemas.

En general, el vocabulario XML para declarar elementos de la IU sigue de cerca la estructura y la denominación de las clases y los métodos, en los que los nombres de los elementos coinciden con los nombres de las clases y los nombres de los atributos coinciden con los métodos. De hecho, la correspondencia generalmente es tan directa que se puede deducir qué atributo XML corresponde a un método de clase, o deducir qué

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



clase corresponde a un elemento XML determinado. No obstante, hay que tener en cuenta que no todo el vocabulario es idéntico, ya que en algunos casos hay pequeñas diferencias de denominación.

Escribir en XML

Al usar vocabulario XML de Android, se puede crear rápidamente diseños de IU y de los elementos de pantalla que contienen, de la misma manera que se crean páginas web en HTML, con una serie de elementos anidados.

Cada archivo de diseño debe contener exactamente un elemento raíz, que debe ser un objeto *View* o *ViewGroup*. Una vez que se haya definido el elemento raíz, se puede agregar *widgets* u objetos de diseño adicionales como elementos secundarios para crear gradualmente una jerarquía de vistas que defina el diseño propio del programador.

Después de declarar el diseño en XML, se debe guardar el archivo con la extensión “.xml” en el directorio “res/layout” del proyecto de Android para que pueda compilarse correctamente.

Carga el recurso XML

Cuando se está dispuesto a compilar la aplicación, cada archivo de diseño XML se compila en un recurso *View*. Se debe cargar el recurso de diseño desde el código de la aplicación, en la implementación de *callbackActivity.onCreate()*. Para hacerlo, es necesario llamar a *setContentview()*, pasarle la referencia al recurso de diseño en forma de *R.layout.nombre_del_layout*, como en el siguiente ejemplo:

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.nombre_del_layout);  
}
```

El framework de Android llama al método *callback onCreate()* en la activity cuando se lanza la activity en cuestión.

Atributos

Todos los objetos *View* y *ViewGroup* admiten su propia variedad de atributos XML. Algunos atributos son específicos para un objeto *View* (por ejemplo, *TextView* admite el atributo *textSize*), pero a esos atributos también los heredan otros objetos *View* que

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



podrían extender esta clase. Algunos son comunes para todos los objetos *View* ya que se heredan desde la clase *View* raíz (como el atributo *id*). Otros atributos se consideran "parámetros de diseño" y son atributos que describen ciertas orientaciones de diseño del objeto *View*, tal como lo define el objeto principal *ViewGroup* de ese objeto.

Identificador (ID)

Cualquier objeto *View* puede tener un ID con número entero asociado a él para identificar de forma exclusiva la vista en un árbol. Cuando se compila la aplicación, este ID se considera un número entero, pero el ID generalmente se asigna en el archivo XML del diseño como un tipo "string", en el atributo ID. Este es un atributo XML común a todos los objetos *View* (definidos por la clase *View*) y se usa con mucha frecuencia. La sintaxis para un ID dentro de una etiqueta XML es la siguiente:

```
android:id="@+id/mi_boton"
```

El símbolo arroba (@) al comienzo del tipo "string" indica que el analizador de XML debe analizar y expandir el resto del "string" de ID e identificarlo como un recurso de ID. El símbolo más (+) significa que es un nuevo nombre de recurso que se debe crear y agregar a nuestros recursos (en el archivo R.java). El framework de Android ofrece otros recursos de ID. Al hacer referencia a un ID de recurso de Android, no se necesita el símbolo (+), pero es necesario agregar el espacio de nombres de paquete "android" de la siguiente manera:

```
android:id="@android:id/vacio"
```

Con el espacio de nombres de paquete "android" establecido, continua la referencia a un ID de la clase de recursos "android.R", en lugar de la clase de recursos local.

Para crear vistas y hacer referencia a ellas desde la aplicación, se puede seguir este patrón común:

1. Definir una vista/widget en el archivo de diseño y asignarle un ID único:

```
<Button android:id="@+id/my_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/my_button_text"/>
```

2. Luego, crear una instancia del objeto *View* y capturarlo desde el diseño (generalmente en el método *onCreate()*):

```
Button mi_boton = (Button) findViewById(R.id.mi_boton);
```

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:

Definir ID para objetos *View* es importante cuando se crea un *RelativeLayout*. En un diseño relativo, las vistas del mismo nivel pueden definir su diseño en función de otra vista del mismo nivel, que se identifica con un ID único.

No es necesario que un ID sea único en todo el árbol, pero debe ser único dentro de la parte del árbol en la que se está buscando (que a menudo puede ser el árbol completo, por lo que es mejor que, en lo posible, sea totalmente único).

Parámetros de diseño

Los atributos de diseño XML denominados *layout_algo* definen parámetros de diseño para el objeto *View* que son adecuados para el objeto *ViewGroup* en el que reside.

Cada clase *ViewGroup* implementa una clase anidada que extiende *ViewGroup.LayoutParams*. Esta subclase contiene tipos de propiedad que definen el tamaño y la posición de cada vista secundaria, según resulte apropiado para el grupo de vistas. Como se observa en la imagen 10, el grupo de vistas principal define parámetros de diseño para cada vista secundaria (incluido el grupo de vistas secundario).

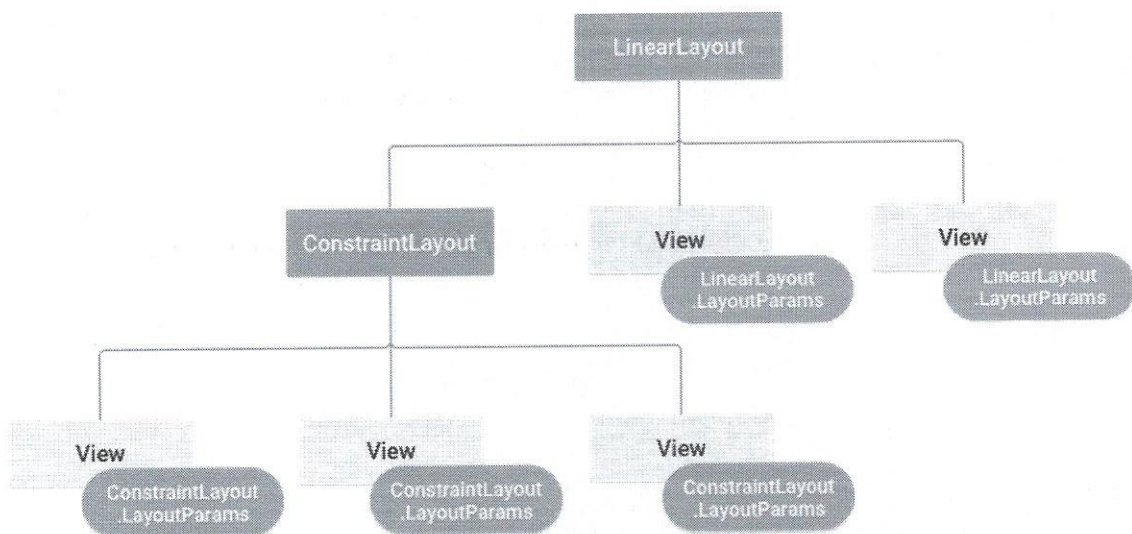


Imagen 10. Visualización de una jerarquía de vistas con parámetros de diseño asociados con cada vista.

Fuente: Recuperado de <https://developer.android.com/guide/topics/ui/declaring-layout> (2018)

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:



Es preciso tener en cuenta que cada subclase *LayoutParams* tiene su propia sintaxis para configurar valores. Cada elemento secundario debe definir *LayoutParams* adecuados para su elemento primario, aunque también puede definir diferentes *LayoutParams* para sus propios elementos secundarios.

Todos los grupos de vistas incluyen un ancho y una altura, y cada vista se debe definirlos. Muchos *LayoutParams* también incluyen márgenes y bordes opcionales.

Se puede especificar el ancho y la altura con medidas exactas, aunque probablemente no se quiera hacerlo con demasiada frecuencia. Generalmente se utilizan una de estas constantes para establecer el ancho o la altura:

- *wrap_content*: indica a la vista que modifique su tamaño conforme a los requisitos de este contenido.
- *match_parent*: indica a la vista que se agrande tanto como lo permita su grupo de vistas principal, generalmente ocupando todo el ancho o el alto.

En general, no se recomienda especificar el ancho y la altura de un diseño con unidades absolutas como píxeles. En cambio, el uso de medidas relativas como unidades de píxeles independientes de la densidad (*dp*), *wrap_content*, o *match_parent*, es un mejor enfoque, ya que ayuda a garantizar que la aplicación se muestre correctamente en dispositivos con pantallas de diferentes tamaños (sea responsiva).

Posición del diseño

La geometría de una vista es la de un rectángulo. Una vista tiene una ubicación expresada como un par de coordenadas *izquierda* y *superior*, y dos dimensiones, expresadas como un ancho y una altura. La unidad para la ubicación y las dimensiones es el pixel.

Es posible recuperar la ubicación de una vista al invocar los métodos *getLeft()* y *getTop()*. El primero devuelve la coordenada izquierda, o X, del rectángulo que representa la vista. El segundo devuelve la coordenada superior, o Y, del rectángulo que representa la vista. Ambos métodos devuelven la ubicación de la vista respecto de su elemento primario. Por ejemplo, cuando *getLeft()* devuelve "20", significa que la vista se encuentra a 20 píxeles a la derecha del borde izquierdo de su elemento primario directo.

Además, se ofrecen varios métodos convenientes para evitar cálculos innecesarios, y se denominan *getRight()* y *getBottom()*. Estos métodos devuelven las

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------

coordenadas de los bordes derecho y superior del rectángulo que representa la vista. Por ejemplo, llamar a `getRight()` es similar al siguiente cálculo: `getLeft() + getWidth()`.

Tamaño, relleno y márgenes

El tamaño de una vista se expresa con un ancho y una altura. En realidad, una vista tiene dos pares de valores de ancho y altura.

El primer par se conoce como *ancho medido* y *altura medida*. Estas dimensiones definen cuán grande quiere ser una vista dentro de su elemento primario. Las dimensiones medidas se pueden obtener llamando a `getMeasuredWidth()` y a `getMeasuredHeight()`.

El segundo par se conoce simplemente como *ancho* y *altura*, o algunas veces *ancho de dibujo* y *altura de dibujo*. Estas dimensiones definen el tamaño real de la vista en la pantalla, al momento de dibujarlas y después del diseño. Estos valores pueden ser diferentes del ancho y la altura medidos, pero no necesariamente. El ancho y la altura se pueden obtener llamando a `getWidth()` y `getHeight()`.

Para medir estas dimensiones, una vista considera su relleno. Éste se expresa en píxeles para las partes izquierda, superior, derecha e inferior de la vista. El relleno se puede usar para desplazar el contenido de la vista una determinada cantidad de píxeles. Por ejemplo, un relleno izquierdo de 2 empuja el contenido de la vista 2 píxeles hacia la derecha del borde izquierdo. Además, se puede ajustar usando el método `setPadding(int, int, int, int)` y se puede consultar llamando a `getPaddingLeft()`, `getPaddingTop()`, `getPaddingRight()`, y `getPaddingBottom()`.

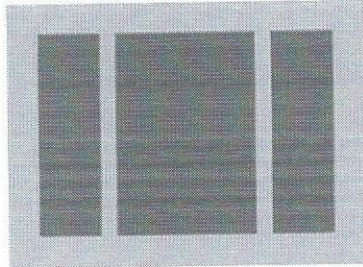
Si bien una vista puede definir un relleno, no proporciona ningún tipo de soporte para márgenes. No obstante, los grupos de vistas sí lo proporcionan en `ViewGroup.MarginLayoutParams`.

Diseños comunes

Cada subclase de la clase `ViewGroup` proporciona una manera única de mostrar las vistas que anidan en ella. Aquí se muestran algunos de los tipos de diseño más comunes integrados en la plataforma Android.

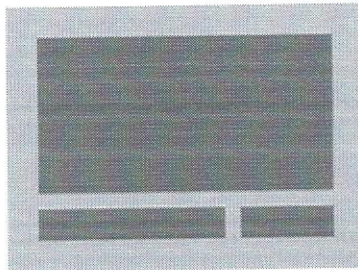
Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------

Diseño lineal



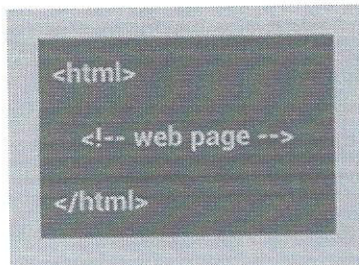
Un diseño que organiza sus elementos secundarios en una sola fila horizontal o vertical. Si la longitud de la ventana supera la longitud de la pantalla, crea una barra de desplazamiento.

Diseño relativo



Permite especificar la ubicación de los objetos secundarios en función de ellos mismos (el objeto secundario A a la izquierda del objeto secundario B) o en función del elemento primario (alineado con la parte superior del elemento primario).

Vista web



Muestra páginas web.

Creación de diseños con un adaptador

Cuando el contenido del diseño sea dinámico o no sea predeterminado, se puede emplear un diseño con la subclase *AdapterView* para completar el diseño con vistas durante el tiempo de ejecución. Una subclase de la clase *AdapterView* usa un *Adapter* para enlazar datos con su diseño. El *Adapter* se comporta como intermediario entre la fuente de datos y el diseño *AdapterView*; el *Adapter* recupera los datos (de una fuente como una matriz o una consulta a la base de datos, o mediante un servicio web RESTful

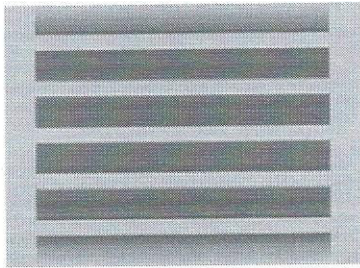
Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:



como es el caso del actual proyecto) y convierte cada entrada en una vista que el programador puede agregar al diseño *AdapterView*.

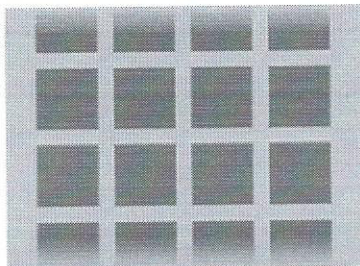
Los diseños comunes respaldados por un adaptador incluyen:

Vista de lista



Muestra una sola lista de columnas desplazable.

Vista de cuadrícula



Muestra una cuadrícula desplazable de columnas y filas.

Relleno de una vista del adaptador con datos

Se puede completar una *AdapterView*, como *ListView* o *GridView*, enlazando la instancia *AdapterView* con un *Adapter*, que recupera datos de una fuente externa y crea una *View* que representa cada entrada de datos.

Android proporciona varias subclases de *Adapter* que resultan útiles para recuperar diferentes tipos de datos y generar vistas para una *AdapterView*. Los dos adaptadores más comunes son los siguientes:

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



ArrayAdapter:

Se usa este adaptador cuando la fuente de datos es una matriz. Según la configuración predeterminada, *ArrayAdapter* crea una vista para cada elemento de la matriz llamando a *toString()* en cada elemento y disponiendo los contenidos en una *TextView*.

Por ejemplo, si hay una matriz de *strings* que se desea visualizar en una *ListView*, se inicializa un nuevo *ArrayAdapter* usando un constructor para especificar el diseño de cada *string* y la matriz de *strings*:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
    android.R.layout.simple_list_item_1, myStringArray);
```

Los argumentos para este constructor son los siguientes:

- El *Context* de la app.
- El diseño que contiene una *TextView* para cada *string* de la matriz.
- La matriz de *strings*.

Luego, simplemente se llama a *setAdapter()* en el *ListView*:

```
ListView listView = (ListView) findViewById(R.id.listView);  
listView.setAdapter(adapter);
```

SimpleCursorAdapter:

Se debe aplicar este adaptador cuando los datos provengan de un *Cursor*, que es una colección de filas y contiene un número determinado de registros, por ejemplo, para exponer los resultados de una consulta *SQLiteDatabase*. Se debe especificar un diseño para cada fila en el *Cursor* y qué columnas del *Cursor* se deben insertar en qué vistas del diseño. Por ejemplo, si se desea crear una lista de nombres y números de teléfono de personas, se puede realizar una consulta que muestre un *Cursor* con una fila para cada persona y columnas para los nombres y los números. Luego, es preciso crear una matriz de tipo *strings* que especifique las columnas del *Cursor* que se desee en el diseño para cada resultado y una matriz con valores enteros que especifique las vistas correspondientes en las que se deba colocar cada columna.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



Luego, el *SimpleCursorAdapter* crea una vista para cada fila en el *Cursor* usando el diseño proporcionado al insertar cada elemento *fromColumns* en la vista *toViews* correspondiente.

Si durante el ciclo de vida de la aplicación se cambian los datos subyacentes que lee el adaptador, es necesario llamar a *notifyDataSetChanged()*. Esto es significativo, ya que le notificará a la vista anexada que se modificaron los datos y que debe actualizarse.

Manejo de eventos de clic

La adaptación que se realice de cualquiera de las formas explicadas anteriormente puede responder a eventos de clic en cada elemento de una *AdapterView* al implementar la interfaz *AdapterView.OnItemClickListener*. Al hacer esto, se puede especificar qué acción realizar al seleccionar un elemento de la lista o grilla, por ejemplo, según el caso.

AsyncTask (Android Studio)

El manejo de hilos para los procesos puede generar comportamientos no deseados, por eso es preciso detallar los beneficios de una gran opción para las aplicaciones Android. La idea es evaluar entre varias opciones la mejor alternativa para implementar concurrencia entre las tareas que se ejecutan.

Para programar la multitarea en Android, se puede usar la clase *Thread* de Java con los métodos *synchronized* y siendo el desarrollador quien controle todo, pero gracias a la clase especial llamada *AsyncTask*, automatiza el control y lo optimiza, lo que resulta conveniente. Esta clase actúa de editor y se utiliza al hacer un *extends* (extender, heredar) de ella. Al hacerlo, se generarán ciertos métodos que estarán vacíos para sobrescribir o implementar en ellos.

Al crearse el objeto de la clase *AsyncTask* se llama, para empezar, a su primer método *onPreExecute()* que se ejecuta sobre el hilo principal. Al terminar este *AsyncTask* crea un hilo secundario y ejecuta su trabajo dentro de *doInBackground()*. Durante la ejecución en segundo plano se puede realizar llamadas al hilo principal (por ejemplo, para incrementar la barra de carga a medida que se ejecuta el hilo en segundo plano) desde *doInBackground()*, y con ayuda de *publishProgress()*, a un método sobrescrito llamado *onProgressUpdate()*. No es obligatorio llamar a *publishProgress()*, pudiendo no invocarlo nunca con lo que no se entrará en *onProgressUpdate()*, o

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------

llamándolo las veces que se requiera. Al terminar de ejecutarse `doInBackground()` se invoca de inmediato a `onPostExecute()`, y se da por acabado a este hilo en segundo plano.

Esta valoración mantendrá la sensación multitarea, evitará bloqueos y optimizará el hilo principal de las aplicaciones. La clase `AsyncTask` posee métodos que permitirán coordinar la ejecución de las tareas que se desean ubicar en segundo plano. Estos métodos tienen los siguientes propósitos:

- `onPreExecute()`: En este método se incluyen todas aquellas instrucciones que se ejecutarán antes de iniciar la tarea en segundo plano. Normalmente es la inicialización de variables, objetos y la preparación de componentes de la interfaz.
- `doInBackground(Parámetros...)`: Recibe los parámetros de entrada para ejecutar las instrucciones específicas que irán en segundo plano, luego de que ha terminado `onPreExecute()`. Dentro de él se puede invocar un método auxiliar llamado `publishProgress()`, el cual transmitirá unidades de progreso al hilo principal. Estas unidades miden cuánto tiempo falta para terminar la tarea, de acuerdo a la velocidad y prioridad que se está ejecutando.
- `onProgressUpdate(Progreso...)`: Este método se ejecuta en el hilo de UI luego de que `publishProgress()` ha sido llamado. Su ejecución se prolongará lo necesario hasta que la tarea en segundo plano haya sido terminada. Recibe las unidades de progreso, así que se puede utilizar algún View para mostrarlas al usuario para que éste sea consciente de la cantidad de tiempo que debe esperar.
- `onPostExecute(Resultados...)`: En este método se pueden publicar todos los resultados retornados por `doInBackground()` hacia el hilo principal.
- `onCancelled()`: Ejecuta las instrucciones deseadas para que se realicen al cancelar la tarea asíncrona.

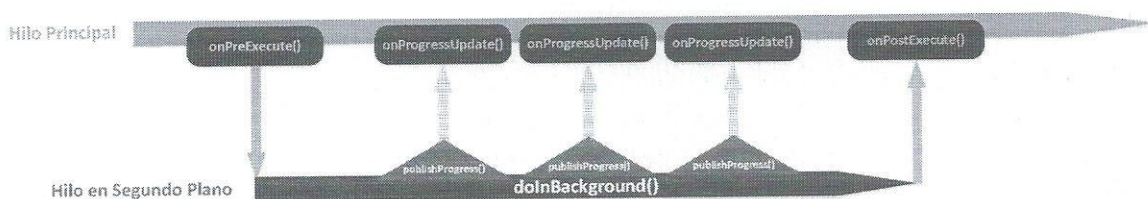


Imagen 11. Estructura de la clase `AsyncTask`.

Fuente: Recuperado <https://jarroba.com/async-task-en-android/> (2013)

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:



Servicios web RESTful

Esta tecnología es una alternativa más simple a los ya considerados servicios obsoletos SOAP y WSDL por muchas de las grandes empresas de software actuales (Facebook, Google, Yahoo, entre otros), por lo que muchos de los proveedores de Web 2.0 (también llamada Web Social) - comprendido por los sitios web que facilitan la compartición de información, la interoperabilidad, el diseño centrado en el usuario y la colaboración en la WWW, permite a los usuarios interactuar y colaborar entre ellos a modo de comunidad virtual - migran a este modelo de tecnología orientado a los recursos, que es más sencillo de utilizar.

JSON (JavaScript Object Notation)

Es el formato que se utilizará en el proyecto para intercambiar datos entre la aplicación y la API. JSON es un formato de texto ligero para representación de datos estructurados o intercambio de datos como se mencionó anteriormente, desarrollado por Douglas Crockford (EE.UU). Sigue la sintaxis de objeto estándar de JavaScript, pero se lo considera un formato de lenguaje independiente, por lo que muchos sistemas y ambientes de programación disponen de la capacidad de leer, analizar, "parsear" (convertir una cadena de un formato a un objeto nativo), "stringify" (convertir un objeto nativo a cadena) y generar JSON.

La estructura de JSON es una cadena donde se puede incluir distintos tipos de datos básicos como números, *strings*, *arrays*, booleanos, y más, de tal forma que se establezca una jerarquía de datos. Cabe destacar que un objeto JSON puede ser almacenado en su propio archivo, el cual es un archivo de tipo texto con extensión ".json" y una MIME (Multipurpose Internet Mail Extensions) type de "application/json", que son una serie de convenciones dirigidas al intercambio de archivos a través de internet de forma transparente para el usuario.

Interfaz de Programación de Aplicaciones (API)

Es una especificación formal sobre cómo un módulo de un software determinado se comunica e interactúa con otro. Otra forma de definirlo es como un conjunto de funciones y procedimientos que cumplen varias funciones como capa de abstracción en que los programas o sitios web intercambian datos. Este intercambio generalmente se efectúa mediante formatos como JSON o XML.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------



Las API's les sirven a los desarrolladores para consumir datos de otras aplicaciones/sitios web, como también para ofrecer datos con un formato estándar a otros desarrolladores o al mismo desarrollador para que los utilice en sus propios sitios web/aplicaciones, y para aplicaciones que se ejecuten en dispositivos móviles. Es importante destacar que los cambios en el código fuente no deberían afectar al API.

Hay muchos sitios web que proveen sus API's para que los desarrolladores les den determinados usos, por ejemplo:

- Google: para consumir google maps y sus recursos
- Twitter: para acceder a los datos de los usuarios y su estado
- The Movie Database (TMDb): para acceder a información de películas y series
- Spotify: para acceder a información de artistas musicales y sus álbumes/canciones

Representational State Transfer (REST)

Es un tipo de arquitectura de desarrollo web que se apoya en el protocolo estándar HTTP (Protocolo de Transferencia de HiperTexto), que permite realizar peticiones sin estado (lo que proporciona mayor rendimiento) y se compone de una serie de reglas que son requisito exclusivo para el diseño de la arquitectura de una API. Es mediante las peticiones HTTP que se hacen las llamadas a la API con la siguiente estructura:

El llamado "recurso" que es representado por la URL, por ejemplo:
<http://www.servidorprueba.com/api/pacientes>

La "operación" está representada por el método (HTTP Verbs), por ejemplo:
GET <http://www.servidorprueba.com/api/pacientes>

Funcionalidad REST

Para desarrollar API's REST los aspectos claves que hay que conocer y dominar son:

- Métodos HTTP
- Códigos de estado
- Aceptación de tipos de contenido
- Reglas de una arquitectura REST

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------



- Autenticación

El código de estado HTTP representa el resultado obtenido como respuesta, por ejemplo:

200 OK HTTP/1.1
404 NOT FOUND HTTP/1.1

Métodos

Si se realiza un CRUD (Create, Read, Update, Delete, como funciones básicas en bases de datos), se deben utilizar para operar los HTTP verbs de forma adecuada para cuidar la semántica y manipular los recursos.

GET: Para consultar y leer recursos

POST: Para crear recursos

PUT: Para editar recursos

DELETE: Para eliminar recursos.

PATCH: Para editar partes concretas de un recurso.

Por ejemplo, para un recurso de pacientes:

“GET /api/pacientes” permite acceder al listado de pacientes (obtener recursos)

“POST /api/pacientes” permite crear un paciente nuevo (crea un nuevo recurso)

“GET /api/pacientes/id” permite acceder al detalle de un paciente (obtener recurso)

“PUT /api/pacientes/id” permite editar el paciente, sustituyendo la totalidad de la información anterior por la nueva (actualizar recurso).

“DELETE /api/pacientes/id” permite eliminar el paciente (eliminar recurso)

“PATCH /api/pacientes/id” permite modificar cierta información del paciente (actualizar datos concretos del recurso).

Debido al desconocimiento o el soporte de ciertos navegadores, los desarrolladores web han usado durante los últimos años únicamente los métodos GET y POST para realizar las acciones anteriormente mencionadas. Si se trabaja con REST, esto sería un error de base conceptual y puede dar problemas incluso a la hora de nombrar los recursos, obligando al desarrollador a poner verbos en las URL.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------



Creación de recursos

Método POST:

La URL estará abierta, el recurso todavía no existe y por tanto no tiene id.

Respuesta a la creación de recursos

La convención en REST es devolverla en la respuesta como valor de la cabecera HTTP Location. Resultados posibles de códigos de estado:

- 403 (Acceso prohibido)
- 400 (petición incorrecta, falta un campo o su valor no es válido)
- 500 (Error del lado del servidor al intentar crear el recurso, como puede ser por indisponibilidad o que se haya detenido el servicio de la base de datos)
- 201 (Recurso creado correctamente)

Actualización de recursos

- Método PUT:

Según la ortodoxia REST, actualizar significaría cambiar todos los datos o campos

- Método PATCH:

Es un método moderno estándar HTTP (año 2010) pensado para cambiar solo ciertos datos. Muchos entornos de programación REST todavía no lo soportan

Resultados posibles de códigos de estado:

- 201 (Recurso creado, cuando se le pasa como parámetro el id deseado al servidor)
- 200 (Recurso modificado correctamente)

Hay que tener en cuenta que no sólo importa el código de estado al recibir una respuesta por parte del servidor, ya que indica solamente que la petición ha sido realizada correctamente, sino que también hay que observar el cuerpo de respuesta

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



que puede avisar de errores y no el recurso que se está solicitando en la URL o cuerpo de envío. Por ejemplo:

Petición: PUT /api/pacientes/id

Respuesta: Status Code 200

Content:

```
{
```

```
  success: false,
```

```
  code: 734,
```

```
  error: "datos insuficientes"
```

```
}
```

Eliminar recursos

Método DELETE:

Elimina el registro especificado de la base de datos. Algunos resultados posibles de códigos de estado:

- 200 OK
- 404 Not found
- 500 Server error

Tras ejecutar DELETE con éxito, las siguientes peticiones GET a la URL del recurso deberían devolver la respuesta del servidor con el código de estado 404.

Tipos y formatos de contenido

HTTP permite especificar en qué formato se necesita recibir el recurso, pudiendo indicar varios en orden de preferencia, para ello se utiliza el header (cabecera) "Accept". De esta manera la API devolverá el recurso en el primer formato disponible y, de no poder mostrar el recurso en ninguno de los formatos indicados por el cliente mediante el header Accept, devolverá el código de estado "HTTP 406". En la respuesta, devolverá el header "Content-Type", para que el cliente sepa qué formato se devuelve, por ejemplo:

Petición: GET /api/pacientes/id

Accept: application/pdf, application/json

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



Respuesta: Status Code 200
Content-Type: application/pdf

Reglas de una arquitectura REST

- Interfaz uniforme
- Cacheable
- Separación de cliente y servidor
- Sistema de Capas
- Código bajo demanda (opcional)
- Nombre de los recursos
- Formato de salida

Interfaz Uniforme

La interfaz de basa en recursos (por ejemplo, el recurso Pacientes).

El servidor mandará los datos (vía *html*, *json*, *xml*, etc.), pero lo que tenga como implementación internamente (la base de datos y el *backend*, por ejemplo) para el cliente es transparente. La representación del recurso que le llega a éste, será suficiente para poder modificar o borrar el recurso suponiendo que el usuario que lo solicite tenga permisos, y es necesario enviarle un parámetro "id" al menos.

Cacheable

En la web los clientes pueden cachear las respuestas del servidor. Dichas respuestas se deben marcar de forma implícita o explícita como *cacheables* o no. Es así que en futuras peticiones, el cliente sabrá si puede reutilizar o no los datos que ya ha obtenido. Si se ahorran solicitudes, se mejorará la escalabilidad de la aplicación y el rendimiento del lado del cliente (se evita principalmente la latencia).

Separación de cliente y servidor

- El cliente y servidor están separados, su unión es mediante la interfaz uniforme.
- Los desarrollos en *frontend* y *backend* se hacen por separado, teniendo en cuenta la API.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------



- Mientras la interfaz no cambie, se pueden cambiar el cliente o el servidor sin problemas.

Sistema de capas

El cliente puede estar conectado mediante la interfaz al servidor o a un intermediario, para él es irrelevante y desconocido, al cliente sólo le debe preocupar que la API REST funcione correctamente. Este uso de capas o servidores intermedios puede servir para aumentar la escalabilidad (sistemas de balanceo de carga, cachés) o para implementar políticas de seguridad.

Código bajo demanda

Los servidores pueden ser capaces de aumentar o definir cierta funcionalidad al cliente transfiriéndole cierta lógica que pueda ejecutar, como los opcionales:

- Componentes compilados como *applets* de Java
- JavaScript en cliente.

Nombre de los recursos

Plural mejor que singular, para lograr uniformidad:

- Obtención de un listado de clientes: GET /v1/pacientes
- Obtención de un cliente en particular: GET /v1/pacientes/id

Se recomienda:

- Utilizar URL's lo más cortas posibles
- Evitar guiones y guiones bajos
- Utilizar nombres y no verbos
- Estructura jerárquica: /v1/pacientes/id/turnos/id

Se pueden producir múltiples errores en la llamada al API:

- Falta de permisos
- Errores de validación
- Error interno de servidor.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------



- Debe devolver un código de estado HTTP con los *requests*.
- Añade un mensaje de error si es necesario.

Formato de salida

En función de la petición la API podría devolver un tipo de formato diferente. Es necesario revisar el "Accept header":

```
GET /v1/geocode HTTP/1.1
Host: api.geocod.io
Accept: application/json
```

Se aconseja utilizar JSON porque es sencillo y simple, en el caso de XML se debe evitar si no es un requerimiento, ya que es más complicado de manipular, por ejemplo, por sus *schemas* y *namespaces*.

Autenticación y validación en API REST

Métodos de autenticación:

- Basada en cookies, la más utilizada
 - El servidor guarda la cookie para autenticar al usuario en cada request.
 - Habrá que tener un almacén de sesiones: en BBDD o Redis por ejemplo.
- Basada en tokens, se confía en un token firmado que se envía al servidor en cada petición.
 - Es un valor que autentica al usuario en el servidor.
 - Se consigue luego de hacer login mediante usuario y contraseña.
 - Distintas formas de generarlo, por ejemplo, habiendo almacenado el Hash en una BBDD asociado al usuario y luego haciendo una comprobación de coincidencia.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------

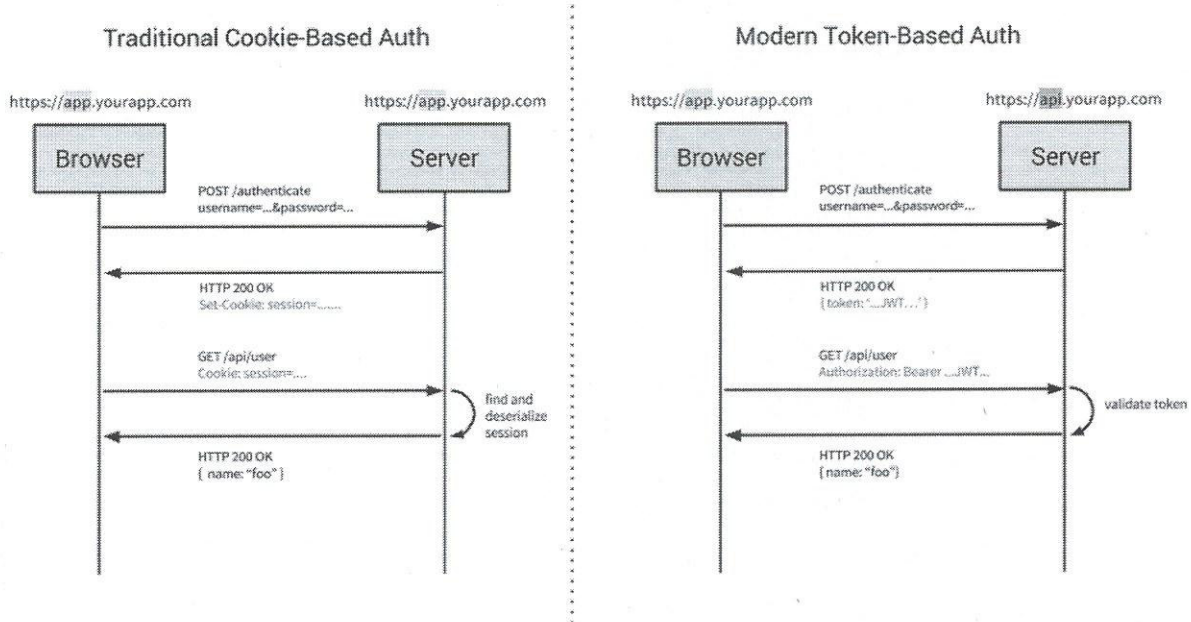


Imagen 12. Ejemplos de metodologías de autenticación en API REST.

Fuente: Recuperado <https://juanda.gitbooks.io/webapps/content/api/arquitectura-api-rest.html> (2018)

Postman

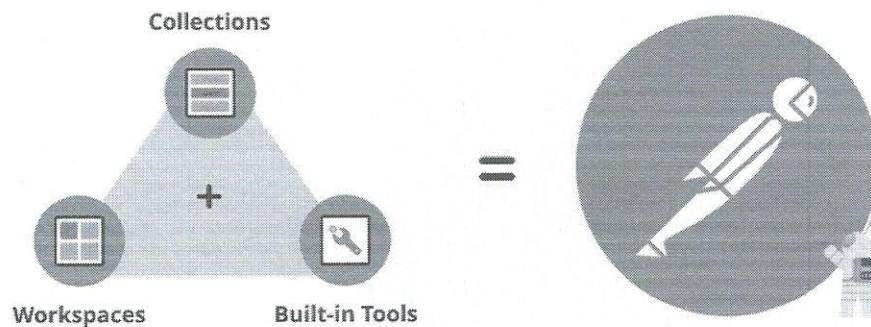


Imagen 13. Postman's API Development Environment (ADE): Collections, Workspaces & Tools.

Fuente: Recuperado <https://www.getpostman.com/> (2018)

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:



Postman es una herramienta que permite a los desarrolladores realizar peticiones HTTP a cualquier API, muy útil para hacer pruebas mientras se programa código fuente, de esta manera ofrece la chance de comprobar el adecuado funcionamiento del servicio que se necesita consumir mediante la API para luego integrar las correctas peticiones al código y de esta manera la aplicación que se esté desarrollando funcionará de manera esperada. Permite al desarrollador mantener una colección actualizada de todas las llamadas de un servicio o una colección que permita probar rápidamente la integración con API's de terceros. Además, posibilita exportar toda la colección en un fichero ".json" y permite al equipo de desarrollo versionar la colección en el sistema de control de versiones, y que sea un documento colaborativo que ayude a todos los componentes del equipo de desarrollo a tener su colección actualizada de una manera inmediata.

Surgió originariamente como una extensión para el navegador Google Chrome. Actualmente dispone de aplicaciones nativas para MAC y Windows y se encuentran trabajando en una aplicación nativa para Linux (disponible en versión beta).

Está compuesto por diferentes herramientas y utilidades gratuitas (en la versión free) que permiten realizar tareas diferentes dentro del mundo API REST: creación de peticiones a API's internas o de terceros, elaboración de pruebas para validar el comportamiento de API's, posibilidad de crear entornos de trabajo diferentes (con variables globales y locales), y todo ello con la posibilidad de ser compartido con otros compañeros del equipo de manera gratuita (exportación de toda esta información mediante URL en formato JSON). Además, dispone de un modo *cloud* colaborativo (de pago) para que los integrantes de los equipos de trabajo puedan desarrollar colecciones para API's sincronizadas en la nube, de modo que la integración sea más inmediata y sincronizada.

Colecciones de API's

El interés fundamental de Postman es que se lo utilice como una herramienta para hacer peticiones a APIs y generar colecciones de peticiones que permitan probarlas de una manera rápida y sencilla.

Las colecciones son carpetas donde se almacenan las peticiones y que permiten ser estructuradas por recursos, módulos o como el equipo de trabajo lo desee. En la imagen 14 puede verse un ejemplo de estructura de una colección de prueba del proyecto desarrollado:

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------

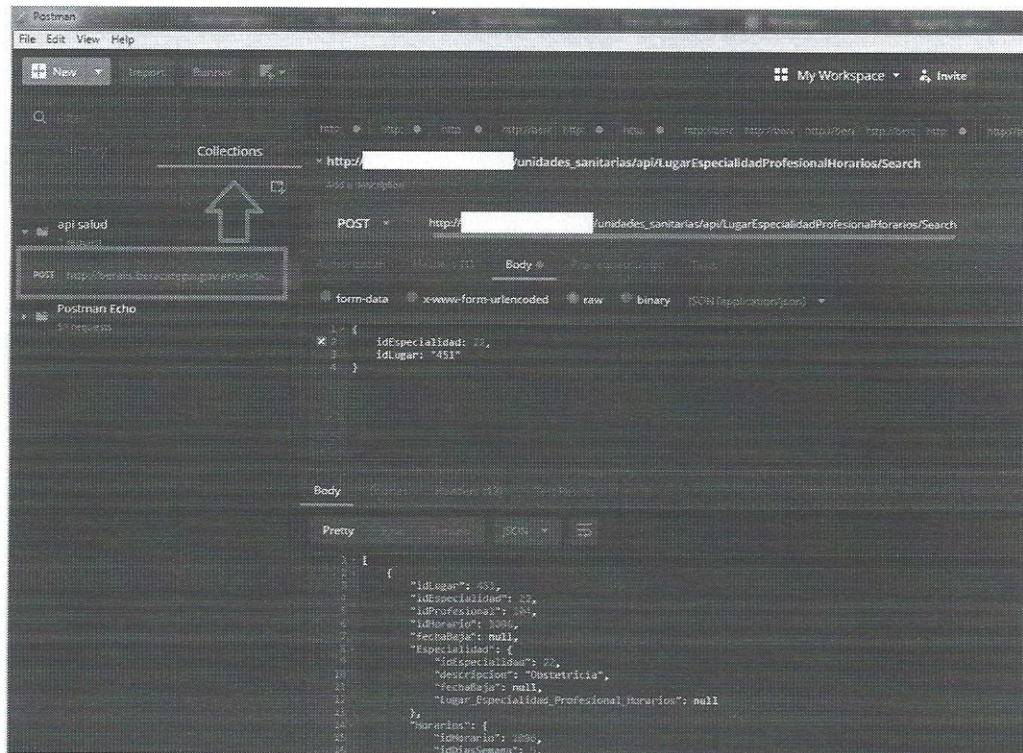


Imagen 14. Ejemplo de estructura de colección.
Fuente: Elaboración propia, basada en la práctica.

Como se observa, se puede definir el tipo de petición (GET, POST, etc.), tokens de autenticación, cabeceras asociadas a la petición, etc., todo de una manera muy sencilla. Todas las llamadas almacenadas en nuestra colección pueden ser exportadas a múltiples lenguajes haciendo clic en el apartado “Generate Code”.

Entornos de trabajo

Uno de los puntos más potentes de Postman es la posibilidad de definir todas las variables que se necesiten y clasificarlas por entornos de trabajo. Esto es muy útil cuando se tienen diferentes proyectos o cuando se necesita configurar múltiples entornos para un mismo proyecto (por ejemplo, diferentes cabeceras, URL's de cada uno de los entornos de trabajo – local, preproducción o incluso producción, etc.).

Desarrollo de pruebas de API's y automatización en Jenkins

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------

Otra de las funcionalidades interesantes, sobre todo para los QA's de los equipos, es la posibilidad de automatizar de una manera sencilla pruebas de integración para los proyectos. En cada una de las llamadas, Postman proporciona una pestaña "tests" donde se puede configurar las pruebas que se deseen.

Postman se apoya en código Javascript para programar sus pruebas. No es necesario saber programar en este lenguaje porque Postman ofrece un conjunto de *snippets* de pruebas que permiten que desarrollar las mismas sea sencillo.

Uno de los aspectos más demandados, y que Postman ha desarrollado, es la posibilidad de automatizar las pruebas para que el software IC ejecute las pruebas dentro del flujo de construcción de nuevas versiones. En el blog de Postman se explica paso a paso cómo integrarlos en Jenkins.

Documentar API

En cada colección que se genere en una API, Postman da la posibilidad de generar una documentación web de acceso libre donde resalta todas las llamadas con los *headers* definidos y, a su lado, un ejemplo de la llamada para ser probada inmediatamente con cURL:

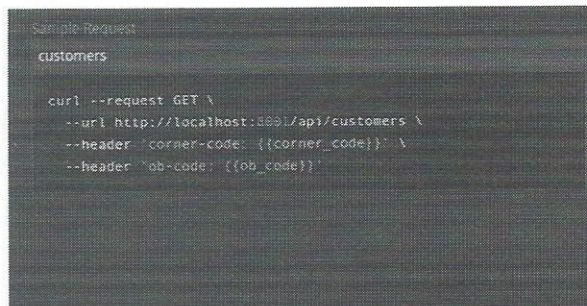
GET customers

`http://localhost:8001/api/customers`

HEADERS

OB-CODE `{{ob_code}}`

CORNER-CODE `{{corner_code}}`



```
Sample Request
customers

curl --request GET \
  --url http://localhost:8001/api/customers \
  --header 'corner-code: {{corner_code}}' \
  --header 'ob-code: {{ob_code}}'
```

Imagen 15. Ejemplo de generación de documentación web y su llamada con headers.

Fuente: Elaboración propia, basada en la práctica.

Para ver la documentación, en la colección se debe dirigir al símbolo ">" y en el menú que aparece, seleccionar "View Docs":

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------



Diseño, desarrollo e implementación

Explicada la metodología, herramientas y tecnologías utilizadas, se está en condiciones de adentrarse en el diseño, desarrollo e implementación de la aplicación móvil nativa para dispositivos Android SO, orientada a maximizar la eficiencia en la gestión de turnos para ser atendidos en los Centros de Atención Primarios de Salud del partido de Berazategui.

El sistema finalizado debe cumplir con las siguientes características:

Sesión

- Registro al sistema mediante un formulario de datos del usuario
- Confirmación del registro vía e-mail
- Inicio de sesión (login) con validaciones
- Extras:
 - Menú lateral desplegable con opciones de secciones o Nav Bar
 - Perfil de usuario con información editable
 - Selección de idiomas

Mapas y ubicación

- Visualizar mapas con la ubicación del usuario y la ubicación de los centros de salud (CAPS).
- Obtener información relevante de los centros de salud (nombre, dirección, barrio, guardia).
- Obtener la ubicación actual del usuario en todo momento.
- Obtener centro de salud más cercano a la ubicación actual del usuario.
- Extra:
 - Buscador rápido de CAPS.

Turnos

- Permitir obtener mediante filtros (especialidad, lugar, profesional), los turnos disponibles del mes en el que se encuentre.
- Seleccionar un turno disponible a través de un calendario que le proporcione al usuario la información requerida para reservarlo.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------



- Visualizar los turnos reservados en una sección o módulo de la aplicación.
- Permitir cancelar un turno en la sección de turnos reservados.

Calendario

- Visualizar los turnos solicitados a través de un calendario intuitivo.
- Los turnos reservados deben agendarse en el calendario nativo del dispositivo para recordarle el evento mediante alarmas y así evitar el ausentismo.

Extras

- Mostrar notificaciones Toast

Sesión

Módulos Registro y Login

Como prácticamente toda aplicación que requiera interacciones con usuarios y otras tareas específicas, es necesario establecer un módulo de registro de los mismos, otro para el inicio de sesión o login, y una vez dentro del sistema poder ingresar a otro módulo de perfil de usuario, donde se muestran los datos del usuario y se permite modificar dicha información. En el caso del registro del usuario, una vez ingresado los datos y confirmados se le envía un correo electrónico a la cuenta proporcionada para validar la activación del registro. En la imagen 16 se visualiza lo recientemente mencionado.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------

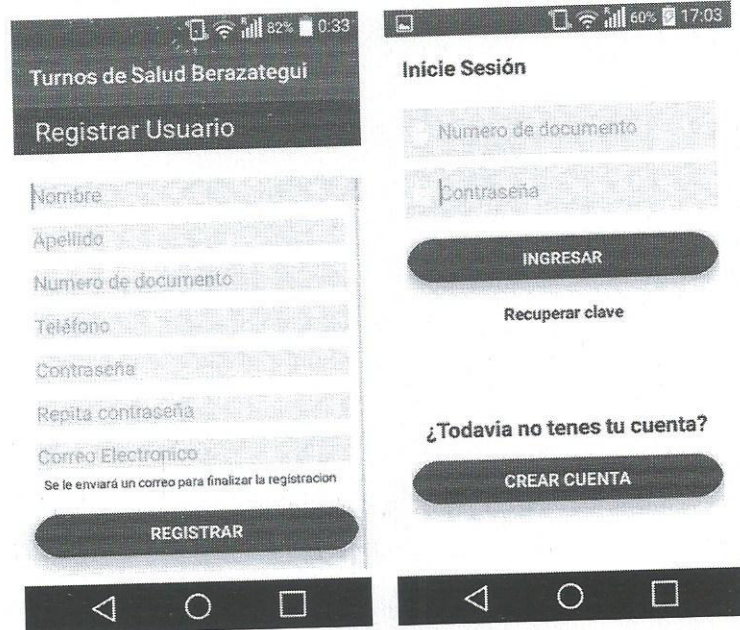


Imagen 16. Vistas de registro y login.

Fuente: Elaboración propia, basada en la práctica.

El envío de correo se logra mediante la clase implementada *EnviarMail* que extiende de *AsyncTask* para sincronizar hilos de procesos y, también, usa el protocolo de red SMTP (Simple Mail Transfer Protocol) utilizado para el intercambio de mensajes de correo electrónico entre dispositivos, que dispone de propiedades de autenticación del para host, puerto y encriptaciones con el protocolo de seguridad SSL (Secure Sockets Layer), que permite a los datos viajar de forma segura por internet a través de claves digitales.

La utilización de la clase *EnviarMail* se ha utilizado con distintos parámetros:

1. Contexto en el que se encuentra la aplicación
2. Cuenta de e-mail ingresada por el usuario a la cual se envía una confirmación a la casilla de la misma.
3. Asunto del e-mail
4. Mensaje en el cuerpo del e-mail

En el código se ejecuta de la siguiente manera:

```
EnviarMail enviomail = new EnviarMail(getApplicationContext(), mailkeulstro, "App Turnos Salud Berazategui", id usuario.toString() + " " +
```

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



```
getResources().getString(R.string.confirmacion_registro);
envionMail.execute();
```

Módulos Nav Bar, Perfil de Usuario y Selección de Idioma

Al ingresar al sistema, se tiene un menú lateral "nav bar" que se accede desde el botón superior izquierdo de la "app bar" o deslizando la pantalla de izquierda a derecha. Esto se implementa desde el *MainActivity* y se puede generar de forma predeterminada al crear un nuevo proyecto en Android Studio.

Respecto al perfil de usuario, se trae la información del mismo desde `../api/Usuario/Search` y se le pasa el ID como parámetro al cuerpo de la petición para traer los datos de los otros campos que se visualizan en la imagen 17, los cuales se pueden actualizar mediante `../api/Usuario/Update`.

Y, por último, la selección de idioma se lleva a cabo con el uso de un cuadro de dialogo (*AlertDialog.Builder*), que se define como una pequeña ventana que le indica al usuario que debe tomar una decisión para continuar, como en este caso seleccionar una opción, o en otros ingresar información adicional. Por otro lado, se necesita del objeto *Locale* que es el que se encarga de representar una región geográfica, política o cultural específica, en el presente caso se lo utiliza para la primera representación y referenciar lenguajes de diferentes regiones (en/es/it).

```
private void mostrarDialogIdioma() {
    AlertDialog.Builder b = new AlertDialog.Builder(this);
    b.setTitle(getResources().getString(R.string.seleccion_idioma));
    //cotiene los idiomas del array de string.xml
    String[] types = getResources().getStringArray(R.array.languages);
    b.setItems(types, new DialogInterface.OnClickListener() {

        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
            switch (which) {
                case 0:
                    locale = new Locale("en");
                    config.locale = locale;
                    break;
                case 1:
                    locale = new Locale("es");
                    config.locale = locale;
                    break;
                case 2:
                    locale = new Locale("it");
                    config.locale = locale;
                    break;
            }
            getResources().updateConfiguration(config, null);
            Intent idiomasAlert = new Intent(MainActivity.this, MainActivity.class);
            startActivity(idiomasAlert);
        }
    });
}
```

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:



```
Toast.makeText(getApplicationContext(), getString(R.string.toast_idioma),  
Toast.LENGTH_LONG).show();  
finish();  
}  
o.show();
```

También se consideran las referencias de cadenas de texto que se alojan en “../res/values/strings/strings.xml”, en este caso se mostrará solo el menú del *AlertDialog* que es de tipo *string-array*, y otro que es de tipo *string*, ya que el documento es extenso. Lo siguiente es mostrar las vistas de estos módulos descritos en la imagen 17.

```
<string-array name="languages">  
  <item>@string/idioma_ingles</item>  
  <item>@string/idioma_español</item>  
  <item>@string/idioma_italiano</item>  
</string-array>  
<string name="seleccion_idioma">Cambiar idioma</string>
```

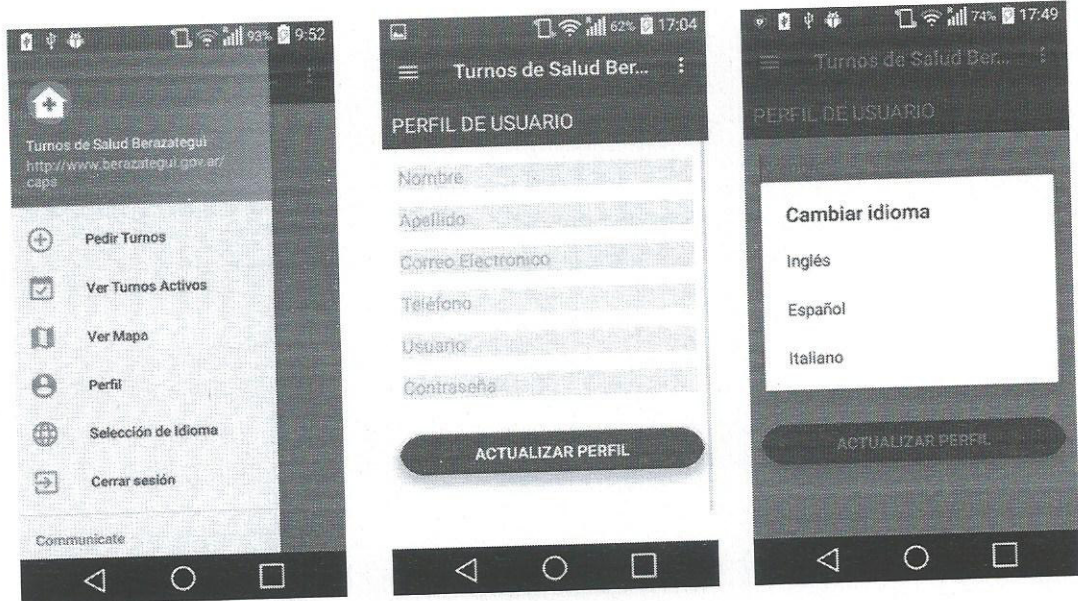


Imagen 17. Vista de nav bar con el logo de la app y sus ítems de rápido acceso, vista de perfil del usuario y vista de selección de idioma. Fuente: Elaboración propia, basada en la práctica.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



Notificaciones Toast

La clase Toast permite mostrar un mensaje rápido superpuesto a la vista, independientemente sea un *activity* o un *fragment*, por un corto periodo de tiempo y sin pausar la ejecución de la aplicación. Este tipo de mensaje es a modo informativo o de aviso para el usuario basado en alguna acción que se haya seleccionado o ejecutado, por lo que, el usuario no puede interactuar con la misma y no tiene acceso para efectuar modificaciones

En la aplicación se utilizó esta clase para mostrar algunos mensajes específicos:

- Aviso de envío de e-mail para confirmar cuenta de correo al registrarse.
- Bienvenida al usuario que inicia sesión.
- Cuando el usuario reserva un turno.
- Cuando el turno no pudo reservarse por algún motivo.
- Cuando el turno ha sido cancelado por el usuario.

Por ejemplo, cuando un usuario ingresa su número de documento y contraseña para entrar al sistema, se evalúan coincidencias en base a un condicional (if/else) que al hacer *match* entre ellos y los datos del servidor, se da la bienvenida o se rechaza en caso contrario, ambos mediante una notificación Toast:

```

if (documento.equals(doc) && contraseña.equals(pass)) {
    String nombres = jsonObject.optString("nombres");
    Toast.makeText(getApplicationContext(), getResources().getString(R.string.str_welco
    me) + " " + nombres.toString() + "!", Toast.LENGTH_LONG).show();
} else {
    Toast.makeText(getApplicationContext(), "Usuario o contraseña" + " " +
    "incorrecta!", Toast.LENGTH_SHORT).show();
}

```

Mapas y ubicación

Con el SDK de Google Maps para Android, se pueden agregar mapas basados en los datos de Google Maps a una aplicación. La API controla automáticamente el acceso a los servidores de Google Maps, la descarga de datos, la visualización del mapa y la respuesta a los gestos del mapa. También se puede usar llamadas a la API para agregar marcadores, polígonos y superposiciones a un mapa básico, y para cambiar la vista del usuario de un área de mapa en particular. Estos objetos proporcionan información adicional para las ubicaciones del mapa y permiten la interacción del usuario con el mapa. La API le permite agregar estos gráficos a un mapa:

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



- Iconos anclados a posiciones específicas en el mapa (marcadores).
- Conjuntos de segmentos de línea (polilíneas).
- Segmentos cerrados (polígonos).
- Gráficos de mapa de bits anclados a posiciones específicas en el mapa (superposiciones de terreno).
- Conjuntos de imágenes que se muestran en la parte superior de los mosaicos del mapa base (Superposiciones de mosaico).

Para poder hacer uso de los mapas, es necesario disponer de una cuenta de Google y registrar el proyecto en Google Cloud Platform Console que requiere su utilización, de esta forma se le proporciona al usuario una "API key" para agregar a la aplicación en desarrollo. Además de esto es necesario añadir diferentes permisos al *manifest* del proyecto y las librerías necesarias para usar los servicios a *build.gradle*, como se observa a continuación:

Permisos y API key de Google Maps en AndroidManifest.xml:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

```
<meta-data
  android:name="com.google.android.gms.version"
  android:value="@integer/google_play_services_version" />
<meta-data
  android:name="com.google.android.gms.version"
  android:value="@integer/google_play_services_version" />
```

Dependencias del módulo app de build.gradle:

```
implementation 'com.google.android.gms:play-services-maps:11.0.0'
implementation 'com.google.android.gms:play-services-location:11.0.0'
implementation 'com.google.android.gms:play-services-location:11.0.0'
```

La utilización de mapas de Google es fundamental para este proyecto, ya que es necesario establecer principalmente marcadores con las ubicaciones del usuario y las de los centros de salud y sus interacciones. Es propicio que el usuario tenga activada la función de ubicación GPS del dispositivo para una mejor experiencia de usuario.

Módulo Marcadores con Información

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



Se detallará parte del código que utiliza una estrategia para ir añadiendo marcadores de los CAPS y mostrar la información relevante de los lugares cuando el usuario vaya seleccionándolos. Cabe destacar que la información proporcionada en los marcadores suele estar limitada por un título y una descripción (snippet), pero en este caso se implementó un método como alternativa para poder mostrar varias líneas en la descripción mediante el método *getInfoWindow* y *getInfoContents*.

```
public void adapterInfoMarkers(ArrayList<Lugar> arrayListLugares) throws IOException {
    String apiUrl = "http://servidor.iaza.una.edu.ar:8080/usuarios/usuarios/GetAPI";

    StrictMode.ThreadPolicy policy = new
    StrictMode.ThreadPolicy.Builder().permitAll().build();
    StrictMode.setThreadPolicy(policy);
    URL url = null;
    HttpURLConnection conn;
    try {
        url = new URL(apiUrl);
        conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("POST");
        conn.connect();
        BufferedReader in = new BufferedReader(new
        InputStreamReader(conn.getInputStream()));
        String inputLine;
        StringBuffer response = new StringBuffer();
        String json = "";
        while ((inputLine = in.readLine()) != null) {
            response.append(inputLine);
        }
        json = response.toString();

        JSONArray jsonArray = null;
        jsonArray = new JSONArray(json);
        for (int i=0; i<jsonArray.length(); i++){
            JSONObject jsonObject = jsonArray.getJSONObject(i);
            String nro_puerta = jsonObject.optString("nro_puerta");
            String barrio_aux = jsonObject.optString("barrio_aux");
            String guardia = jsonObject.optString("estadoGuardia");
            String direccion = jsonObject.getString("direccion_ed");
            String descripcion = jsonObject.getString("descripcion");
            for (int j=0; j<arrayListLugares.toArray().length; j++){
                Lugar lugar = arrayListLugares.get(j);
                if (descripcion == lugar.getNombre() ||
                descripcion.equals(lugar.getNombre()))
                    if (guardia.equals("1") || guardia == "1"){
                        guardia = "SI";
                    }
                else{
                    guardia = "No";
                }
                MarkerOptions markerOptions = new
                MarkerOptions().position(lugar.getLatLng()).title(descripcion)
                .snippet(direccion + "\n" + nro_puerta + "\n" + "barrio" + "\n"
                + barrio_aux + "\n" + "Guardia: " + guardia) //se le pone a líneas ahora
                .icon(BitmapDescriptorFactory.DefaultMarker(BitmapDescriptorFactory.DEFAULT_GREEN));
                mMap.setInfoWindowAdapter(new GoogleMap.InfoWindowAdapter() {
```

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



```
@Override
public View getInfowindow(Marker arg0) {
    return null;
}

//se usa para salto de línea del snippet
@Override
public View getInfoContents(Marker marker) {
    Context context = getContext();
    LinearLayout info = new LinearLayout(context);
    info.setOrientation(LinearLayout.VERTICAL);
    TextView title = new TextView(context);
    title.setTextColor(Color.BLACK);
    title.setGravity(Gravity.CENTER);
    title.setTypeface(null, Typeface.BOLD);
    title.setText(marker.getTitle());
    TextView snippet = new TextView(context);
    snippet.setTextColor(Color.GREEN);
    snippet.setText(marker.getSnippet());
    info.addView(title);
    info.addView(snippet);
    return info;
}

}

} catch (MalformedURLException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} catch (JSONException e) {
    e.printStackTrace();
}
}
```

En el metodo *adapterInfoMarkers* se requirió hacer una petición de recursos a la API, más precisamente de todos los lugares (que corresponden a los centros de salud) mediante ".../api/Lugar/GetAll", realizando iteraciones con los resultados obtenidos en formato JSON para mostrar en la vista distintos atributos relevantes como: la descripción del lugar (como título), la dirección exacta (calles y número de puerta), el nombre del barrio y si dispone de guardia (todos éstos como snippet o información que refuerza al título). El resultado visual se muestra en la imagen 18:

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



Imagen 18. Vista de mapa: ubicaciones del usuario y de los CAPS con su información detallada al ser seleccionada por el usuario.

Fuente: Elaboración propia, basada en la práctica.

Módulo Centro más cercano

Lo siguiente a detallar es la implementación de la acción realizada por el botón "centro más cercano". Básicamente mide todas las distancias a partir de sus coordenadas, entre los CAPS y la ubicación actual del usuario, tomando la menor distancia y moviendo la cámara y zoom del mapa hacia el marcador que se corresponda con el CAPS más cercano, gracias a un algoritmo desarrollado específicamente para realizar esta tarea.

```

BotonMasCercano.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        LatLng ubicacionUsuario = new LatLng(latActual, lngActual);
        agregarMarcadorMiUbicacion(latActual, lngActual);
        int distancia = 0;
        String cadena = "";
        double dis01 = distanceBetween(ubicacionUsuario, coordCAPS01);
        double dis02 = distanceBetween(ubicacionUsuario, coordCAPS02);
        double dis03 = distanceBetween(ubicacionUsuario, coordCAPS03);
        double dis04 = distanceBetween(ubicacionUsuario, coordCAPS04);
    }
});
    
```

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



```
String cadenaListaDistancias = "";  
listaDis.add(dis01);  
listaDis.add(dis02);  
listaDis.add(dis03);  
listaDis.add(dis04);  
  
double[] listaDistancias = new double[listaDis.size()];  
for (int i = 0; i < listaDistancias.length; i++) {  
    listaDistancias[i] = listaDis.get(i);  
}  
  
double max, min;  
int j;  
min = max = listaDistancias[0];  
  
for (j = 0; j < listaDistancias.length; j++)  
  
    if (min > listaDistancias[j])  
        min = listaDistancias[j];  
    if (max < listaDistancias[j])  
        max = listaDistancias[j];  
  
    //mensaje en pantalla para ver la distancia en metros  
    Toast.makeText(getApplicationContext(), "" + min, Toast.LENGTH_SHORT).show();  
    if (min == listaDistancias[0]){  
        CameraUpdate ubicacionCercana =  
CameraUpdateFactory.newLatLngZoom(coordCAPS1, 10);  
        mMap.animateCamera(ubicacionCercana);  
    }  
    else if (min == listaDistancias[1]){  
        CameraUpdate ubicacionCercana =  
CameraUpdateFactory.newLatLngZoom(coordCAPS2, 10);  
        mMap.animateCamera(ubicacionCercana);  
    }  
    else if (min == listaDistancias[2]){  
        CameraUpdate ubicacionCercana =  
CameraUpdateFactory.newLatLngZoom(coordCAPS3, 16);  
        mMap.animateCamera(ubicacionCercana);  
    }  
    else if (min == listaDistancias[3]){  
        CameraUpdate ubicacionCercana =  
CameraUpdateFactory.newLatLngZoom(coordCAPS4, 16);  
        mMap.animateCamera(ubicacionCercana);  
    }  
}
```

El método *distanceBetween* devuelve la distancia entre los dos puntos (coordenadas) *seteados* como parámetros al presionar el botón "centro más cercano". Luego se añade esas distancias a una lista que es preciso convertirla a tipo *double* para recorrerla con un iterador *for* y obtener los resultados que se requieren. En este caso se obtiene la distancia máxima y la mínima, siendo solo necesaria la mínima.

```
private double distanceBetween(LatLng latLng1, LatLng latLng2) {  
    Location loc1 = new Location(LocationManager.GPS_PROVIDER);  
    Location loc2 = new Location(LocationManager.GPS_PROVIDER);  
    loc1.setLatitude(latLng1.getLatitude());
```

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------

```
loc1.setLongitude(latLng1.longitude);
loc2.setLatitude(latLng2.latitude);
loc2.setLongitude(latLng2.longitude);
double distancia = loc1.distanceTo(loc2);
return distancia;
```

Módulo Búsqueda de CAPS

En este módulo se trae la información del servidor desde “../api/Lugar/GetAll” para luego listar solo la descripción de cada lugar al presionar el botón de búsqueda, siendo ítems seleccionables que al ser presionados por el usuario mueven la cámara al marcador solicitado y con un zoom establecido, en la siguiente imagen (19) sólo se listan algunos lugares a modo de ejemplo:

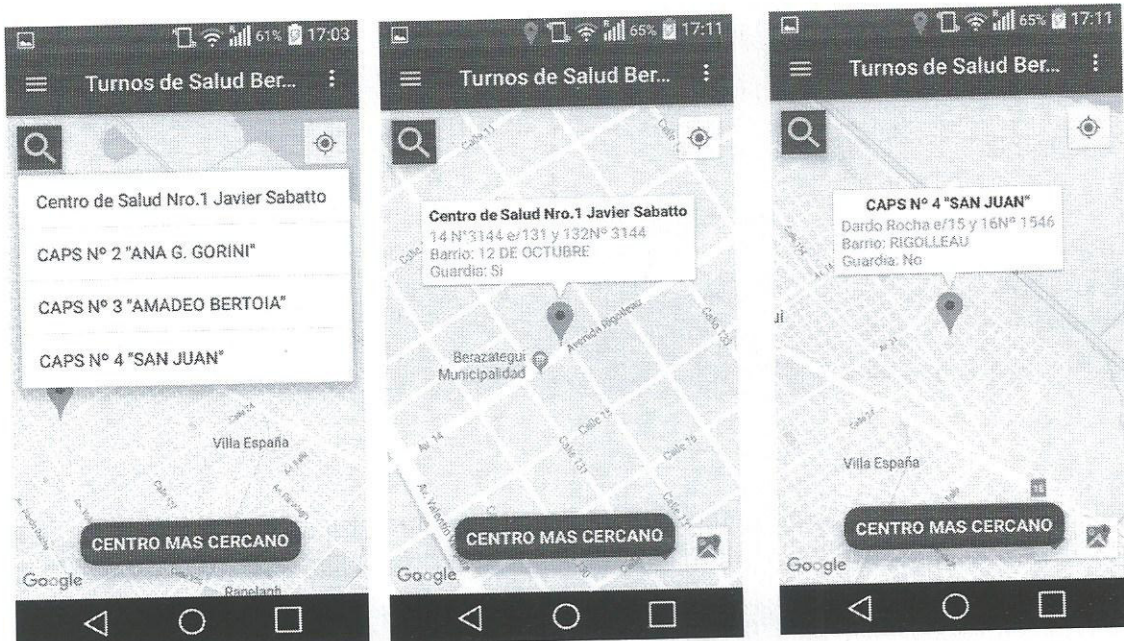


Imagen 19. Vista de mapa: buscador rápido de CAPS y el detalle del seleccionado.

Fuente: Elaboración propia, basada en la práctica.

El método `inicializarLista` utiliza un `adapter` para generar una lista de búsqueda rápida con los lugares cargados en `arrayListLugares`, que se corresponden con los lugares (CAPS), como se visualiza en la imagen anterior.

```
private void inicializarLista() {
    listView.setAdapter(new ArrayAdapter<>(getContext(),
    android.R.layout.simple_list_item_1, arrayListLugares));
    listView.setVisibility(GONE);
}
```

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



```
ImageButtonSearch.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        if(listView.getVisibility() == GONE){  
            listView.setVisibility(View.VISIBLE);  
        } else {  
            listView.setVisibility(View.GONE);  
        }  
    }  
});
```

Turnos

Como primeras pruebas de petición de recursos, se utilizó el software Postman, creando una colección con las peticiones necesarias para traer información con el método POST para luego utilizar dichas peticiones en la implementación de la aplicación. A continuación, se listan las URL probadas con Postman y obteniendo la información requerida:

Traer todos los registros

[http://servidor/unidades sanitarias/api/Paciente/GetAll](http://servidor/unidades_sanitarias/api/Paciente/GetAll)

[http://servidor/unidades sanitarias/api/Usuario/GetAll](http://servidor/unidades_sanitarias/api/Usuario/GetAll)

[http://servidor/unidades sanitarias/api/Especialidad/GetAll](http://servidor/unidades_sanitarias/api/Especialidad/GetAll)

[http://servidor/unidades sanitarias/api/Lugar/GetAll](http://servidor/unidades_sanitarias/api/Lugar/GetAll)

[http://servidor/unidades sanitarias/api/Profesional/GetAll](http://servidor/unidades_sanitarias/api/Profesional/GetAll)

[http://servidor/unidades sanitarias/api/Horarios/GetAll](http://servidor/unidades_sanitarias/api/Horarios/GetAll)

[http://servidor/unidades sanitarias/api/Turnos/GetAll](http://servidor/unidades_sanitarias/api/Turnos/GetAll)

Filtrar registros

Ejemplos:

[http://servidor/unidades sanitarias/api/Profesional/Search](http://servidor/unidades_sanitarias/api/Profesional/Search)

{ nombre: "marcelo" }

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------

http://servidor/unidades_sanitarias/api/Especialidad/Search

{ descripcion: "Clínico" }

http://servidor/unidades_sanitarias/api/Turnos/Search

{ idLugar: "33", idEspecialidad: "5", idProfesional: "9" }

http://servidor/unidades_sanitarias/api/Lugar/Search

{ Descripcion: "33" }

http://.../unidades_sanitarias/api/LugarEspecialidadProfesionalHorarios/Search

{ idLugar: "33", idEspecialidad: "5" }

A continuación, se mostrará un ejemplo práctico con Postman para luego trasladarlo al código realizado en Java con el beneficio de sus herramientas:

```

POST http://.../unidades_sanitarias/api/Lugar/Search
Body
  form-data
  x-www-form-urlencoded
  raw
  binary
  JSON (application/json)
  {
    "Descripcion": "28"
  }

Body
  Pretty
  [
    {
      "idLugar": 489,
      "idReferente": 39,
      "Descripcion": "CAPS Nº 28",
      "idCalle": 249,
      "FEATID_EjeCalle": 108970,
      "FEATID_Lugar": 8223497,
      "noPuerta": "2589",
      "telefono": "",
      "poseeGuardia": 0,
      "barrio_aux": "BARRIO LAS HERMANAS",
      "especialidades_aux": "Clínica Médica, Pediatría, Ginecología, Obstetricia, Enfermería",
      "observaciones_aux": "",
      "direccion_aux": "75 e/125 y 126",
      "localidad_aux": "Cutiérrez",
      "coordenadas": null,
      "fechaBaja": null,
      "Compania": null,
      "Referente": {
        "idReferente": 39,
    
```

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------

Imagen 20. Ejemplo del uso del Search de la API con body.
Fuente: Elaboración propia, basada en la práctica.

Como se observa en la imagen 20, en el cuerpo de la petición (body) se envía un parámetro como filtro de búsqueda en formato JSON, obteniéndose como respuesta otro body con los resultados de la búsqueda, pero de la respuesta del servidor.

También se puede observar en la siguiente imagen (21), la cabecera de la petición y por debajo la de respuesta, ésta nos muestra los métodos permitidos (GET, PUT, POST, DELETE, OPTIONS), el tipo de contenido (application/json), la codificación del mismo (charset=utf-8), la fecha, software y versión del servidor web, etc.

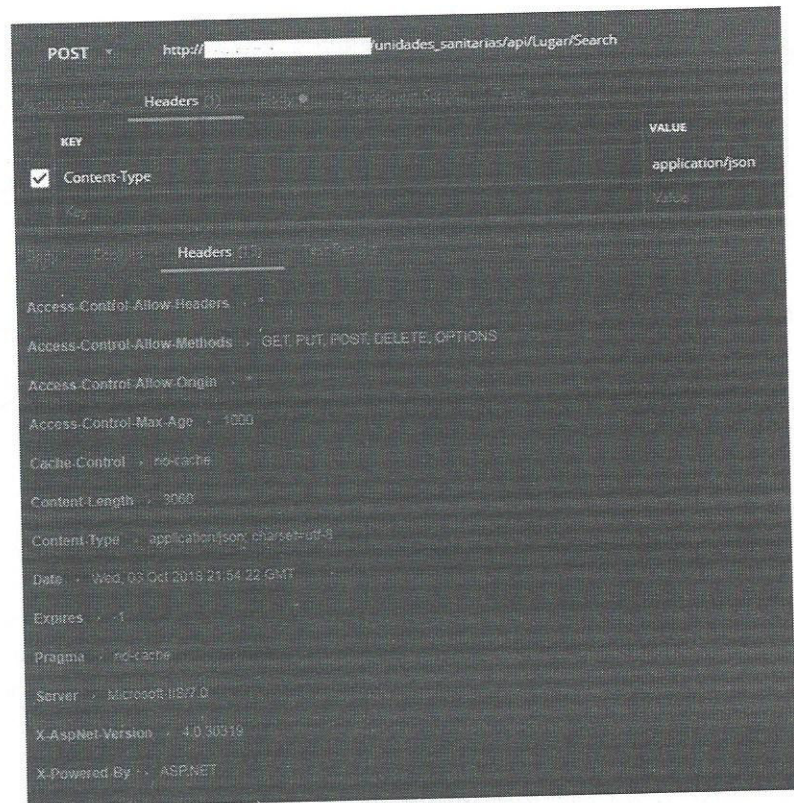


Imagen 21. Cabecera de respuesta de la petición (servidor).
Fuente: Elaboración propia, basada en la práctica.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:



Otra herramienta muy útil que dispone Postman es la posibilidad de generar *snippets* (partes reutilizables de código fuente, como métodos funcionales que se pueden integrar en módulos más grandes aportando funcionalidad) de determinada petición que se realice. Esto es muy útil sobre todo para trasladar el cuerpo de la petición al lenguaje de programación que se esté utilizando en el desarrollo, en este caso se genera para Java y con una función de servicio de cliente HTTP, además brinda información sobre cómo se puede implementar la petición y cómo enviar el cuerpo y cabecera (header) de la misma, el resultado obtenido se visualiza en la imagen 22:

```
GENERATE CODE SNIPPETS  
Java OK HTTP  
Copy to Clipboard  
1 OkHttpClient client = new OkHttpClient();  
2  
3 MediaType mediaType = MediaType.parse("application/json");  
4 RequestBody body = RequestBody.create(mediaType, "{\n\t\"Descripcion\": \"28\n\"}");  
5 Request request = Request.Builder()  
6   .url("http://[redacted]unidades_sanitarias/api/Lugar/Search")  
7   .post(body)  
8   .addHeader("Content-Type", "application/json")  
9   .addHeader("Cache-Control", "no-cache")  
10  .addHeader("Postman-Token", "8ecc00f4-98a2-4010-8540-9674504ddc51")  
11  .build();  
12  
13 Response response = client.newCall(request).execute();
```

Imagen 22. Snippet generado a partir de la petición.
Fuente: Elaboración propia, basada en la práctica.

Gracias a esta información brindada por la generación del snippet, se está en condición de implementarlo en el IDE Android Studio. Se crea un método con dos parámetros: *serverUrl*, en el cual se introduce un valor tipo *string* con la URL a la cual va dirigida la petición, y *jsonObject* que es donde se introduce el *body* en formato JSON.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



```

public String getJsonLugarEspecialidadProfesionalHorarios2(String serverUrl, String jsonObject){

    StringBuilder sb = new StringBuilder();
    profesionalClaveValor.clear();
    String http = serverUrl;

    HttpURLConnection urlConnection1 = null;
    try {
        URL url = new URL(http);
        urlConnection1 = (HttpURLConnection) url.openConnection();
        urlConnection1.setDoOutput(true);
        urlConnection1.setRequestMethod("POST");
        urlConnection1.setUseCaches(false);
        urlConnection1.setConnectTimeout(5000);
        urlConnection1.setReadTimeout(5000);
        urlConnection1.setRequestProperty("Content-Type", "application/json");
        urlConnection1.connect();

        OutputStreamWriter out = new OutputStreamWriter(urlConnection1.getOutputStream());
        out.write(jsonObject);
        out.close();
    }
}

```

Imagen 23. Método que realiza la petición en la implementación de la app.
Fuente: Elaboración propia, basada en la práctica.

Parámetro *jsonObject* del método implementado que representa el *body* de la petición a la API a partir del *snippet* generado por Postman:

```

{"\n\"idLugar\": \" + lugarClaveValor.getId() + "\", \"\n\"idEspecialidad\": \"
+ especialidadClaveValor.getId() + \"\n\""}

```

Modulo Filtrar Turnos (spinners anidados)

Con las pruebas realizadas con Postman, se está en condiciones de elaborar los filtros para obtener los turnos que se requieran con precisión. Esto se diseñó en el código a modo de "spinner anidados", de tal manera que a medida que se seleccionen un campo, se filtre el siguiente consecutivo con la información pertinente.

Los *spinners* ofrecen una manera rápida de seleccionar un valor de un conjunto. En el estado predeterminado, un *spinner* exhibe su valor actualmente seleccionado. Al tocar el *spinner*, se visualiza un menú desplegable con todos los demás valores disponibles, de los cuales el usuario puede seleccionar solo uno. Puede agregarlo al diseño con el objeto *Spinner* en la lógica de negocio, y en el diseño XML con un elemento *<Spinner>*. Para rellenarlo con una lista de opciones, es necesario especificar un *SpinnerAdapter* en el código fuente de la *Activity* o el *Fragment*.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:

Las opciones que se proporcionen para el *spinner* pueden provenir de cualquier fuente, pero deben brindarse mediante un *SpinnerAdapter* (como un *ArrayAdapter*) si están disponibles en una matriz o un *CursorAdapter* si están disponibles a partir de una consulta de la base de datos.

En la imagen 24 se muestra una captura de pantalla de la vista preliminar de selección de turnos, y la mejora realizada para la interfaz de usuario (UI):

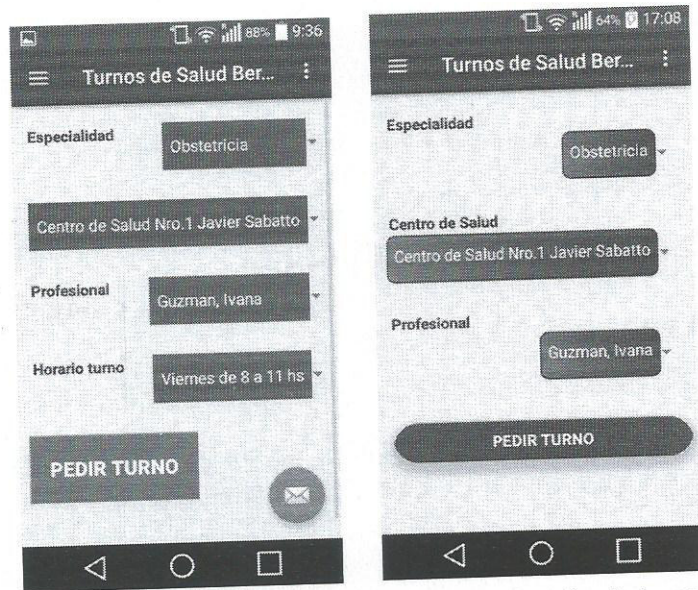


Imagen 24. Vista preliminar y vista mejorada de selección de turnos.
 Fuente: Elaboración propia, basada en la práctica.

El diseño de *simple_spinner_item* se proporciona mediante la plataforma y es el diseño predeterminado que se utiliza a menos que se desee definir un propio diseño para la apariencia del *spinner*. En este caso, se modificó el mismo para adquirir una apariencia mejorada estéticamente. En la imagen 25 se muestra cómo se ven los filtros, y luego cómo se implementaron en código fuente:

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:

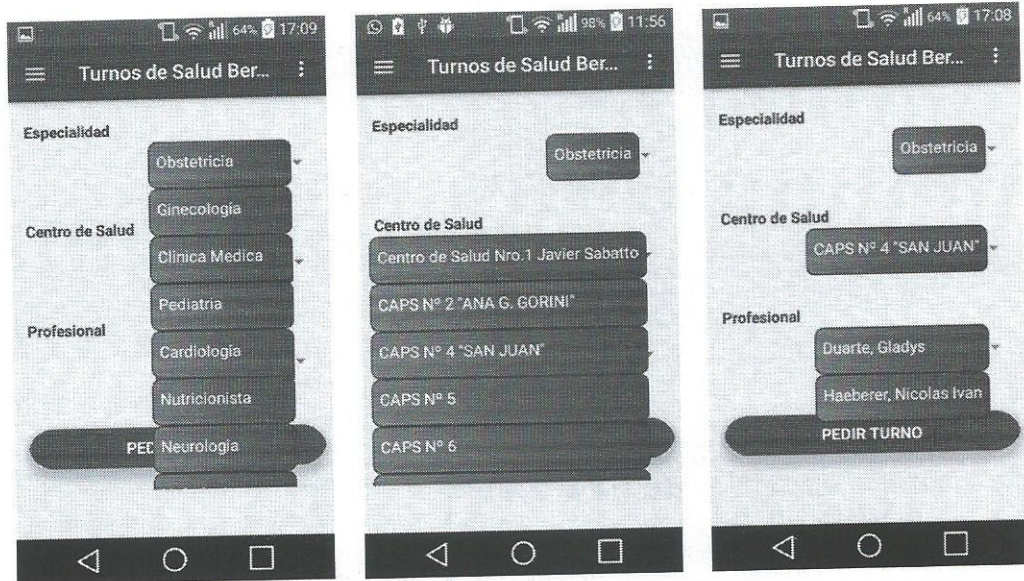


Imagen 25. Vistas secuenciales del funcionamiento de los filtros con spinners anidados.
Fuente: Elaboración propia, basada en la práctica.

Se declaran los spinners y se asignan a una vista ya creada en el layout:

```
final Spinner spinnerEspecialidad = (Spinner) vista.findViewById(R.id.spinner_especialidad);
final Spinner spinnerLugar = (Spinner) vista.findViewById(R.id.spinner_lugar);
final Spinner spinnerProfesional = (Spinner) vista.findViewById(R.id.spinner_profesional);
```

Se implementan spinners en forma anidada para ir filtrando según ítem seleccionado, a medida que se ejecutan distintos métodos se realizan las peticiones a la API que toman los datos requeridos y los agregan en *ArrayList* de clases tipo *Clave-Valor*, éstos contienen como parámetro "clave" su ID y en "valor" descripciones varias. Se muestra el código de los spinners anidados implementados y luego el código de una clase POJO (Plain Old Java Object es una instancia de una clase simple que no extiende ni implementa nada en especial y que no depende del framework utilizado, es puramente Java) de tipo *Clave-Valor* que se utiliza para guardar los datos del campo "Especialidad" con el método *getEspecialidad()*. Como alternativa se pudo haber utilizado el mapeo con *HashMap* o *TreeMap* para realizar esta acción, pero como los valores varían en cantidad de usa clase constructora a otra, se decidió implementarlos de esta manera.

```
getEspecialidad();
//comienzan los spinner anidados
spinnerEspecialidad.setAdapter(new ArrayAdapter<EspecialidadClaveValor>(getActivity(),
R.layout.support_spinner_dropdown_item, especialidadClaveValor));
spinnerEspecialidad.setOnItemClickListener(new AdapterView.OnItemClickListener() {
```

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------

```

@Override
public void onItemClick(AdapterView<?> adapterView, View view, int pos, long id)
{
    final EspecialidadClaveValor especialidadClaveValor = (EspecialidadClaveValor)
adapterView.getSelectedItemAt();
    getJsonLugarEspecialidadProfesionalHorarios("http://servidor
/unidades_sanitarias/esp/lugarEspecialidadProfesionalHorarios/Search",
"\n\tidEspecialidad:" + especialidadClaveValor.getId() + "\n");

    spinnerLugar.setAdapter(new ArrayAdapter<LugarClaveValor>(getActivity(),
R.layout.support_simple_spinner_dropdown_item, lugarClaveValor));
    spinnerLugar.setOnItemClickListener(new AdapterView.OnItemClickListener() {

        @Override
        public void onItemClick(AdapterView<?> adapterView, final View view, int pos,
long id) {

            final LugarClaveValor lugarClaveValor = (LugarClaveValor)
adapterView.getSelectedItemAt();
            profesionalClaveValor = new ArrayList<>();
            spinnerProfesional.setVisibility(vista.VISIBLE);
            getJsonLugarEspecialidadProfesionalHorarios2("http://servidor
/unidades_sanitarias/esp/lugarEspecialidadProfesionalHorarios/Search", "\n\tidLugar: " +
lugarClaveValor.getId() + "\n" + "\n\tidEspecialidad: " + especialidadClaveValor.getId() +
"\n");

            spinnerProfesional.setAdapter(new
ArrayAdapter<ProfesionalClaveValor>(getActivity(),
R.layout.support_simple_spinner_dropdown_item, profesionalClaveValor));
            spinnerProfesional.setOnItemClickListener(new
AdapterView.OnItemClickListener() {

                @Override
                public void onItemClick(AdapterView<?> adapterView, View view,
int pos, long id) {

                    dialog.dismiss();
                    final ProfesionalClaveValor profesionalClaveValor =
(ProfesionalClaveValor) adapterView.getSelectedItemAt();
                    botonPedirTurno.setVisibility(vista.VISIBLE);
                    botonPedirTurno.setOnClickListener(new View.OnClickListener() {

                        @Override
                        public void onClick(View v) {
                            confirmDialogDisponibles(especialidadClaveValor,
lugarClaveValor, profesionalClaveValor);
                        }
                    });
                }
            });
            //PROFESIONAL
        }
    });
}

@Override
public void onNothingSelected(AdapterView<?> adapterView) {
    botonPedirTurno.setVisibility(vista.GONE);
    spinnerProfesional.setVisibility(vista.GONE);
}

//LUGAR
@Override
public void onNothingSelected(AdapterView<?> parent)
{
    botonPedirTurno.setVisibility(vista.GONE);
    spinnerProfesional.setVisibility(vista.GONE);
}
}

```

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------

```
    }  
    //ESPECIALIDAD  
    @Override  
    public void onNothingSelected(AdapterView<?> parent) {  
    }  
}  
return vista;
```

```
public class EspecialidadClaveValor {  
    private String id;  
    private String descripcion;  
  
    public EspecialidadClaveValor(String id, String name) {  
        this.id = id;  
        this.descripcion = name;  
    }  
  
    public String getId() {  
        return id;  
    }  
  
    public void setId(String id) {  
        this.id = id;  
    }  
  
    public String getDescripcion() {  
        return descripcion;  
    }  
  
    public void setDescripcion(String descripcion) {  
        this.descripcion = descripcion;  
    }  
  
    @Override  
    public String toString() {  
        return descripcion;  
    }  
  
    @Override  
    public boolean equals(Object obj) {  
        if(obj instanceof EspecialidadClaveValor){  
            EspecialidadClaveValor c = (EspecialidadClaveValor) obj;  
            if(c.getDescripcion().equals(descripcion) && c.getId()==id) return true;  
        }  
        return false;  
    }  
}
```

Modulo Turnos Disponibles (lista)

Al seleccionar todos los campos y presionar el boton "pedir turno" se redirige a otra vista con los datos solicitados. Como primera opción, se muestran los resultados en forma de lista para ser seleccionado sólo uno mediante el uso de eventos *listeners* en el código fuente.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------

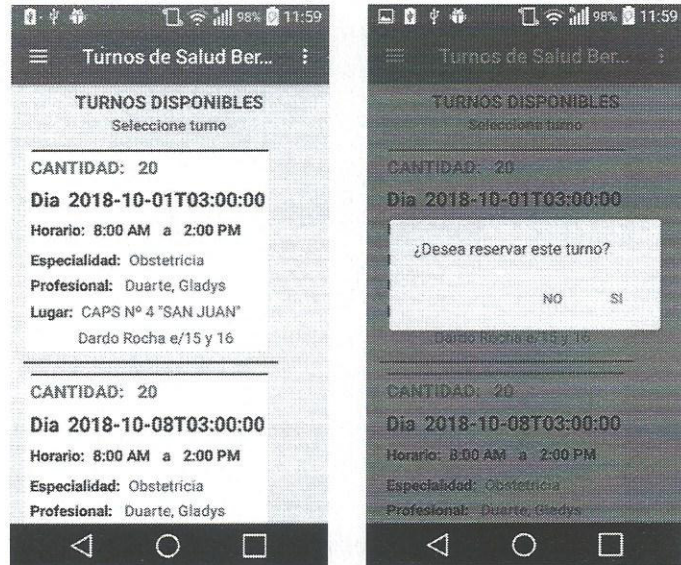


Imagen 26. Vistas secuenciales preliminares de los resultados de los turnos solicitados en forma de lista y su posterior selección con para reserva.
Fuente: Elaboración propia, basada en la práctica.

El almacenamiento de datos en esta acción se realizó mediante la clase *SharedPreferences*, que se utiliza para persistencia de datos en Android (no es el único método) al pasar de una pantalla a otra, por ejemplo. También es muy útil para guardar un número limitado de datos que sea necesario cargar en alguna otra instancia de la aplicación. En este caso, se guardan los datos en un *Fragment* de los turnos disponibles filtrados y se cargan en otro *Fragment* que los muestra en forma de lista mediante un *Adapter*.

Guardado de datos:

```
SharedPreferences sharedPreferences =
getActivity().getSharedPreferences("TurnosDisponibles", getActivity().MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putString("id_especialidad", especialidadClaveValor.getId());
editor.putString("id_lugar", lugarClaveValor.getId());
editor.putString("id_profesional", profesionalClaveValor.getId());
editor.putString("descripcion_especialidad", especialidadClaveValor.getDescripcion());
editor.putString("descripcion_lugar", lugarClaveValor.getDescripcion());
editor.putString("descripcion_profesional", profesionalClaveValor.getApellido() + " " +
profesionalClaveValor.getNombre());
editor.putString("direccion_lugar", lugarClaveValor.getDireccion());
editor.commit();
```

Carga de datos y guardados en variables para luego ser utilizadas:

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



```
SharedPreferences prefUsuario = getContext().getSharedPreferences("TurnosDisponibles",  
MODE_PRIVATE);  
final String especialidad = prefUsuario.getString("descripcion_especialidad","");  
final String lugar = prefUsuario.getString("descripcion_lugar","");  
final String profesional = prefUsuario.getString("descripcion_profesional","");  
String domicilio = prefUsuario.getString("direccion_lugar","");
```

Estas variables con los datos guardados son utilizadas para rellenar los turnos disponibles a mostrar mediante la clase implementada *TurnoDisponible* y adaptadas con *AdaptadorTurnos* (adapter) mediante un *RecyclerView* como se detalla a continuación con código extraído de la aplicación.

Declaraciones:

```
ArrayList<TurnoDisponible> listaTurnosDisponibles;  
RecyclerView recyclerViewTurnosDisponibles;  
recyclerViewTurnosDisponibles = (RecyclerView) vista.findViewById(R.id.recyclerId);  
recyclerViewTurnosDisponibles.setLayoutManager(new LinearLayoutManager(getContext()));
```

En la iteración (*for*) cada turno disponible leído se guarda en un *ArrayList*, todo esto dentro del método *doInBackground()* de la clase *AsyncTask*:

```
TurnoDisponible turnoDisponible = new TurnoDisponible(id_turno, fecha, horaDesde, horaHasta,  
cantidad, especialidad, lugar, domicilio, profesional);  
listaTurnosDisponibles.add(turnoDisponible);
```

Luego, en el método *onPostExecute()* de la clase *AsyncTask* se consulta si la lista está vacía para hacer la adaptación de la información a la vista, y en caso de presionar un elemento de la lista que corresponde a un turno, se pregunta al usuario mediante un *AlertDialog* si desea reservar el turno (pasándole como parámetros el ID del turno a reservar, la posición del ítem seleccionado y el adapter propio):

```
if(listaTurnosDisponibles != null) {  
    final AdaptadorTurnos adapter = new AdaptadorTurnos(listaTurnosDisponibles);  
    adapter.notifyDataSetChanged();  
    recyclerViewTurnosDisponibles.setAdapter(adapter);  
  
    getActivity().setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_UNSPECIFIED);  
    adapter.setOnClickListeners(new View.OnClickListener() {  
        @Override  
        public void onClick(View view) {  
            String idTurnoAReservar =  
listaTurnosDisponibles.get(recyclerViewTurnosDisponibles.getChildAdapterPosition(view)).getId();  
            int posicion = recyclerViewTurnosDisponibles.getChildAdapterPosition(view);  
            InterfaceComunicacionFragmenta.enviarTurnoDisponible(listaTurnosDisponibles.get(recyclerViewTurnosDisponibles.getChildAdapterPosition(view)));  
            confirmDialog(idTurnoAReservar, posicion, adapter);  
        }  
    });  
}
```

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------



La clase *TurnoDisponible* es un *JavaBean* (o simplemente *Bean*), se trata de una clase simple en Java que cumple con ciertas normas con los nombres de sus propiedades y métodos. Debe tener un constructor sin argumentos, tiene declarados todos sus atributos como privados y para cada uno de ellos un método *setter* y *getter*, añadiéndole la palabra "set" o "get" al nombre del atributo y deben ser serializables (implementa la clase *Serializable*, y en este caso también *Parcelable*). Mediante estos *JavaBeans*, se desarrollan modelos de objetos para la aplicación.

Para lograr adaptar la información a la vista, se utilizó el patrón *Adapter*, que se conforma de una *interface* que hace de comunicación entre los *Fragments* y luego se efectúa la adaptación mediante la clase *AdaptadorTurnos*:

```
public interface ComunicacionFragments {
    public void enviarTurnoDisponible(TurnoDisponible turnoDisponible);
}
```

```
public class AdaptadorTurnos extends
RecyclerView.Adapter<AdaptadorTurnos.TurnosDisponiblesViewHolder> implements
View.OnClickListener {

    ArrayList<TurnoDisponible> listaTurnosDisponibles;
    private View.OnClickListener listener;

    public AdaptadorTurnos(ArrayList<TurnoDisponible> listaTurnosDisponibles) {
        this.listaTurnosDisponibles = listaTurnosDisponibles;
    }

    @Override
    public TurnosDisponiblesViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view =
        LayoutInflater.from(parent.getContext()).inflate(R.layout.item_turno_disponible, null,
        false);
        view.setOnClickListener(this);
        return new TurnosDisponiblesViewHolder(view);
    }

    @Override
    public void onBindViewHolder(TurnosDisponiblesViewHolder holder, int position) {
        holder.textFecha.setText(listaTurnosDisponibles.get(position).getFecha());
        holder.textHoraDesde.setText(listaTurnosDisponibles.get(position).getHoraDesde());
        holder.textHoraHasta.setText(listaTurnosDisponibles.get(position).getHoraHasta());
        holder.textCantidad.setText(listaTurnosDisponibles.get(position).getCantidad());
        holder.textProfesional.setText(listaTurnosDisponibles.get(position).getProfesional());
        holder.textLugar.setText(listaTurnosDisponibles.get(position).getLugar());
        holder.textDireccion.setText(listaTurnosDisponibles.get(position).getDomicilio());
        holder.textEspecialidad.setText(listaTurnosDisponibles.get(position).getEspecialidad());
    }

    @Override
    public int getItemCount() {
        return listaTurnosDisponibles.size();
    }
}
```

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------

```

public void setOnClickListener(View.OnClickListener listener){
    this.listener = listener;
}

@Override
public void onClick(View view) {
    if (listener != null) {
        listener.onClick(view);
    }
}

public class TurnosDisponiblesViewHolder extends RecyclerView.ViewHolder{
    TextView textID, textFecha, textHoraDesde, textHoraHasta, textCantidad,
    textDireccion, textProfesional, textLugar, textEspecialidad, textDisponible,
    textNoDisponible;

    public TurnosDisponiblesViewHolder(View itemView) {
        super(itemView);
        textFecha = (TextView) itemView.findViewById(R.id.dinamico_fecha);
        textCantidad = (TextView) itemView.findViewById(R.id.dinamico_cantidad);
        textHoraDesde = (TextView) itemView.findViewById(R.id.dinamico_horaDesde);
        textHoraHasta = (TextView) itemView.findViewById(R.id.dinamico_horaHasta);
        textDireccion = (TextView) itemView.findViewById(R.id.dinamico_direccion);
        textEspecialidad = (TextView) itemView.findViewById(R.id.dinamico_especialidad);
        textLugar = (TextView) itemView.findViewById(R.id.dinamico_lugar);
        textProfesional = (TextView) itemView.findViewById(R.id.dinamico_profesional);
    }
}

```

Módulo Turnos Activos (reservados)

De forma similar a los turnos disponibles se genera un listado de turnos activos, al cual se puede acceder desde el menú *nav bar* o lateral. Este listado muestra los turnos disponibles seleccionados y confirmados por el usuario, por lo que ya fueron reservados, de modo que se van almacenando en la sección o *Fragment* "turnos activos" y al acceder se ofrece la información relevante de los mismos como también la posibilidad de ser cancelados individualmente. La diferencia en la implementación de este módulo, respecto de Turnos Disponibles, es que en este caso los datos debieron ser insertados como registros en la base de datos mediante el uso de la API con `"/api/TurnoPaciente/Insert"` al confirmar la reserva en el calendario, con el siguiente método acotado para mostrar que se envía en el cuerpo de la petición (body):

```

public String insertTurnoPaciente(String idTurno, String idPaciente, String idLugar,
    AdapterTurnos adapter, final int posicion){
    String apiRest = "http://192.168.1.100/unidades_sanitarias/api/TurnoPaciente/Insert";
    final SimpleDateFormat fecha = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss");
    String fechaActual = fecha.format(new Date());
    String body = "\n\tidPaciente: " + idPaciente + "," + "\n\tidTurno: " + idTurno + "," +
        "\n\tfechaAsignacion: " + fechaActual + "," + "\n\tfechaAsistencia: " + null +
        "\n\tasignado: " + true + "," + "\n\tret: " + false + "," + "\n\tCOPSA: " +
        false + "," + "\n\tesSporTurno: " + false + "," + "\n\tidLugar: " + idLugar + ",";
}

```

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------

```
+ "\n\tfechaBaja: " + null + "," + "\n\tHistoriaClínica: " + "[ ]" + "," +
"\n\t";
```

Una vez insertados los turnos, en `TurnosActivosFragment` se trae esta información para ser visualizada en su sección con `"/api/TurnoPaciente/Search"` y al body se le pasa `idPaciente` e `idTurno`.

Como se observa en la imagen 27, fue necesario *parsear* y formatear la fecha para una mejor UI. Al seleccionar un ítem, que referencia un turno, ofrece la chance de eliminarlo, o mejor dicho desde la perspectiva del usuario, cancelar la reserva. Esto se logra accediendo al recurso con `"/api/TurnoPaciente/Delete"` y pasándole los ID del paciente y turno como parámetros en el cuerpo de la petición a enviar como en el caso anterior.

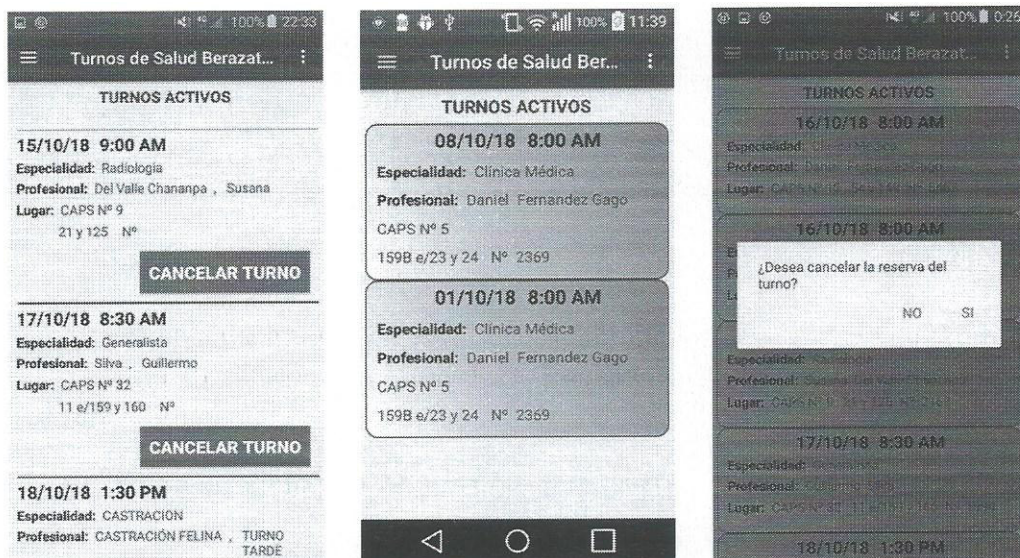


Imagen 27. Vista preliminar y vistas finales del módulo Turnos Activos, de izquierda a derecha.

Fuente: Elaboración propia, basada en la práctica.

Calendario

Para interactuar con un calendario se utilizó una librería de Android llamada *CompactCalendarView* que proporciona una vista de calendario compacta como la que se usa en los calendarios de Google.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:



CompactCalendarView es una vista de calendario simple que proporciona desplazamiento entre meses. Está basado en las clases de fecha y calendario de Java. Proporciona una API simple para consultar fechas y escuchas (*listeners* en Android) para eventos específicos. Por ejemplo, cuando el calendario se ha desplazado a un nuevo mes o se ha seleccionado un día. Todavía se encuentra en desarrollo activo, especificado en el repositorio de GitHub: <https://github.com/SundeepK/CompactCalendarView>

Es necesario especificar en las dependencias de build.gradle el uso de la librería con:

```
implementation 'com.github.SundeepK:CompactCalendarView:1.0.0'
```

Es posible cambiar el aspecto de la vista a través de algunas propiedades. Esto incluye el color de fondo, el color del texto, el tamaño del texto del día actual y el color del primer día del mes.

Es posible también establecer la configuración regional para que el calendario configure automáticamente los nombres de las columnas del día de la semana.

```
CompactCalendarView compactCalendarView = (CompactCalendarView)  
findViewById(R.id.compactcalendar_view);  
compactCalendarView.setLocale(Locale.SPANISH); //o getDefault()  
compactCalendarView.setUseThreeLetterAbbreviation(true);
```

Modulo Turnos Disponibles (calendario)

Para complementar o suplantar la vista en forma de lista de los turnos disponibles, se buscó cómo realizar la muestra de resultados sobre un calendario, a partir de un selector como se observa en la imagen 28 utilizando un *AlerDialog*. El color azul oscuro representa la fecha actual. Para mostrar los turnos disponibles se realizó la generación de eventos de calendario por cada turno filtrado para ser mostrado con color verde, y el color amarillo representa la fecha seleccionada por el usuario, lo cual es intuitivo para el mismo. Se detallan a continuación:

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------

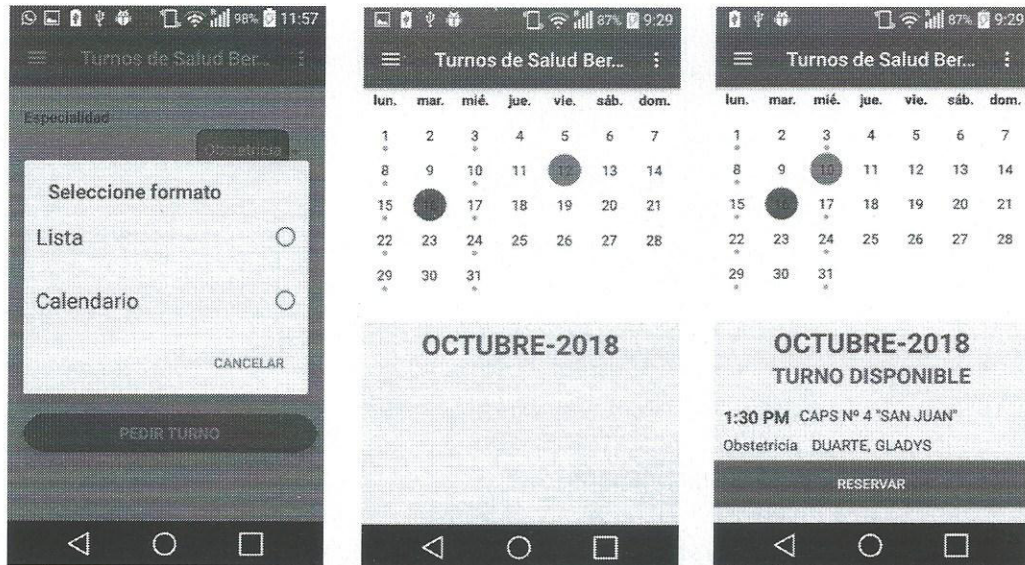


Imagen 28. Vistas preliminares secuenciales de los resultados de los turnos solicitados sobre el calendario.

Fuente: Elaboración propia, basada en la práctica.

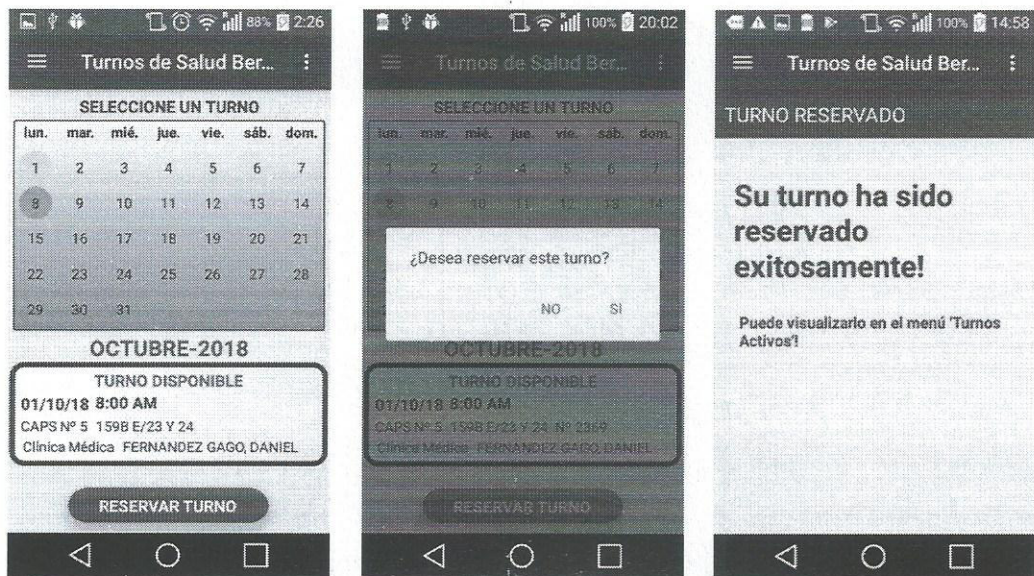


Imagen 29. Vistas finales y secuenciales del proceso de reserva de un turno sobre el calendario.

Fuente: Elaboración propia, basada en la práctica.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



Es importante mencionar las conversiones que debieron realizarse de los formatos de fechas, fue necesario realizarlas con la ayuda de la aplicación web <https://www.epochconverter.com/>, la cual ofrece información sobre cómo están compuestos los formatos y también cómo implementarlos en el código fuente con varios lenguajes de programación.

Java proporciona una clase llamada *SimpleDateFormat*, que permite *parsear* (texto a fecha), *formatear* (fecha a texto) y *normalizar* las fechas según sus requisitos al entorno local. Dicho esto, primero es necesario tomar el *string* del objeto JSON con campo "fecha" del turno, el cual tiene un formato específico en la base de datos con el patrón "yyyy-MM-dd'T'HH:mm:ss", por lo que es preciso hacer el *parseo*, formateo y finalmente obteniendo el resultado que se desea mostrar en la aplicación con el *string* "fechaFinal", de modo que se visualice con el formato "dd/MM/yy" (dia/mes/año). A continuación, se muestra este proceso logrado en el código fuente:

```
Final String fecha = jsonObject.getString("fecha");
SimpleDateFormat parseador = new SimpleDateFormat("yyyy-MM-dd"); // parseo
SimpleDateFormat formateador = new SimpleDateFormat("dd/MM/yy"); // de formato
Date date = parseador.parse(fecha);
String fechaFinal = formateador.format(date);
```

Para situar los turnos disponibles en el calendario se debió obtener el *datetime* (que es de tipo *long*) de cada turno para añadirlo a cada evento y luego agregarlo a la vista del calendario como se observa en el siguiente fragmento de código. Luego mediante un *listener* del mismo se realizan las acciones necesarias para que interactúe el usuario.

```
long dateTime = date.getTime();
Event event0 = new Event(Color.GREEN, dateTime, "Evento");
listaEventos.add(event0);
TurnoDisponibleCalendario turnoDisponibleCalendario = new
TurnoDisponibleCalendario(id_turno, fecha, horaDesde, horaHasta, especialidad, id_lugar,
lugar, domicilio, profesional, date, event0);
listaTurnosDisponibles.add(turnoDisponibleCalendario);
compactCalendarView.addEvent(turnoDisponibleCalendario.getEvent());
```

Módulo Recordatorio de Turno

Este módulo define la creación de eventos en Google Calendar, que es una agenda y calendario electrónico que permite sincronizarlo con los contactos de Gmail con el objetivo de invitarlos y compartir eventos. Utilizando esta herramienta y mediante la sincronización de una cuenta de usuario de Gmail, se añadirán las reservas de turnos como eventos de dicho calendario de forma interactiva y con la opción de establecer

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------



recordatorios. Otra característica es que funciona con todos los calendarios del dispositivo, por lo que el calendario nativo del dispositivo visualizará también estos eventos.

Para lograr esto, y justo en el momento en que la petición “../api/TurnoPaciente/Insert” devuelva *true* (condición de aceptación de reserva de turno), se llama al siguiente método:

```

public void recordatorioCalendario(String fechaParseada, int dateYear, int dateMonth, int
dateDay, long dateTime, String hora, String minuto, String descripcion, String localizacion)
{
    Calendar cal = Calendar.getInstance();
    boolean val = false; //controla el ciclo while
    Intent intent = null;
    while (val == false) {
        try {
            cal.set(Calendar.YEAR, dateYear); // pone año, mes y día
            cal.set(Calendar.MONTH, dateMonth);
            cal.set(Calendar.DAY_OF_MONTH, dateDay);
            cal.set(Calendar.HOUR_OF_DAY, Integer.parseInt(hora)); // pone hora y minutos
            cal.set(Calendar.MINUTE, Integer.parseInt(minuto));
            intent = new Intent(Intent.ACTION_EDIT);
            intent.setType("vnd.android.cursor.item/event");
            intent.putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME, cal.getTimeInMillis());
            intent.putExtra(CalendarContract.EXTRA_EVENT_END_TIME, cal.getTimeInMillis() +
60 * 60 * 1000);
            intent.putExtra(CalendarContract.Events.ALL_DAY, true);
            intent.putExtra(CalendarContract.Events.TITLE, "Recuerda asistir al turno el " +
fechaParseada + " [" + "]"); // mensaje a mostrar del evento, luego una descripción y ubicación
            intent.putExtra(CalendarContract.Events.DESCRPTION, descripcion);
            intent.putExtra(CalendarContract.Events.EVENT_LOCATION, localizacion);
            startActivity(intent);
            val = true;
        } catch (Exception e) {}
    }
}

```

Al ejecutarse este método, aparece en pantalla un breve asistente con la opción de guardar el turno como evento y establecer un recordatorio, todo esto situando la información del turno en los diferentes campos. Si se acepta guardar, lo muestra como se observa en las imágenes 30 y 31.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------

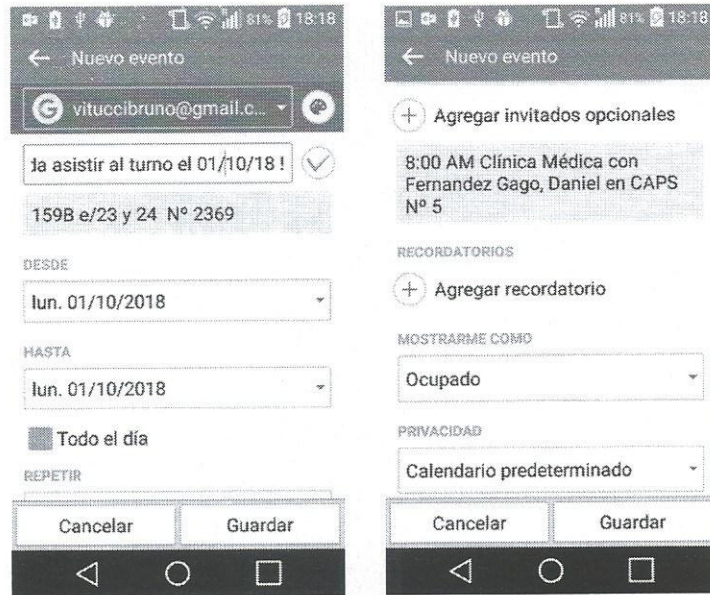


Imagen 30. Vistas secuenciales del turno reservado, agregado como un evento al calendario de Google y al nativo del dispositivo.
Fuente: Elaboración propia, basada en la práctica.

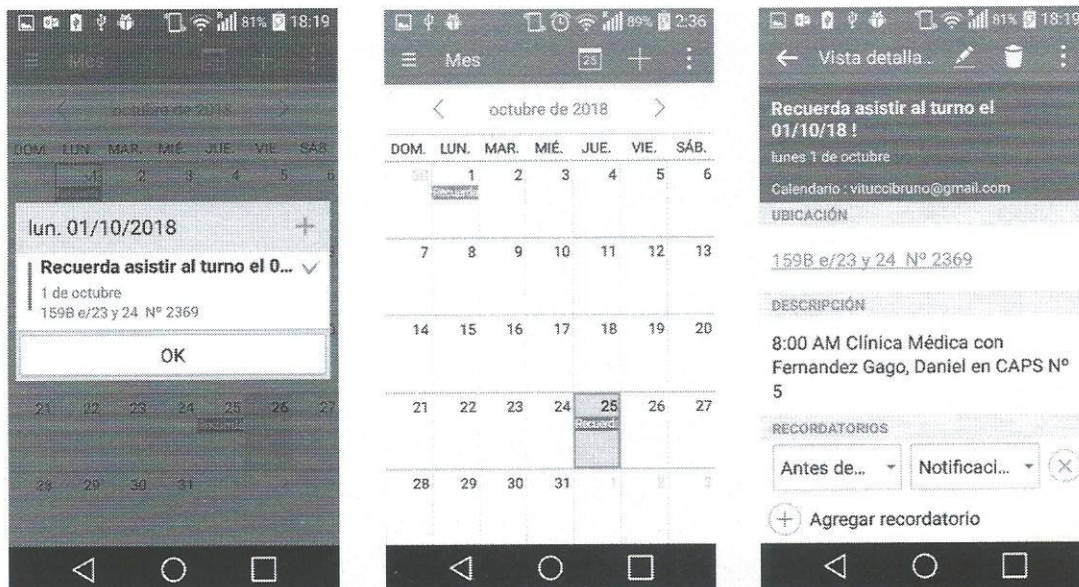


Imagen 31. Vistas secuenciales del evento recordatorio del turno, el cual puede mostrar información sobre el mismo y su ubicación en mapa.
Fuente: Elaboración propia, basada en la práctica.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



También se puede visualizar esta información desde la app móvil de Google Calendar o desde la app web, como se demuestra en la imagen 32.

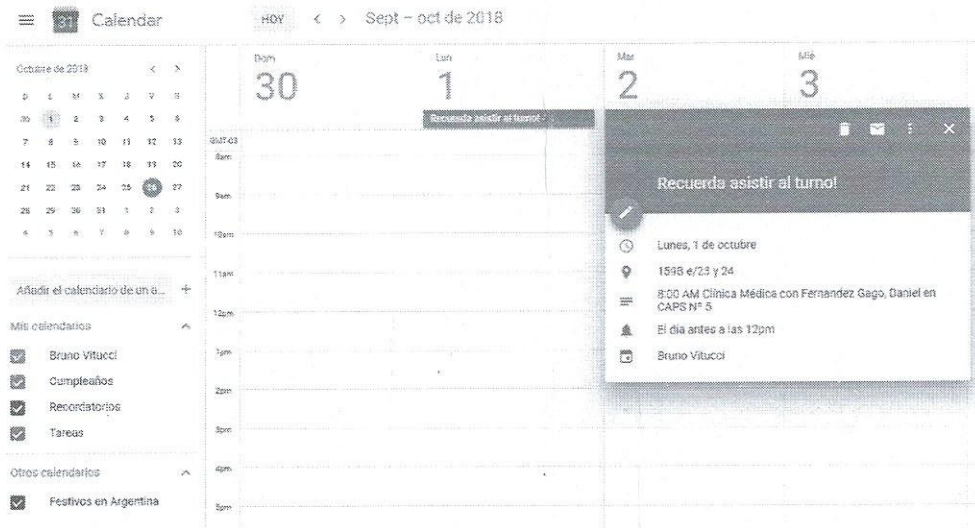


Imagen 32. Vista de la app web de Google Calendar con el turno guardado como evento.

Fuente: Recuperado de <https://calendar.google.com/calendar/> (2018)

Refactorización de código

Para finalizar, se realizó una refactorización de código fuente, que implica eliminar clases y archivos XML en desuso, editar nombres de archivos y métodos para que describan mejor su propósito, como también borrar código comentado, dejar solo lo necesario y que describa que es lo que hacen determinadas líneas o funcionalidades.

Previo a producción, el proyecto pasa por el ambiente de *testing* del área de sistemas, quienes aplican pruebas funcionales y unitarias sobre la app para finalmente presentarla a los *stakeholders*.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------

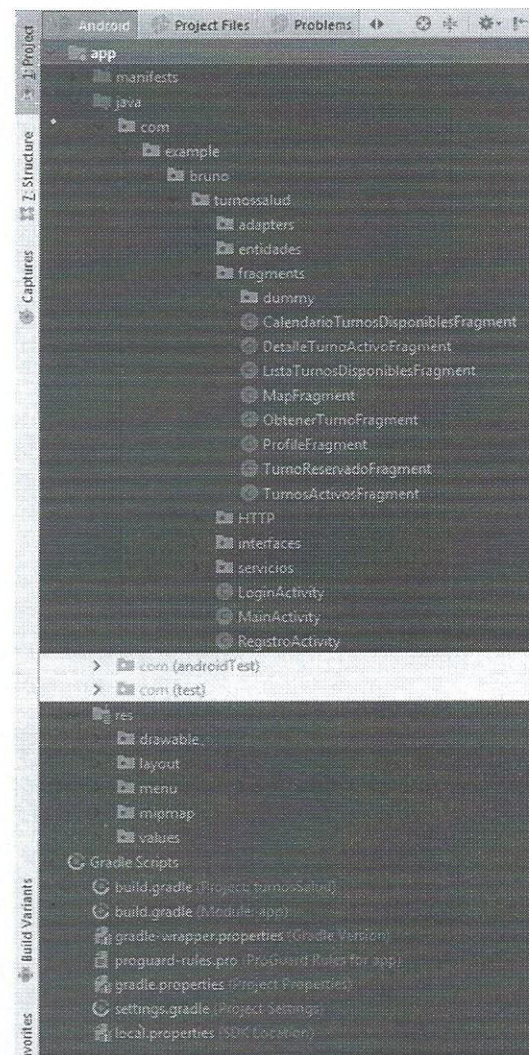


Imagen 33. Estructura del proyecto manejada por el IDE.
 Fuente: Elaboración propia, basada en la práctica.

Mejoras futuras para el sistema en desarrollo

Finalizado el desarrollo del sistema propuesto y observando las funcionalidades que posee actualmente, se sugieren otras para mejorar y reforzar lo ya realizado.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



Una de las posibles mejoras es que el usuario tenga mayor interacción con el mapa, que le permita conocer las especialidades del CAPS seleccionado, y también solicitar turnos desde este módulo, trasladando la información desde el mapa hasta el módulo de selección de especialidad, lugar y profesional, de manera que al usuario sólo le reste escoger el profesional para pedir los turnos disponibles.

Otra mejora para el mismo módulo, en cuanto a la funcionalidad que facilita conocer el CAPS más cercano al usuario, consiste en ofrecer la información de la distancia entre ellos mediante notificaciones Toast o AlertDialog. Esta información debe implementarse de modo tal que se muestre en pantalla dependiendo de la distancia, por ejemplo:

- Menor a 100 metros: debe mostrarse en metros.
- Mayor a 100 metros y menor a 1000 metros: debe mostrarse en cuadas.
- Mayor a 1000 metros: debe mostrarse en kilómetros y cuadas.

Además, sería posible añadir los horarios de las guardias de cada CAPS, como también las ubicaciones de las farmacias de turno, con sus respectivos horarios. Valores que actualmente no se encuentran implementados en la base de datos.

Por último, ha surgido la idea de establecer especialidades o CAPS como favoritos para el usuario de la aplicación, siendo los más solicitados o recurrentes del mismo. Esta diferenciación respecto de las especialidades/CAPS restantes se puede determinar a través de un icono distintivo.

Conclusiones

El desarrollo del proyecto realizado permitió de manera progresiva aclarar ciertos aspectos que en un principio se vislumbraban por falta de conocimiento del sistema con el que se debió trabajar conjuntamente con desarrolladores del área de Sistemas. Dichos aspectos tuvieron que ser analizados y estudiados minuciosamente para lograr llevar a cabo la integración al nuevo sistema propuesto. Esto ayuda a entender que es fundamental la comunicación entre los interesados del proyecto, el análisis del E.R.S. (Especificación de Requisitos de Software) para conocer cómo debe comportarse el sistema que se pretende desarrollar, como también la capacitación e interiorización del desarrollador en materia de herramientas a utilizar y documentación sobre las mismas.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------

Por otro lado, es grato concientizarse sobre los resultados obtenidos y los beneficios sociales que acarrea el disponer de la herramienta desarrollada, la cual habilita una alternativa moderna para que los ciudadanos puedan recibir información de los centros de salud de la ciudad, obtener turnos para ser atendidos, como también disponer de recordatorios para evitar ausentismo, todo al alcance del usuario en pos de las ventajas que brindan la tecnología móvil y los sistemas de información.

Reflexión sobre la Práctica Profesional Supervisada como espacio de formación

En esta sección se propicia la reflexión personal brindada por la Práctica Profesional Supervisada en la formación del futuro Ingeniero en Informática.

Los conocimientos adquiridos en esta etapa son vastos, destaco principalmente la experiencia de trabajar con un equipo de desarrollo de software en un ámbito profesional, permitiéndome conocer y respetar la jerarquía de cargos dentro de la Secretaria de Modernización del Estado, que apunta a fomentar el uso de sistemas informáticos como también resolver problemas de dicha índole a las instituciones del partido de Berazategui para los ciudadanos. Además, esta experiencia me permitió obtener una amplia visualización de oportunidades para aplicar los conocimientos adquiridos gracias a la Práctica como también en el marco universitario.

Algunos de los contenidos, que se han recuperado del proceso de formación académica hasta el momento, fueron los siguientes:

- Análisis de bases de datos relacionales, gracias al aporte de Base de Datos I y Base de Datos II.
- Análisis de sistemas existentes para la implementación de una aplicación, por las contribuciones de las asignaturas Proyecto de Software, Ingeniería de Software II, y Administración de Proyectos.
- Patrones de diseño, estructuración y reutilización de código, aportes de Metodologías de Programación I.
- Desarrollo e implementación de aplicaciones, contenidos adquiridos de Algoritmos y Programación, Complejidad Temporal, Estructuras de Datos y Algoritmos, y Aplicaciones Móviles.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------

- Manejo de hilos en los procesos, por parte de Sistemas Operativos I, Sistemas Operativos II, y Sistemas Distribuidos

He logrado consolidar distintos aprendizajes que proporcionan facilitar mi futuro desempeño profesional, como por ejemplo, conocer distintos perfiles laborales de los interesados del proyecto desarrollado para orientarme en base a sus posibles conocimientos, de esta manera se pueden establecer relaciones personales y profesionales con ellos. Teniendo relación con lo mencionado, la interacción con el equipo de desarrollo ha sido muy satisfactoria, permitiéndome aprender cómo desenvolverme y actuar en el entorno generando soluciones viables para cada situación. Además, he aprendido que las oportunidades de aplicar mis conocimientos son prácticamente ilimitadas, ya que la T.I. (Tecnología de la Información) está en auge desde hace años y sigue en crecimiento.

Respecto a las *fortalezas* identificadas en el transcurso de mis tareas realizadas, he notado que la formación académica ayudó considerablemente a alcanzar los objetivos propuestos, percibiendo cómo convergen los conocimientos obtenidos en la formación, como también el autoaprendizaje y la voluntad para llevar a cabo las responsabilidades que me confirieron, llegando a realizarlas en tiempo y forma. Como *oportunidades de mejora* se destaca la falta de utilización de herramientas y tecnologías que pudieron mejorar el desempeño del proyecto y del resultado final, algunas de ellas fueron nombradas como alternativas para el desarrollo, y si hubiera probado trabajar con dichas herramientas probablemente los resultados serían óptimos, aunque, eventualmente, también se hubieran presentado algunas desventajas.

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	----------------------------	----------------------



Bibliografía

- Bosetti, G. (09/02/2013). "Design Patterns with UML: Adapter Pattern". Disponible en:
<<http://design-patterns-with-uml.blogspot.com/2013/02/adapter-pattern.html>>.
[Consulta: 15 de septiembre de 2018].
- Compañía Exes (s.f.). "Área de Programación y Desarrollo: Curso de Introducción a Java".
Disponible en: <http://mundojava.net/el-lenguaje-java.html?Pg=java_inicial_4.html>. [Consulta: 02 de septiembre de 2018].
- Eckel, B. (1998). Piensa en Java (Edición española). España, Pearson Educación, 2002.
- El Titular (s.f.). "Lenguajes: Java". Disponible en:
<<https://lenguajesdeprogramacion.net/java/>>. [Consulta: 02 de septiembre de 2018].
- Gamma, E. (1995). Design Patterns: Elements of Reusable Object-Oriented Software (1st Edition). London, Addison-Wesley.
- García, D. (28/02/2014). "Patrones Estructurales: Patrón Adapter". Disponible en:
<<https://danielggarcia.wordpress.com/2014/02/28/patrones-estructurales-i-patron-adapter-wrapper/>>. [Consulta: 15 de septiembre de 2018].
- Garzas, J. (22/11/2011). "Kanban para la gestión de proyectos". Disponible en:
<<http://www.javiergarzas.com/2011/11/kanban.html>>. [Consulta: 2 de septiembre de 2018].
- Java™ Platform Standard Ed. 7 (s.f.). "Class SimpleDateFormat". Disponible en:
<<https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html>>.
[Consulta: 15 de octubre de 2018].
- Misja.com (s.f.). "Convert from human readable date to epoch". Disponible en:
<<https://www.epochconverter.com/>>. [Consulta: 15 de octubre de 2018].
- Moisset, D. (s.f.). "Android Ya". Disponible en:
<<http://www.tutorialesprogramacionya.com/javaya/androidya/>>. [Consulta: 10 de septiembre de 2018].

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa:
-------------------	-------------------	-------------------------	----------------------



Oracle Corporation [US] (s.f.). Disponible en:
<<https://www.oracle.com/technetwork/topics/newtojava/documentation/index.html>>.
[Consulta: 29 de agosto de 2018].

Perry, J. S. (03/12/2012, IBM developerWorks). "Introducción a la programación Java, parte 1". Disponible en: <<https://www.ibm.com/developerworks/ssa/java/tutorials/j-introtojava1/index.html>>. [Consulta: 04 de septiembre de 2018].

Sitio oficial de Android Developers (s.f.). "Guías para desarrolladores". Disponible en:
<<https://developer.android.com/studio/>>. [Consulta: 1 de septiembre de 2018].

Sitio oficial de Google Developers (s.f.). "La API de Google para Android". Disponible en:
<<https://developers.google.com/android/>>. [Consulta: 1 de septiembre de 2018].

TIOBE, the software quality company (2018). "TIOBE Index for October 2018". Disponible en: <<https://www.tiobe.com/tiobe-index/>>. [Consulta: 20 de octubre de 2018].

Firma Estudiante:	Firma tutor UNAJ:	Firma tutor TAPTA UNAJ:	Firma tutor Empresa: