ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
(GRADO EN INGENIERÍA DE COMPUTADORES)

# APLICACIÓN ESTADÍSTICA MÓVIL PARA DATOS ABIERTOS

# MOBILE STATISTICAL APPLICATION FOR OPEN DATA

Realizado por
**Juan de Dios Paredes Arjona**
Tutorizado por
**Daniel Garrido Márquez**
Departamento
**Lenguajes y Ciencias de la Computación**

UNIVERSIDAD DE MÁLAGA
MÁLAGA, Diciembre, 2016

Fecha defensa:
El Secretario del Tribunal

**Resumen:**

Este TFG nace a partir del movimiento de Datos Abiertos que podemos encontrar en gran número de entidades a nivel mundial y sobre todo, en las administraciones públicas. Este movimiento se basa en la difusión de datos, según las leyes de cada país, que hacen estas entidades para que puedan ser tratados de manera automática por cualquier programa de forma libre y sin restricciones. Tenemos por objetivo realizar una aplicación móvil que permita realizar análisis estadísticos de estos datos, centrándose en el formato CSV, que es el formato más utilizado en estos momentos.

Hemos implementado dicha aplicación, con éxito, en entorno Android, siguiendo las pautas de un proceso software de Desarrollo Iterativo e Incremental (IID), pasando así, por etapas retroalimentadas de capturas de requisitos, análisis, diseño, implementación y pruebas, tal y como se detalla en la presente memoria.

Se obtiene un producto final completo que puede abordar cualquieras datos en formato CSV. Además podemos crear y analizar nuestro propios datos creados en este formato. Tiene limitaciones de memoria debido a que estos ficheros pueden consumir gran cantidades de memoria, pero esperamos que esta restricción sea superada por la evolución de capacidad física que tengan los móviles en un futuro próximo.

**Palabras Clave:**

Datos Abiertos, Android, Estadística, Ficheros CSV, Móvil, App, Aplicación, Teléfono, Celular y Smarthphone.

**Abstract:**

This TFG born from the Open Data movement that can be found in many institutions worldwide and especially in government. This movement is based on the dissemination of data, according to the laws of each country, which make these entities so that they can be handled automatically by any program freely and without restrictions.

We aim to make a mobile application that allows statistical analysis of these data, focusing on the CSV format, which is the most widely used format right now.

We have implemented the application, successfully, in Android environment, following the guidelines of software Iterative Development and Incremental (IID) process and happening, fed back stages of catches of requirements, analysis, design, implementation and testing, as detailed herein.

We complete end product that can address data in CSV format no matter who is obtained. We can also create and analyze our own data created in this format. Memory has limitations because these files can consume large amounts of memory, but we hope that this restriction is overcome by the evolution of physical capacity that are mobile in the future.

**Keywords:**

Open Data, Android, Statistics, CSV Files, Mobile, Cell, App, Application, Phone and Smarthphone.

**ÍNDICE**:

# Introducción

En esta introducción hablaremos de la motivación y objetivos del proyecto, de la estructura de esta memoria y haremos un breve estudio de la tecnología usada

El concepto datos abiertos (open data, en inglés) es una filosofía y práctica que persigue que determinados tipos de datos estén disponibles de forma libre para todo el mundo, sin restricciones de derechos de autor, de patentes o de otros mecanismos de control. Tiene una ética similar a otros movimientos y comunidades abiertos, como el software libre, el código abierto (open source, en inglés) y el acceso libre (open access, en inglés). La utilización de estándares puede facilitar a cualquier persona acceder a estos datos y también el tratamiento automático de esta información.

Son numerosas las administraciones que actualmente ponen a disposición de los ciudadanos datos públicos: ayuntamientos, comunidades autónomas y la propia administración central. Por ejemplo, tenemos el portal web para datos abiertos del ayuntamiento de Málaga (http://datosabiertos.malaga.eu/), Junta de Andalucía ( http://www.juntadeandalucia.es/datosabiertos/portal.html) o Gobierno de España ( http://datos.gob.es/). Todo ello, con el objetivo de conseguir una mayor transparencia de la actividad gubernativa. También existen reflejos de este movimiento en Europa, concretamente en la Unión Europea (http://europa.eu/publications/open-data/index_es.htm) y en el mundo, como por ejemplo en el banco mundial (http://datos.bancomundial.org/).

En este contexto, es de utilidad la realización de aplicaciones que manejen estos datos de manera automática y pueden suponer una ventaja para el ciudadano. En este sentido, en este trabajo se realizará una aplicación móvil que permite recuperar y visualizar estos datos para su análisis estadístico (siempre que estos datos sean susceptibles a realizar este tipo de análisis).

El objetivo de este TFG es realizar un programa para Smartphone que se encargue de realizar estadísticas descriptivas de los Datos Abiertos ofrecidos a los ciudadano aprovechando la cobertura que de ellos hacen las diferentes asociaciones e instituciones entre las que se encuentran las Administraciones Públicas encargadas de publicar estos datos en una variedad de formatos para su diferente tratamiento de forma automática. Nuestra aplicación se centrará en el formato ‘csv’, que es quizás el más utilizado para difundir estos datos.

La presente memoria se estructura en cuatro capítulos, el primero de ellos nos explican los concepto estadísticos utilizados en la App, el segundo nos explica los conceptos más generales de Android y está pensado para personas que no conozcan este plataforma de programación, y el tercero capítulo nos explican qué son los archivos CSV y por último capítulo cuatro nos explica las fases por las cuáles ha pasado este TFG. Además se explican las conclusiones a las que hemos

llegado en éste TFG y las posibles mejoras del trabajo. Complementariamente se aporta como anexos un completísimo manual de usuario y una descripción de los formatos de fecha aceptados por la App

La tecnología a usar es la plataforma de programación Android aprovechando su funcionalidad y la enorme expansión que tiene su SO Android. Ésta funcionalidad permite la creación de aplicaciones como la nuestra permitiéndonos explotar los beneficios que supone hacer esta aplicación para un dispositivo móvil tipo smarthphone.

# Capítulo 1: Medidas de estadística descriptivas

En esta sección vamos explicar las medidas de estadística descriptiva utilizada por nuestra aplicación Open Data y haremos hincapié en aquellas más desconocidas. Supondremos que el usuario conoce las medidas de la Media, la Mediana y la Moda.

Las medidas descriptivas son valores numéricos calculados a partir de la muestra y que nos resumen la información contenida en ella. Podemos dividir, las medidas usadas en nuestra aplicación en tres grandes grupos:

- **De Centralización**: Indican valores con respecto a los que los datos de la muestra parecen agruparse. En nuestra aplicación utilizaremos la Media, Mediana y la Moda
- **De Dispersión**: Indican la mayor o menor concentración de los datos con respecto a las medidas de centralización. En nuestra aplicación utilizamos la Varianza, Desviación Típica, Coeficiente de Variación, Error Estándar de la Media y Rango.
- **De Relación Lineal**: Indican si existen alguna relación lineal entre los valores de dos muestras. En nuestra aplicación usamos la Covarianza y el Coeficiente de Correlación de Pearson.

Iremos directamente a las Medidas de Dispersión pues son estás más desconocidas por el lector y a continuación definiremos la Medidas de Relación Lineal.

## Medidas de Dispersión:

Las medidas de tendencia central tienen como objetivo el sintetizar los datos en un valor representativo; las medidas de dispersión nos dicen hasta qué punto estas medidas de tendencia central son representativas como síntesis de ésta información. Las medidas de dispersión cuantifican la separación, la dispersión, la variabilidad de los valores de la distribución respecto al valor central. Distinguimos entre medidas de dispersión absolutas, que no son comparables entre diferentes muestras y las relativas que nos permitirán comparar varias muestras.

### Medidas de Dispersión Absolutas:

#### *Varianza:*

Es el promedio del cuadrado de las distancias entre cada observación y la media aritmética del conjunto de observaciones. Se representa por S^2

$$s^2 = \frac{\sum_i (x_i - \bar{x})^2 n_i}{n}$$

Haciendo operaciones en la fórmula anterior obtenemos otra fórmula para calcular la varianza:

$$s^2 = \frac{\sum_i x_i^2 n_i}{n} - \bar{x}^2$$

### *Desviación Típica:*

La varianza viene dada por las mismas unidades que la variable pero al cuadrado, para evitar este problema podemos usar como medida de dispersión la desviación típica que se define como la raíz cuadrada positiva de la varianza

$$s = \sqrt{s^2}$$

Para estimar la desviación típica de una población a partir de los datos de una muestra se utiliza la fórmula (cuasi desviación típica):

$$s = \sqrt{\frac{\sum_i (x_i - \bar{x})^2 n_i}{n-1}}$$

### *Rango:*

Es la diferencia entre el valor de las observaciones mayor y el menor. Re = xmax – xmin

## Medidas de Dispersión Relativas:

### *Error Estándar de la Media:*

Esta medida nos sirve para indicar cuanto de exacta es la media de una muestra con respecto a la media real de la población en el supuesto que hagamos varias muestras sobre la misma población y queramos averiguar el grado de exactitud de la media hallada en cada muestra. Los valores más bajos, nos hacen sospechar que la media calculada en esta muestra se aproxima más a la media real de toda la población y los valores más altos nos hace sospechar lo contrario.

$$\sqrt{\frac{\sigma^2}{n}} = \frac{\sigma}{\sqrt{n}}$$

Cuando mayor es el tamaño de la muestra tomada sobre la población y más agrupadas estén estos valores, no dará un error estándar pequeño como es lógico; por el contrario cuanto menor sea el tamaño de la muestra y mayor sea las diferencias entre valores, mayor será este error estándar de la media.

### *Coeficiente de Variación de Pearson:*

Cuando se quiere comparar el grado de dispersión de dos distribuciones que no vienen dadas en las mismas unidades o que las medias no son iguales se utiliza el coeficiente de variación de Pearson que se define como el cociente entre la desviación típica y el valor absoluto de la media aritmética.

$$CV = \frac{S}{|\bar{x}|}$$

CV representa el número de veces que la desviación típica contiene a la media aritmética y por lo tanto cuanto mayor es CV mayor es la dispersión y menor la representatividad de la media.

## Medidas de Relación Lineal

Estas medidas, Covarianza y Coeficiente de Correlación de Pearson, tratan de identificar si existe una relación lineal entre dos muestras, es decir, en qué medida existe una relación cuando crecen los valores de ambas muestras o cuando decrecen. Si para valores grandes de una muestra la corresponden valores grandes de la otra muestra diremos que estamos ante una correlación directa y si ocurre que para valores más grandes de la una de las muestras le corresponde valores más pequeños de la otra muestra entonces estamos antes una correlación inversa. A continuación definimos con más detallas ambas medidas:

### *Covarianza*

La covarianza de una variable bidimensional es la media aritmética de los productos de las desviaciones de cada una de las variables respecto a sus medias respectivas.

Se representa por sxy o σxy.

$$\sigma_{xy} = \frac{\sum f_i(x_i - \bar{x})(y_i - \bar{y})}{N}$$

$$\sigma_{xy} = \frac{\sum f_i x_i y_i}{N} - \bar{x}\bar{y}$$

La covarianza indica el sentido de la correlación entre las variables

Si $\boldsymbol{\sigma_{xy} > 0}$ la correlación es directa.

Si $\boldsymbol{\sigma_{xy} < 0}$ la correlación es inversa.

Se interpreta de la siguiente manera:

Cuando a grandes valores de una de las variables suelen mayoritariamente corresponderles los grandes de la otra y ocurre que para sus valores pequeños ocurre que le corresponde mayoritariamente valores pequeños de la otra, se dice que tienden a mostrar similar comportamiento lo que se refleja en un valor positivo de la covarianza y decimos que estamos ante una correlación directa. Por el contrario, cuando a los mayores valores de una variable suelen corresponder en general los menores de la otra y viceversa, expresando un comportamiento opuesto, la covarianza es negativa y decimos que estamos ante una correlación inversa.

La covarianza presenta como inconveniente, el hecho de que su valor depende de la escala elegida para los ejes. Es decir, la covarianza variará si expresamos la altura en metros o en centímetros. También variará si el dinero lo expresamos en euros o en dólares.

### *Coeficiente de Correlación de Pearson*

Mide lo mismo que la Covarianza pero en este caso esta magnitud no depende de la escala de los valores de la muestra. El valor del Coeficiente de Correlación de Pearson oscila entre el valor 1 y -1.

Se define por su fórmula:

$$\rho = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

Es decir, se define ‘p’ como el coeficiente de correlación de Pearson como la Covarianza de dos variables, entre el producto de sus desviaciones típicas. Y se interpreta de la siguiente manera:

- Si es 1, existe una correlación positiva perfecta. El índice indica una dependencia total entre las dos variables denominada relación directa: cuando una de ellas aumenta, la otra también lo hace en proporción constante.
- Si 0 <coeficiente < 1, existe una correlación positiva.
- Si coeficiente = 0, no existe relación lineal. Pero esto no necesariamente implica que las variables son independientes: pueden existir todavía relaciones no lineales entre las dos variables.
- Si -1 < coeficiente < 0, existe una correlación negativa.
- Si coeficiente = -1, existe una correlación negativa perfecta. El índice indica una dependencia total entre las dos variables llamada relación inversa: cuando una de ellas aumenta, la otra disminuye en proporción constante.

# Capítulo 2: Introducción a Android

El presente capítulo trata de hacer una muy breve introducción a Android, suponiendo que el lector tiene conocimientos de programación en Java y de Sistemas Operativos. Definiremos Android desde el punto de Sistema Operativo y desde el punto de una plataforma de programación. Citaremos algunas las consideraciones de la aplicación Open Data. Definiremos qué es una Vista, una Actividad, una Intención y qué es la Máquina Virtual Dalvik y el Ciclo de Vida de una Actividad. Aunque se podría hablar mucho más de Android, creemos que lo que se explica aquí es suficiente para entender las referencias de la memoria que se hace a la terminología asociada a Android.

## Android

Desde el punto de los sistema operativos Android es como se conoce a un tipo Sistema Operativo diseñado para dispositivos móviles aunque puede encontrarse en sistema empotrados como puede ser en lavadoras, frigoríficos, etc. Es un sistema abierto basado en Linux y es Multiplataforma pues posee su máquina virtual optimizada (equivalente a la de Java).

Desde el punto de las plataformas de programación: Android es una plataforma de programación pensada para el diseño de aplicaciones que corran bajo su propio S.O. conocido, también, como Android. Se suele definir con el número de API o el nombre de un postre en inglés.

Como vemos el nombre de Android puede referenciarse al S.O. o a la plataforma de desarrollo, ambas palabras parece que se confunden. Pero en realidad no es así, pues siempre que se lanza una nueva versión de plataforma de desarrollo es porque se lanza a su vez una nueva versión del S.O. Con lo que para una API concreta le corresponde una versión de S.O. concreta.

## Consideraciones Android de Open Data

La aplicación Open Data necesita como mínimo la versión 4.1 del S.O. de Android (plataforma API 16, Jelly Bean) y funcionará en todas las versiones posteriores. Esta versión mínimo eligió prediciendo que para el día de presentación de esta TFG apenas existirían en el mercado versiones anteriores a la 4.1.

## Vista

Una vista es un elemento del interfaz del usuario. Por ejemplo: un botón, una línea de texto… Una Actividad está compuesta por una o más vistas.

## Actividad

Definimos el concepto de Actividad: en Android una Actividad es una pantalla del Interfaz del Usuario y tiene asociada una subclase de la clase Activity. Es en ésta subclase donde definiremos todo el código de Android asociado a la actividad, como layouts, líneas de texto, eventos, etc. Por lo general no todo el código se escribe en esta clase, aunque se podría, y viene ayudado de otra forma alternativa de definir elementos de visualización como son los archivos XML. Con lo que nuestro código de la Actividad tiene mezclados código de las API, más archivos XML.

## Intención

Intención: Una intención es con concepto asociado a la programación, y es la voluntad de una aplicación de hacer algo. Esta voluntad se comunica al S.O. y este reacciona adecuadamente. P.ej. Una intención de una aplicación sería podría ser hacer una llamada por teléfono, se define en el código de la aplicación y el S.O. reaccionará para facilitar esa llamada.

## Máquina Virtual Dalvik

Máquina Virtual Dalvik: es una implementación del Máquina Virtual Java optimizada para dispositivos móviles. Aunque la sintaxis del código fuente de las actividades y demás código es igual a Java, las API de Android no son compatibles con todos los códigos Java, así, por ejemplo, no admite el AWT entre otros.

## Ciclo de vida de una actividad

El ciclo de vida de una aplicación Android es bastante diferente al ciclo de vida de una aplicación en otros S.O., como Windows. La mayor diferencia es que, en Android el ciclo de vida es controlado principalmente por el sistema, en lugar de ser controlado directamente por el usuario.

Son las actividades las que realmente controlan el ciclo de vida de las aplicaciones, dado que el usuario no cambia de aplicación, sino de actividad. El sistema mantiene una pila con las actividades previamente visualizadas, de forma que el usuario puede regresar a la actividad anterior pulsando la tecla “retorno”.

Una aplicación Android corre dentro de su propio proceso Linux. Este proceso se crea con la aplicación y continuará vivo hasta que ya no sea requerido y el sistema reclame su memoria para asignársela a otra aplicación.

Una característica importante, y poco usual, de Android es que la destrucción de un proceso no es controlada directamente por la aplicación, sino que es el sistema el que determina cuándo destruir el proceso. Lo hace basándose en el conocimiento que tiene de las partes de la aplicación que están corriendo (actividades y servicios), en la importancia de dichas partes para el usuario y en cuánta memoria disponible hay en un determinado momento.

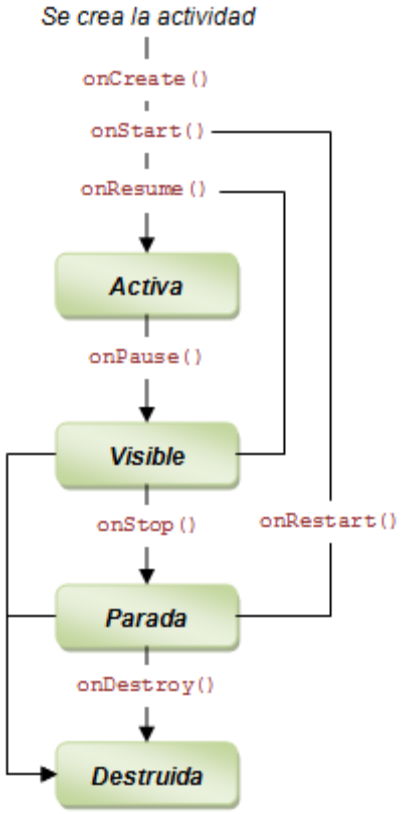
Si tras eliminar el proceso de una aplicación, el usuario vuelve a ella, se crea de nuevo el proceso, pero se habrá perdido el estado que tenía esa aplicación. En estos casos, será responsabilidad del programador almacenar el estado de las actividades, si queremos que cuando sean reiniciadas conserven su estado.

Como vemos, Android es sensible al ciclo de vida de una actividad; por lo tanto, necesitas comprender y manejar los eventos relacionados con el ciclo de vida si quieres crear aplicaciones estables.

Una actividad en Android puede estar en uno de estos cuatro estados:

- **Activa** (Running): La actividad está encima de la pila, lo que quiere decir que es visible y tiene el foco.
- **Visible** (Paused): La actividad es visible pero no tiene el foco. Se alcanza este estado cuando pasa a activa otra actividad con alguna parte transparente o que no ocupa toda la pantalla. Cuando una actividad está tapada por completo, pasa a estar parada.
- **Parada** (Stopped): Cuando la actividad no es visible. El programador debe guardar el estado de la interfaz de usuario, preferencias, etc.
- **Destruida** (Destroyed): Cuando la actividad termina al invocarse el método finish(), o es matada por el sistema.

Cada vez que una actividad cambia de estado se van a generar eventos que podrán ser capturados por ciertos métodos de la actividad. A continuación se muestra un esquema que ilustra los métodos que capturan estos eventos.




*Ciclo de vida de una actividad.*

- **onCreate(Bundle)**: Se llama en la creación de la actividad. Se utiliza para realizar todo tipo de inicializaciones, como la creación de la interfaz de usuario o la inicialización de estructuras de datos. Puede recibir información de estado de la actividad (en una instancia de la clase Bundle), por si se reanuda desde una actividad que ha sido destruida y vuelta a crear.
- **onStart()**: Nos indica que la actividad está a punto de ser mostrada al usuario.
- **onResume()**: Se llama cuando la actividad va a comenzar a interactuar con el usuario. Es un buen lugar para lanzar las animaciones y la música.
- **onPause**(): Indica que la actividad está a punto de ser lanzada a segundo plano, normalmente porque otra actividad es lanzada. Es el lugar adecuado para detener animaciones, música o almacenar los datos que estaban en edición.
- **onStop**(): La actividad ya no va a ser visible para el usuario. Ojo si hay muy poca memoria, es posible que la actividad se destruya sin llamar a este método.
- **onRestart**(): Indica que la actividad va a volver a ser representada después de haber pasado por onStop().
- **onDestroy()**: Se llama antes de que la actividad sea totalmente destruida. Por ejemplo, cuando el usuario pulsa el botón de volver o cuando se llama al método finish(). Ojo si hay muy poca memoria, es posible que la actividad se destruya sin llamar a este método.

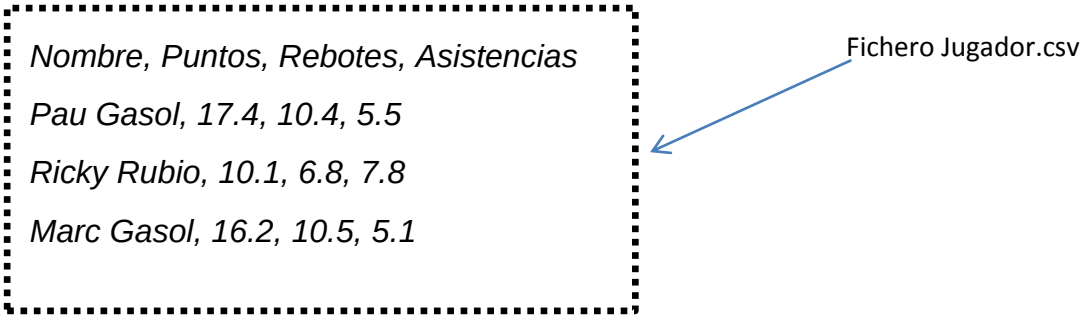
## Capítulo 3: Ficheros csv

Este es el tipo de formato de documento que lee la aplicación Open Data. A continuación vamos a explicarlo con un sencillo ejemplo.

Los archivos **CSV** (del inglés *comma-separated values*) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas (o punto y coma en donde la coma es el separador decimal: Argentina, Brasil...) y las filas por saltos de línea. Los campos que contengan una coma, un salto de línea o una comilla doble deben ser encerrados entre comillas dobles.

El formato CSV es muy sencillo y no indica un juego de caracteres concreto, ni cómo van situados los bytes, ni el formato para el salto de línea. La primera fila que encontramos en el fichero se corresponde con los nombres de los campos y las siguientes filas se corresponden con los registros que contienen, cada uno, un valor para cada campo.

Veamos un ejemplo de fichero CSV, al que llamaremos 'Jugador.csv', este fichero podría ser creado fácilmente con el Bloc de Notas de Windows:




Este fichero se correspondería con la tabla:

| *Nombre* | *Puntos* | *Rebotes* | *Asistencias* |
|---|---|---|---|
| *Pau Gasol* | *17.4* | *10.4* | *5.5* |
| *Ricky Rubio* | *10.1* | *6.8* | *7.8* |
| *Marc Gasol* | *16.2* | *10.5* | *5.1* |

Nuestra aplicación puede leer datos separados por comas o por punto y coma y es responsabilidad del usuario configurarla adecuadamente para leer el fichero correctamente.

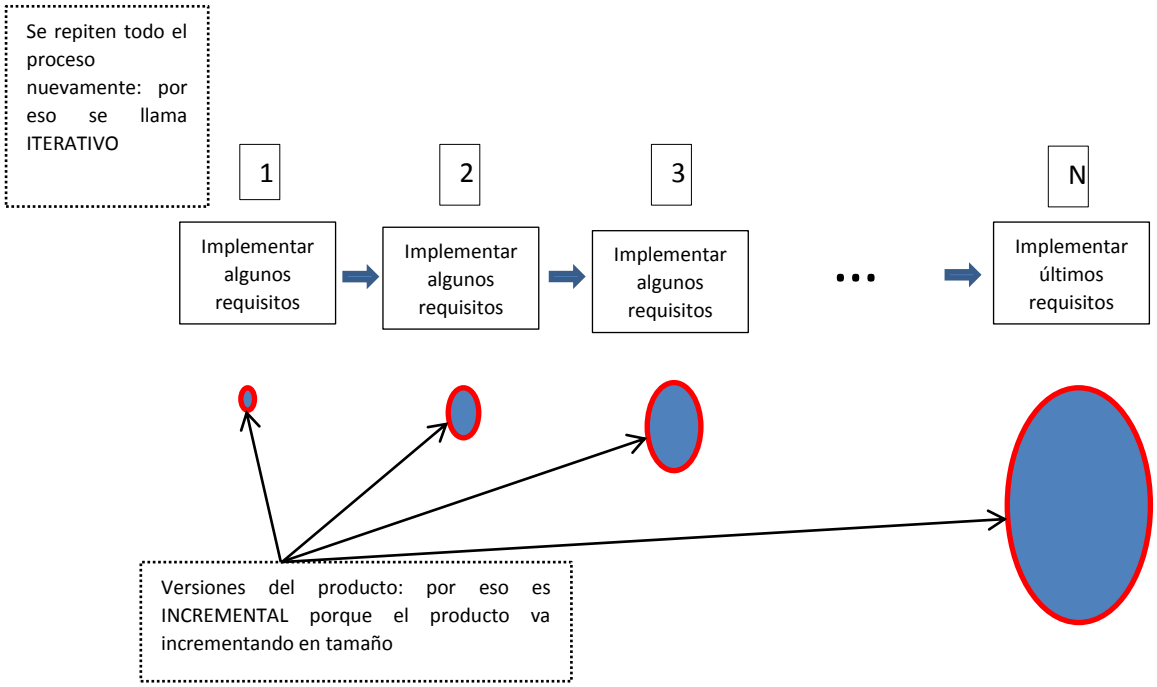
# Capítulo 4: Fases del Proyecto

En este capítulo vamos a ver explicar qué Proceso de Desarrollo Software hemos usado, además de explicar, con ejemplos, las fases de Captura de Requisitos, Análisis, Diseño, Implementación y Pruebas.

Como punto inicial partimos de que nunca hemos desarrollado un producto parecido al que se pide y de que carecíamos de experiencia. Además la tecnología utilizada es nueva para nosotros, lo que aumenta la novedad e imprevisibilidad, con lo que hemos de afrontar y gestionar un desarrollo innovador teniendo en cuenta que tuvimos que: retrasar las especificaciones detalladas, retrasar las estimaciones de tiempo y esfuerzo, usar etapas de realimentación y pensar en una tasa de cambios de requisitos alta. Es por todo ello que decidimos seguir las prácticas de los Métodos de Desarrollo Iterativos e Incrementales (IID). Entre estas prácticas, de las esenciales, decimos usar las siguientes:

**Desarrollo Iterativo:**

En IID el ciclo de vida global se componen de varias iteraciones en secuencia. Cada iteración es un 'mini proyecto' compuesto por actividades de captura de requisitos, análisis de requisitos, diseño, programación y prueba. Cada 'mini proyecto' da como resultado una versión estable, integrada y probada. Y es la última versión, la de la última iteración, la que da como resultado el producto final. A continuación vemos un gráfico explicativo de esta práctica.




Para el desarrollo de nuestra App hemos usado cuatro iteraciones, pasando por las sucesivas etapas de captura de requisitos, análisis, diseño, implementación y

pruebas, en cada iteración. Cada iteración se corresponde con una aproximación, más completa, a la solución final. En la primera nos centremos en definir, aún más, el interfaz de usuario general de la aplicación. En la segunda dividimos el proyecto en cuatro módulos e hicimos que funcionasen. Se conserva esta división en subsistemas en el producto final (subsistemas de: lectura de datos, selección, cálculo y modificación de datos). En la tercera iteración se hizo posible la comparación de muestras entre sí y refinamos la versión anterior, además de añadir algunos requisitos complementarios como la posibilidad de compartir los resultados. Y en la cuarto iteración se retocó el producto anterior y se facilitó el uso y la escritura de la url por parte del usuario a través de la inclusión de un navegador personalizado.

**Desarrollo Incremental y Evolutivo:**

Esta característica está casi siempre emparejada a la anterior. En cada iteración se hacen muchas cosas (como el refinamiento de requisitos…) y entre ellas está el aumento de la funcionalidad del sistema objeto. Es por ello que se habla de desarrollo incremental, es decir, en cada iteración se va incrementando la funcionalidad del producto.

**Dirigido por el Cliente:**

Está característica nos dice que es el cliente quién decide qué requisitos son los que hay que desarrollar en cada iteración. En nuestro caso el papel de Cliente ha sido realizado mi tutor y yo mismo.

**Requisitos Evolutivos:**

Al comienzo de cada iteración hemos revisado y refinado los diferentes requisitos.

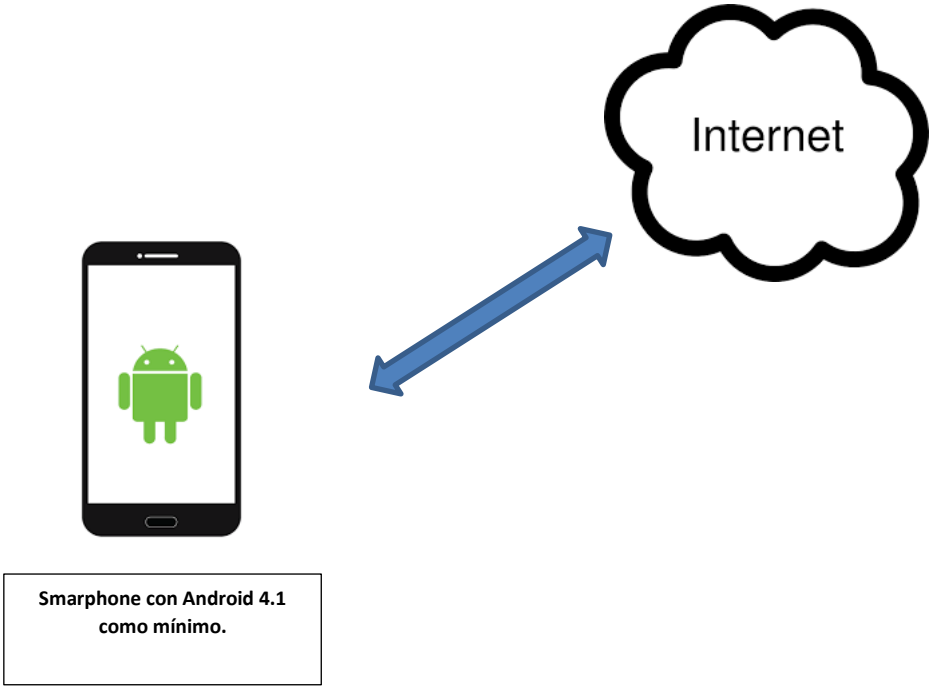
**Planificación Adaptativa:**

La verdadera planificación del proyecto, precisa y detallada, era verdaderamente difícil darla al comienzo. Ha sido cuando teníamos un 20% del trabajo cuando podíamos hacerla.

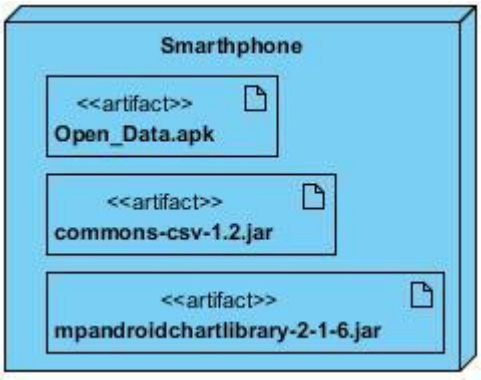
**Centrado en la Arquitectura:**

Debemos de definir, sobre todos en sistema muy complejos, la arquitectura lo antes posible pues nos va a determinar, en términos de planificación todo el proyecto. Si tenemos dudas en la arquitectura a utilizar, la estrategia será abordar los casos de uso más determinante desde este punto de vista para poder decidir la arquitectura adecuada lo antes posible. En nuestro caso decidir esto fue muy fácil, pues era una arquitectura muy conocida y muy usada, además que los requisitos no influenciaron de manera determinante.

Nuestro diagrama de despliegue para describir la estructura física del sistema es el siguiente:




Y dentro del smarthphone podemos representar su despliegue de la siguiente manera:




En nuestro caso, nuestra arquitectura es bien simple: el programa está diseñado para correr en un Smarphone con acceso a internet. Y este Smarphone trabaja bajo el SO Android 4.1 como mínimo, pudiendo trabajar en cualquiera de las versiones posteriores. Cómo vemos en el segundo diagrama el sistema usa dos bibliotecas, una para leer correctamente archivos CSV y otra para representar gráficos en un Smarthphone. El artefacto Open_Data.apk está asociado al paquete android 'com.example.modulo_lectura_datos' dentro del cual encontraremos los archivos interpretables de android correspondientes a la App creada.

El uso de la plataforma Android API 16 hizo posible plantearnos los casos de uso: Compartir Gráfico, Compartir Cálculos y Compartir Fichero. Como vemos aquí

es una clara influencia de la Arquitectura sobre el Modelo de Casos de Uso. Los Casos de Uso también puede influir sobre la Arquitectura, pero no es ese nuestro caso.

A continuación detallamos las diferentes fases del proyecto en cada iteración.

## Captura de requisitos:

En esta etapa, e iterativamente en cuatro entrevistas, además de los requisitos señalados en el Anteproyecto, hemos ido captando diferentes requisitos. Algunos imprescindibles que provienen del Anteproyecto y otros los hemos decidido añadirlos, sobre la marcha, para enriquecer el producto final. Es una etapa dirigida sobre todo al cliente, donde usamos el lenguaje natural y diagramas de caso de uso muy fácilmente entendibles por el cliente y el analista.

### Requisitos Funcionales y No Funcionales:

Esta división de requisitos nos facilita el desarrollo del producto final al distinguir, los requisitos funcionales de los que no lo son. Es decir distinguir lo que sistema debe realizar sin tener en cuenta restricciones físicas (requisitos funcionales) y de los tienen sí tienen en cuenta estas restricciones (requisitos no funcionales). Estos últimos puede influenciar a los otros.

Los requisitos funcionales, nos va a servir, para interactuar y potenciar la comunicación con el cliente. En nuestro producto, los podemos ver en la siguiente tabla:

| REQUISITOS FUNCIONALES | DESCRIPCIÓN |
|---|---|
| VISUALIZAR GRÁFICOS | El usuario podrá visualizar la muestra de datos o las diferentes muestras de datos (en este segundo caso para poder compararlas entre sí) en diferentes gráficos: Gráfico de Líneas, Gráfico de Barras, Gráfico de Barras Horizontal y Gráfico de Tarta. En el gráfico de Líneas y cuando se selecciona una sola muestra se mostrará gráficamente el valor de la media y la mediana. |
| COMPARTIR GRÁFICOS | El usuario podrá compartir el estado actual de los gráficos que sean visibles al usuario en ese momento. |
| VISUALIZAR CÁLCULOS | El usuario podrá visualizar los cálculos estadísticos por cada campo seleccionado: Media, Moda, Mediana, Varianza, Desviación Típica, Rango, Error Estándar de la Media, Coeficiente de Variación de Pearson. |
| COMPARTIR CÁLCULOS | El usuario podrá compartir los datos calculados de las diferentes muestras. Se exceptúan la |

| | |
|---|---|
| | Covarianza y el Coeficiente de Correlación de Pearson debido a sus características singulares. |
| CALCULAR RELACIÓN LÍNEAL | El usuario podrá calcular la Covarianza y Coeficiente de Correlación de Pearson de dos muestras. |
| VISUALIZAR LOS DATOS | El usuario podrá visualizar el estado actual de los datos de entrada (fichero csv). Se indicará cuando estos datos han sido filtrado y/o editados. |
| EDITAR LOS DATOS | El usuario podrá eliminar, insertar, añadir y editar los registros que conforman el estado actual de los datos de entrada (fichero csv). |
| FILTRAR DATOS | El usuario podrá filtrar los registros del estado actual de los datos de entrada con respecto a cualquier campo. Podremos filtrar campos numéricos, de cadenas de caracteres y con formato de fecha. |
| EXPORTAR DATOS | El usuario podrá salvar en cualquier momento el estado actual de los datos de entrada. Esto se salvará en un fichero CSV que podrá almacenarse tanto en la memoria interna como en la externa. |
| IMPORTAR DATOS | El usuario podrá leer los datos desde Internet. Se facilitará esta lectura con un Navegador personalizado encargado de averiguar la URL del fichero CSV a tratar, que podrá arrancarse desde la aplicación o desde cualquier otro navegador compartiendo el contenido web del mismo. También facilitará la introducción de URL con un Historial de carga. |
| CREAR ARCHIVO | El usuario podrá crear un archivo nuevo CSV que posteriormente se podrá editar para rellenar sus campos. |
| ELIMINAR FICHERO | El usuario podrá eliminar ficheros almacenados en el dispositivo en la memoria interna o en la memoria externa. |
| COMPARTIR FICHERO | El usuario podrá compartir ficheros almacenados en la memoria interna del dispositivo |
| ENTRADA DE DATOS | El usuario podrá introducir los datos a través de un fichero CSV colgado en internet, o desde el almacenamiento del dispositivo (desde la memoria interna o desde la memoria externa) o directamente desde teclado. |
| AVISO DE DATOS NO | Se advertirá al usuario cuando quiere |

| | |
|---|---|
| GUARDADOS | abandonar la aplicación y el estado actual de los datos de entrada no ha sido guardado con anterioridad. El usuario podrá retroceder y guardar estos datos. |
| ACCESO A LOS FICHEROS DESCARGADOS | El usuario podrá tener acceso a los ficheros CSV descargados en la memoria externa por cualquier otra aplicación, como podría ser otro navegador que descargase estos ficheros. |
| ENTRADA DE TECLADO | Cuando los datos de entrada sean suministrados directamente por el teclado, la aplicación pasará directamente a mostrar los resultados, sin que se puedan modificar estos datos ni guardarlos. |
| IMÁGENES A GALERÍA | Se debe permitir llevar las fotos de los gráficos a la galería del dispositivo móvil. |

Vemos una tabla con los requisitos No funcionales, es decir, aquellos que se imponen a los funcionales, basándose en lo físico:

| NO FUNCIONALES | DESCRIPCIÓN |
|---|---|
| MAX TAMAÑO DE FICHERO | El usuario podrá configurar que la aplicación no acepte ficheros a partir de un tamaño máximo en bytes, debido a que estos pueden hacer que el sistema no responda. |
| VENTANA DE VISUALIZACIÓN | El usuario podrá configurar cuando registros se muestran como máximo a la vez cuando visualizamos el estado actual de los datos (fichero csv), debido a que si son demasiados, pueden hacer que el sistema no responda. |
| SOLO UNA INSTANCIA DE LA APLICACIÓN | No podrá solaparse dos instancias de la aplicación por razones de consistencias de datos y porque se usan atributos estáticos que funcionan como grandes variables globales. |
| ANDROID | Mínima versión 4.1. |

**CASOS DE USO:**

Veamos una de las principales técnicas para comunicarse con el cliente y demás personas participantes en el proyecto: son los casos de uso. Permiten la captura de requisitos de una forma muy intuitiva y muy cercana al lenguaje natural. Tampoco nos interesa tratar de especificar mucho en este modelo pues entorpecería la comunicación con cliente. Se trata de una descripción del sistema en su más alto

nivel. Nos servirán como guía para las demás etapas y debemos mantener su trazabilidad.

Nuestro Modelos de Casos de Uso es el siguiente:


USUARIO
VER GRÁFICAS
COMPARTIR GRÁFICOS
<<Extend>>
A GALERIA
<<Extend>>
CALCULO ESTADÍSTICO
COMPARTIR CÁLCULOS
<<Extend>>
CALCULAR RELACIÓN LINEAL
<<Extend>>
VISUALIZAR DATOS
USAR NAVEGADOR PERSONALIZADO
EDITAR DATOS
FILTRAR DATOS
EXPORTAR DATOS
IMPORTAR DATOS
CREAR ARCHIVO
ELIMINAR ARCHIVO
COMPARTIR ARCHIVO

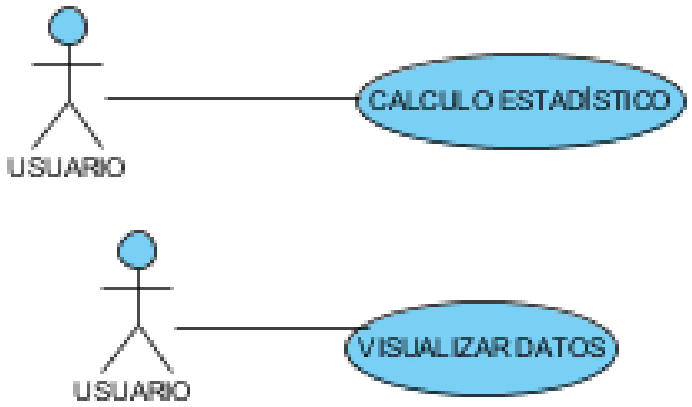
Los casos de uso suelen ser un subconjunto amplio de los requisitos funcionales. Captan requisitos a grandes detalles, sin entrar, en la complejidad física de su implementación. Estos han ido creciendo en cada iteración y algunos se han visto influenciados por la arquitectura software como ya hemos visto.

Por otra parte este modelo de casos de uso será el punto de partida para el desarrollador, y éste podrá usar un modelo de casos de uso más completo, menos intuitivo, que le permita entrar en mayor detalle. Usando más relaciones tipo <<include>> y <<extends>>.

## Análisis:

La siguiente fase, como las demás, ha creciendo y refinándose en cada iteración. Es la de análisis. Como su propio nombre indica, es estudiar los requisitos en sí relacionándolos con los demás actores y clases de análisis que vayan apareciendo en esta fase.

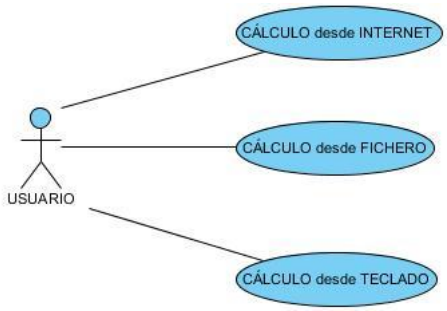
Vamos a ver la realización parcial de dos casos de uso, es decir, la realización de dos escenarios concretos de dos casos de uso concretos. Tener en cuenta que cada caso de uso puede tener varios escenarios posibles. En nuestro ejemplo vamos a realizar el análisis del escenario principal de éxito del caso de uso 'CÁLCULO ESTADÍSTICO' y el análisis del escenario principal de éxito del caso de uso 'VISUALIZAR DATOS'.




Además dentro de estos casos de uso, puede haber otros casos de uso. Nosotros supondremos que el nivel superior está enfocado para la comunicación con el cliente y creamos otro nivel de casos de uso en la que estos están enfocados para el desarrollador. Aunque se podría agrupar todos estos casos de uso en su nivel más superior, decidimos no hacerlo con objeto de no complicar demasiado el modelo de casos de uso al que tiene acceso el cliente final. Ni usar demasiados las relaciones <<include>> ni <<extend>> que nos aleja del lenguaje natural y de la

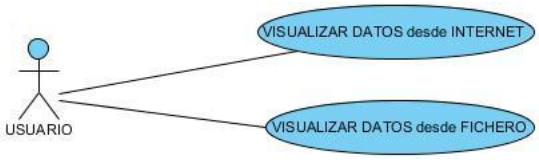
facilidad de compresión del cliente de lo que se espera que haga el sistema (la aplicación).

Así dentro del caso de uso ‘CALCULO ESTADISTICO’ tenemos a su vez el siguiente nivel de casos de usos:

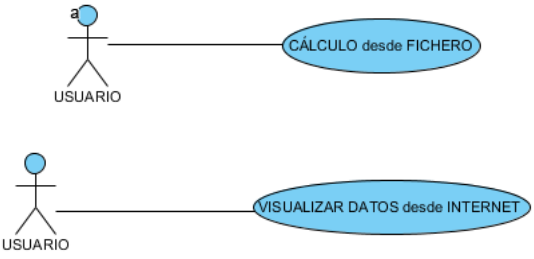



Como vemos en diagrama superior, el Usuario puede introducir los datos para su cálculo de tres formas: a través de una dirección de internet, seleccionando un fichero almacenado en memoria del dispositivo, o bien introduciendo los datos directamente desde el teclado.

Dentro del caso de uso ‘VISUALIZAR DATOS’ nos encontramos que podemos visualizar los datos desde un archivo interno del móvil o descargado a través de internet. El siguiente diagrama de uso viene a expresar esto.




Para el caso de uso ‘CÁLCULO ESTADÍSTICO’ vamos a realizar el caso en el que el usuario optar por analizar un fichero almacenado en el dispositivo y para el caso de uso ‘VISUALIZAR DATOS’ vamos a realizar cuando el usuario decido ver el contenido de un fichero colgado en internet. Realizaremos solo este dos subcasos por no extender en exceso este documento que supone la memoria.




Los casos de uso textual de los escenarios principales de éxito pueden ser los siguientes:

CASO DE USO: CÁLCULO ESTADÍSTICO desde FICHERO

CONTEXTO DE USO: El Usuario quiere visualizar los datos estadísticos de los campos seleccionados a través de un fichero local.

ÁMBITO: Sistema caja negra
NIVEL: Usuario
ACTOR PRINCIPAL: El usuario
PARTICIPANTES Y OBJETIVOS:
Usuario: Se le muestra los datos estadísticos de los campos del fichero seleccionado por el mismo.
PRECONDICIONES:
El usuario se encuentra ante el interfaz de usuario de carga de datos y podemos averiguar el tamaño, en bytes, del fichero CSV. Además existen al menos un fichero para seleccionar.
GARANTÍAS MÍNIMAS:
El sistema reaccionará de manera estable.
GARANTÍAS DE ÉXITO:
El sistema visualiza los cálculos en el móvil.

ESCENARIO PRINCIPAL DE ÉXITO:

1. El usuario selecciona en el sistema el fichero a analizar.
2. El sistema muestra los campos del fichero
3. El usuario selecciona los campos para analizar.
4. El sistema muestra los cálculos de los campos seleccionados.

EXTENSIONES:
2.a. El sistema advierte que el fichero seleccionado es demasiado grande para ser tratado. El caso de uso termina.
3.a. El usuario no selecciona ningún campo numérico. El caso de uso termina.

**CASO DE USO: VISUALIZAR DATOS desde INTERNET**

**CONTEXTO DE USO:** El Usuario quiere visualizar los datos del fichero CSV seleccionado por el mismo a través de internet

**ÁMBITO:** Sistema caja negra
**NIVEL:** Usuario
**ACTOR PRINCIPAL**: El usuario
**PARTICIPANTES Y OBJETIVOS:**
Usuario: Se le muestra los datos del fichero internet seleccionado por el mismo.
**PRECONDICIONES:**
Existe el Fichero deseado por el Usuario en internet. Además podemos averiguar el tamaño de este fichero.
**GARANTÍAS MÍNIMAS:**
El sistema reaccionará de manera estable.

**GARANTÍAS DE ÉXITO:**

El sistema visualiza los datos de la url en el móvil.

**ESCENARIO PRINCIPAL DE ÉXITO:**

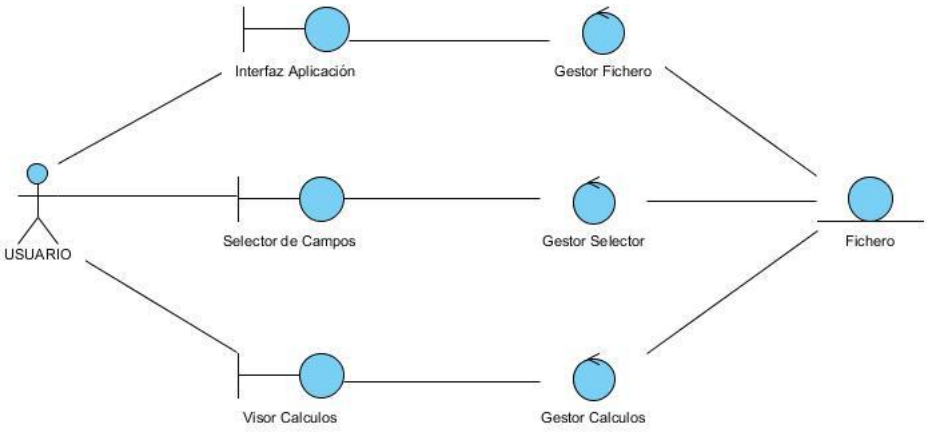
1. El usuario introduce la url del fichero a analizar
2. El sistema muestra los campos del fichero
3. El usuario pide que se le muestre el fichero.
4. El sistema muestra el fichero.

**EXTENSIONES:**

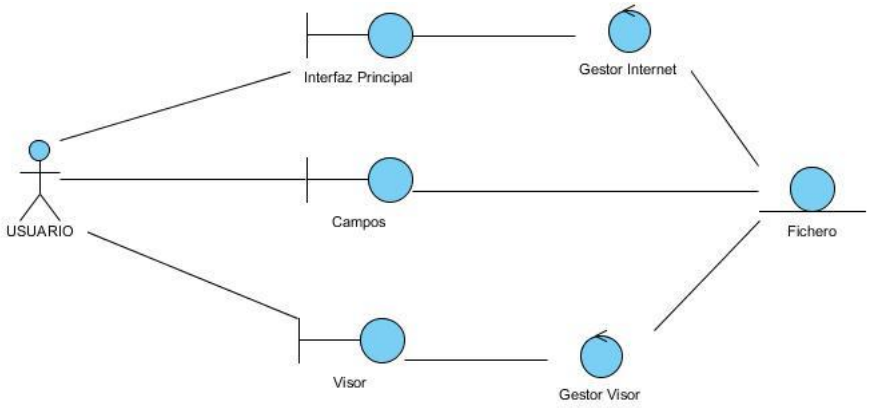
2.a. El sistema advierte que el fichero seleccionado es demasiado grande para ser tratado. El caso de uso termina.

Los diagramas de clases de análisis son los siguientes:

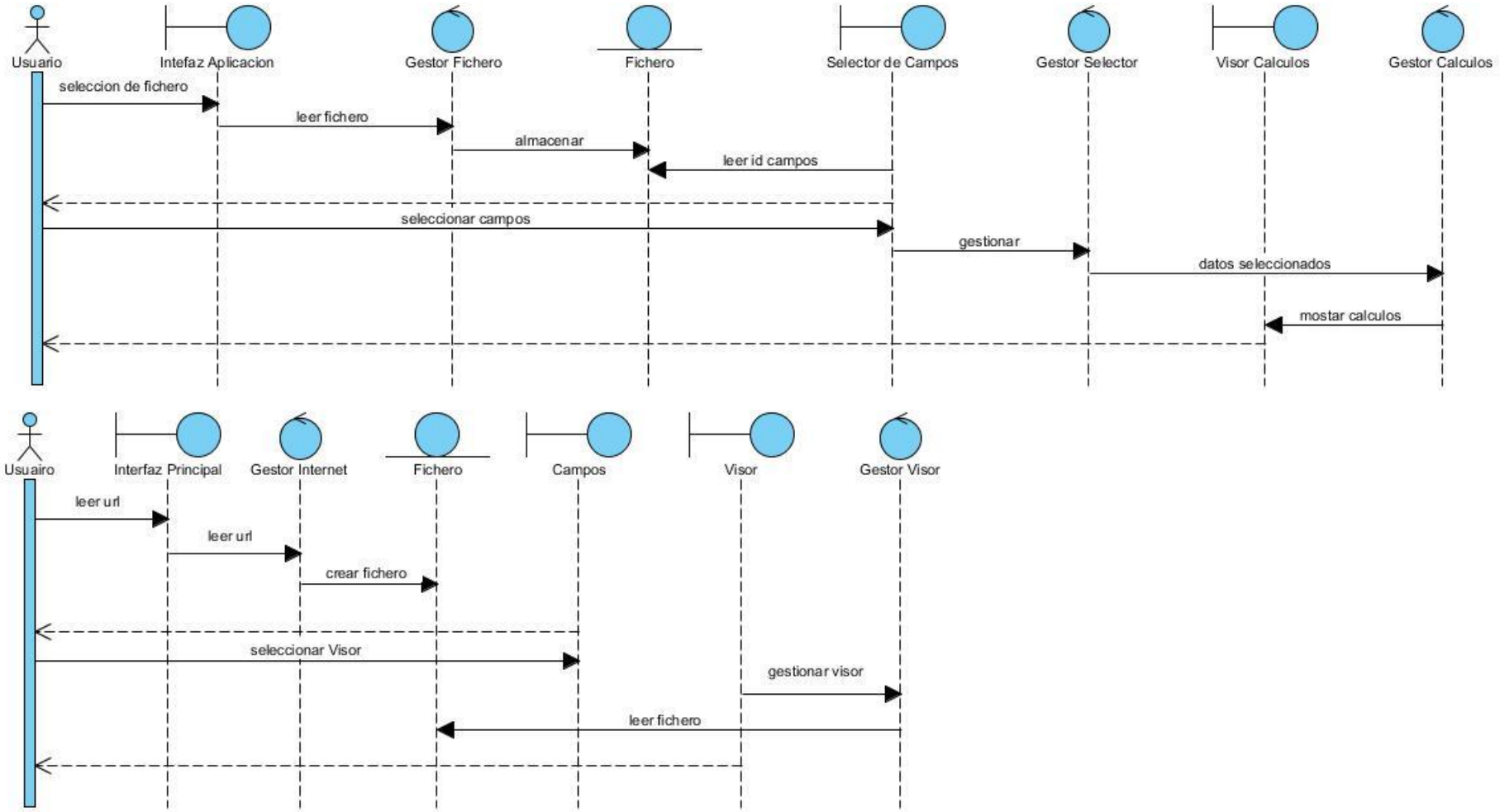
Para el subcaso de uso 'CÁLCULO desde Fichero'




Y para el subcaso de uso 'VISUALIZAR DATOS desde INTERNET':




Para aclarar las relaciones entre las clases de arriba se realiza los diagramas de secuencia de análisis siguientes para cada subcaso:


Usuario
Intefaz Aplicacion
Gestor Fichero
Fichero
Selector de Campos
Gestor Selector
Visor Calculos
Gestor Calculos
seleccion de fichero
leer fichero
almacenar
leer id campos
seleccionar campos
gestionar
datos seleccionados
mostar calculos
Usuairo
Interfaz Principal
Gestor Internet
Fichero
Campos
Visor
Gestor Visor
leer url
leer url
crear fichero
seleccionar Visor
gestionar visor
leer fichero

Estos tres tipos de artefactos de análisis sirven para refinar aún más los requisitos captados en la primera fase. Damos un primer paso para la implementación y permite realizar, en el diseño, los casos de uso. En el caso de uso textual empezamos a entender en como interacciona el sistema con el usuario para lograr su objetivo. El diagrama de clases de análisis sirve para dar un primer paso e introducir las clases en su nivel más abstracto. El diagrama de secuencia de análisis de este escenario sirve para identificar la comunicación entre clases genéricas para llevar a cabo la realización de éste.

Todavía no hemos despegado desde el punto de vista de la implementación. Estamos en una fase donde estamos más cerca de la abstracción que de su realidad física de la arquitectura. Se podría decir que estamos todavía más cerca del lenguaje natural, propio del cliente, que del lenguaje técnico propio del programador.

De manera iterativa e incremental se realizarían todos los casos de uso, aumentando el tamaño del producto desde el punto de vista funcional. Los casos de uso realizados en esta etapa podrían sufrir alteraciones de requisitos o refinamientos.

## Diseño:

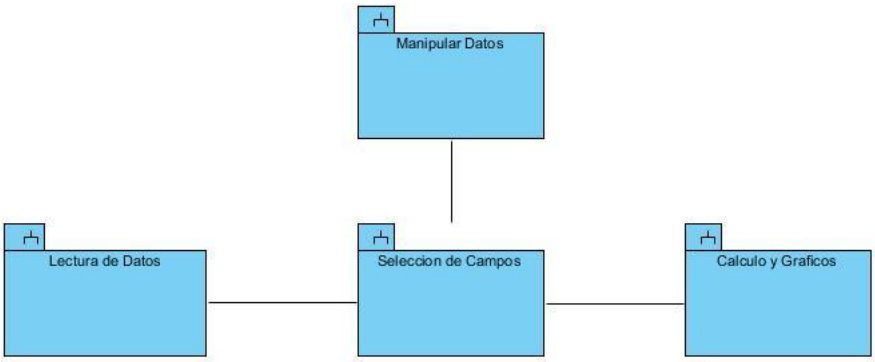
Esta etapa es una etapa intermedia entre la abstracción del problema que tiene el cliente y la abstracción del problema que tiene el programador. Está pensada para facilitar el mantenimiento del producto final y es decisiva, en la construcción de un software de calidad.

Para la realización de un caso de uso de análisis, usamos dos tipos de artefactos principalmente: El diagrama de clases de diseño del producto y los diagramas de secuencia que se hacen a partir de este diagrama de clases.

### DIAGRAMA DE CLASES DE DISEÑO:

El diagrama de clases de diseño sirve para detectar las relaciones entre las clases que van a aparecer en esta etapa, muchas de estas clases de diseño se corresponden con una clase de análisis.

Comencemos definiendo el diagrama de clases de diseño en una composición de Subsistemas relacionados entre sí.




En este diagrama podemos ver que el producto se puede descomponer en cuatro subsistemas:

- **Lectura de Datos**: este subsistema es el encargado de llevar los datos de entrada a la memoria de una manera correcta.
- **Selección de Campos**: es el subsistema encargado de hacer posible que el usuario seleccione los campos del fichero que desea tratar de manera estadística.
- **Cálculo y Gráficos**: es el subsistema encargado de explotar los resultados de los campos seleccionados en el anterior subsistema.
- **Manipular Datos**: este subsistema es el encargado de manipular los datos del fichero existentes en la memoria asignada a la App.

Cada una de estos subsistemas contiene una vista parcial del diagrama de clases de diseño total.

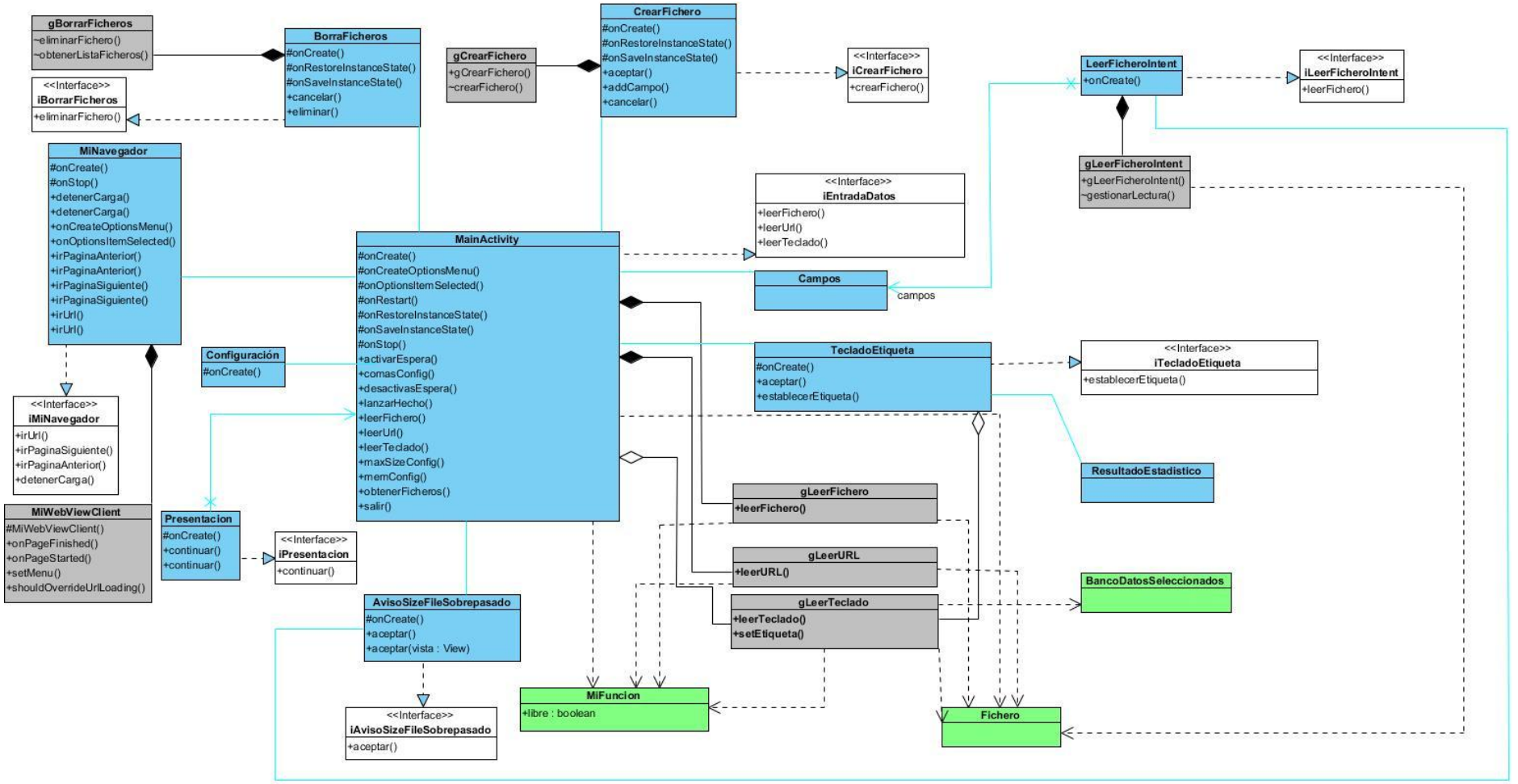
Un aspecto destacable de su diseño general, es el uso masivo de interfaces de las actividades. Este permite definir de antemano la funcionalidad deseada de cada actividad y es una buena técnica para una posible división del trabaja entre distintos programadores. Facilita mucho las potenciales modificaciones de funcionalidades de la aplicación y la organización del trabajo a realizar.

También es importante la división de trabajo que permite las diferentes clases gestoras, haciendo posible la posible división de trabajo entre el diseñador de interfaz de usuario y el trabajo puro de programación que supone la gestión de la funcionalidad.

Otro aspecto, fruto de un buen diseño, es prácticamente la ausencia, salvo los imprescindibles, de atributos de clases que pueden ser accedidos directamente. En su lugar son privados pero gestionados por ‘getters ‘y ‘setters’.

**Subsistema Lectura De Datos:**

Veamos primero el subsistema: ‘Lectura de Datos’ pensado pare leer correctamente los ficheros CSV introducidos por el usuario o los datos directamente introducidos por el teclado. El siguiente diagrama nos muestra su estructura en clases:


gBorrarFicheros
~eliminarFichero()
~obtenerListaFicheros()
BorraFicheros
#onCreate()
#onRestoreInstanceState()
#onSaveInstanceState()
+cancelar()
+eliminar()
<<Interface>>
iBorrarFicheros
+eliminarFichero()
gCrearFichero
+gCrearFichero()
~crearFichero()
CrearFichero
#onCreate()
#onRestoreInstanceState()
#onSaveInstanceState()
+aceptar()
+addCampo()
+cancelar()
<<Interface>>
iCrearFichero
+crearFichero()
LeerFicheroIntent
+onCreate()
<<Interface>>
iLeerFicheroIntent
+leerFichero()
gLeerFicheroIntent
+gLeerFicheroIntent()
~gestionarLectura()
MiNavegador
#onCreate()
#onStop()
+detenerCarga()
+detenerCarga()
+onCreateOptionsMenu()
+onOptionsItemSelected()
+irPaginaAnterior()
+irPaginaAnterior()
+irPaginaSiguiente()
+irPaginaSiguiente()
+irUrl()
+irUrl()
<<Interface>>
iEntradaDatos
+leerFichero()
+leerUrl()
+leerTeclado()
MainActivity
#onCreate()
#onCreateOptionsMenu()
#onOptionsItemSelected()
#onRestart()
#onRestoreInstanceState()
#onSaveInstanceState()
#onStop()
+activarEspera()
+comasConfig()
+desactivasEspera()
+lanzarHecho()
+leerFichero()
+leerUrl()
+leerTeclado()
+maxSizeConfig()
+memConfig()
+obtenerFicheros()
+salir()
Campos
campos
Configuración
#onCreate()
TecladoEtiqueta
#onCreate()
+aceptar()
+establecerEtiqueta()
<<Interface>>
iTecladoEtiqueta
+establecerEtiqueta()
ResultadoEstadistico
<<Interface>>
iMiNavegador
+irUrl()
+irPaginaSiguiente()
+irPaginaAnterior()
+detenerCarga()
MiWebViewClient
#MiWebViewClient()
+onPageFinished()
+onPageStarted()
+setMenu()
+shouldOverrideUrlLoading()
Presentacion
#onCreate()
+continuar()
+continuar()
<<Interface>>
iPresentacion
+continuar()
gLeerFichero
+leerFichero()
gLeerURL
+leerURL()
gLeerTeclado
+leerTeclado()
+setEtiqueta()
BancoDatosSeleccionados
AvisoSizeFileSobrepasado
#onCreate()
+aceptar()
+aceptar(vista : View)
<<Interface>>
iAvisoSizeFileSobrepasado
+aceptar()
MiFuncion
+libre : boolean
Fichero

Las actividades son las clases coloreadas de color azul claro. Cada actividad tiene asociado un interfaz que implementa (color blanco) y varias, una o ninguna clase gestora (color gris) que lleva el peso de la implementación y que suele corresponderse con las clases controladoras de la etapa de análisis. El que tenga una, varias o ninguna clases gestoras asociadas depende de la complejidad de esta gestión.

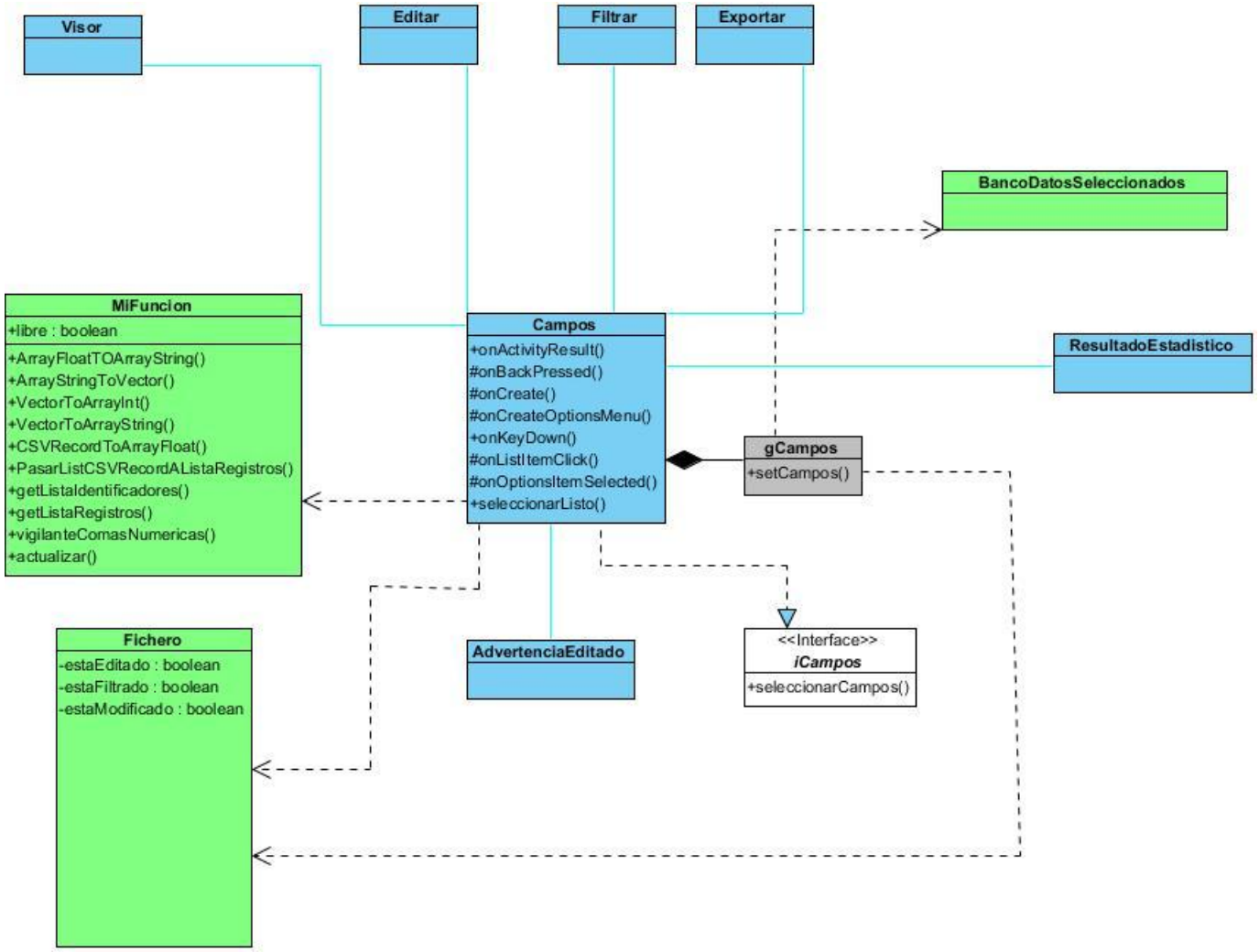
Las actividades se relacionan entre sí mediante relaciones de asociación (color cian) que representan que unas abren a través de un evento. Las otras clases son clases 'puras' (color verde) necesarias para el buen diseño. Algunas de las cuáles se corresponden con las clases entidad de la fase de análisis

Vemos que la actividad '*MainActivity'* es la encarga de cargar los datos. Tiene asociadas tres clases gestoras. Cada una encarga de llevar a cabo el correspondiente tipo de entrada que puede hacer el usuario y que están definidos en el interfaz '*iEntradaDatos'*. Estás son '*leerFichero*()', '*leerUrl*()' y '*leerTeclado*()'. Estos métodos llevan los datos del fichero a la clase '*Fichero'* encargada de almacenarlo en memoria y de soportar las operaciones que hacen las demás clases sobre estos datos. Señalar que el diseño de la funcionalidad de leer por teclado sigue un camino diferente que el diseño para leer desde internet o desde el almacenamiento del dispositivo móvil.

La clase '*MiFuncion'* es una clase que ofrece un tipo de funcionalidad común a todas las de los diferentes subsistemas y clases.

**Subsistema Seleccionar Campos:**

El subsistema Seleccionar Campos se encarga de hacer posible que el usuario seleccione qué campos del fichero desea analizar. El diagrama de clases de subsistema Seleccionar Campos es el siguiente:


Visor
Editar
Filtrar
Exportar
BancoDatosSeleccionados
MiFuncion
+libre : boolean
+ArrayFloatTOArrayString()
+ArrayStringToVector()
+VectorToArrayInt()
+VectorToArrayString()
+CSVRecordToArrayFloat()
+PasarListCSVRecordAListaRegistros()
+getListaIdentificadores()
+getListaRegistros()
+vigilanteComasNumericas()
+actualizar()
Campos
+onActivityResult()
#onBackPressed()
#onCreate()
#onCreateOptionsMenu()
+onKeyDown()
#onListItemClick()
#onOptionsItemSelected()
+seleccionarListo()
ResultadoEstadistico
gCampos
+setCampos()
Fichero
-estaEditado : boolean
-estaFiltrado : boolean
-estaModificado : boolean
AdvertenciaEditado
<<Interface>>
iCampos
+seleccionarCampos()

En este subsistema existe una actividad principal, llamada '*Campos'* que implemente el interfaz '*iCampos'* que define la funcionalidad esencial de la actividad. A esta actividad le corresponde la clase gestora '*gCampos'*. Esta actividad es la encargada de seleccionar los campos numéricos que el usuario le indica. Para ello lee los datos de la clase '*Fichero'* encargada de almacenar en memoria el fichero csv y los almacena en la clase '*BancoDatosSeleccionados'* que va a ser utilizada por el Subsistema 'Cálculo y Gráficos'.

Comentar que por temas tecnológicos, '*Fichero'* y '*BancoDatosSeleccionados'* son como grandes variables globales (tienen atributos estáticos para almacenar esta información).

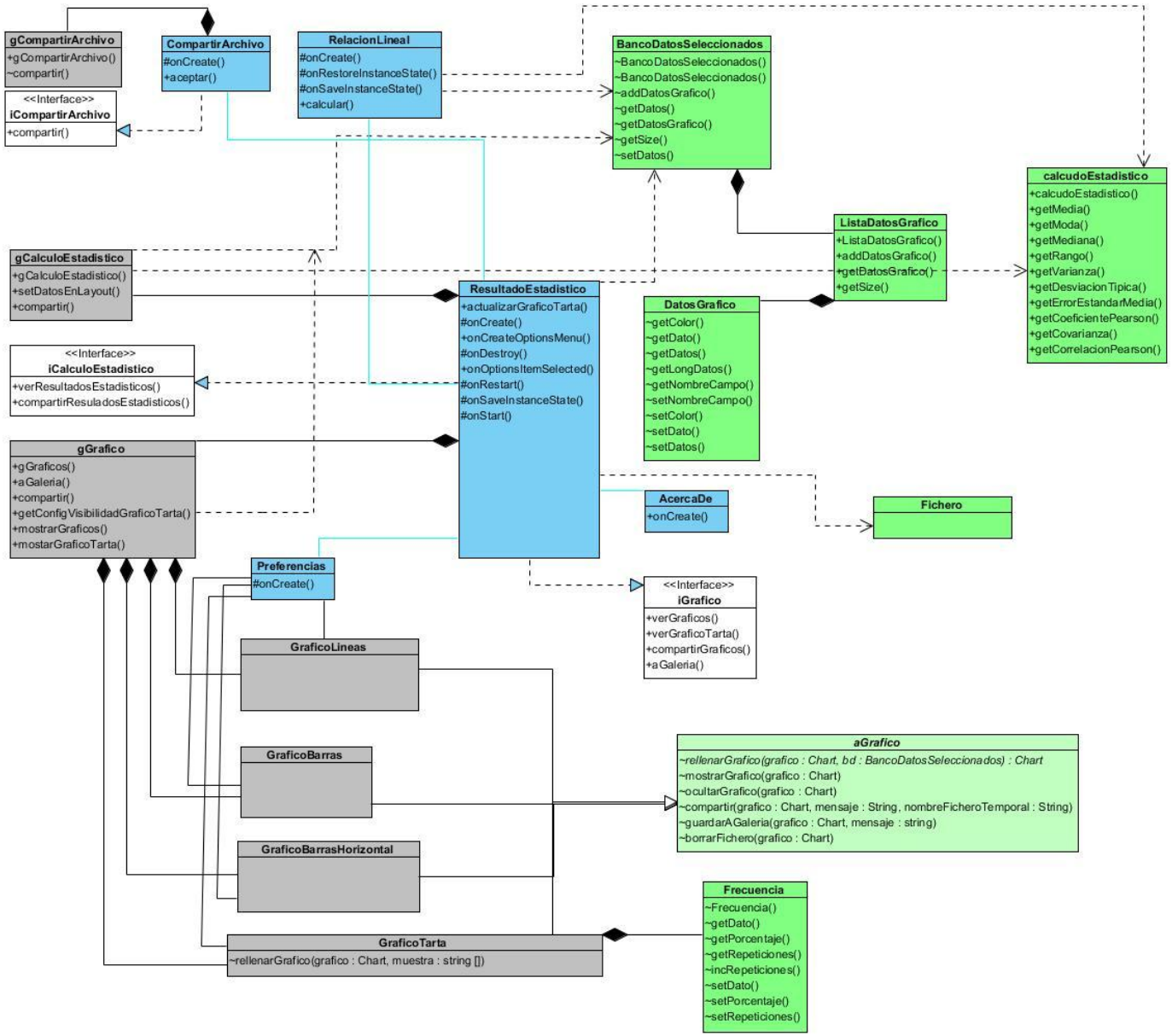
Es necesaria una forma de asegurarnos que no pueda haber dos instancias de la aplicación funcionando paralelamente, debido, al uso de atributos estáticos (variables globales). Este hecho es controlado por el atributo '*libre'* de la clase '*MiFuncion'*. Cuando una Instancia de la aplicación trata de colarse a otra, esta variable sirve como barrera para que no pueda hacerlo.

Otros atributos importantes, son los de la clase '*Fichero*' que indican cuando un fichero ha sido modificado, filtrado o editado. Se usará para advertirnos que se quiere salir de la aplicación y no se han guardado los datos modificados, cuando lo han sido. En este caso se lanzará la actividad '*AdvertenciaEditado'*.

Cuando todo vaya bien, la actividad '*Campos*' hará paso a la actividad '*ResultadoEstadistico*' definido en el subsistema '*Cálculo y Gráficos*' que es el encargado de explotar los resultados obtenidos de cara al usuario.

**Subsistema Cálculo y Gráficos:**

El subsistema 'Cálculo y Gráficos' es el encargado visualizar los datos estadísticos y los gráficos y explotar estos datos-resultados. Su subdiagrama de clases es el siguiente:


gCompartirArchivo
+gCompartirArchivo()
~compartir()
CompartirArchivo
#onCreate()
+aceptar()
RelacionLineal
#onCreate()
#onRestoreInstanceState()
#onSaveInstanceState()
+calcular()
BancoDatosSeleccionados
~BancoDatosSeleccionados()
~BancoDatosSeleccionados()
~addDatosGrafico()
~getDatos()
~getDatosGrafico()
~getSize()
~setDatos()
<<Interface>>
iCompartirArchivo
+compartir()
calcudoEstadistico
+calcudoEstadistico()
+getMedia()
+getModa()
+getMediana()
+getRango()
+getVarianza()
+getDesviacionTipica()
+getErrorEstandarMedia()
+getCoeficientePearson()
+getCovarianza()
+getCorrelacionPearson()
ListaDatosGrafico
+ListaDatosGrafico()
+addDatosGrafico()
+getDatosGrafico()
+getSize()
gCalculoEstadistico
+gCalculoEstadistico()
+setDatosEnLayout()
+compartir()
ResultadoEstadistico
+actualizarGraficoTarta()
#onCreate()
+onCreateOptionsMenu()
#onDestroy()
+onOptionsItemSelected()
#onRestart()
#onSaveInstanceState()
#onStart()
DatosGrafico
~getColor()
~getDato()
~getDatos()
~getLongDatos()
~getNombreCampo()
~setNombreCampo()
~setColor()
~setDato()
~setDatos()
<<Interface>>
iCalculoEstadistico
+verResultadosEstadisticos()
+compartirResuladosEstadisticos()
gGrafico
+gGraficos()
+aGaleria()
+compartir()
+getConfigVisibilidadGraficoTarta()
+mostrarGraficos()
+mostarGraficoTarta()
AcercaDe
+onCreate()
Fichero
Preferencias
#onCreate()
<<Interface>>
iGrafico
+verGraficos()
+verGraficoTarta()
+compartirGraficos()
+aGaleria()
GraficoLineas
GraficoBarras
GraficoBarrasHorizontal
GraficoTarta
~rellenarGrafico(grafico : Chart, muestra : string [])
aGrafico
~rellenarGrafico(grafico : Chart, bd : BancoDatosSeleccionados) : Chart
~mostrarGrafico(grafico : Chart)
~ocultarGrafico(grafico : Chart)
~compartir(grafico : Chart, mensaje : String, nombreFicheroTemporal : String)
~guardarAGaleria(grafico : Chart, mensaje : string)
~borrarFichero(grafico : Chart)
Frecuencia
~Frecuencia()
~getDato()
~getPorcentaje()
~getRepeticiones()
~incRepeticiones()
~setDato()
~setPorcentaje()
~setRepeticiones()

En este diagrama merece la pena destacar la implementación de dos interfaces de la actividad '*ResultadoEstadistico'*. Cada uno encargados de definir una parte de la funcionalidad de esta actividad. Esta funcionalidad está claramente influenciada por la arquitectura software utilizada, pues la biblioteca *MPAndroidChart* no permite la selección individual de un gráfico, así el interfaz 'i*Grafico*' definer el método '*compartirGraficos*()' que comparte uno a uno los gráficos que se visualizan en un momento en la actividad.

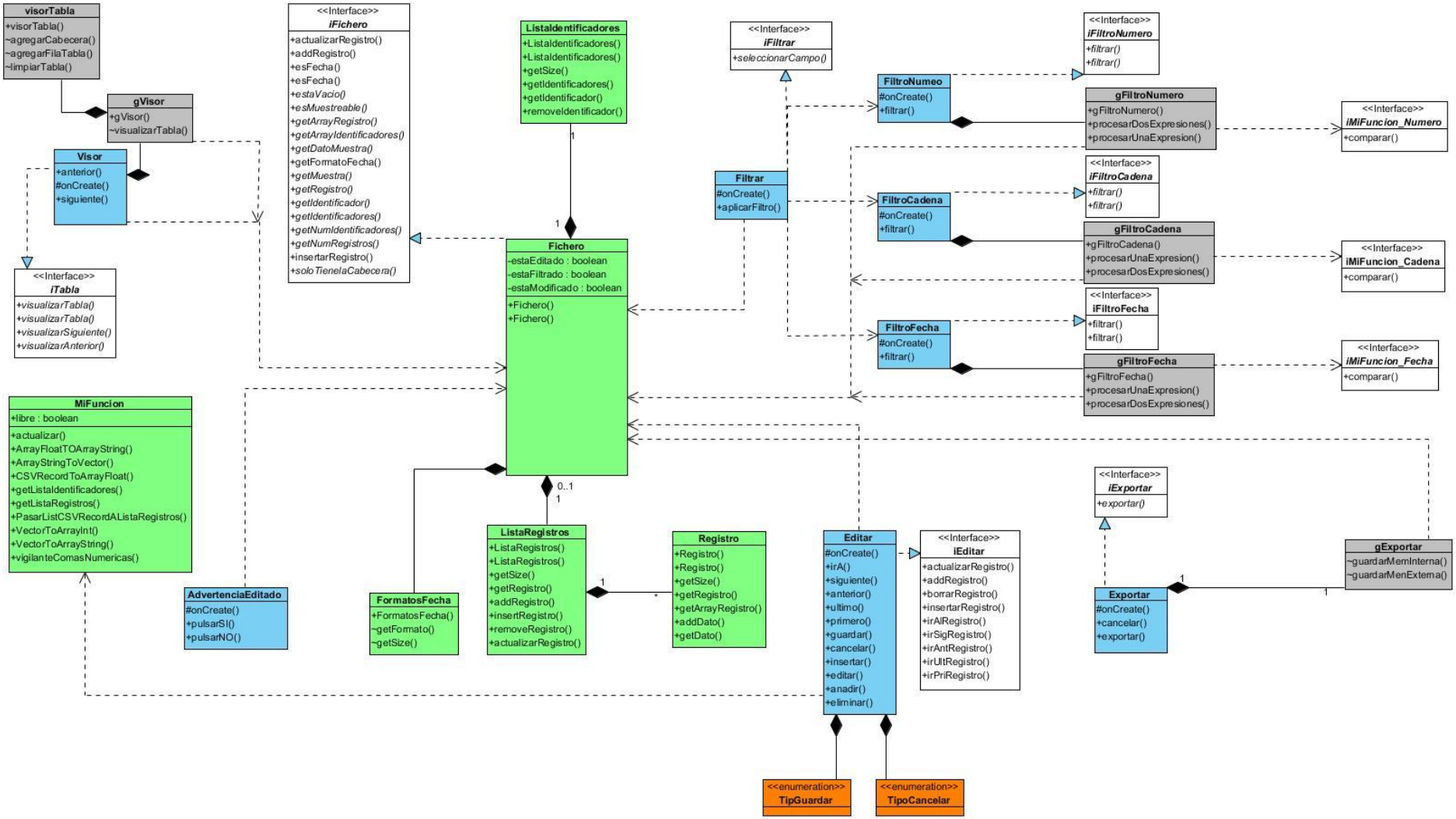
Desde el punto de vista de la programación, y beneficiándose de la reutilización de código, existe la clase abstracta '*aGrafico'* heredada por cada uno de los gráficos que hay y gestionada a su vez por la clase gestora '*gGrafico'*. Este es un diseño inteligente de esta gestión. También destacar que el Gráfico Tarta se comporta de manera diferente que los otros gráficos y tiene un método adicional que define el interfaz con el que va a interactuar con el contenido del 'Fichero'. El resto de gráficos leen sus datos de la clase '*BancoDatosSeleccionados'*. Estás relaciones vienen expresada en el diagrama con dos dependencias: para el gráfico de tarta, hay una dependencia de '*ResultadosEstadisticos'* a la clase 'Fichero' y para la segunda hay una dependencia de la clase gestora '*gGrafico'* con la clase '*BancoDatosSeleccionados'*.

Es de destacar, en esta diagrama la clase '*BancoDatosSeleccionados'* que funciona como una gran variable global donde van a poder leer el resto de actividades y gestores de este subsistema, y que contiene los datos de las muestras seleccionados por el usuario en la actividad 'Campos' hayan o no sido modificados.

También destacar la clase entidad '*calculoEstadístico'* que soporta las operaciones matemáticas-estadísticas de esta aplicación.

**Subsistema Modificar Datos:**

El siguiente diagrama de clases se corresponde con el subsistema 'Modificar Datos':


visorTabla
+visorTabla()
~agregarCabecera()
~agregarFilaTabla()
~limpiarTabla()
gVisor
+gVisor()
~visualizarTabla()
Visor
+anterior()
#onCreate()
+siguiente()
<<Interface>>
iTabla
+visualizarTabla()
+visualizarTabla()
+visualizarSiguiente()
+visualizarAnterior()
<<Interface>>
iFichero
+actualizarRegistro()
+addRegistro()
+esFecha()
+esFecha()
+estaVacio()
+esMuestreable()
+getArrayRegistro()
+getArrayIdentificadores()
+getDatoMuestra()
+getFormatoFecha()
+getMuestra()
+getRegistro()
+getIdentificador()
+getIdentificadores()
+getNumIdentificadores()
+getNumRegistros()
+insertarRegistro()
+soloTieneIaCabecera()
ListaIdentificadores
+ListaIdentificadores()
+ListaIdentificadores()
+getSize()
+getIdentificadores()
+getIdentificador()
+removeIdentificador()
1
1
Fichero
-estaEditado : boolean
-estaFiltrado : boolean
-estaModificado : boolean
+Fichero()
+Fichero()
<<Interface>>
iFiltrar
+seleccionarCampo()
Filtrar
#onCreate()
+aplicarFiltro()
FiltroNumeo
#onCreate()
+filtrar()
<<Interface>>
iFiltroNumero
+filtrar()
+filtrar()
gFiltroNumero
+gFiltroNumero()
+procesarDosExpresiones()
+procesarUnaExpresion()
<<Interface>>
iMiFuncion_Numero
+comparar()
FiltroCadena
#onCreate()
+filtrar()
<<Interface>>
iFiltroCadena
+filtrar()
+filtrar()
gFiltroCadena
+gFiltroCadena()
+procesarUnaExpresion()
+procesarDosExpresiones()
<<Interface>>
iMiFuncion_Cadena
+comparar()
FiltroFecha
#onCreate()
+filtrar()
<<Interface>>
iFiltroFecha
+filtrar()
+filtrar()
gFiltroFecha
+gFiltroFecha()
+procesarUnaExpresion()
+procesarDosExpresiones()
<<Interface>>
iMiFuncion_Fecha
+comparar()
MiFuncion
+libre : boolean
+actualizar()
+ArrayFloatTOArrayString()
+ArrayStringToVector()
+CSVRecordToArrayFloat()
+getListaIdentificadores()
+getListaRegistros()
+PasarListCSVRecordAListaRegistros()
+VectorToArrayInt()
+VectorToArrayString()
+vigilanteComasNumericas()
AdvertenciaEditado
#onCreate()
+pulsarSI()
+pulsarNO()
FormatosFecha
+FormatosFecha()
~getFormato()
~getSize()
0..1
1
ListaRegistros
+ListaRegistros()
+ListaRegistros()
+getSize()
+getRegistro()
+addRegistro()
+insertRegistro()
+removeRegistro()
+actualizarRegistro()
1
*
Registro
+Registro()
+Registro()
+getSize()
+getRegistro()
+getArrayRegistro()
+addDato()
+getDato()
Editar
#onCreate()
+irA()
+siguiente()
+anterior()
+ultimo()
+primero()
+guardar()
+cancelar()
+insertar()
+editar()
+anadir()
+eliminar()
<<Interface>>
iEditar
+actualizarRegistro()
+addRegistro()
+borrarRegistro()
+insertarRegistro()
+irAlRegistro()
+irSigRegistro()
+irAntRegistro()
+irUltRegistro()
+irPriRegistro()
<<enumeration>>
TipGuardar
<<enumeration>>
TipoCancelar
<<Interface>>
iExportar
+exportar()
Exportar
#onCreate()
+cancelar()
+exportar()
1
1
gExportar
~guardarMemInterna()
~guardarMenExterna()

Este subsistema cumple la función de modificar y guardar los datos del fichero por parte del usuario.

De esta diagrama destacamos la clase '*Fichero'* y como se componen de al menos de una lista de campos identificadores y de cero o una lista de registros que contiene los datos del fichero. Ésta es la clase más importante del diagrama y va a soportar todas las operaciones que se pueden hacer con el fichero descargado. Es como una gran variable global que van a poder tener acceso el resto de actividades y de clases gestoras.

En este diagrama cada Actividad tiene una clase gestora de uso exclusivo salvo la actividad '*Editar'* pues su gestión es muy simple, y puede ser gestionada desde la misma actividad.

La actividad *Filtro* se encarga, además de presentar el interfaz de usuario, de analizar el campo seleccionado y aplicar el tipo de filtro adecuado de manera automática. Este hecho se muestra en las relaciones de dependencia que tiene sobre las actividades '*FiltroNumero'*, '*FiltroCadena'* y '*FiltroFecha'*. Cada uno de estos Filtros tiene asociado su interfaz y su clase gestora.

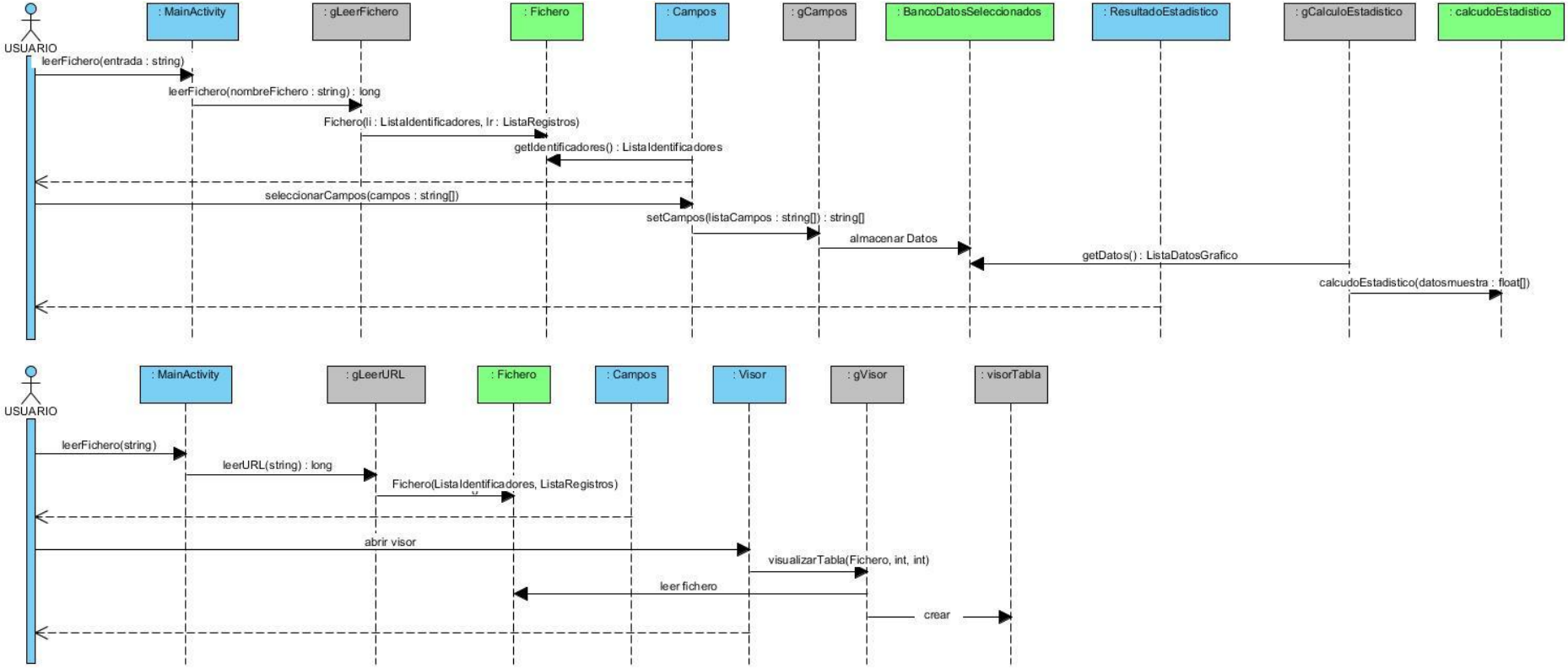
Otro detalle importante del diseño y que simplifica más la codificación evitando cientos de código repetidos es el uso de punteros de función: '*MiFuncion_Numero*' , '*MiFuncion_Cadena*', '*MiFuncion_Fecha*', son interfaces que unidos con el uso de clases anónimas permite ahorrarnos mucho código en la gestión de los filtros. Se explicará con mayor detalle en la sección de Programación.

Un método importante es '*vigilanteComasNumericas*()' de la clase '*MiFunción'* y es el encargado de asegurarse de que todos los datos numéricos del fichero tenga el formato decimal punto y no coma. Esto permite la utilización casi directa de funciones matemáticas para estos datos. Así, por ejemplo, la aplicación se encuentra que un dato numérico es '8,76' lo cambia por '8.76'.

**DIAGRAMAS DE SECUENCIA DE DISEÑO:**

Ahora nos toca definir el diagrama de secuencia del escenario principal de éxito de caso de uso 'CÁLCULOS desde FICHERO' para cuando el usuario ha seleccionado un fichero almacenado en el dispositivo y el mismo tipo de diagrama para el escenario principal de éxito para el caso de uso 'VISUALIZAR DATOS desde INTERNET' cuando el usuario quiere ver el contenido de un fichero descargado de internet.

Como vemos aparecen nuevas clases que no aparecían en los diagramas de secuencia de análisis.


USUARIO
: MainActivity
: gLeerFichero
: Fichero
: Campos
: gCampos
: BancoDatosSeleccionados
: ResultadoEstadistico
: gCalculoEstadistico
: calcudoEstadistico
leerFichero(entrada : string)
leerFichero(nombreFichero : string) : long
Fichero(li : ListaIdentificadores, lr : ListaRegistros)
getIdentificadores() : ListaIdentificadores
seleccionarCampos(campos : string[])
setCampos(listaCampos : string[]) : string[]
almacenar Datos
getDatos() : ListaDatosGrafico
calcudoEstadistico(datosmuestra : float[])
USUARIO
: MainActivity
: gLeerURL
: Fichero
: Campos
: Visor
: gVisor
: visorTabla
leerFichero(string)
leerURL(string) : long
Fichero(ListaIdentificadores, ListaRegistros)
abrir visor
visualizarTabla(Fichero, int, int)
leer fichero
crear

## Programación:

En esta etapa se codifica lo anteriormente diseñado. Se debe de implementar las clases de diseño una a una respetando sus métodos y atributos definidos en la anterior fase. Es una etapa muy técnica donde no se suele interaccionar con el cliente final.

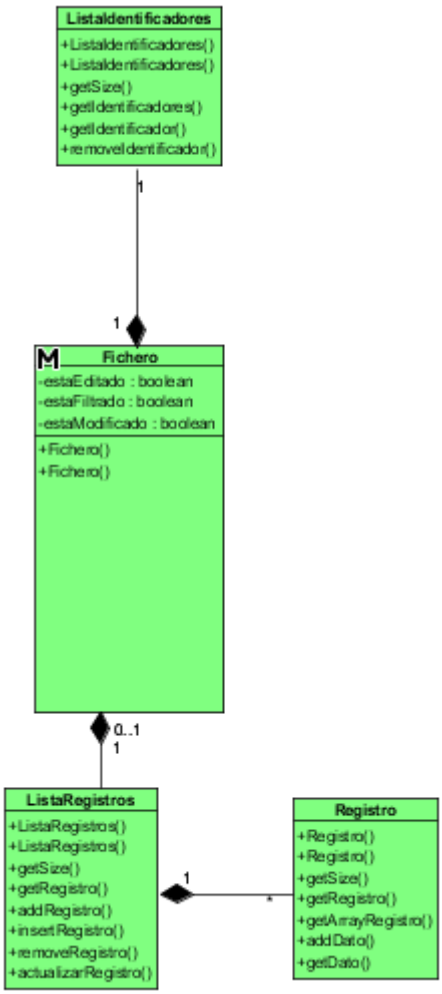
Vamos a detallar solo algunos de los aspectos más importantes que nos hemos encontrado al afrontar las sucesivas etapas de programación de las iteraciones IID. Veremos: El uso de bibliotecas, atributos estáticos y diseños de los atributos estáticos más importantes, uso de punteros de función, herencia de clases abstracta, multiplicidad de implementación de interfaces, conversión de los datos numéricos a formato decimal utilizable de manera directa por Android y rotaciones de la pantalla y similares.

- **Uso de Bibliotecas:**

En nuestra aplicación se usan dos librerías, una para leer correctamente fichero CSV y otra para crear y gestionar gráficos estadísticos especialmente pensados para móviles. La primera se llama Commons CSV de Apache, y la segunda se llama MPandroidchart.

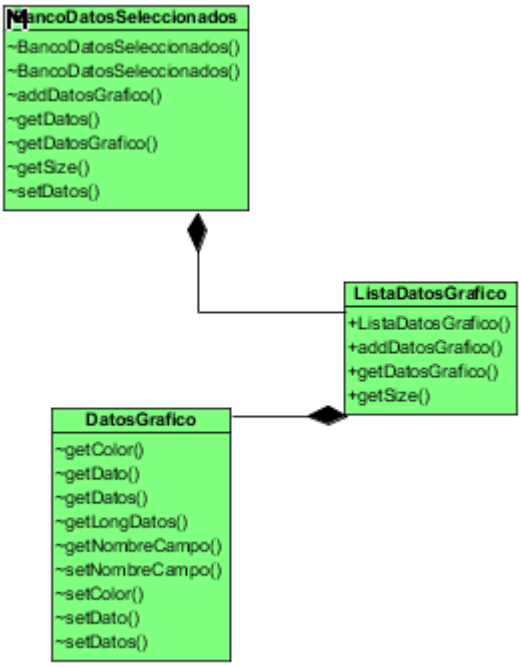
- **Atributos Estáticos:**

Una de las imposiciones de la arquitectura software es que la memoria que crea el S.O. para una actividad es independiente de la que crea para otra actividad, incluso si ambas pertenecen a la misma aplicación. Aunque existen mecanismos de comunicación entre actividades no podemos enviar un objeto de una actividad a otra. Esto nos plantea el problema, de usar atributos y métodos estáticos. El ejemplo más importante es la clase 'Fichero': posee atributos estáticos y en especial los atributos que almacena el fichero son estáticos y privados. Éstos dos atributos son: '*listaIdentificadores'* y '*listaRegistros'*, son en realidad uno '*List <String>'* y otro una '*List <Lista <String>>'*, aunque este detalle queda convenientemente oculto a través de la abstracción (encapsulamiento) de la POO, como se indica en el siguiente parte del diagrama y con todas la ventajas que esto supone de cara a posibles modificaciones futuras:




Aquí vemos que la estructura de almacenamiento del fichero esta convenientemente encapsulada y abstraída por las clases *'ListaRegistros'*, '*Registros'*, '*ListaIdentificadores'*. Como hemos dicho esto supone muchas ventajas de cara al mantenimiento y modificación futura de la aplicación pues permite modificar la estructura del fichero con un mínimo impacto en la aplicación en cuanto a efectos colaterales y con el mínimo esfuerzo.

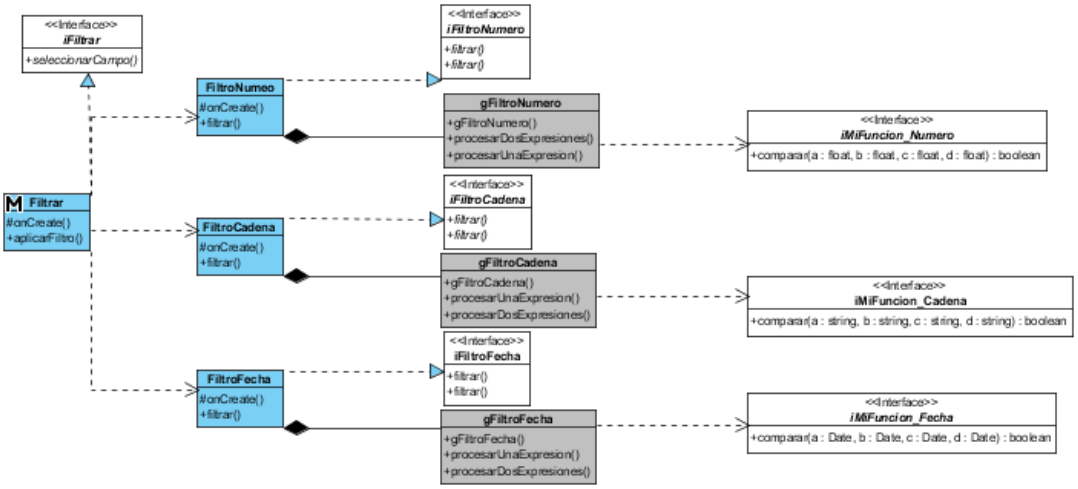
Otra clase muy utilizada y con atributos estáticos, por la misma causa es la clase '*BancoDatosSeleccionados'*, va a ser la encargada de almacenar la lista de muestras que el usuario ha elegido para su cálculo y representación gráfica y se crea cuando el usuario selecciona los campos a tratar de manera estadística:




También convenientemente abstraída y con todas las ventajas que esto supone. En el diagrama de clases de arriba no aparecen los atributos y los métodos privados, pues no se sirven directamente al resto del mundo. Sí podemos contar que la clase '*DatosGrafico'*, que es equivalente a un registro que contiene un atributo tipo '*String'* para almacenar un Identificador de Campo, otro atributo tipo '*array'* de decimales, que contiene la muestra correspondiente y otro atributo de tipo '*Color'*, para almacenar el color con que se dibujará esta muestra en los diferentes gráficos, salvo el de Tarta que tiene un tratamiento diferente. La clase '*ListaDatosGrafico'* encapsula una lista de objetos '*DatosGrafico'* especificado anteriormente, y la clase '*BancoDatosSeleccionados'* usa '*ListaDatosGrafico'* para una mayor abstracción de clases.

- **Punteros de Función:**

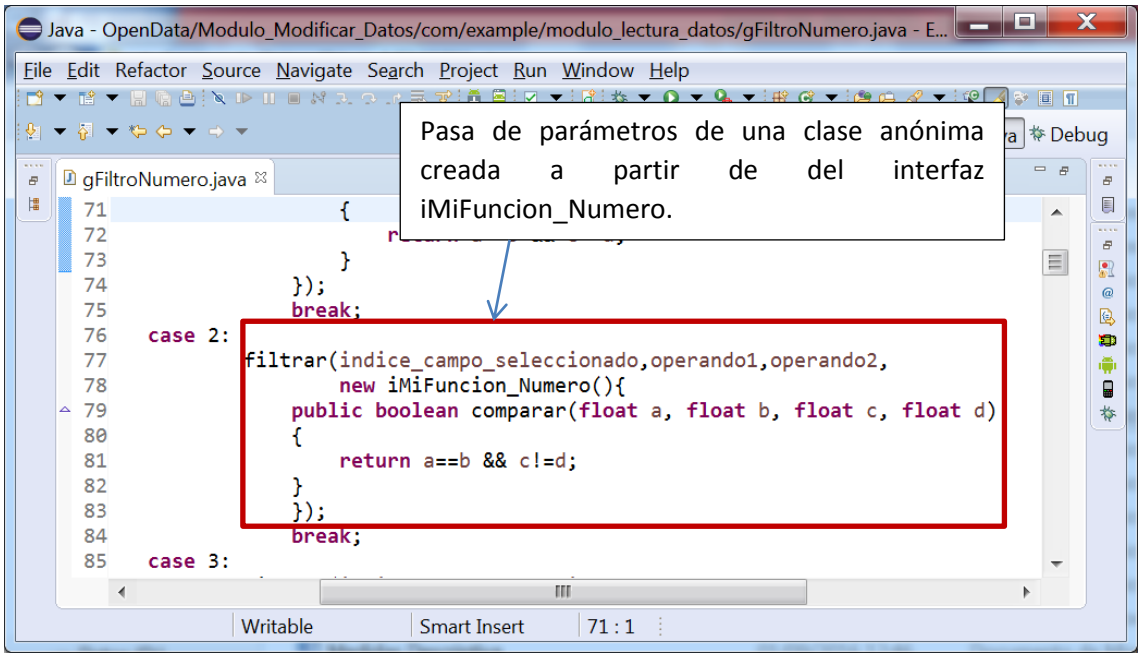
Otro aspecto destacable, desde el punto de vista de la programación, es el uso de punteros de función cuando se gestiona cualquier filtro, ya sea numérico, de cadenas o de fechas. Podemos crear punteros funciones definiendo en un interfaz la función y rescatándola en la creación de diferentes clases anónimas. Esto nos permite ahorrarnos una gran cantidad de código redundante, de difícil localización de errores y modificaciones futuras. El siguiente diagrama muestra estos punteros a funciones:



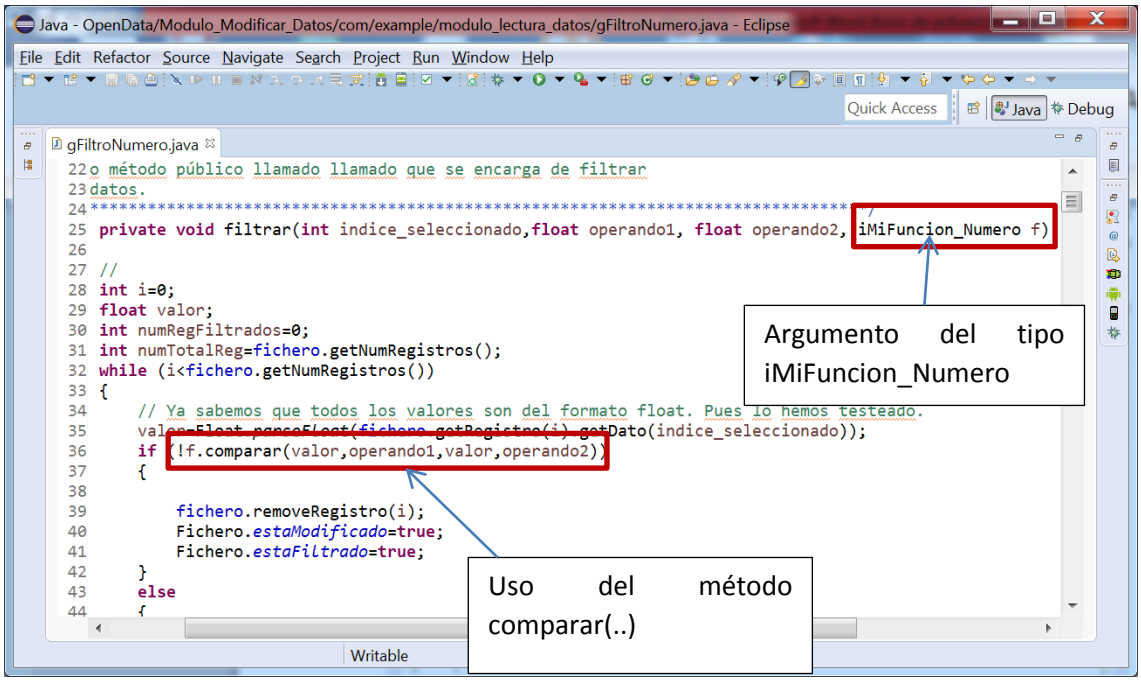

Vemos diagrama muestra estos uso de punteros de función: para decimales es ‘iMiFuncion_Numero’ y su método ‘comparar(float a, float b, float c, float d)’; para String ‘iMiFuncion_Cadena’ y su método ‘comparar (String a, String b, String c, String d)’ y para Fechas ‘iMiFuncion_Fecha’ y su método abstracto ‘comparar (Date a, Date b, Date c, Date d)’, todas ellas devuelve un valor booleano que va a ser determinante a la hora de filtrar datos.

```
public interface iMiFuncion_Numero {
public boolean comparar(float a, float b,float c, float d);
}

public interface iMiFuncion_Cadena {
public boolean comparar(String a, String b,String c, String d);
}

public interface iMiFuncion_Fecha {
      public boolean comparar(Date a, Date b,Date c, Date d);
}
```

Veamos a continuación como usa estos interfaces las clases anónima:




Se define el constructor del interfaz y dentro de éste la implementación de su método abstracto. Con lo que creamos una clase anónima que implementa solo este interfaz. Esto se le pasa como parámetro a un método que espera esta clase anónima y la usa. Vemos en el ejemplo de abajo la función filtrar al que se le pasa un parámetro del tipo *'iMiFuncion_Numero'*. Más abajo en el cuerpo del método filtrar nos encontramos el uso del método comparar de esta clase anónima.

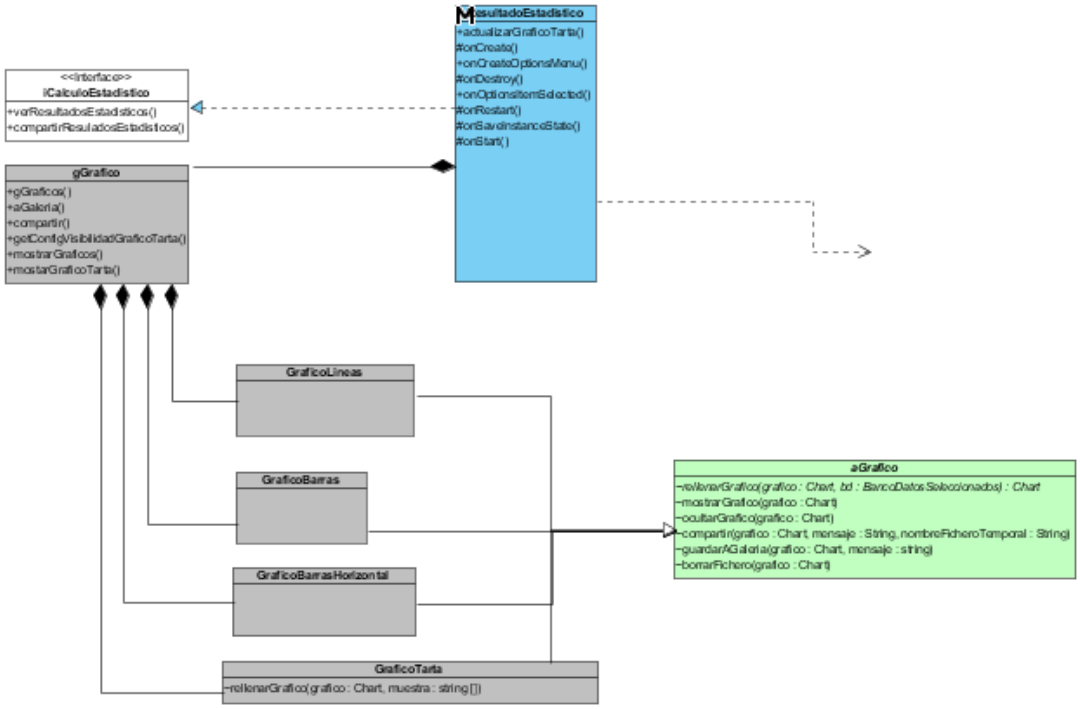



Con lo que solo se codifica una vez este método filtrar.

Sin punteros a función se debería codificar un método filtrar por cada variación de la función que se le aplica para filtrar. Esto nos supuso ahorrarnos en gran cantidad de código redundante.

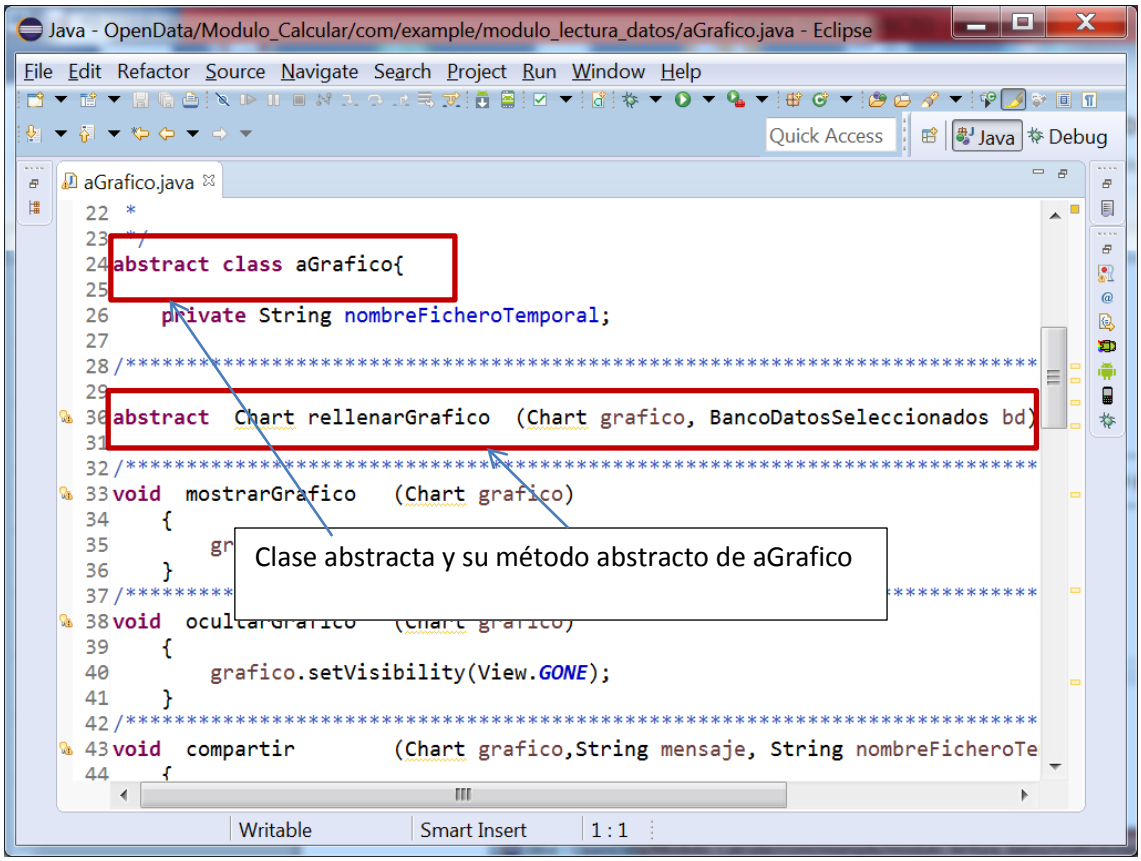
- **Uso de clases Abstractas:**

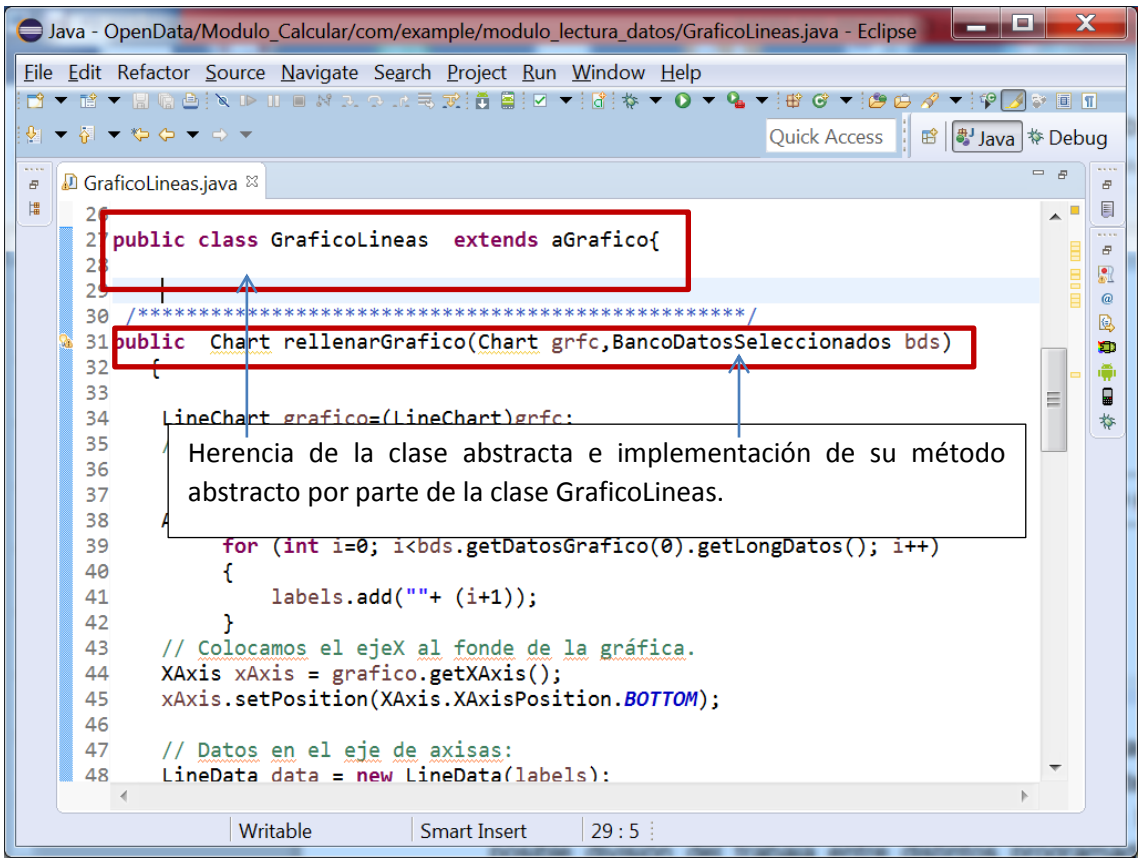
Otro aspecto importante desde el punto de vista del diseño y su posterior codificación es el uso de la clase abstracta '*aGrafico'* que define de manera común las operaciones que se pueden hacer con los diferentes gráficos y deja el método abstracto para rellenar el gráfico de muestras para que lo implemente cada gráfico de manera particular. El siguiente diagrama muestra esta circunstancia:




Vemos en este diagrama que los cuatro gráficos heredan de la clase abstracta '*aGrafico'*. Estos gráficos son gestionados por '*gGrafico'* que resuelve la funcionalidad planteada por la actividad '*ResultadoEstadistico'* encargada de representar el interfaz de usuario principal donde se muestran los resultados gráficos entre otros cálculos. Señalar que el gráfico de tarta funciona de manera diferente y define su propio método '*rellenarGrafico*()', diferente al método abstracto.

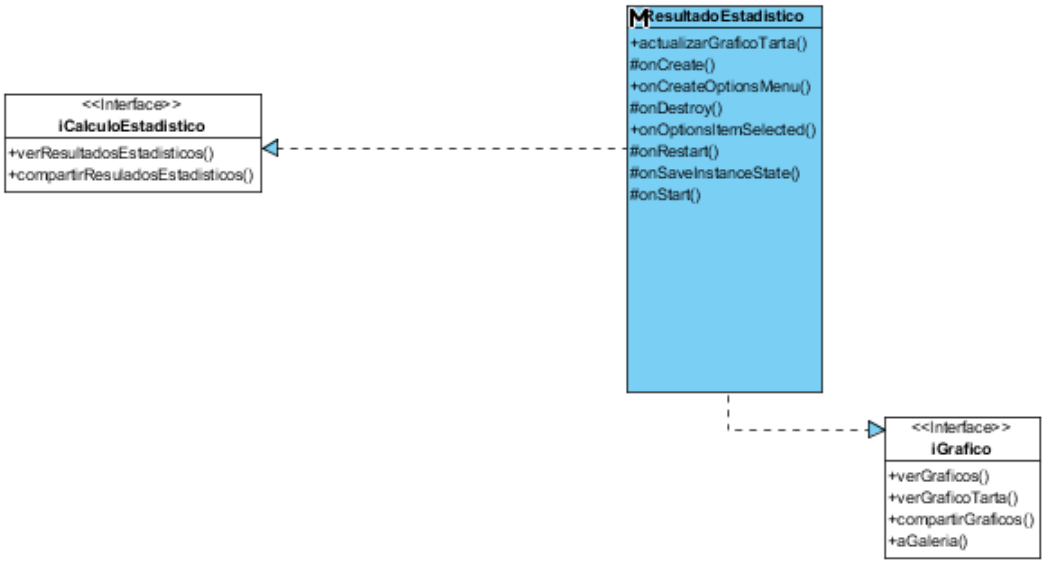
Señalamos aquí el métodos ´'guardarA*Galeria*()' y '*compartir*()' de la clase '*gGrafico*()', estos método están impuestos por la funcionalidad de la librería MPAndroidChart, esta librería sacrifica la posibilidad de seleccionar aisladamente un gráfico a favor de ser interactivo. Los métodos '*guardarAGaleria*()' y '*compartir*()' trabajan sobre los gráficos mostrados actualmente al usuario; el usuario es el responsable de tener en cada momento visibles los gráficos que quiere compartir o mandar a la galería del móvil.

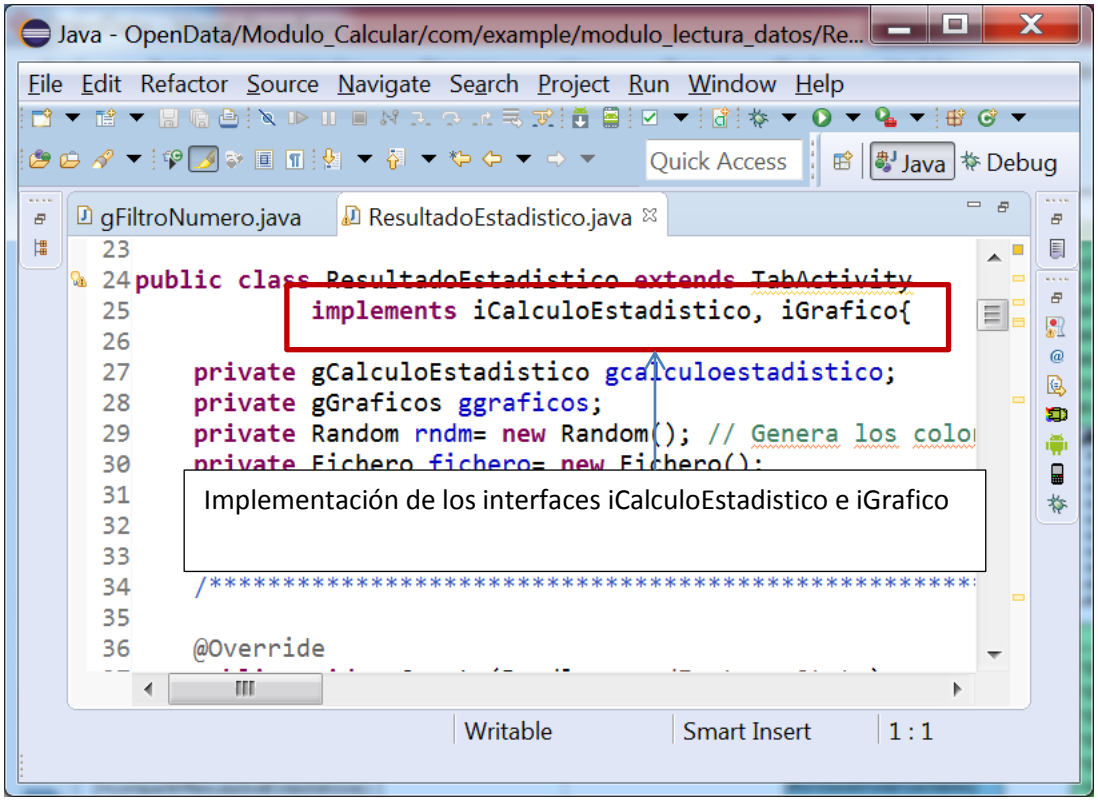

Java - OpenData/Modulo_Calcular/com/example/modulo_lectura_datos/aGrafico.java - Eclipse
File Edit Refactor Source Navigate Search Project Run Window Help
Quick Access
Java
Debug
aGrafico.java
abstract class aGrafico{
private String nombreFicheroTemporal;
abstract Chart rellenarGrafico (Chart grafico, BancoDatosSeleccionados bd)
void mostrarGrafico (Chart grafico)
void compartir (Chart grafico,String mensaje, String nombreFicheroTe
Clase abstracta y su método abstracto de aGrafico
Writable
Smart Insert
1 : 1


Java - OpenData/Modulo_Calcular/com/example/modulo_lectura_datos/GraficoLineas.java - Eclipse
File Edit Refactor Source Navigate Search Project Run Window Help
Quick Access
Java
Debug
GraficoLineas.java
public class GraficoLineas extends aGrafico{
public Chart rellenarGrafico(Chart grfc,BancoDatosSeleccionados bds)
for (int i=0; i<bds.getDatosGrafico(0).getLongDatos(); i++)
labels.add(""+ (i+1));
// Colocamos el ejeX al fonde de la gráfica.
XAxis xAxis = grafico.getXAxis();
xAxis.setPosition(XAxis.XAxisPosition.BOTTOM);
// Datos en el eje de axisas:
Herencia de la clase abstracta e implementación de su método abstracto por parte de la clase GraficoLineas.
Writable
Smart Insert
29 : 5

- **Implementación de Varios Interfaces:**

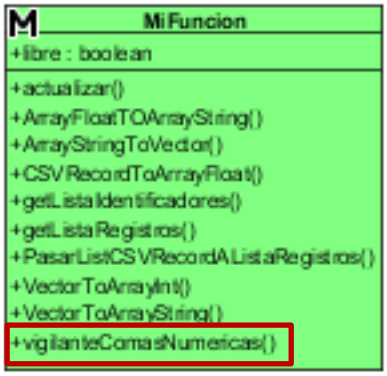
Otro aspecto de la programación es la posibilidad de que una clase puede implementar más de un interfaz. Esto es lo que le ocurre a la actividad '*ResultadoEstadistico'*. A continuación vemos las secciones más significativas de su diagrama de clases y de su codificación:



- **Conversión interna en formato decimal android:**

Esto consiste en convertir todo lo leído por la aplicación a formato decimal android (o java). Así si editamos o leemos un fichero que tiene los números decimales con coma, éste es automáticamente convertido al formato interno de android, que es de número decimales convertido por punto. Esta conversión es necesaria y ocurre constantemente en el programa como al pasar los datos del fichero a memoria, al editar un registro, etc. La raíz de esta funcionalidad es el método '*vigilanteComasNuméricas*()' de la clase '*MiFuncion'*.




- **Rotaciones de la Pantalla:**

Cuando rotamos la pantalla del dispositivo o consultamos un menú entre otras acciones, la actividad se reinicializa. Tuvimos que retocar algunas actividades para que funcionaran correctamente. En muchos casos tuvimos que sobrescribir los métodos '*onSaveInstanceState*()' y '*onRestoresInstanceState*()' para guardar y recuperar el estado anterior de la actividad para esta clase de eventos. En algún caso tuvimos que retocar el método '*onCreate*()'. El uso de argumentos en estos métodos lo permite. Utilizando un objeto '*Bundle'* para guardar o recuperar estos datos.

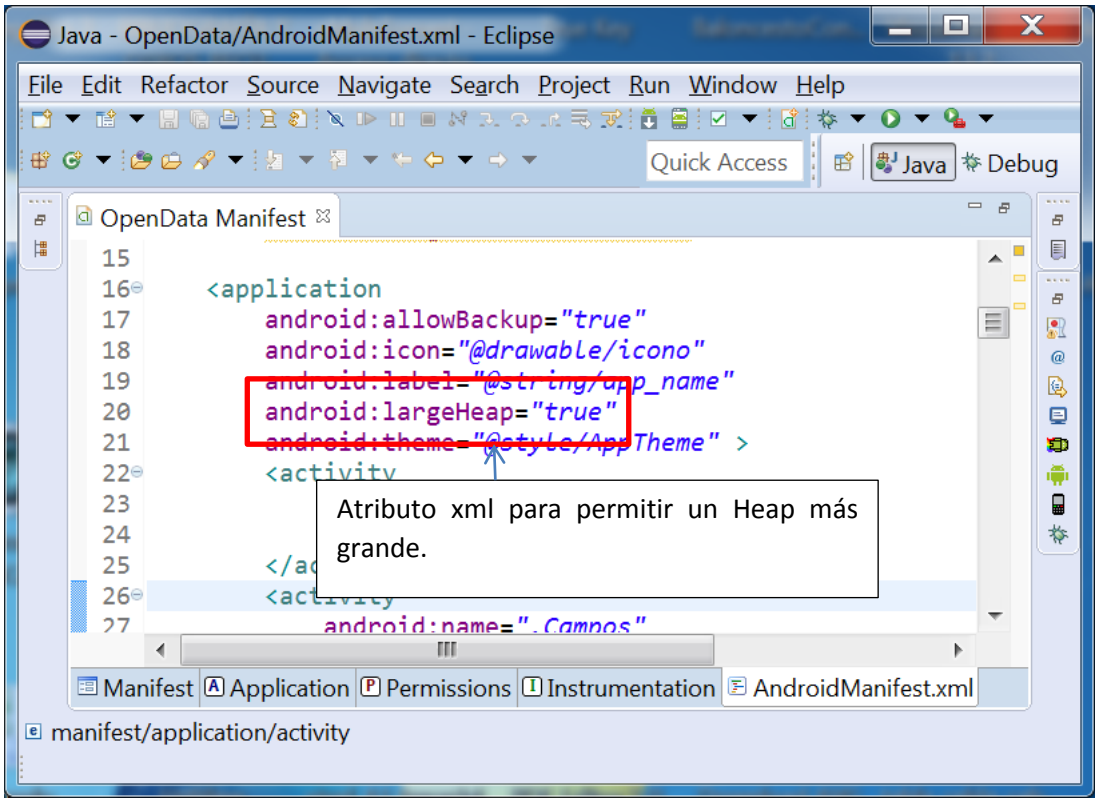
## Pruebas:

La fase final de cada iteración y la última que da pié al producto final es la fase de Pruebas. Ha pasado muchas veces que había fallos en la codificación y se resolvieron adecuadamente. Lo que vamos a citar aquí son los aspectos y situaciones más importantes que nos hemos encontrado en esta fase.

- **Memoria Rebasada**

Se observa que el software funciona bien para cuando tiene que descargar ficheros .csv relativamente pequeños o mejor dicho no muy grandes. Estos ficheros son pasados a memoria a una estructura de Vector de Vectores de 'String'. Estos objetos son almacenados en la parte de memoria 'Heap' (Montículo) y este 'Heap' tiene un tamaño máximo fijado en tiempo de compilación, si sobrepasa este tamaño la aplicación falla y deja de funcionar. He probado descargando ficheros del Ayuntamiento de Málaga como el dedicado a la Lecturas de cuadros electrónicos de Mayo 2016 con un peso de 13,8 MB y al intentar pasar estos datos a memoria, quedan rebasados la capacidad de memoria del montículo y el programa no sigue, lanza una error llamada 'outofMemoryError'. También nos falla en los ficheros de líneas y horarios de bus Horarios que ocupan 16,5 MB.

Para intentar paliar este defecto podemos definir que tenga una cabecera más grande de lo normal dentro de los atributos XML de la aplicación: '*android:largeHeap="true"*'.




El error 'outOfMemoryError' hace que el programa se detenga y es debido a que la máquina virtual Dalvik llena completamente la parte de memoria Heap (Montículo) que le asigna a la aplicación. No hay una buena o mala solución para esta error, debido a que no se garantiza que el estado de la máquina virtual vuelta a un estado estable. Algunos programadores la capturan pero depende mucho de lo que se quiera hacer pues como he dicho no se garantiza que podemos llevar al programa a un buen estado tras capturar este error. En mi aplicación ocurre este error por dos causas, principalmente: Una cuando se trata de abrir un fichero relativamente

grande (por ejemplo uno de 10MB), y otro cuando se llama al visor de fichero csv y el fichero correspondiente tiene una gran cantidad de registros para visualizar, esto hace que se gaste muchos recursos en visualizarlos todos a la vez en una tabla.

Para solucionar o paliar la primera causa del error decidimos que el usuario pudiese especificar el máximo tamaño del fichero que se desea cargar. Si es un tamaño relativamente pequeño no tendrá problema y la aplicación podrá mostrarlo. Es el usuario el responsable de administrar este tamaño y ajustarlo a las demanda de memoria de su dispositivo móvil. Para ver cómo se hace se puede ir al Manual de Usuario incluido en esta memoria como anexo.

La segunda causa de error lo intentamos solucionar de la misma manera que el primero, es decir, dejando al usuario que pueda especificar el máximo de registros del fichero que se puede visualizar de una vez. Con ello no renunciamos que queramos ver el fichero entero sino que aceptamos que esto puede dar problemas y decidimos visualizarlo por tramos. Esto también viene explicado en el manual de usuario.

- **Tabla de caracteres usada en la aplicación:**

Para leer los ficheros se utiliza el código UTF-8 que es el más internacional que existe y cubre por ejemplo las tildes y la 'ñ'. Hay algunos ficheros que usan un formato distinto y hace que, por ejemplo, las tildes se lean mal. Esto, sin embargo, no es un gran problema pues los programas más populares como Excel pueden tener el mismo problema.

- **Menú Contextuales a los Gráficos:**

Pensemos en ponerles menús contextuales a los gráficos pero no encontremos nada, ningún ejemplo que fuese la prueba de que se puede hacer. Al contrario nos informemos por internet y este es un tema que no se aborda en la actual versión de la biblioteca MPAndroidChart. Parece ser que no te deja pues el gráfico puede ser zoombeado con los dedos y hacer que capturase un evento de pulsación larga entorpecería estas acciones que podemos hacer con estos gráficos interactivos. Quizás en futuras versiones se pueda hacer pero actualmente todo indica a que no se puede hacer.

- **Gráficos: leyendas.**

La biblioteca MPAndroidChart permite un número ilimitados de leyendas que se corresponde con cada campo numérico seleccionado. Por defecto, las recortar si son muchas, aunque dejas la opción de indicarle a la biblioteca que no las recorte, sino que crezca con una nueva línea cuando sea necesario, pero esta última opción tiene un gran inconveniente y es que al crear una nueva línea, disminuye en la

misma cantidad de líneas el recuadro que visualiza el gráfico en sí (las líneas o las barras). Mi opción fue dejarlo por defecto y asumir que no comparemos demasiadas muestras a la vez.

- **Distintos Tamaños de Pantalla:**

Eclipse nos proporciona una vista donde podemos ver la mayoría de los tamaños de pantalla existente en el mercado, incluso para las tabletas. La aplicación desarrollada ha sido probada, en el simulador, en diversos tamaños de pantalla. Las distintas vistas de la aplicación son elementales y no han dado casi ningún problema en visualizarlas en distintos tamaños de pantalla. Tuvimos que retocar la actividad inicial donde ser cargan los datos, pues en la pantallas de 2.5' no se veían bien y en la opción edición de datos cargados tuvimos que poner el espacio que ocupan los distintos '*layouts'* en porcentaje de ocupación en pantalla.

- **Distintas Resoluciones de Pantalla:**

Al no tener, prácticamente ninguna imagen, no tuve que realizar ningún cambio para que se adaptase a las distintas resoluciones de pantalla más comunes que nos podemos encontrar.

# CONCLUSIONES y POSIBLES MEJORAS:

En esta sección vamos a explicar las conclusiones llegadas al finalizar la aplicación y indicaremos las posibles mejoras que puede hacerse.

Actualmente existen infinidad de portales de las administraciones públicas que ofrecen datos abiertos en formato ‘csv’, susceptibles al trato estadístico y se espera que este movimiento crezca más. Nuestra herramienta trata de abarcar a todos los ficheros ‘csv’ y no se especializa en ninguna en concreto. Es cierto que no es fácil encontrar un fichero ideal que maximice los que nuestra aplicación hace, pero cualquier persona de cualquier país, en un momento determinado puede demandar una aplicación como la nuestra. Fácil de manejar, eficiente con relación a los grandes tamaños de datos. Estos fichero ‘csv’ tiene una estructura de tabla, ideal para su trato estadístico lo que potencialmente facilita enormemente su análisis y compresión. Nuestra aplicación no es un programa tan difundido con los de Microsoft pero rebaza y supera a muchos programas estadísticos que hay en el mercado, en cuanto al trato de ficheros ‘csv y además le añade un fuerte componente de comunicación al usuario al desenvolverse en el ambiente de los teléfonos móviles.

La aplicación Open Data debe de cuidar el tamaño de los datos a analizar. Si el fichero ‘csv’ es demasiado grande, la aplicación podría no funcionar por falta de memoria. Como solución se deja al usuario especificar el máximo tamaño de fichero a utilizar y la máxima ventana de visualización para cualquier fichero.

Hemos tenidos que utilizar, durante el desarrollo del producto, atributos privados estáticos (equivalentes a variables globales) de las clases de diseño. Y esto no es deseable desde el punto de vista de la programación. Esto es debido porque Android no permite comunicar un objeto entre dos actividades de la misma aplicación. Y como consecuencia de esto, hemos hecho que la aplicación no permita instanciarse más de una vez en el móvil, pues entramos en los efectos colaterales del uso de atributos estáticos.

En cuanto a las posibles mejoras, es de esperar que la evolución de los móviles y los aumentos de sus prestaciones, además de capacidades determinen las mejoras futuras. Y también la opinión de los futuros usuarios y su espíritu de colaboración nos aportarán más ideas para agregárselas a nuestra aplicación.

Es de esperar que las capacidades físicas de los dispositivos móviles crezcan de manera que las restricciones de memoria que conllevan el uso de ficheros ‘csv’ de gran tamaño se vean superadas. Con lo que en futuras versiones de Android este programa funcionará prácticamente sin límites en cuanto al consumo de recursos.

En cuanto a la imposibilidad de comunicarse objetos genéricos entre dos actividades, espero que esto se solucione en futuras versiones de Android. Y en el

futuro contemplar la posibilidad de modificar la aplicación para que pueda soportar más de dos instancias de la misma.

# BIBLIOGRAFÍA:


[1] Concepto de Datos Abiertos:
http://es.wikipedia.org/wiki/Datos_abiertos
[2] Portal de Datos Abiertos de la Junta de Andalucía:
http://www.juntadeandalucia.es/datosabiertos/portal.html
[3] Portal de Datos Abiertos del Ayuntamiento de Málaga:
http://datosabiertos.malaga.eu/
[4] Portal de Datos Abiertos del Estado Español:
http://datos.gob.es/
[5] Sitio Oficial para Desarrolladores Android:
http://developer.android.com
[6] Libro 'El gran libro de Android' de Jesús Tomás Gironés y su sitio web asociado:
http://www.androidcurso.com
[7] Foro de Dudas de Programación.
http://stackoverflow.com/
[8] Libro 'El gran libro de Android' Avanzado. Autores varios.

[9] Libro 'Android 4: desarrollo profesional' de McCraken Scott.

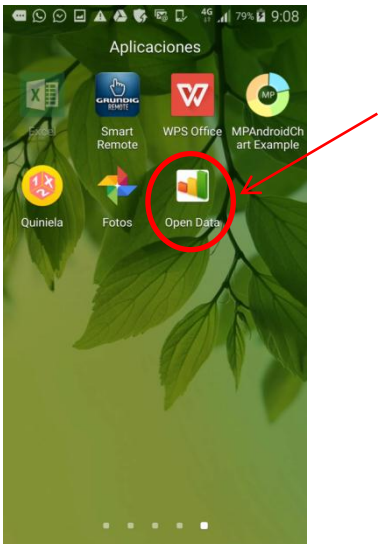
[10] Sitio web de la biblioteca Common CSV.
https://commons.apache.org/proper/commons-csv/
[11] MPAndroidChart (Biblioteca para Gráficos Estadísticos para Android)
https://github.com/PhilJay/MPAndroidChart

# ANEXO A: MANUAL DE USUARIO



## 1. INTRODUCCIÓN A LA APLICACIÓN:

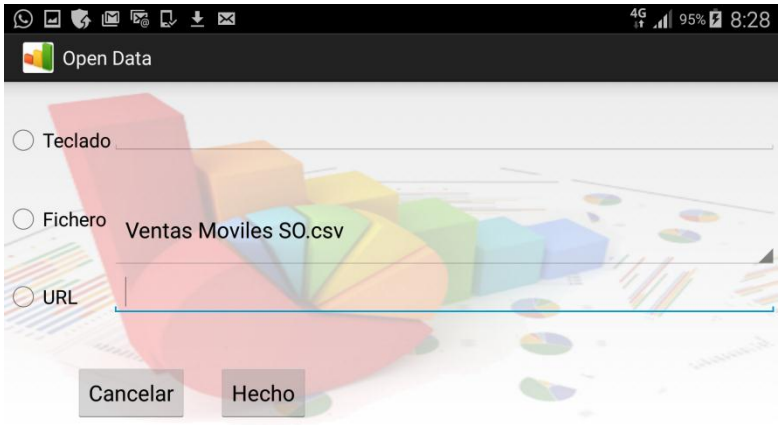
Nuestra aplicación se llama Open Data, y tiene el icono asociado que se muestra en la imagen siguiente:




Al pulsar sobre el icono, lo primero que nos encontramos es la pantalla de presentación: esta tiene un gráfico y un botón 'Continuar' que sirve adentrarnos al interior de la aplicación.

## 2. CARGA DE LOS DATOS DE ENTRADA Y NAVEGADOR PERSONALIZADO.

La pantalla siguiente nos muestra un interfaz pidiéndonos una de las tres formas para cargar datos numéricos en formato CSV.




Estos datos deben de cumplir los requisitos exigidos para ser tratados adecuadamente por la aplicación: es decir en un documento en formato abierto sencillo en forma de tabla, en las que las columnas se separan por comas (o punto y coma donde el separador es el Separador decimal: Argentina, España, Brasil…) y las filas por salto de línea. Los campos que contengan una coma, un salto de línea o una comilla doble deben de ser encerrados por comillas dobles.

Como vemos en la imagen podemos seleccionar solo una opción de las tres alternativas: Teclado, Fichero o URL. Veamos a continuación las tres opciones:

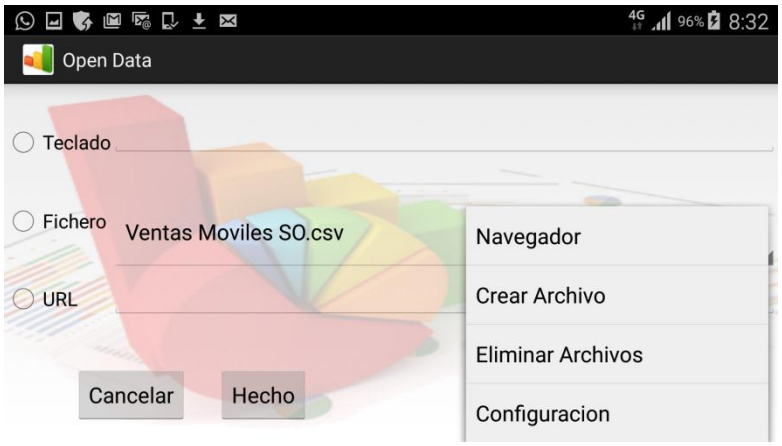
- **URL:**

Comencemos con la opción más interesante, la opción URL: permite descargar directamente un fichero CSV desde su dirección web de descarga. Tenemos varias opciones para introducir la dirección URL:

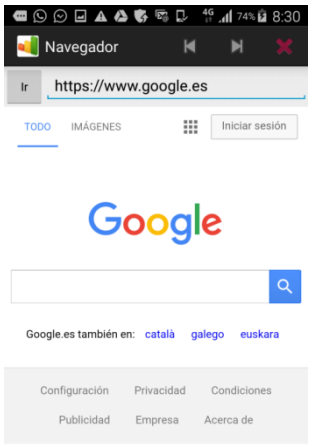
- ✓ **Introducir directamente la dirección por el usuario:** Se trata como su propio nombre indica de escribir directamente la dirección URL en la línea de texto.
- ✓ **Hacer un Copiar y Pegar de la URL:** Como su propio nombre indica es copiar en el portapapeles la URL previamente visualizada con cualquier navegador comercial y seguidamente pegarla en esta línea de texto.

- ✓ **Seleccionar una URL del Historial:** cuando empezamos a escribir en esta línea de texto aparecerán emergentemente las direcciones URL que se hayan cargado previamente, con los que podemos ahorrarnos tener que repetir urls.

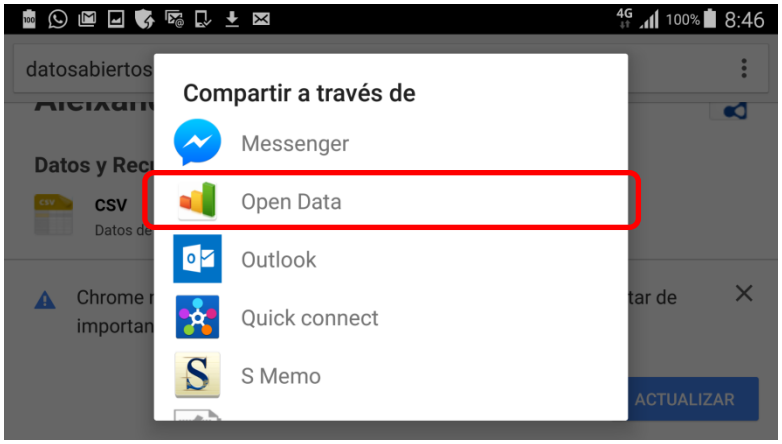
- ✓ **Utilizar el Navegador Personalizado:** Consiste en usar el menú asociado a esta actividad y arrancar un navegador personalizado. Este navegador tiene la característica principal de que si pulsamos en un enlace correspondiente a la descarga de un fichero CSV, nos arranca la aplicación Open Data con su correspondiente línea de texto rellena con esta URL pulsada por el usuario en este navegador personalizado. En la imagen siguiente se ve el menú asociado a esta actividad, podemos distinguir la opción Navegador, que permite arranca a este Navegador Personalizado.




A continuación describimos el Navegador Personalizado visualizando la web: www.google.es. Posee en la parte superior el logo de nuestra aplicación Open Data, un título y tres controles para ir a la página anterior, otro para ir a la página siguiente y otro para interrumpir la carga de la web. En la siguiente línea, bajo esta parte superior, se presenta una barra de direcciones con historial que nos permite introducir cualquier URL y cargarla pulsando el botón Ir. Además este Navegador posee un indicador del estado de cargar y como hemos dicho antes, cuando pulsamos en un enlace asociado a una URL de descarga de un fichero CSV, arranca la actividad principal proponiendo este enlace en la línea de texto URL. Lo que hace la vida más fácil a la hora de introducir cualquier URL en la aplicación.




- ✓ **Compartir Página desde cualquier Navegador del Mercado:** Esta es la opción ideal y consiste en navegar normalmente en cualquier navegador comercial y cuando estemos en una página web cercana al enlace de descarga CSV deseado, compartamos esta web a nuestra aplicación Open Data que responderá arrancando su Navegador personalizado. En el que podemos continuar nuestra exploración de la web y cuando pulsemos sobre un enlace de descarga CSV, arranque la aplicación Open Data, proponiendo en la línea de texto la URL, este enlace CSV.




En la imagen anterior aparece indicado como aparece la aplicación Open Data al compartir la página web en el navegador comercial Chrome. Solo tenemos que seleccionar esta opción y se nos mostrará el Navegador Personalizado de la aplicación Open Data.

El fichero será analizado y tratado adecuadamente. La opción URL se realiza en 'Background' pero el interfaz de usuario no tiene más remedio que esperar a que termine pues eso datos son fundamentales para la secuencia del programa. La opción tiene efecto al pulsar el botón Hecho.

- ✓ **Descargar el Fichero:** Esta es la forma más simple de cargar un fichero CSV de internet. Se trata de descargar, como se hace normalmente, el fichero CSV con cualquier navegador del mercado. La aplicación está hecha para que este fichero pueda ser leído desde la opción 'FICHERO' cuando está configurado para leer la memoria externa.

- **FICHERO:**

La siguiente opción interesante es la de fichero, pensada para poder trabajar offline. Se trata de mostrar una lista desplegable con todos los ficheros CSV guardados por la aplicación. Si seleccionamos uno y pulsamos el botón Hecho cargaremos el correspondiente fichero en la aplicación, con lo que se realizará un análisis muy similar a la anterior opción de URL pero en este caso no accede a la red de internet. Al igual que el anterior caso el IU no le queda más remedio que esperar a que termine el análisis.

Podemos cargar los ficheros desde el almacenamiento interno que asigna Android a la nuestra aplicación o bien optar por cargar los ficheros situados en donde Android descarga por defecto cualquier fichero, que llamaremos almacenamiento externo.

- **TECLADO**:

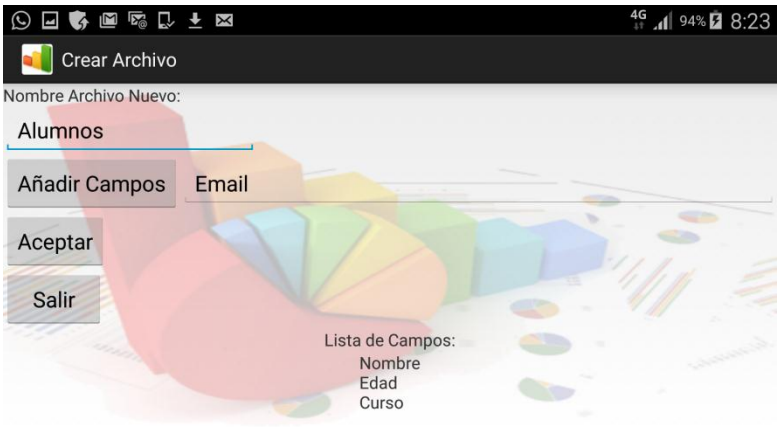
Consiste en que el usuario introduzca directamente los datos sobre esta línea de texto. Estos datos pueden separarse por coma o punto y coma, según lo configure el usuario. Esta opción no pasa por el mismo camino que las anteriores, aunque también se analiza de forma similar. Se pedirá un identificador relacionado a estos datos introducidos para darles un significado a los mismos.

# 3. GESTIÓN DE ARCHIVOS

En la pantalla de carga podemos hacer una pequeña y simplificada gestión de archivos. Así podemos eliminar uno previamente almacenado en memoria interna o externa o crear uno nuevo. A continuación veremos estas opciones:

## 3.1. Creación de un Archivo Nuevo:

Para crear un archivo nuevo debemos de darle un nombre y crear al menos un identificador de campo, aunque podremos crear cuantos queramos. Este paso es muy importante ya que va a definir toda la estructura del fichero CSV y debemos de ser muy exactos a la hora de crear estos identificadores de campo pues no podremos modificarlos posteriormente. A continuación vemos un ejemplo ilustrativo:

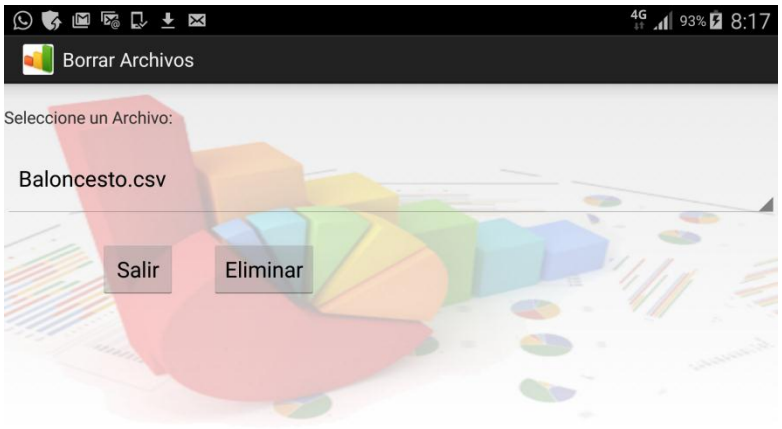



En la imagen anterior vemos una captura de pantalla de esta opción. En primer lugar aparece el nombre que queremos darle al fichero, en este caso concreto se llamará ‘Alumnos’ y no hará falta introducirle la extensión ‘.csv’, pues se hace automáticamente. En la parte inferior aparece unas etiquetas que van modificándose conforme vamos añadiendo más campos al fichero ‘Alumnos’, en este caso se han creado los campos ‘Nombre’, ‘Edad’ y ‘Curso’ y está a punto de crearse el campo ‘Email’, para ello debemos pulsar el botón ‘Añadir Campos’ y se añadirá automáticamente a la lista anterior. El botón Aceptar confirma todos los cambios y el Cancelar retrocede y finaliza esta pantalla. El fichero nuevo se guardará en memoria interna o externa según la configuración general.

Una vez creada el archivo CSV con la estructura básica de los campos, podemos añadirles los registros que queramos, editando éste. Para editarlos primero creamos el fichero solo con los campos y luego lo cargamos en la aplicación para editarlos posteriormente (La opción Edición se explica en su correspondiente apartado).

### 3.2. Eliminar Archivos:

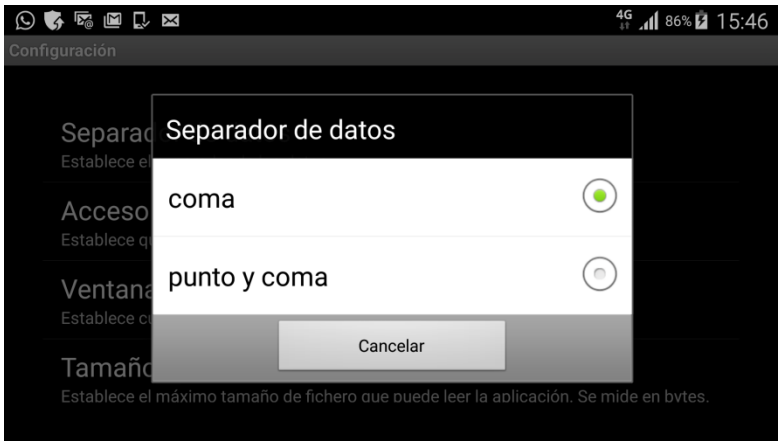
Para eliminar un archivo solo debemos de seleccionarlo de una lista desplegable y pulsar el correspondiente botón. Se podrán eliminar tanto los archivos almacenados en memoria interna como los almacenados en memoria externa según lo tengo configurado el usuario en ese momento. A continuación vemos una captura de pantalla de esta opción:




## 4. CONFIGURACIÓN GENERAL:

En esta sección describimos una serie de parámetros que sirven para configurar la aplicación. Se carga desde la actividad que pide la entrada para carga de datos, seleccionando del menú la opción 'Configuración'. Distinguimos cuatro secciones:

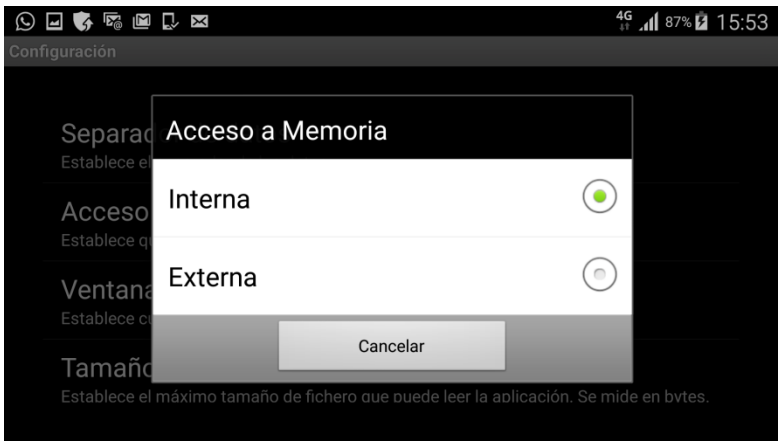
- **Separador de datos**: Es una opción fundamental para poder cargar datos de manera correcta. Existen dos alternativas, valores separado por coma o valores separados por punto y coma. Por defecto el valor es por coma. Estas opciones reflejan la definición de CSV que hemos explicado en esta memoria. Antes de cargar los datos deberíamos saber de antemano cómo están separados entre ellos. En la imagen siguiente vemos su cuadro de diálogo:



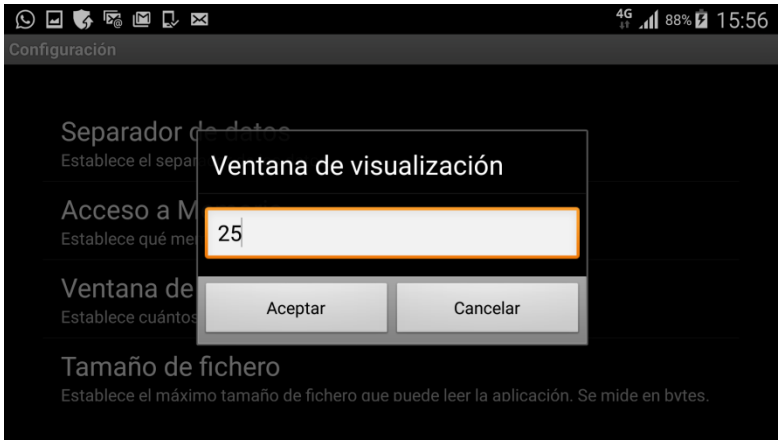

- **Acceso a Memoria**: Nos permite definir de dónde se leerán los archivos almacenados en el dispositivo, desde memoria interna o desde memoria externa. Los archivos almacenados en la memoria interna son los que crean la aplicación ella misma, sólo son accesibles por la misma aplicación y se eliminan al desinstalarla. Los archivos almacenados en memoria externa, también los crea la aplicación, pero en este caso pueden ser accedidos por cualquier otra aplicación y no se eliminan cuando se desístale. La zona

donde se ha definido esta memoria externa corresponde a la carpeta que Android usa por defecto para descargar cualquier archivo, con lo que se podría descargar un fichero CSV por otra aplicación y ser visible esta descarga por Open Data a través de su memoria externa.

Según hayamos definidos el uso de una u otra tipo de memoria, cuando leamos, creamos o eliminemos cualquier archivo, se realizará en la memoria en uso que hayamos definido en este opción.



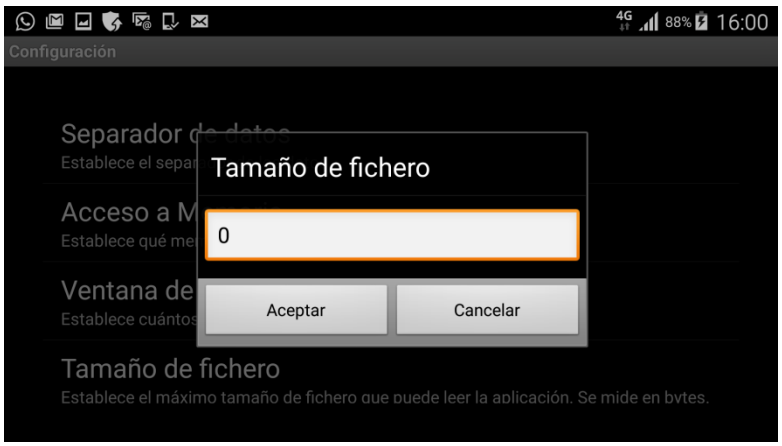

- **Ventana de visualización**: Esta opción se usa para indicar al visor de archivos cuantos registros nos muestra de una vez como máximo. Por efecto es 25 aunque puede ser cualquier número entero positivo. Si introducimos el valor 0 la aplicación intentará visualizar todos los registros del archivo a la vez.




- **Tamaño de Fichero**: Indica el máximo Tamaño del fichero, en bytes, que la aplicación está dispuesta a soportar. Su razón de existir es evitar en la medida de lo posible que la aplicación se comporte de manera ineficiente o se pare debido al gran tamaño de fichero de carga. Un cero como valor (que es su valor por defecto), indica que se intentará cargar cualquier fichero

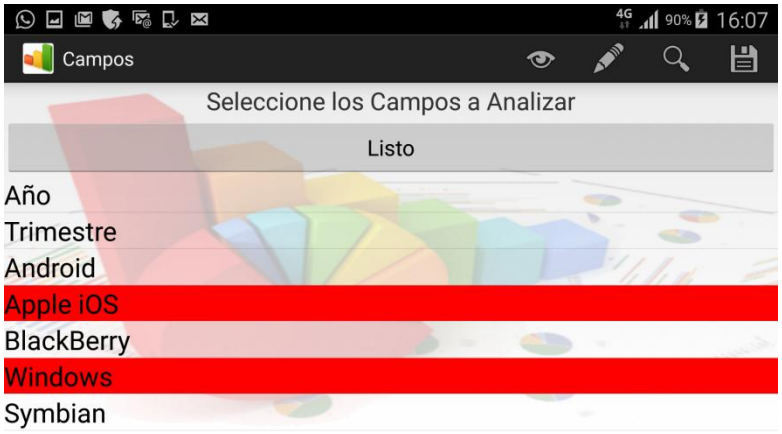
independientemente de su tamaño. Si no se puede averiguar el tamaño del fichero (como ocurre en algunas URLS) se intentará cargar sin pasar por esta barrera.

En el caso que no se admita el fichero, nos saldrá un mensaje indicando esta situación.




## 5. SELECCIÓN DE CAMPOS A MUESTREAR:

Una vez seleccionado el Archivo a analizar ya sea desde la memoria del dispositivo o desde internet, pulsamos el botón 'Hecho' y pasamos a una nueva actividad encargada de recoger los campos seleccionados por el usuario que quiere analizar. Estos campos se corresponden exactamente con las cabeceras del archivo CSV. En la imagen siguiente vemos un ejemplo:




En este ejemplo se muestra siete campos de los cuales seleccionamos dos (Apple iOS y Windows) marcados en rojo. Como vemos podemos seleccionar más de un campo al mismo tiempo con objeto de poder comparar sus muestras asociadas. Si un campo no es numérico no es muestreable y no se puede seleccionar (se indicaría con el oportuno mensaje de advertencia). Una vez que

decidamos los campos a muestrear pulsamos el botón ‘Listo’ para ver los resultados de lo seleccionado.

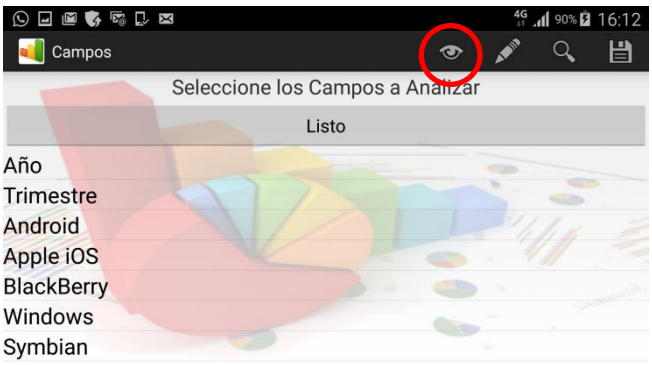
En este pantalla también se puede visualizar el contenido actual del fichero, editarlo, filtrarlo y guardarlo en memoria. Es decir podemos manipular los datos cargados en memoria principal antes de ser tratados estadísticamente.

Si el fichero asociado no tienes campos muestreables, el fichero no puede ser tratado por nuestra aplicación, sí podríamos manipular sus datos pero no podemos calcular ningún resultado.

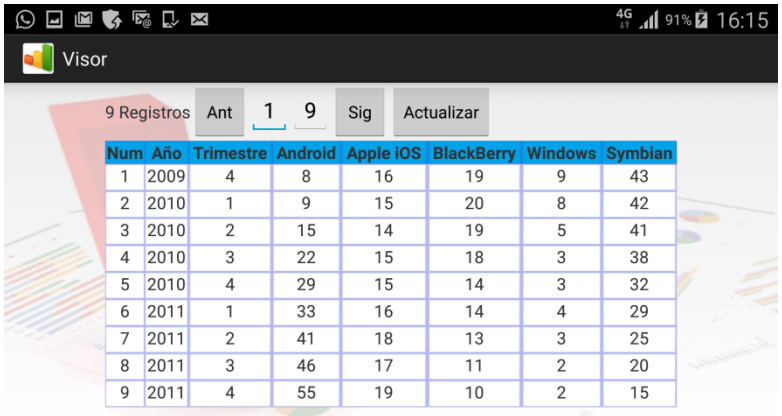
Cabe mencionar que esta opción no es aplicable cuando los datos son introducidos directamente por teclado.

## 6. VISOR DE DATOS:

Esta opción está disponible cuando estamos en la pantalla para seleccionar el/los campo/s a muestrear. Aparece en el menú como una acción directa con el icono de un ojo:




Nos aparece un IU como en la siguiente imagen:



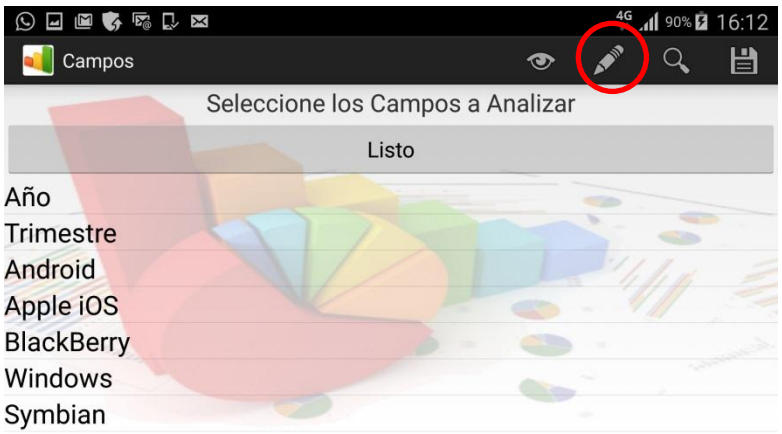

| Num | Año | Trimestre | Android | Apple iOS | BlackBerry | Windows | Symbian |
|---|---|---|---|---|---|---|---|
| 1 | 2009 | 4 | 8 | 16 | 19 | 9 | 43 |
| 2 | 2010 | 1 | 9 | 15 | 20 | 8 | 42 |
| 3 | 2010 | 2 | 15 | 14 | 19 | 5 | 41 |
| 4 | 2010 | 3 | 22 | 15 | 18 | 3 | 38 |
| 5 | 2010 | 4 | 29 | 15 | 14 | 3 | 32 |
| 6 | 2011 | 1 | 33 | 16 | 14 | 4 | 29 |
| 7 | 2011 | 2 | 41 | 18 | 13 | 3 | 25 |
| 8 | 2011 | 3 | 46 | 17 | 11 | 2 | 20 |
| 9 | 2011 | 4 | 55 | 19 | 10 | 2 | 15 |

Aparece una tabla con los primeros registros del archivo. Podemos desplazarnos tanto horizontalmente como verticalmente en esta pantalla. En la parte superior se encuentra los controles.

Se visualiza el número de registros al mismo tiempo que hemos definido en la configuración (en este ejemplo se visualizar 9 registros, que son todos los del archivo). En la parte superior se indica el número total de registros del archivo, tres botones y dos líneas de texto que sirven para navegar por el archivo. Podemos ir a saltos, a través de los botones 'Ant' y 'Sig' o directamente colocando el rango de registros que deseamos ver en las líneas de texto y pulsando el botón 'Actualizar'. En este último caso la lista se recalcula teniendo en cuenta el máximo de registros a visualizar en cada momento y el rango colocado en la líneas de texto numérico. Si el rango es igual o menor a éste máximo, entonces el rango se muestra correctamente pero si es mayor, entonces se recorta a esté máximo definido en la configuración de la aplicación.

## 7. EDICIÓN DE DATOS:

Ahora lo que vamos a explicar es la forma que tiene la aplicación de editar los datos que nos hemos traído desde el fichero ya sea descargado de internet o previamente guardado en el almacenamiento del dispositivo.




En la parte superior se señala el icono asociado a esta actividad y a continuación mostramos el Interfaz de Usuario encargado de esta funcionalidad:


En la parte izquierda de la pantalla se visualizan uno a uno los registros actualmente en memoria. En la parte derecha de la pantalla muestra las opciones de visualización y edición que podemos realizar actualmente con el registro mostrado en la parte izquierda. El campo “Número de Registro” no es un campo real, es un campo calculado que nos facilita la navegación a través de los registros.

Para visualizar los diferentes registros tenemos las opciones de:

- Ir a:

Nos lleva al registro correspondiente al número insertado en este campo. Si introducimos un número fuera del rango nos advierte del error.

- Siguiente:

Nos lleva al siguiente registro del registro visualizado actualmente.

- Anterior

Nos lleva al anterior registro del registro visualizado actualmente.

- Primero

Nos lleva al primer registro.

- Último

Nos lleva al último registro.

*Estás opciones de: Primero, Último, Siguiente y Anterior se habilitan y deshabilitan adecuadamente.

Para Editar los Registros tenemos las siguientes opciones:

- Añadir

Añade un nuevo registro al final de la lista. Al pulsar esta opción nos muestra todos los campos vacíos y el número de Registro con el número siguiente al último. Debemos de pulsar el botón Guardar para confirmar el cambio o Cancelar en caso de no confirmarlo.

- Insertar

Inserta un nuevo registro en la posición actual del campo “Número de Registros”. Al igual que la opción anterior se debe pulsar el botón Guardar para confirmar los cambios o Cancelar en caso contrario.

- Eliminar

El botón eliminar elimina directamente el dato visualizado actualmente. En esta primera versión no te pide confirmación sino los elimina directamente. Ha de tratarse este botón con cuidado.

- Editar

El botón editar permite editar el registro visualizado actualmente. Al igual que el botón Insertar y Añadir debemos usar el botón Guardar para confirmar el cambio si los descartamos usaremos el botón Cancelar.

- Guardar

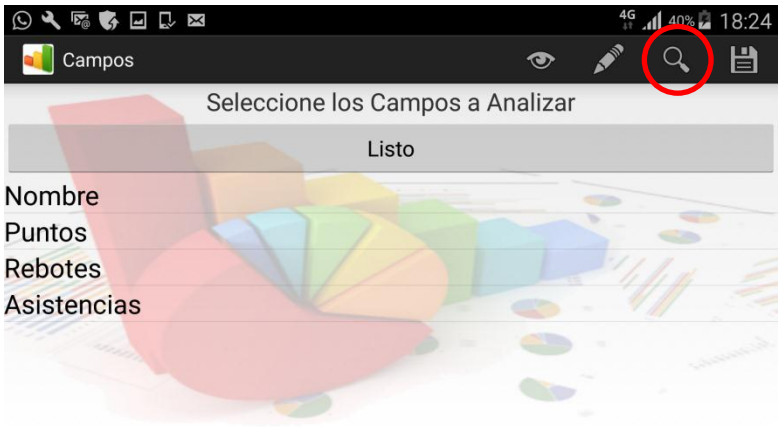
El botón Guardar guarda los cambios actuales en memoria RAM. Para guardarlos en memoria secundaria interna del dispositivo debemos usar la opción ‘Guardar Archivo’ que explicaremos más adelante.

- Cancelar

El botón Cancelar descarta los cambios efectuados sobre el registro actual o descarta Añadir un nuevo registro a la lista.

## 8. FILTROS:

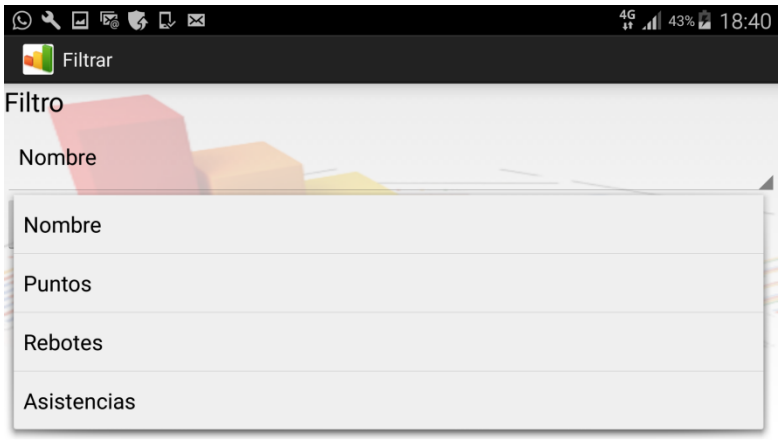
Esta opción está disponible inmediatamente después de la carga de datos y cuando el usuario puede seleccionar los campos numéricos para su análisis. Se activa pulsando el icono de lupa al lado del icono de edición.




Usaremos para ilustrar el funcionamiento de los filtros de tipo numérico y los de tipo cadena los siguientes datos que se muestran en la pantalla siguiente:

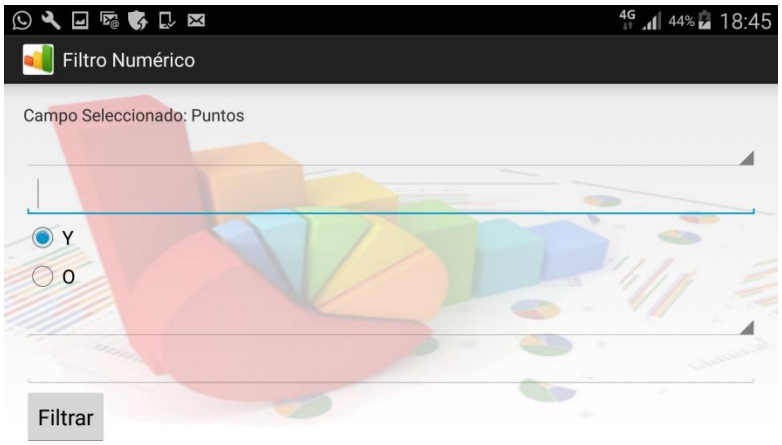

La opción Filtrar permite filtrar el estado actual de los registros de los datos según un patrón o dos. Este filtro distingue tres tipos de campos: Numérico, de Cadenas y de Fecha.

Lo primero que se nos pide es que seleccionemos el campo del fichero a partir del cual queremos que se haga el filtro. En el ejemplo de arriba vemos que el fichero tiene cuatro campos: ‘Nombre’, ‘Puntos’, ‘Rebotes’, ‘Asistencias’, ‘y el filtro se aplicará a un solo uno de ellos.




El programa detecta automáticamente el tipo de filtro a aplicar: si el campo es numérico se le aplicará el filtro específico para datos numéricos, si es una fecha si le aplicará el filtro específico para fechas y si es una cadena de caracteres se le aplicará el filtro específico para caracteres. En la gráfica de arriba vamos a aplicar un filtro al campo 'Puntos.

En la siguiente gráfico se ve la siguiente pantalla: En esta se observa un texto que nos recuerda el campo seleccionado en el paso anterior (en nuestro caso el campo 'Puntos') y nos muestra la estructura a la que se van a someter los filtro: Nos encontramos a dos listas desplegables y dos área de texto. La primera lista desplegable permite seleccionar la opción de filtro que se va a aplicar y la siguiente área de texto, el valor a comparar en la muestra que será introducido directamente por el usuario. Si no rellenamos nada más el filtro es aplicado. Si queremos introducir dos filtros a aplicar debemos de seleccionar la segunda lista desplegable e introducir el segundo valor a comparar en el filtro. La botones 'Y' y 'O' son las operaciones lógicas que se aplican en el caso que el usuario elija filtrar con dos filtros a la vez, estas operaciones lógicas enlazan la forma lógica que se van a combinar cuando metemos dos filtros a la vez.

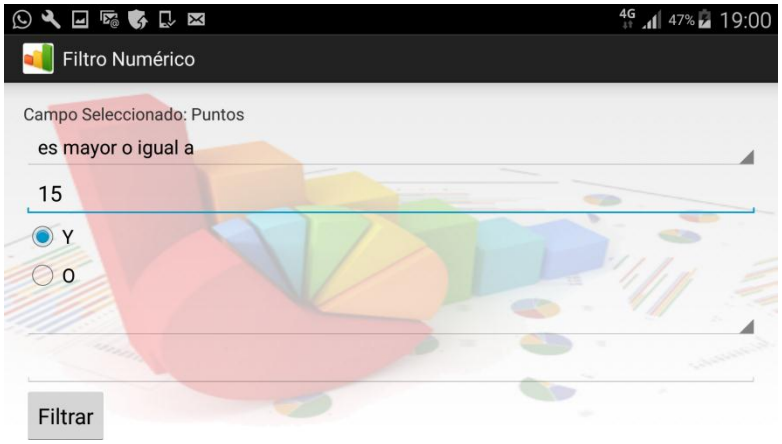



Al pulsar el botón Filtro el programa actúa y realiza el correspondiente filtro. En la parte inferior aparecerá un mensaje indicando cuantos registros han cumplido el filtro del total actualmente en memoria. El botón filtro no finaliza esta pantalla y podemos volver a filtrar a partir de lo filtrado anteriormente.

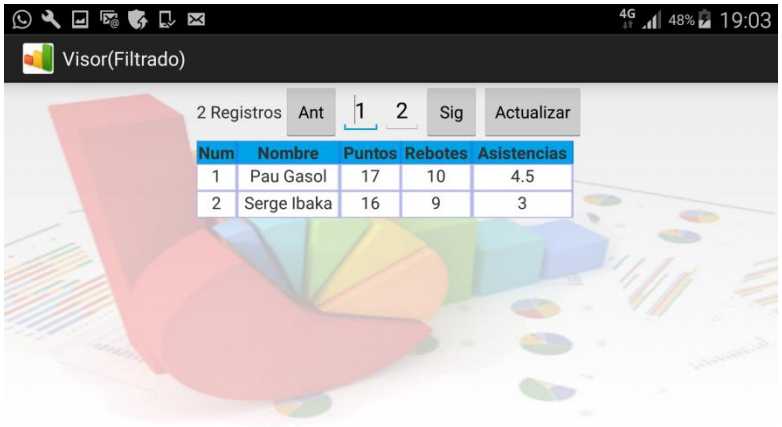
**Filtro Numérico:**

Como hemos dicho si el campo seleccionado es un campo de números entonces se le aplica automáticamente el filtro de tipo numérico como es el caso de la figura anterior.

Veamos un ejemplo intuitivo para averiguar qué jugadores puntúan más o igual de quince puntos:




El resultado de este filtro ejemplo se muestra en es figura:




Al seleccionar solo la primera lista desplegable e introducir un valor numérico estamos diciéndole a la aplicación que solo aplique un criterio.

Las diferentes opciones que tenemos para filtrar un campo numérico son las siguientes.

- Es igual a:

Esta opción filtra todos los registros del campo seleccionado que sean igual al valor numérico introducido como valor en la correspondiente línea de texto.

- No es igual a:

  Esta opción es la contraria que la anterior.

- Es mayor a:

Esta opción filtra todos los registros del campo seleccionado que sea mayor al valor numérico introducido como valor en la correspondiente línea de texto.

- Es mayor o igual a:

  Esta opción es similar a la anterior y además filtra los valores que sean iguales.

- Es menor a:

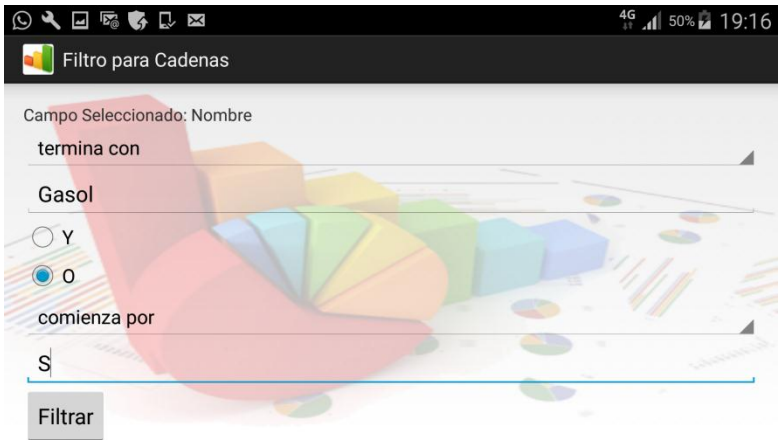
Esta opción filtra todos los registros del campo seleccionado que sean menores al valor numérico introducido como valor en la correspondiente línea de texto.

- Es menor o igual a

  Esta opción es similar a la anterior y además filtra los valores que sean iguales.

**Filtro de Cadena de Caracteres:**

En la siguiente imagen se muestra en ejemplo de cómo funciona el filtro de cadenas:




Se pide realizar un filtro sobre el campo ‘Nombre’ de todos los jugadores cuyo nombre terminen en ‘Gasol’ o bien empiecen con la letra ‘S’.

A continuación presentamos el resultado:


Las diferentes opciones son que nos podemos encontrar en la lista desplegables son:

- Es igual a:

Esta opción filtra todos los registros que coincidan con el valor dado. Se ignoran si están es mayúscula o en minúscula.

- No es igual a:

Es opción es la contraria que la anterior.

- Comienza por:

Esta opción filtra todos los registros cuya valor campo comienza con la cadena de caracteres introducida como valor. Se distinguen las mayúsculas y las minúsculas.

- Termina con:

Esta opción filtra todos los registros cuyo valor campo terminan con la cadena de caracteres introducida como valor. Se distinguen si el valor introducido y los del campo están en minúscula o en mayúscula

- Contiene:

Esta opción filtra a todos los registros cuyo valor campo contiene la cadena de caracteres introducida como valor. Se distinguen si el valor introducido y el campo están en minúscula o en mayúscula.

- No contiene:

Esta opción es la contraria de la anterior descrita. Se distinguen mayúsculas y minúsculas.

- Es mayor que:

Esta opción filtra todos los registros que alfabéticamente hablando sean mayores que la cadena de caracteres introducida es decir los que van después en el alfabeto. Se ignoran mayúsculas y minúsculas.

- Es mayor o igual a:

Esta opción es la misma que la anterior pero en este caso filtran además las que los registros que tengan en su campo el mismo valor introducido como cadena de caracteres.

- Es menor que:

Esta opción filtra todos los registros que sean alfabéticamente hablando sean menores que la cadena de caracteres introducida como valor es decir los que están antes en el alfabeto. Se ignoran mayúsculas y minúsculas.

- Es menor o igual a:

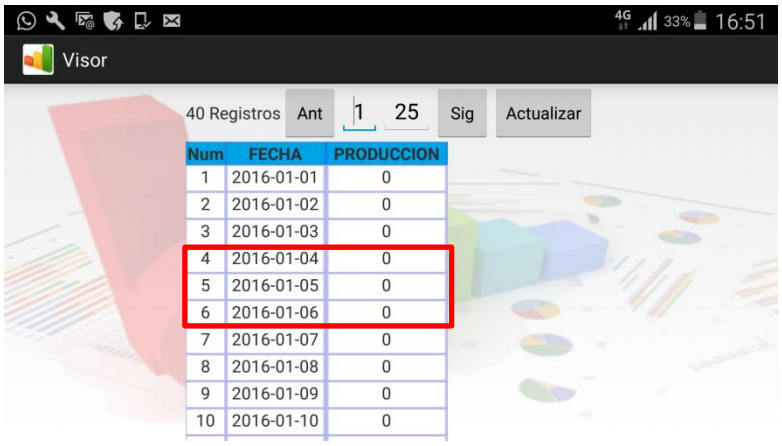
Esta opción es la misma que la anterior pero en este caso filtran además los registros que tengan en sus campos el mismo valor introducido como cadena de caracteres.

**Filtro de Fechas:**

La aplicación detecta automáticamente cuando un campo se le aplica el filtro de Fechas si todos sus campos son fechas. El filtro es muy parecido que los anteriores.

Veamos con un ejemplo:

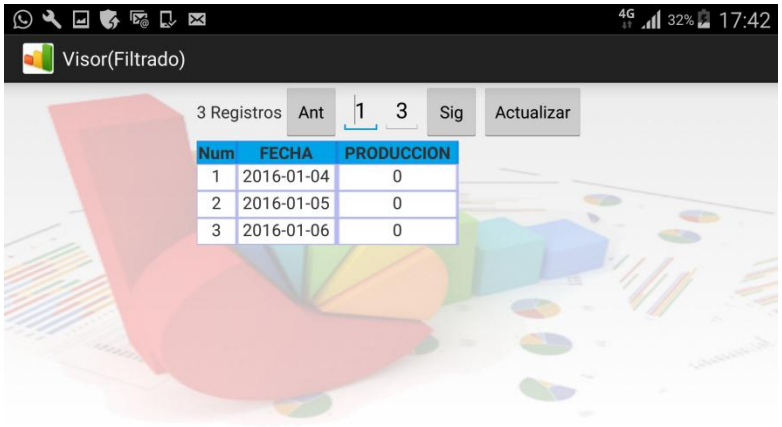
Sea los datos de entrada que visualizamos




Supongamos que queremos filtrar los registros 4, 5 y 6, tal y como indicamos en la figura de arriba.

Tendremos que irnos a Filtrar y seleccionar el campo Fecha una vez hecho esto, podríamos filtrarlos con la siguiente lógica:


Y el resultado correspondiente se muestra en la siguiente pantalla:




Recuerda que el campo Num no es un Campo Real sino que está ahí para facilitarnos la lectura de los diferentes registros.

Lo formatos de fechas aceptados por las líneas de texto de este filtro están en el anexo correspondiente de esta memoria.

Las diferentes opciones son:

- Es igual a:

  Ocurre cuando las fechas son iguales.

- No es igual a:

  Ocurre cuando las fechas no son iguales.

- Es Posterior a:

  Ocurre con los datos de los registros que sean posteriores a la fecha introducida por el usuario.

- Es Posterior o Igual a:

  Lo mismo que el anterior pero además incluye el caso que sean iguales.

- Es Anterior a:

Ocurre con los datos de los registros que sean anteriores a la fecha introducida por el usuario.

- Es Anterior o Igual a:

Lo mismo que el anterior pero además incluye el caso que sean iguales.

## 9. SALVAR A MEMORIA SECUNDARIA:

Esta opción permite Salvar en memoria interna o en memoria externa el estado actual de registros del fichero cargado previamente ya sea por internet o desde el interior del dispositivo. Debemos de seleccionar el icono en forma de diskette que aparece junto al icono de Filtros. El interfaz de usuario que aparece se muestra en la siguiente imagen:


Introducimos el nombre de fichero que queremos darle (la extensión CSV es opcional) en la línea de texto de cuadro de diálogo.

Debemos seleccionar la opción de Almacenamiento Interno o Almacenamiento Externo, introducir un nombre de Fichero y pulsar el botón Guardar. Si queremos volver a la pantalla anterior sin guardar el estado actual de registros pulsaremos Cancelar.

Si hemos seleccionado Al. Interno (Almacenamiento Interno) el fichero se almacenará en el espacio que Android asigna a esta aplicación. Este espacio tiene como característica que sólo lo puede acceder la aplicación Open Data y en caso de que se desístale, estos ficheros son eliminados.

Sin embargo si seleccionamos Al. Externo (Almacenamiento Externo) ocurre: que estos ficheros se puede acceder por cualquier aplicación y no se borrar cuando

ésta de desístale. El programa almacenará en la carpeta por defecto Android para las descargar (carpeta download).

Cuando queramos salir de la aplicación sin guardar los datos y siendo estos manipulados, el sistema nos mostrará un mensaje de advertencia indiciando este hecho con objeto de prevenir posibles olvidos de usar esta opción.

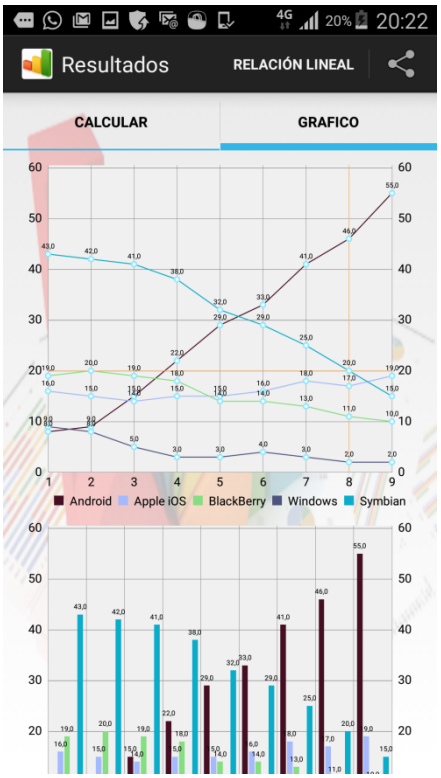
## 10. RESULTADOS:

Una vez tratados los datos de entradas, seleccionados los campos que queremos analizar y pulsando el botón 'Listo' aparecerá la pantalla 'Resultados' mostrándonos los cálculos estadísticos automáticos por cada campo y el/los gráfico/s correspondiente/s, además de otras opciones.

Lo primero que nos aparecen son los cálculos de las diferentes fórmulas estadísticas aplicadas por cada campo: así aparecerán la Media, Moda, Mediana, Rango, Varianza, Desviación Típica, Error Estándar de la Media y Coeficiente de Variación de Pearson.

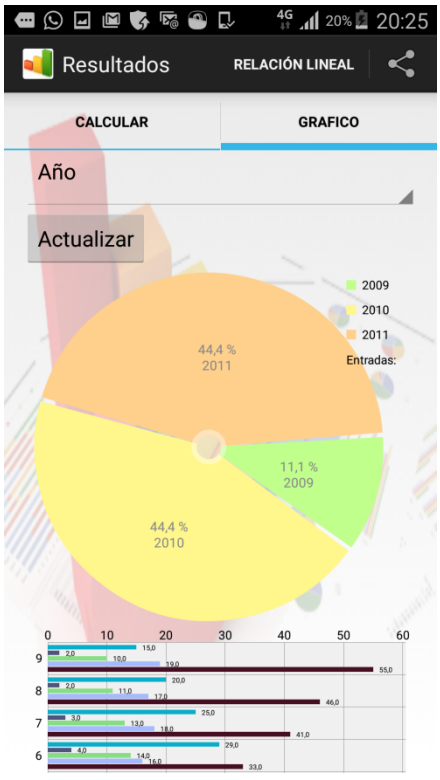

Podemos desplazarnos verticalmente por la pantalla sin problemas.

En la otra pestaña de la actividad 'Resultados' aparece la de Gráficos que nos muestra los gráficos asociados a estos campos y si se ha seleccionado más de un campo previamente aparecen las gráficas comparativas como es el caso de la imagen siguiente:




Se mostrarán tres gráficos posibles: de líneas, de barras y de barras horizontal. Todos ellos interactivos y podemos hacerles zoom pero siempre en el mismo tamaño del marco. También podemos controlar qué gráficos se muestran y cuáles no.

Adicionalmente se muestra un gráfico de tarta donde se calcula el tanto por ciento de ocurrencias de los distintos valores de una determinada columna de la entrada CSV. En la imagen de abajo está calculado el porcentaje de muestra de cada ocurrencia distinta en el campo ‘Año’. Podemos cambiar dinámicamente campo asociado a este gráfico seleccionando otro distinto en  lista desplegable colocada en la parte superior de este gráfico y pulsando el botón ‘Actualizar’. Podemos girar este gráfico o pulsar en alguna parte de él para que nos muestre que tanto por ciento corresponde al valor sobre el que pulsamos.




En esta pantalla existe un botón llamado ‘Relación Lineal’ en la barra de acciones. Se trata de calcular la Covarianza y del Coeficiente de Correlación de Pearson de dos muestras, previamente seleccionadas por el usuario. Al pulsarlo sale un cuadro de diálogo que permite seleccionar las mencionadas muestras y realizar los dos cálculos correspondientes. Estas muestras son siempre muestreables, es decir numéricas. El resultado se indica en la misma pantalla.

Existe el botón compartir que lo que hace es compartir cada una de las fotos actual de los gráficos que estén activos en ese momento. Se trata de una opción del menú que por su relevancia se decido colocar como botón de acceso directo.

El menú de esta pantalla tiene las siguientes opciones:

- Compartir Números:

Como su propio nombre indica comparte los datos estadísticos calculados para cada campo y que están en la primera pestaña.

- Compartir Archivo:

Permite transferir cualquier archivo almacenado en memoria interna a otra aplicación del móvil o a otro dispositivo.

- A Galería:

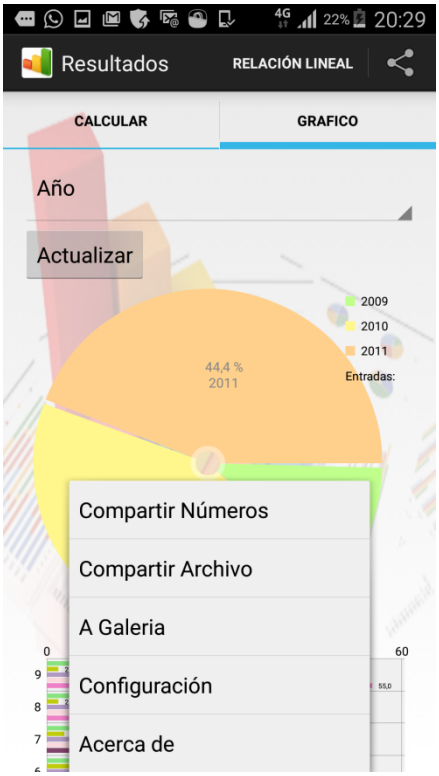
Manda las fotos de los estados actuales de los gráficos a la galería.

- Configuración:

Controla qué gráficos se muestran en cada momento. Es importante pues cuando compartamos sus ‘fotos’ se compartirán solos los que estén activos en esta opción.

- Acerca de:

Esta opción muestra información sobre la aplicación: la versión y sus creadores.



## ANEXO B: Formatos de Fecha aceptados por la aplicación

Los siguientes formatos de fechas son los aceptados por la aplicación y eso significa que son los que son aceptados en los ficheros csv, como en la entrada de usuario del filtro correspondiente a un campo de formato fecha. Cada letra indica un número y la letra ‘d’ indica que va a ser tratado como una número de la cifra día, ‘M’ un número de la cifra mes, ‘y’ un número de la cifra año, ‘H’ un número de la cifra hora, ‘m’ un número de la cifra minuto y ‘s’ un número de la cifra segundo.

| Fechas separadas por barra invertida: | Fechas separadas por guión: |
|---|---|
| "dd/MM/yyyy HH:mm:ss”<br>"dd/MM/yyyy HH:mm"<br>"dd/MM/yyyy HH"<br>"dd/MM/yyyy" | "dd-MM-yyyy HH:mm:ss"<br>"dd-MM-yyyy HH:mm"<br>"dd-MM-yyyy HH"<br>"dd-MM-yyyy" |
| "dd/yyyy/MM HH:mm:ss"<br>"dd/yyyy/MM HH:mm"<br>"dd/yyyy/MM HH"<br>"dd/yyyy/MM" | "dd-yyyy-MM HH:mm:ss"<br>"dd-yyyy-MM HH:mm"<br>"dd-yyyy-MM HH"<br>"dd-yyyy-MM" |
| "MM/dd/yyyy HH:mm:ss"<br>"MM/dd/yyyy HH:mm"<br>"MM/dd/yyyy HH"<br>"MM/dd/yyyy" | "MM-dd-yyyy HH:mm:ss"<br>"MM-dd-yyyy HH:mm"<br>"MM-dd-yyyy HH"<br>"MM-dd-yyyy" |
| "yyyy/dd/MM HH:mm:ss"<br>"yyyy/dd/MM HH:mm"<br>"yyyy/dd/MM HH"<br>"yyyy/dd/MM" | "yyyy-dd-MM HH:mm:ss"<br>"yyyy-dd-MM HH:mm"<br>"yyyy-dd-MM HH"<br>"yyyy-dd-MM" |
| "MM/yyyy/dd HH:mm:ss"<br>"MM/yyyy/dd HH:mm"<br>"MM/yyyy/dd HH"<br>"MM/yyyy/dd" | "MM-yyyy-dd HH:mm:ss"<br>"MM-yyyy-dd HH:mm"<br>"MM-yyyy-dd HH"<br>"MM-yyyy-dd" |
| "yyyy/MM/dd HH:mm:ss"<br>"yyyy/MM/dd HH:mm"<br>"yyyy/MM/dd HH"<br>"yyyy/MM/dd" | "yyyy-MM-dd HH:mm:ss"<br>"yyyy-MM-dd HH:mm"<br>"yyyy-MM-dd HH"<br>"yyyy-MM-dd" |

Así por ejemplo:

Podríamos introducir la fecha 14/08/2016 cuyo formato asociado sería "dd/MM/yyyy", es decir: catorce se agosto del 2016 o bien introducir la fecha de 09/10/2016 12:30:50 cuyo formato asociado sería "dd-MM-yyyy HH:mm:ss", es decir, nueve de noviembre del 2016 a las doce y media y cincuenta segundos.