Public Health Data Dissemination Strategies: Development of Widgets and APIs for a Case Study with INCUCAI

Carlos Bateca, Roxana Martínez, Santiago Acha, Victoria Agnelli

Centro de Altos Estudios en Tecnología Informática (CAETI)
Universidad Abierta Interamericana (UAI)
Buenos Aires, Argentina
{roxana.martinez, santiago.acha}@uai.edu.ar;
{CarlosAlberto.BatecaCaicedo,
Victoria.Agnelli}@alumnos.uai.edu.ar

Abstract. Access to accurate and timely information is crucial today, especially in public health. In the context of organ and tissue donation, data enables the tracking of patients on waiting lists, improves resource planning, optimizes service quality, and promotes transparency. In Argentina, INCUCAI centralizes and disseminates this information, making it essential for citizens to effectively access data to raise awareness about organ donation and reduce the number of patients waiting for a transplant. Technological advancements and the increasing use of mobile devices have created new opportunities for the dissemination of public data. In this context, widgets emerge as an innovative solution to provide immediate access to relevant information on mobile devices. This paper proposes the development of a platform that extracts key indicators from the INCUCAI website, stores them in a SQL Server database, and enables their real-time visualization through a widget on Android devices. To achieve this, an API is implemented to facilitate communication between the database and the application. The main objective is to enhance access to organ donation data through an accessible and efficient technological solution, promoting transparency and citizen engagement in public health matters.

Keywords: Open Data, Public Health, Organ Donation, Widgets, APIs, Transparency.

Estrategias de Difusión de Datos de Salud Pública: Desarrollo de Widgets y APIs para Caso de Estudio con INCUCAI

Resumen. El acceso a información precisa y oportuna es clave en la sociedad actual, especialmente en el ámbito de la salud pública. En materia de donación de órganos y tejidos, los datos permiten conocer la cantidad de pacientes en lista de espera, mejorar la planificación de recursos, optimizar la calidad del servicio y fomentar la transparencia. En Argentina, el INCUCAI centraliza y difunde

esta información, siendo fundamental que los ciudadanos accedan de manera eficaz a los datos para concientizar sobre la donación de órganos y reducir la cantidad de pacientes en espera. El avance tecnológico y el crecimiento del uso de dispositivos móviles han abierto nuevas oportunidades para la difusión de datos públicos. En este contexto, los widgets se presentan como una solución innovadora para brindar acceso inmediato a información relevante en dispositivos móviles. Este trabajo propone el desarrollo de una plataforma que extrae indicadores clave desde el sitio web de INCUCAI, almacenándolos en un servidor SQL Server y permitiendo su visualización en tiempo real a través de un widget en dispositivos Android. Para ello, se implementa una API que facilita la comunicación entre la base de datos y la aplicación. El objetivo principal es mejorar el acceso a los datos de donación de órganos mediante una solución tecnológica accesible y eficiente, promoviendo la transparencia y la participación ciudadana en temas de salud pública.

Palabras clave: Datos Abiertos, Salud Pública, Donación de Órganos, Widgets, APIs, Transparencia.

1 Introducción

Los datos públicos son de suma importancia para la sociedad en general, estos suponen una simetría en el dialogo ciudadanos-administración y promueve la transparencia y colaboración entre ciudadanos, empresas y sociedad civil. Además, favorece la participación activa en la formulación de políticas por parte de la población [1]. Entre las distintas áreas que abordan los datos públicos, la donación de órganos es una de las más críticas ya que actualmente representa una alternativa de tratamiento para un creciente número de enfermedades, otrora consideradas terminales, reflejando una realidad en donde miles de personas esperan un trasplante mientras el sistema enfrenta desafíos logísticos y éticos que generan una desproporción entre la limitada demanda y la oferta de órganos, lo que ha propiciado prácticas indeseables como el tráfico de seres humanos [2]. En Argentina, el Instituto Nacional Central Único Coordinador de Ablación e Implante (INCUCAI) [3] es quien normatiza, coordina y fiscaliza las actividades de donación y trasplante de órganos, tejidos y células en nuestro país, además provee datos públicos a través de la Central de Reportes y Estadísticas del SINTRA (CRESI) [4]. Sin embargo, la disponibilidad de estos datos a través de su página web no garantiza que sea de interés para la mayoría del público ya que implica el acceso regular a la plataforma y familiarizarse con su uso, aunado con que muchas personas no tienen el habito de buscar esta información por iniciativa propia. Lo que plantea la necesidad de explorar alternativas de difusión automatizadas que sean más amigables y accesibles.

Los widgets (también llamados Gadgets) son pequeñas aplicaciones, basadas en Web 2.0, que pueden ser instaladas en dispositivos móviles y tienen como objetivo dar fácil acceso a funciones que son utilizadas frecuentemente, así como también el proveer información visual en tiempo real [5]. Esta tecnología permite interactuar con servicios e información distribuida en Internet; pueden ser vistosos relojes en pantalla, notas, calculadoras, calendarios, agendas, juegos, ventanas con información del clima en su ciudad [6] y pueden ser desarrollados para distintas plataformas, sin embargo en

una sociedad impensable de imaginar sin Internet, el 99.5% de los internautas se conectan a la misma a través del teléfono móvil, siendo preciso destacar que el principal tipo de conexión a la Red, por banda ancha, es la conexión móvil a través de un dispositivo [7] y considerando que según el Centro de Investigación Pew (2019), el 98% de la población española posee un dispositivo móvil —el 80% son smartphones—, alcanzando la cifra más alta la franja de edad comprendida entre los 18 y los 34 años (95%) [8], resulta atractivo en términos de visibilidad el desarrollo de widgets para dispositivos móviles ya que tienen el potencial de llegar a más personas de forma casi instantánea, lo que es crucial en escenarios críticos como lo son las donaciones de órganos donde es de gran importancia un trabajo activo de diagnóstico prematuro y de un adecuado mantenimiento para evitar la pérdida de órganos y tejidos trasplantables, debido a que el mantenimiento del potencial donante es una labor de asistencia crítica, además de sumar una alternativa de información para la sociedad sobre un tema profundo, complicado y difícil de concientizar a la población de lo importante que es expresar en vida la voluntad de donar sus órganos y tejidos, logrando de esta forma mejorar la calidad de vida de muchas personas y salvar la vida de otras tantas [9]. A partir de este contexto en el presente documento se describe la implementación de widgets para la visualización de diferentes indicadores relevantes extraídos desde datos públicos del sitio web INCUCAI, en particular se aplicará sobre plataformas móviles específicamente Android. En las siguientes secciones se presenta la organización del contenido, describiendo en primer lugar los trabajos relacionados donde se ha sido de utilidad la implementación de widgets en diferentes áreas profesionales. Posteriormente en la sección 3 se especifican los procesos involucrados para la preparación de la información que será utilizada asi como las herramientas utilizadas junto con los pasos realizados durante la implementación del widget, seguidamente en la sección 4 se discuten los resultados obtenidos. Por último, se presentan conclusiones y futuras líneas de trabajo.

2 Trabajos Relacionados

El uso de aplicaciones móviles se ha convertido en un recurso indispensable de nuestras actividades diarias. Actualmente son utilizadas con más frecuencia para facilitar diversas actividades humanas, no solamente para entretenimiento sino también como instrumento de apoyo para estudiantes con el fin de optimizar, complementar y compartir las sesiones presenciales, favoreciendo la comunicación y socialización [10]. De esta forma se logró transformar sus smartphones en una herramienta de apoyo al aprendizaje en lugar de distractores, teniendo en cuenta que la mayor parte del tiempo es destinada a las Redes Sociales y Navegadores; su mayor dominio de conocimiento en herramientas Web lo aplican a las redes sociales y al Chat [10]. Otro caso de uso refiere a la utilización de widgets para la gestión y consulta de datos almacenados desde gráficas, mapas y tablas que faciliten la comprensión del usuario, permitiendo la selección de los diferentes tipos de filtros aplicables a los datos de entrada, según sea el tipo de variable a procesar (dato numérico, de tipo discreto, texto, fechas, etc.) mediante una interfaz amigable que permita al usuario con una serie de clics realizar estos filtros o consultas complejas sin necesidad de conocer el lenguaje de consulta a

la base de datos del que dispone y sus particularidades para luego ser visualizados geográficamente sobre un mapa tanto con coordenadas (puntos en el mapa) como con polígonos [11]. Estos datos de entrada además pueden ser enviados desde sensores digitales que monitoreen condiciones específicas de productos que requieran cuidados especiales como por ejemplo el caso presentado en [12], en que él se muestra la trazabilidad en el transporte de camarones, teniendo acceso a proyectos de manera remota desde una aplicación móvil, y usando widgets para ajustar los controles y visualizar datos de temperatura, humedad, energía, trazas y localización en el mapa. El recibir notificaciones por mensaje de texto o correo electrónico basado a eventos específicos como anomalías en el entorno para que se activen los disparadores y tomar solución mediante la aplicación. Estos ejemplos muestran el potencial que tienen estas aplicaciones en cuanto a la transmisión de información en escenarios donde a pesar de la existencia del dato su acceso no es lo suficientemente intuitivo para el usuario, por lo que estas soluciones tecnológicas se presentan como alternativas más atractivas y fáciles para el público.

Siguiendo los antecedentes mencionados, nos enfocaremos en la implementación de widgets para la visualización de indicadores relevantes sobre las donaciones de órganos en la Argentina con datos tomados del INCUCAI.

3 Metodología

Origen de datos:

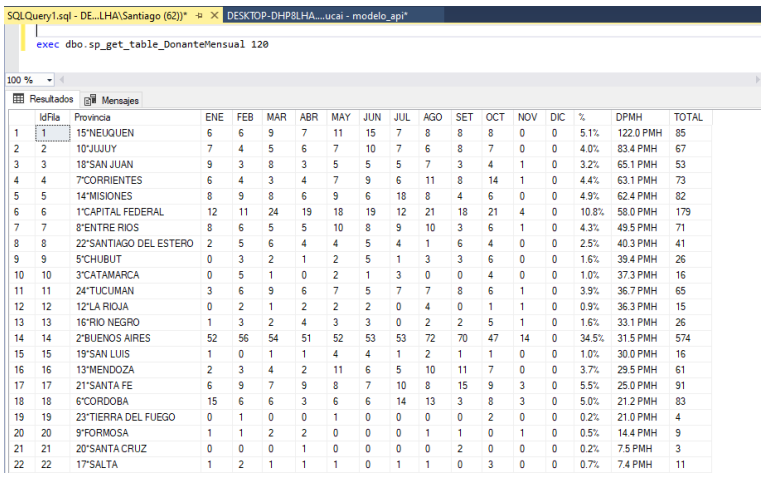
Este trabajo utiliza como fuente de información datos extraídos del sitio web del Instituto Nacional Central Único Coordinador de Ablación e Implante (INCUCAI) [3] correspondientes a los reportes del Centro Regional de Información en Salud (CRESI) [4] específicamente el “reporte de donantes de órganos y tejidos comparativo mensual” agrupados por provincia de establecimiento origen.

En la Figura 1, se muestra un detalle en el que luego de analizar la tabla visualizada, y estimando que la cantidad de columnas podría cambiar en el tiempo, afectando directamente la cantidad de filas, se optó por un modelo de almacenamiento dinámico, es decir, de cada tabla de resultado, sin importar la cantidad de filas y de columnas que posea, guardamos en una base de datos de desarrollo propio (SQL Server 2022 exress), con las columnas en una tabla a parte que se llama “DMColumna” y guardamos en la tabla “DMColumnaFilaValor” el producto cartesiano de cada valor, estas tablas nos permitirán reconstruir la tabla “virtual” a partir de los datos dinámicos.

Provincia	ENE	FEB	MAR	ABR	MAY	JUN	JUL	AGO	SET	OCT	NOV	DIC	%	DPMH	TOTAL
15*NEUQUEN	10	7	11	2	-	-	-	-	-	-	-	-	6.6%	43.1 PMH	30
10*JUJUY	10	3	4	2	-	-	-	-	-	-	-	-	4.2%	23.6 PMH	19
14*MISIONES	10	8	8	1	-	-	-	-	-	-	-	-	5.9%	20.5 PMH	27
7*CORRIENTES	7	5	6	2	-	-	-	-	-	-	-	-	4.4%	17.3 PMH	20
1*CAPITAL FEDERAL	23	13	11	5	-	-	-	-	-	-	-	-	11.5%	16.9 PMH	52
18*SAN JUAN	4	3	5	-	-	-	-	-	-	-	-	-	2.6%	14.7 PMH	12
19*SAN LUIS	3	3	1	-	-	-	-	-	-	-	-	-	1.5%	13.1 PMH	7
24*TUCUMAN	3	7	5	4	-	-	-	-	-	-	-	-	4.2%	10.7 PMH	19
5*CHUBUT	4	2	1	-	-	-	-	-	-	-	-	-	1.5%	10.6 PMH	7
8*ENTRE RIOS	4	3	6	1	-	-	-	-	-	-	-	-	3.1%	9.8 PMH	14
13*MENDOZA	6	5	7	1	-	-	-	-	-	-	-	-	4.2%	9.2 PMH	19
2*BUENOS AIRES	48	42	55	10	-	-	-	-	-	-	-	-	34.1%	8.5 PMH	155
21*SANTA FE	7	10	11	-	-	-	-	-	-	-	-	-	6.2%	7.7 PMH	28
3*CATAMARCA	2	-	1	-	-	-	-	-	-	-	-	-	0.7%	7.0 PMH	3
6*CORDOBA	6	5	10	-	-	-	-	-	-	-	-	-	4.6%	5.4 PMH	21
23*TIERRA DEL FUEGO	-	1	-	-	-	-	-	-	-	-	-	-	0.2%	5.2 PMH	1
16*RIO NEGRO	2	-	2	-	-	-	-	-	-	-	-	-	0.9%	5.1 PMH	4
22*SANTIAGO DEL ESTERO	-	3	2	-	-	-	-	-	-	-	-	-	1.1%	4.9 PMH	5
17*SALTA	4	1	1	-	-	-	-	-	-	-	-	-	1.3%	4.0 PMH	6
9*FORMOSA	-	2	-	-	-	-	-	-	-	-	-	-	0.4%	3.2 PMH	2
4*CHACO	2	1	-	-	-	-	-	-	-	-	-	-	0.7%	2.4 PMH	3
TOTAL	155	124	147	28	-	-	-	-	-	-	-	-	100.0%	9.6 PMH	454

Figura 1: Reporte INCUCAI [4].

Por otra parte, se realizó el desarrollo de un proceso almacenado propio por los autores (en SQL Server), llamado, proceso “sp_get_table_DonanteMensual” en el que si especificamos un @IdCabecera se genere una reconstrucción “virtual” de la tabla. Este proceso transforma los datos dinámicos a su estructura original, permitiendo visualizar los datos simulando ser la tabla de origen, esto se muestra en la Figura 2.



	IdFila	Provincia	ENE	FEB	MAR	ABR	MAY	JUN	JUL	AGO	SET	OCT	NOV	DIC	%	DPMH	TOTAL
1	1	15*NEUQUEN	6	6	9	7	11	15	7	8	8	8	0	0	5.1%	122.0 PMH	85
2	2	10*JUJUY	7	4	5	6	7	10	7	6	8	7	0	0	4.0%	83.4 PMH	67
3	3	18*SAN JUAN	9	3	8	3	5	5	5	7	3	4	1	0	3.2%	65.1 PMH	53
4	4	7*CORRIENTES	6	4	3	4	7	9	6	11	8	14	1	0	4.4%	63.1 PMH	73
5	5	14*MISIONES	8	9	8	6	9	6	18	8	4	6	0	0	4.9%	62.4 PMH	82
6	6	1*CAPITAL FEDERAL	12	11	24	19	18	19	12	21	18	21	4	0	10.8%	58.0 PMH	179
7	7	8*ENTRE RIOS	8	6	5	5	10	8	9	10	3	6	1	0	4.3%	49.5 PMH	71
8	8	22*SANTIAGO DEL ESTERO	2	5	6	4	4	5	4	1	6	4	0	0	2.5%	40.3 PMH	41
9	9	5*CHUBUT	0	3	2	1	2	5	1	3	3	6	0	0	1.6%	39.4 PMH	26
10	10	3*CATAMARCA	0	5	1	0	2	1	3	0	0	4	0	0	1.0%	37.3 PMH	16
11	11	24*TUCUMAN	3	6	9	6	7	5	7	7	8	6	1	0	3.9%	36.7 PMH	65
12	12	12*LA RIOJA	0	2	1	2	2	2	0	4	0	1	1	0	0.9%	36.3 PMH	15
13	13	16*RIO NEGRO	1	3	2	4	3	3	0	2	2	5	1	0	1.6%	33.1 PMH	26
14	14	2*BUENOS AIRES	52	56	54	51	52	53	53	72	70	47	14	0	34.5%	31.5 PMH	574
15	15	19*SAN LUIS	1	0	1	1	4	4	1	2	1	1	0	0	1.0%	30.0 PMH	16
16	16	13*MENDOZA	2	3	4	2	11	6	5	10	11	7	0	0	3.7%	29.5 PMH	61
17	17	21*SANTA FE	6	9	7	9	8	7	10	8	15	9	3	0	5.5%	25.0 PMH	91
18	18	6*CORDOBA	15	6	6	3	6	6	14	13	3	8	3	0	5.0%	21.2 PMH	83
19	19	23*TIERRA DEL FUEGO	0	1	0	0	1	0	0	0	0	2	0	0	0.2%	21.0 PMH	4
20	20	9*FORMOSA	1	1	2	2	0	0	0	1	1	0	1	0	0.5%	14.4 PMH	9
21	21	20*SANTA CRUZ	0	0	0	1	0	0	0	0	2	0	0	0	0.2%	7.5 PMH	3
22	22	17*SALTA	1	2	1	1	1	0	1	1	0	3	0	0	0.7%	7.4 PMH	11

Figura 2: Reconstrucción de la tabla en la base de datos.

Desarrollo de API

Esta API ha sido desarrollada con el objetivo de gestionar y proporcionar información sobre datos relacionados con la base de INCUCAI. Seguidamente e detalla la arqui-

tectura, los componentes utilizados, las herramientas seleccionadas y los endpoints que la componen. La implementación es de tipo RESTful, utilizando el protocolo HTTP para gestionar las solicitudes y respuestas entre el cliente y el servidor que aloja la base de datos, a través de varias tecnologías. En primer lugar, se utilizó ASP.NET como framework de desarrollo, específicamente la biblioteca Web API, este framework facilita la creación de aplicaciones web y APIs de manera eficiente. El lenguaje de programación elegido fue C#, que se integra de forma natural con ASP.NET y es comúnmente usado en el desarrollo de aplicaciones empresariales. Para gestionar la interacción con la base de datos, se empleó ADO.NET, un conjunto de clases de .NET que permite acceder a bases de datos y realizar consultas [13]. En este caso, se utiliza SqlConnection, SqlCommand y SqlDataReader para ejecutar un procedimiento almacenado en la base de datos SQL Server y obtener los datos relacionados con el total acumulado de donantes. El endpoint tiene la posibilidad ser escalable en el futuro, a medida que se analicen y validen otros reportes, actualmente permite obtener la cantidad de donantes registrada hasta el momento en INCUCAI.

Endpoint: API/DonanteMensual/TotalAcumulado [GET]

La Figura 3 muestra la implementación del endpoint orientada en la API DonanteMensual con TotalAcumulado, el cual expone un método HTTP GET que permite recuperar el total acumulado de donaciones mensuales. Este método establece una conexión con la base de datos utilizando una cadena de conexión configurada, ejecuta un procedimiento almacenado (sp_get_api_DonanteMensual_Total) y, en caso de que existan resultados, mapea los datos obtenidos a un objeto de tipo DMTotal. Este objeto incluye información como el identificador de cabecera, la fecha de alta y el total acumulado. El código muestra el manejo de excepciones y asegura el cierre adecuado de la conexión, promoviendo buenas prácticas en el acceso a datos.

```
   [Route("TotalAcumulado"), HttpGet]
   public DMTotal TotalAcumulado()
   {
      DMTotal total = null;
      SqlConnection conexion = null;
      try
      {
         conexion = new SqlConnection();
         conexion.ConnectionString =
ConfigurationManager.ConnectionStrings["BD"].ConnectionString;
         conexion.Open();
         SqlCommand comando = conexion.CreateCommand();
         comando.CommandType = CommandType.StoredProcedure;
         comando.CommandText = "sp_get_api_DonanteMensual_Total";
         SqlDataReader reader = comando.ExecuteReader();
         if (reader.HasRows)
         {
              reader.Read();
              total = new DMTotal();
              total.IdCabecera =
reader.GetSqlInt32(reader.GetOrdinal("IdCabecera")).Value;
              total.FechaAlta =
reader.GetSqlDateTime(reader.GetOrdinal("FechaAlta")).Value;
              total.Total =
reader.GetSqlInt32(reader.GetOrdinal("Total")).Value;
          }
          conexion.Close();
       }
       catch (Exception ex) {if (conexion != null) conexion.Close();}
       finally {if (conexion != null) conexion.Dispose();}
       return total;
    }
```

Figura 3: Endpoint de la API Reportes total acumulado en el año.

Se han definido los modelos de datos que representan las entidades dentro de la base de datos, el siguiente es un ejemplo del modelo que representa la respuesta del servicio, esto se muestra en la Figura 4.

```
namespace APIIncucai.Models
{
    public class DMTotal
    {
        public int IdCabecera { get; set; }

        public DateTime FechaAlta { get; set; }

        public int Total { get; set; }
    }
}
```

Figura 4: Modelo de respuesta del servicio de la API

Desarrollo de Widget

Para la implementación del widget se utilizó el entorno de desarrollo (IDE) Android Studio versión 2024.2.2 con el lenguaje de programación Kotlin (también es posible hacerlo con JAVA), este IDE nos permitirá escribir el código mientras se compila en un dispositivo Android emulado, donde se visualizará el widget. Al crear un nuevo proyecto el IDE nos ofrece diferentes plantillas para seleccionar según la necesidad, en nuestro caso utilizamos “Empty Activity”. Esta plantilla incluye el archivo MainActivity.java o MainActivity.kt (dependiendo del lenguaje utilizado) que actúa como el punto de entrada lógico de la app, y define el comportamiento que debe tener la pantalla principal, luego se encuentra un layout XML asociado (activity_main.xml) para los elementos de la interfaz de usuario, y finalmente el archivo central del proyecto llamado AndroidManifest.xml encargado de declarar todas las actividades de la app y le indica al sistema Android como se debe ejecutar la aplicación para funcionar correctamente. Estos archivos representan lo mínimo y necesario para crear una aplicación, y solo nos proporcionara la estructura de archivos, carpetas, herramientas, bibliotecas y APIs esenciales para comenzar a desarrollar, evitando la creación automática de módulos que no necesitemos, de esta forma mantenemos limpio el código y reducimos el consumo de recursos. No es estrictamente necesario el uso de una actividad para el funcionamiento del widget, pero esta opción nos permitirá el agregado de más funcionalidades si se requieren utilizando la misma estructura, como por ejemplo la interacción del usuario con el objeto en pantalla.

Para la ejecución y visualización del widget durante el desarrollo, se generó un emulador de Android desde el IDE. A través del asistente integrado podemos configurar las especificaciones que tendrá el dispositivo que se quiere emular, para estas pruebas se especificó phone como tipo de dispositivo por comodidad al momento de hacer las compilaciones, también es posible emular otro tipo de dispositivos, como por ejemplo tablets o dispositivos plegables. Sin embargo, esto implica mayor consumo de recursos ya que suelen ser de gama alta, lo que implica mayores prestaciones y por ende mayores requisitos de hardware necesarios para emularlos. En nuestro caso se buscaba un dispositivo móvil con bordes reducidos y disposición de pantalla amplia, para facilitar la visualización, ubicación y redimensionamiento del objeto. El modelo que más se acercó a nuestras necesidades es el pixel 9 con un panel de 6,24

pulgadas y una relación de aspecto 20:9, resultando en una pantalla más alta y estrecha lo que mejora la experiencia de uso con una sola mano.

En cuanto a la versión del sistema operativo, se eligió Android principalmente por su facilidad de publicación y uso de las aplicaciones, que le hizo ganar popularidad, tanto entre los desarrolladores como entre los usuarios. Esto sumado a que es el sistema que tiene la mayor participación en el mercado lo convierten en una opción que garantiza un mayor alcance entre los usuarios potenciales [14]. Se realizaron pruebas con versiones recientes como Android 15, 14 y 13. Sin embargo durante el desarrollo se buscaba agilidad y fluidez entre cada una de las compilaciones mientras se modificaba y testeaba el código fuente, por esta razón se utilizó Android 7.0 nougat, principalmente por tener un tamaño ligero y su menor uso de recursos frente a versiones más recientes. Esta elección se hace con el fin de evitar problemas presentados por versiones más recientes como cierres de aplicación o demoras a la hora de su ejecución, teniendo en cuenta que la emulación es a nivel local y no en la nube, por lo que existe la limitación dependiendo de los recursos de hardware disponibles del equipo donde se ejecute el IDE. Además, los widgets en Android están disponibles desde la versión 1.5 Cupcake [15], por lo tanto, para nuestro objetivo es indiferente ya que no requerimos el uso de funcionalidades existentes en versiones más recientes de Android que podrían entorpecer y demorar el desarrollo si no se cuenta con el hardware necesario para emularlo.

Una vez definidos los parámetros y configuraciones iniciales, se trabajó sobre la estructura de la plantilla seleccionada ofrecida por el IDE. En la Figura 5 se observa la disposición de carpetas generadas, sobre las cuales se basa el widget desarrollado. La carpeta manifest contiene el archivo manifiest.xml y especifica la estructura fundamental de la aplicación y su interacción con el sistema operativo, acá se especifican permisos importantes como el uso de la cámara y localización, en nuestro caso no están habilitados ya que no son necesarios. En este archivo también figuran los componentes que formaran parte de la aplicación para que esta los reconozca en la ejecución, inicialmente solo contiene la actividad principal que fue generada por la plantilla, por lo que adicionalmente fue necesario registrar en este documento el widget definiendo su comportamiento básico junto con su configuración XML, para que pueda ser agregado a la pantalla de inicio y se actualice automáticamente desde el sistema. De esta forma aparecerá en la lista de widgets disponibles en el dispositivo para que el usuario pueda arrastrarlo y ubicarlo en su pantalla de inicio.

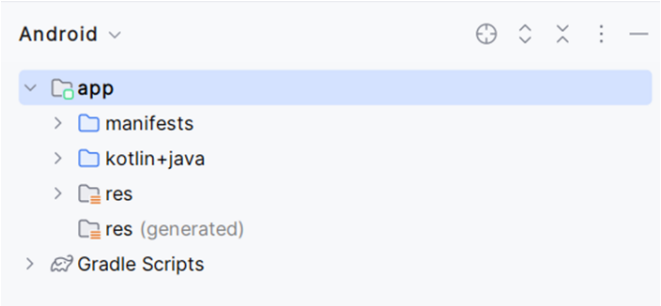


Figura 5: Estructura general, plantilla Empty Activity.

La siguiente carpeta “Kotlin+java” como se observa en la Figura 5, refiere a el código fuente que puede ser escrito en Kotlin o Java, en nuestro caso utilizamos Kotlin ya que es el lenguaje oficial recomendado por Google [16], siendo este basado en Java, pero con ventajas que lo hacen más amigable, conciso y flexible con una curva de aprendizaje menor, además tiene apoyo completo de Google y posee funciones de alto orden y técnicas que Java no contiene [17]. En la Figura 6, se muestra la organización de las carpetas necesarias para la implementación, por defecto la plantilla contiene el archivo “Main Activity” la cual representa la aplicación base que será instalada en el dispositivo y sobre la cual se basará el widget definido en el archivo “DonantesWidget”, en nuestro caso ambas vistas muestran el mismo indicador, sin embargo son independientes y podrían utilizarse para enfoques diferentes pero relacionados, por ejemplo: seria útil visualizar el indicador en el widget mientras que la app muestre información más detallada y extendida, así como establecer configuraciones de preferencia que afecten al propio widget.

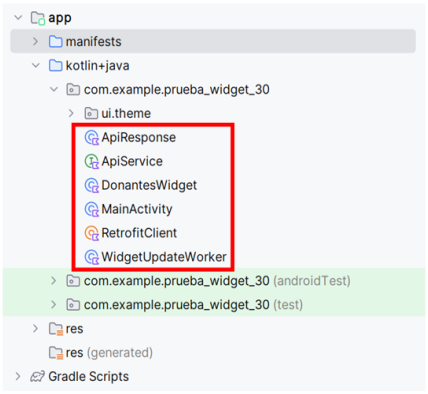


Figura 6: Distribución de archivos para el código fuente

La diferencia principal radica en que la app necesita estar abierta para su uso, se ejecuta en primer plano y consume recursos mientras el usuario o sistema no la finalice. Por otro lado, el widget siempre está visible en la pantalla de inicio o bloqueo, opera en segundo plano y se actualiza periódicamente sin importar que la app este abierta. Además, la interacción con la app es más completa que con el widget. Para la organización general del código se optó por una estructura separada por archivos independientes según su finalidad, buscando independencia funcional entre ellos y siguiendo una cadena de invocación que comienza por el llamado de la API desde el archivo “RetrofitClient” utilizando la biblioteca HTTP para Android Retrofilt definida como un objeto de única instancia para hacer peticiones HTTP a una API REST. Desde esta clase se llama al archivo “ApiService” desde donde se hace el GET a la API, además se incluye la estructura de los campos mapeados en el archivo “ApiResponse” que conforman el JSON resultante. Hasta este punto tendríamos definido la conexión con la API, lo siguiente consiste en mapear los valores del JSON en cada una de las variables que se muestran en el widget, para ello en “WidgetUpdateWorker” definimos una clase para actualizar el widget haciendo

uso de Corrutinas y a través del componente Workmanager diseñado para ejecutar tareas en segundo plano. Esta clase se encarga de obtener datos recientes, formatearlos y asignarlos a cada elemento del objeto visual, que seran mostrarlos en el widget final. Finalmente, desde “DonantesWidget” y “MainActivity” se programa la ejecución a demanda de forma periódica, en nuestro caso utilizamos el tiempo mínimo permitido para los widgets de 30 minutos. Sin embargo, haciendo uso de más recursos como threads se podría reducir la frecuencia de la consulta. Esto se muestra en la Figura 7. Concluida la parte funcional, lo siguiente consiste en diseñar los objetos visuales con los que va a interactuar el usuario y que serán ubicados en la pantalla principal del dispositivo, las configuraciones de diseño son realizadas en la carpeta RES dentro de la organización de la plantilla, en esta ubicación generaremos archivos con las especificaciones visuales en lenguaje XML.

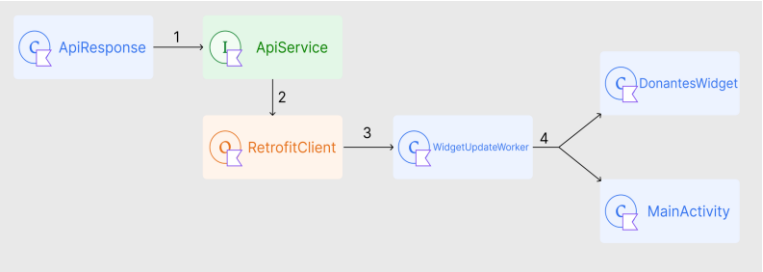


Figura 7: Relación entre archivos para la implementación.

Como primera pieza fundamental necesitamos un elemento padre que pueda ser ubicado y redimensionado en la pantalla principal y que sea capaz de abarcar subelementos de manera ordenada. Para ello generamos un tipo de contenedor unidireccional llamado linearLayauot que organizara los elementos hijos de tipo TextViews en una distribución vertical, estos componentes son de interfaz gráfica (UI) y se utilizan para mostrar información al usuario en forma de títulos, descripciones o mensajes en formato de texto y en escenarios en donde no sea necesario la edición por parte del usuario en cuyo caso usaríamos un elemento de tipo EditText en su lugar. Esta información mostrada en el widget puede ser de origen estático o dinámico según la necesidad, y se define en el código kotlin cuando se llama al objeto TextViews determinado.Cada uno de estos objetos UI se diseñan por bloque de forma independiente, especificando el id del elemento asi como su ubicación dentro del contenedor principal, parámetros de dimensión como: el tamaño de fuente, altura máx./min, ancho máx./min y parámetros de apariencia como: color, tipo de fuente y trasparencia. Esto se muestra en la Figura 8.

Para el widget realizado en este trabajo se definieron cuatro componentes de tipo TextViews, los cuales se pueden apreciar en la Figura 9. El primero se ubica en la parte superior del LinearLayaout y es utilizado como título para mostrar el texto “Cantidad Total de donantes”, A continuación, se dispone el segundo TextView que contiene un numero dinámico en color azul y muestra el indicador obtenido desde la API que representa la cantidad de donantes.

```
<TextView
    android:id="@+id/tv_title"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Cantidad de donantes"
    android:textColor="#FFFFFF"
    android:textSize="16sp"
    android:textStyle="bold"
    android:gravity="center" />
```

Figura 8: Ejemplo de configuración de diseño para un TextView.

El tercer TextView presenta la leyenda “Argentina | STATUS ACTUAL | Año 2025” con el objetivo de contextualizar la información mostrada. Por último, el cuarto elemento muestra la última fecha de actualización del widget en formato DD/MM/YYYY, visualizada en color verde. Este último elemento está contenido dentro de un LinearLayout secundario del mismo tipo que el contenedor principal y su valor se genera de forma dinámica, calculado automáticamente en la actividad principal de la aplicación cada vez que se actualiza el widget.

Figura 9: Diseño final del widget.

Una vez finalizado el desarrollo y luego de verificar que todo funciona correctamente en las pruebas realizadas en el emulador del IDE, se procede a limpiar y reconstruir la aplicación con las herramientas “Clean Project” y “Rebuild Project”, con esto garantizamos la eliminación de elementos temporales que puedan dar errores durante la compilación y así evitar posibles problemas de lanzamiento por cache corrupta, además se hace un compilación de la solución de cero para sincronizar los elementos del archivo Gradle desde donde se gestionan las dependencias, tareas de compilación y configuraciones del proyecto. A partir de este punto la aplicación esta lista para ser exportada e instalada en dispositivos fisicos, para ello AndroidStudio nos presenta dos alternativas. La primera opción genera un archivo instalable APK que podemos compartir con otros dispositivos, esta opción no es ideal para puestas en producción ya que no cumple los requisitos para la publicación en la GoogleStore requiriendo de difusión del archivo APK e instalación ma-

nual por parte del usuario. La otra alternativa ofrecida por el IDE nos permite generar un archivo con formato. abb a la que se debe agregar un certificado de firma requerido para la publicación en la PlayStore oficial de Android. Independientemente de la alternativa utilizada no afecta el uso de la aplicación una vez instalada por lo que para nuestro objetivo optamos por la primera opción debido a la facilidad y rapidez para ser utilizada en otros dispositivos. Sin embargo, al ser un archivo APK y no tener firmas de autenticación es posible que el sistema operativo donde se instale lo interprete como archivo riesgoso o comprometido, y por razones de seguridad propias del sistema hace un análisis del archivo en la Figura 10, para garantizar que sea seguro, esto no ocurriría si la aplicación estuviera alojada en la PlayStore ya que antes de su publicación pasa por estos análisis de seguridad por parte de Google.

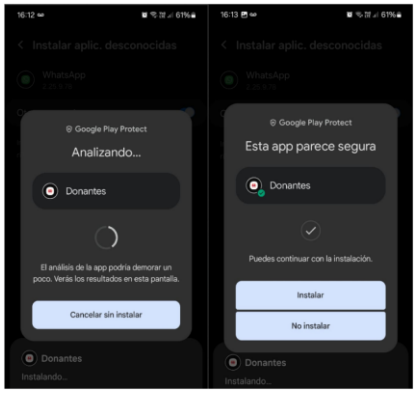


Figura 10: Proceso de instalación en dispositivo físico.

Finalmente, una vez instalada la aplicación podemos utilizar el indicado desarrollado en la propia app ubicada en el cajón de aplicaciones o a través del widget ubicado en la lista de widgets. En la figura 11 podemos observar la aplicación y el widget en funcionamiento sobre un dispositivo físico.

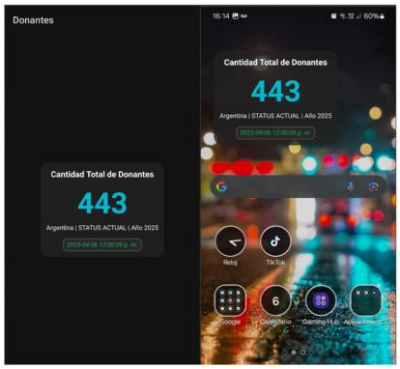


Figura 11: De izquierda a derecha, ejecución de APP y ejecución de Widget.

4 Análisis y discusión de resultados

Los datos integrados a la aplicación desarrollada demostraron ser una forma práctica y eficaz de acceder información desde dispositivos electrónicos, especialmente desde teléfonos móviles, en donde el impacto es mayor debido a su presencia constante en la vida cotidiana, convirtiéndolas en herramientas fundamentales para la actualidad. Los resultados obtenidos evidencian una significativa mejora en la gestión de los procesos de seguimiento para trasplante a partir del desarrollo e implementación del Sistema de Información Especializado (SIE). Este hallazgo se alinea con lo expuesto un trabajo de investigación [18], en el cual se destaca la importancia de los sistemas de información como herramientas clave para la transparencia y la eficiencia en los procesos de procuración y trasplante. Mientras que investigaciones anteriores se han aplicado en áreas como educación y transporte, la propuesta de este trabajo introduce una nueva posibilidad al centrarse en la transparencia y calidad del dato entorno a la donación de órganos, aportando información para la concientización social en el ámbito de la salud pública. En el trabajo citado se menciona que la aceptabilidad del usuario final es uno de los principales factores de éxito en la implementación de tecnologías sanitarias, lo cual también se verifica en esta experiencia. Es por lo que, como líneas futuras, se enfocará en la experiencia del usuario con más funcionalidades que acompañen en este objetivo propuesto. Un aspecto que no es menor es que esta solución podría escalar y tomar datos más complejos que permitan la toma de decisiones en el menor tiempo posible. A nivel técnico se comprobó la viabilidad de integrar información pública extraída desde un sitio web, posteriormente alojada en una base de datos y finalmente consumida desde cualquier dispositivo a través de un widget conectado a una API, pero deberemos continuar con las pruebas. Los resultados obtenidos respaldan la hipótesis de que el SIE representa una herramienta eficaz para optimizar la gestión de pacientes en lista de espera, y refuerzan la necesidad de avanzar en la consolidación de sistemas interoperables que articulen la información sanitaria a nivel nacional, tal como lo propone el marco de referencia y lo que promueve el INCUCAI.

5 Conclusiones

Durante este trabajo se llevo a cabo el desarrollo de una aplicación que habilita un objeto visual (widget) para la visualización de indicadores relevantes en el ámbito de las donaciones de órganos. Esta información tomada desde el sitio oficial INCUAI se extrae, almacena y se brinda a través de una API que luego es consumida por la aplicación instalada desde cualquier dispositivo móvil para finalmente mostrarse en un formato amigable para el usuario a través de su pantalla de inicio. Con este enfoque podemos replicar la información del sitio web y enviarla directamente hasta la palma de la mano del usuario, evitando ingresar y navegar por la pagina web para ubicar el dato como lo haría habitualmente. Por el contrario, se le facilita a la persona la consulta de información de una manera sencilla, similar a como haría para ver la hora en su teléfono. Este tipo de soluciones son útiles para dar a conocer información relevan-

te a la población, ya que no implica sumarle actividades a la rutina diaria de las personas para su consulta, lo que favorece una mayor difusión del contenido.

Entre las posibilidades para futuras etapas podemos mencionar por un lado respecto a la aplicación, la incorporación de más indicadores, así como la habilitación para interactuar con el widget. Por otro lado, respecto a la portabilidad, de acuerdo con la documentación oficial de Google, Android Studio es un IDE multiplataforma lo que significa que una misma solución es de fácil exportación a otros tipos de sistemas operativos, esto implica la posibilidad a futuro de utilizar el desarrollo de este trabajo en dispositivos como APPLE y otros, sin necesidad de reescribir el código fuente.

Referencias

[1] Antonia Ferrer, Fernanda Peset, Rafael Aleixandre. (2011). Accceso a los datos públicos y su reutilización: open data y open government, 20(3), 260-261.
[2] A. Baquero1, J. Alberú. (2011). Desafíos éticos en la práctica de trasplantes en America Latina: Documento de Aguascalientes, 31(3), 276-277.
[3] INCUCAI. (s.f.). Instituto Nacional Central Único Coordinador de Ablación e Implante. Ministerio de Salud. https://www.argentina.gob.ar/salud/incucai
[4] INCUCAI. (s.f.). Sistema Nacional de Información de Procuración y Trasplante de la República Argentina (SINTRA). Instituto Nacional Central Único Coordinador de Ablación e Implante. https://sintra.incucai.gov.ar/
[5] Graciela D. Saltos C., Tiziana María H. (2010). Creación de un contenedor de WIDGETS. http://www.dspace.espol.edu.ec/handle/123456789/19411
[6] Javier García Puga. (2019). Desarrollo de aplicaciones para sistemas móviles con tecnología Symbian y Mobile Widgets.
[7] Julio Ruiz Palmero, Ernesto Colomo Magaña, Enrique Sánchez Rivas¸ Teresa Linde Valenzuela. (2021). Estudio del uso y consumo de dispositivos móviles en univer-sitarios. Digital Education Review, (39), 102.
[8] González Cortés, Córdoba Cabús. (2020). Una semana sin smartphone: usos, abuso y dependencia del teléfono móvil en jóvenes.
[9] García, M. A., Gonzalo, P., & Monter, V. (2023). Conocimientos sobre la gestión del proceso de donación de órganos y tejidos. SALUD, COMUNIDAD Y CIENCIA, 1(2), 1.
[10] Mata, M. E. G., Rocha, J. L. I. T., & Rodríguez, M. E. I. (2018, November). El uso de gadgets y widgets como apoyo para disminuir el rezago y abandono escolar. In Congresos CLABES.
[11] Magaldi Calleja, F. (2015). Desarrollo de widgets integrables en aplicaciones de análisis y visualización de datos.
[12] Segovia Sabando, M. D. L. Á., & Silva Astudillo, D. A. (2022). Diseño e implementación de una plataforma de interpretación de datos de trazabilidad en el transporte de los camarones. ESPOL. FIEC.
[13] Kanjilal, J. (2013). ASP.NET Web API: Build RESTful web applications and services on the .NET framework. Packt Publishing.

[14] Campos Gavilánez, D. M. (2023). Análisis comparativo entre sistemas operativos de dispositivos móviles ANDROID, IPHONE OS y MIUI (Bachelor's thesis).
[15] Sánchez Villafuerte, M. G. (2022). Estudio comparativo entre los sistemas operativos móviles “Android Os y Chrome Os” (Bachelor's thesis, Babahoyo: UTB-FAFI. 2022).
[16] Android Developers. (s.f.). Android Developers. Google.
[17] Martínez Vaca, D. A. (2021). Estudio comparativo de las mejoras del lenguaje de programación kotlin y el lenguaje java en el desarrollo de aplicaciones Android (Bachelor's thesis, BABAHOYO: UTB, 2021).
[18] Bacqué, M. D. C., Vallejos, A., & Bisigniano, L. (2018). Situación del trasplante renal y la donación de órganos en Argentina. *Revista de nefrologia, dialisis y trasplante*, *38*(1), 1-14.