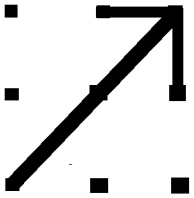
UNIVERSIDADE DA CORUÑA

PROGRAMACIÓN DE SISTEMAS 22/23 Q1



MallaVectores

Autores: Iago Valeiro Castrillón
Alejandro Dopazo López
Alberto Ferreiro Campello
Fecha:*A Coruña, 19 Diciembre 2022*

Índice

Capítulos **Página**

Cuadro 1: Tabla de versiones.

Versión	Fecha	Autor
x	y	
x	y	
x	y	

1. Introducción

La aplicación propuesta consiste en un editor de imágenes vectoriales para dispositivos móviles Android, inicialmente sería para la creación de iconos y basaría su funcionamiento en la interconexión de puntos en una malla cuadriculada, además llevaría integrado un sistema de guardado local mediante una base de datos SQL, y la posibilidad de guardar las imágenes exportadas en la nube con Firebase.

1.1. Objetivos

1. Principal:
    - Creación de imágenes vectoriales a partir de puntos, líneas y curvas.
2. Secundarios:
    - Exportación de las imágenes a formatos de archivo habituales como .svg [1] que permitan compartirlas con otros usuarios.
    - Manejo de los cambios en las imágenes mediante un sistema de guardado local en base de datos SQL [2].

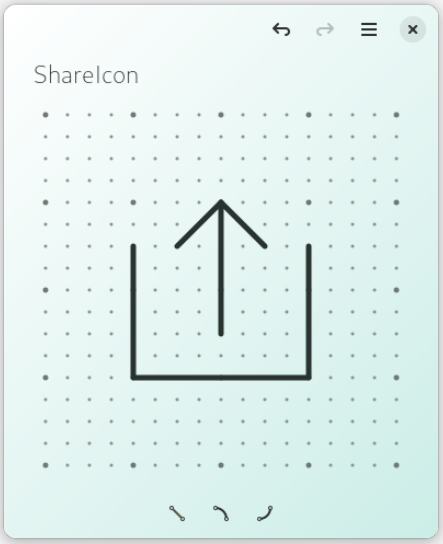
Las dependencias son las especificadas en la sección *Funcionalidades*

1.2. Motivación

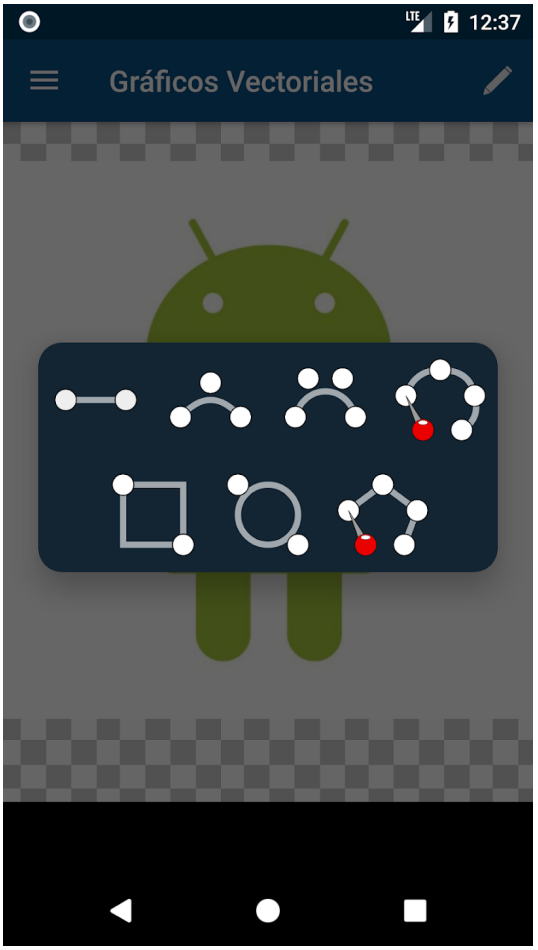
Crear una herramienta con la que poder experimentar fácil y cómodamente con gráficos vectoriales, se trata de un tema de especial interés dada la relativa carencia de herramientas destinadas a estos fines. Es aplicable para, por ejemplo, el diseño de recursos para aplicaciones gráficas, principalmente iconos.

1.3. Trabajo relacionado

- Dot Matrix [3], se trata de la aplicación original que dio pié a este proyecto, consta de un editor de iconos desarrollado para Linux por un equipo de 10 personas en el lenguaje de programación Vala [4]. Se puede ver en la figura 1.(a).
- Gráficos Vectoriales [5] en este caso se trata de una aplicación Android similar a la que se plantea, es bastante completa y se oferta como conversor de archivos .png y .jpg a .svg. Se puede ver en la figura 1.(b).



(a) Dot Matrix



(b) Graficos Vectoriales

Figura 1: Ejemplos de aplicaciones similares.

2. Análisis de requisitos

- La aplicación debe ser capaz de permitir al usuario realizar dibujos utilizando gráficos vectoriales que permitan ser redimensionados sin perder calidad.
- Dichos dibujos pueden incluir líneas rectas y curvas, la aplicación que se desarrolle debe dar soporte a ambas.
- Con el objetivo de facilitar el proceso de dibujado, si dos vértices están muy próximos estos deberían unirse de forma automática. Dicho de otro modo, los puntos de inicio y fin de cada línea deben ser fijos, dispuestos en una cuadrícula de puntos.
- Se espera que la aplicación incluya una opción habitual de los editores de imágenes como es "deshacer".
- La aplicación debe permitir exportar los dibujos a formatos de archivo habituales.
- Dichos archivos deben poderse compartir.

2.1. Funcionalidades

2.1.1. Creación de imágenes vectoriales

Se pueden disponer los siguientes elementos en la pantalla:

- Líneas rectas.
- Curvas de Bézier [6] cúbicas.
- Curvas de Bézier cuadráticas.

Cada tipo de elemento tiene un modo asociado, que se puede seleccionar mediante la barra de botones de la actividad principal. Los modos están identificados por iconos que identifican a las curvas cuadráticas y cúbicas.

El modo actual se muestra al principio de la barra de botones.

Las líneas rectas se dibujan arrastrando entre los dos puntos que vayan a definirla.

Las curvas se definen pulsando primero en cada punto de la recta y luego en sus puntos de control. Las curvas cuadráticas tienen un punto de control y las cúbicas dos.

2.1.2. Exportación a ficheros

Depende de la funcionalidad *Creación de imágenes vectoriales*. Las imágenes generadas se pueden exportar a ficheros externos en formato svg. El formato usado se ceñirá a la especificación SVGTiny [7] excepto las funciones de text, al igual que los VectorDrawable de Android [8].

2.2. Sistema de guardado local

Se implementará un sistema de guardado local mediante una base de datos SQL, esta guardará cada línea del dibujo en formato String lo que permitirá realizar las siguientes acciones:

- Deshacer/rehacer la última acción.
- Eliminar un trazo específico.
- Guardar distintos dibujos.
- Exportar un dibujo concreto a SVG para su guardado en la nube y/o compartir con otras apps.

2.3. Guardado automático de los ficheros

La base de datos local se actualizará cada vez que se añada un nuevo trazo, además habrá un servicio en segundo plano que convertirá las imágenes creadas a un fichero SVG y lo subirá a Firebase [9] de forma automática cada cierto tiempo.

Esta funcionalidad estará inicialmente desactivada y se podrá activar y configurar desde la propia aplicación.

2.4. Prioridades

1. Creación de imágenes vectoriales.
2. Guardado local.
3. Edición de imágenes.
    - Deshacer.
    - Rehacer.
    - Seleccionar y eliminar un trazo.
4. Exportación a ficheros.
5. Compartir a otras apps.
6. Integración con Firebase.
7. Guardado automático en la nube.

3. Planificación inicial

3.1. Iteraciones

Primera iteración: Implementación de la funcionalidad *Exportación a ficheros* y *Creación de imágenes vectoriales*.

Segunda iteración: Implementación de la funcionalidad *Exportación a ficheros*.

Tercera iteración: Implementación de la funcionalidad *Guardado en base de datos*.

Cuarta iteración: Implementación de la funcionalidad *Guardado automático de los ficheros*.

Posiblemente otras iteraciones, dependiendo del curso del desarrollo: Expansión de funcionalidades; por ejemplo, posibilitar la interacción con otros elementos del estándar SVGTiny [7], como rellenos y colores, o posibilitar la exportación a distintos formatos.

3.2. Responsabilidades

Iago Valeiro Castrillón: *Creación de imágenes vectoriales*, *Exportación a ficheros*, *Guardado automático de los ficheros* e *Guardado en base de datos*.

Alejandro Dopazo López: *Creación de imágenes vectoriales*, *Exportación a ficheros*, *Guardado automático de los ficheros* e *Guardado en base de datos*.

Alberto Ferreiro Campello: *Creación de imágenes vectoriales*, *Exportación a ficheros*, *Guardado automático de los ficheros* e *Guardado en base de datos*.

3.3. Hitos

Primer entregable: primera iteración implementada. Pruebas de las funcionalidades implementadas.

Segundo entregable: segunda iteración implementada. Pruebas de las funcionalidades implementadas.

Tercer entregable: tercera iteración implementada. Pruebas de las funcionalidades implementadas.

Entregable final: Pruebas de integración y de usabilidad. Revisión de las iteraciones anteriores.

3.4. Incidencias

- El proyecto transcurre demasiado rápido/despacio:

Se añadirán o quitarán iteraciones al proyecto de acuerdo a lo especificado en la sección 3.1. Los hitos y los entregables se ajustarán apropiadamente.

4. Diseño

4.1. Arquitectura

Se utilizará la arquitectura Clean Code [10] basada en el patrón de diseño model view controller o MVC [11].

4.1.1. Actividad principal

MainActivity, con una barra con botones para seleccionar los modos de dibujo una Vistamalla que muestre la imagen en sí y responda a las acciones y un menú de opciones con:

- Exportación de la imagen actual a un fichero.
- Conectarse a Firebase para el sistema de guardado en la nube.
- Lanzar la actividad Ajustes.

4.1.2. Vistamalla

La vista muestra una malla de puntos y las formas dibujadas por el usuario.
Gestiona su propia interacción con el usuario.
Envía los trazos a base de datos.

4.1.3. Actividad Ajustes

Muestra un sistema de login para utilizar los servicios de Firebase, así como una entrada para cambiar la frecuencia con la que se guarda automáticamente la imagen actual, y un botón para habilitar el servicio GuardadoAutomatico.

4.2. Persistencia

Se almacenarán las imágenes en una base de datos SQL. Además se almacenarán los ajustes del servicio de guardado automático en un fichero local "SharedPreferences".

4.3. Vista

Una actividad principal mostrando un menú con las acciones que se pueden realizar sobre la imagen actual. La actividad contiene el fragmento con la malla de puntos.

Desde el menú de la actividad principal se puede navegar a la actividad de ajustes.

En la actividad de ajustes se muestra una entrada para cambiar la frecuencia con la que se guarda automáticamente la imagen actual y un botón para deshabilitar el servicio de guardado automático.

4.4. Comunicaciones

Las imágenes se guardan en formato String en la base de datos local (un nombre/id y múltiples path, uno por cada trazo). Para enviar una imagen, se extrae de la base de datos y se convierte a SVG, tras la conversión se envía mediante un intent a otras apps, o a la base de datos online en Firebase.

4.5. Sensores

No se debería usar ninguno explícitamente aparte de la entrada por la pantalla táctil.

4.6. Background

4.6.1. Servicio GuardadoAutomatico

Mientras el usuario va haciendo su diseño se irán guardando sus avances en local, además si el usuario lo desea, puede activar el sistema de guardado en la nube en Firebase.

El guardado automático en la nube sucederá mediante el servicio GuardadoAutomatico. Éste sucederá cada vez que pase un período definido en la actividad Ajustes.

5. Avances sobre el entregable anterior

5.1. Diseño

Ya que la librería para trabajar con Git no ofrece soporte para Android, y no se han encontrado alternativas funcionales, se ha quitado del diseño a favor de Firebase.

5.2. Avances

Se ha añadido el icono a la aplicación. Se ha cambiado el modo de selección de línea de texto a iconos. Se ha implementado la arquitectura Clean Code para el acceso a base de datos.

5.3. Hitos

Actualmente la aplicación permite realizar dibujos mediante distintos tipos de trazos, para probar el modo dibujo, simplemente se instala en un dispositivo o emulador y:

- Para dibujar líneas rectas, seleccionamos el modo línea y observamos que ha cambiado la previsualización de herramienta seleccionada, a continuación tocamos donde queremos que inicie y arrastramos hasta donde queremos que acabe, al soltar aparecerá la línea.

- Para las líneas cúbicas y cuadráticas, escogemos el modo igual que con las rectas, y a continuación, para dibujar, tocaremos donde queremos que comience y finalice la línea. En cuanto hayamos colocado el segundo punto aparecerá la línea, tras esto sólo resta añadir un tercer punto (dos en el caso de las cuadráticas), y ya se nos dibujará la línea con la curva aplicada.

Además realiza de forma transparente el almacenamiento de los dibujos en base de datos, para comprobar su correcto funcionamiento simplemente hay que ir a App Inspection -¿Database Inspector, y tras hacer cualquier trazo, seleccionar la tabla Draws y marcar Live Updates, tras esto podremos comprobar que al hacer cualquier trazo, éste se guarda en tiempo real.

5.4. Detalles de la implementación

El diseño de la base de datos en esta implementación es muy sencillo y no se utiliza correctamente, se ha desarrollado un modelo Entidad-Relación sencillo de cara a la siguiente implementación.

5.5. Pruebas

Se ha comprobado el funcionamiento de la app en una tablet de 10"BMXC B801 con Android 8, un Poco F3 con Android 12, y un Google Nexus 5 (emulado) con Android 13. Todo funciona de forma correcta, salvo el guardado local en la tablet el cual no ha sido posible probar dada la versión de Android (no es posible ver las bases de datos en App Inspection con una versión de Android tan antigua).

Referencias

[1] "Svg." Available on: `https://developer.mozilla.org/es/docs/Web/SVG`. Last access on 2022-10-18.

[2] "Sql." Available on: `https://www.iso.org/standard/63555.html`. Last access on 2022-12-19.

[3] "Dot matrix." Available on: `https://github.com/lainsce/dot-matrix/`. Last access on 2022-10-18.

[4] "Vala." Available on: `https://vala.dev/`. Last access on 2022-10-18.

[5] "Gráficos vectoriales." Available on: `https://play.google.com/store/apps/details?id=com.inglesdivino.vectorassetcreator&hl=es_419&gl=US`. Last access on 2022-10-18.

[6] "Curvas de bezier." Available on: `https://learn.microsoft.com/es-es/xamarin/xamarin-forms/user-interface/graphics/skiasharp/curves/beziers`. Last access on 2022-10-18.

[7] "Svgtiny." Available on: https://www.w3.org/TR/SVGTiny12/. Last access on 2022-10-18.

[8] "Vectordrawable." Available on: https://developer.android.com/studio/write/vector-asset-studio#svg-supported. Last access on 2022-10-18.

[9] "Firebase." Available on: https://firebase.google.com/. Last access on 2022-12-19.

[10] Martin, R. C., *CLEAN CODE: A Handbook of Agile Software Craftsmanship*. PRENTICE HALL, 2008.

[11] "Mvc." Available on: https://en.wikipedia.org/wiki/Model\%E2\%80\%93view\%E2\%80\%93controller. Last access on 2022-12-19.