Suite JVG

Victor Domínguez | Jonathan Alonzo | Gabriel Ramos López

Resumen

El presente trabajo tiene como objetivo **desarrollar**, de la forma más eficiente y estructurada posible, una **suite ofimática** que consta de las siguientes aplicaciones: Saber la **batería** actual, nivel de **proximidad** y un **lector de huellas**, contrar los pasos con un **podómetro**, hacer una **vibración**, y una **brújula**, saber el nivel de **rotación** y **gravedad**, y un **acelerómetro**.

El siguiente trabajo, al que llamaré **suite** de ahora en adelante, consiste en un conjunto de **proyectos** realizados de forma individual, testeados y verificados de forma grupal, para poder brindar al usuario una experiencia de calidad con las herramientas expuestas anteriormente. Es una suite creada con **buenas prácticas**, comprobadas por los diferentes integrantes del grupo, y desarrolladas siguiendo las buenas prácticas conocidas por cada uno de los integrantes.

El proyecto consiste en una actividad inicial en la que encontramos en un **menú** una imagen de los integrantes del equipo. Al darle a alguno, la pantalla principal que se mostrará será su **calculadora**. Para acceder a las actividades de cada uno, hay un menú **hamburguesa** que permite acceder tanto a las **actividades**, como a la **información** que quiera poner el desarrollador. El proyecto completo cuenta con tecnología **no vista en clase** que **refuerza** y mejora el rendimiento y la calidad del proyecto.

Palabras clave

Suite, Suite ofimática, Herramientas, Aplicación

Abstract

The present work aims to **develop**, in the most efficient and structured way possible, an **office suite** consisting of the following applications: Knowing the current **battery**, **proximity level** and a **fingerprint reader**, counting steps with a **pedometer**, making a **vibration**, and a **compass**, knowing the **level of rotation** and **gravity**, and an **accelerometer**.

The following work, which I will call **suite** from now on, consists of a set of **projects** made individually, tested and verified as a group, in order to provide the user with a quality experience with the tools described above. It is a suite created with good practices, tested by the different members of the group, and developed following the **good practices** known by each of the members.

The project consists of an initial activity in which we find in a **menu** an image of the team members. When clicking on one of them, the main screen that will be shown will be their **calculator**. To access the activities of each one, there is a **hamburger** menu that allows access to both the **activities** and the information that the developer wants to put. The entire project has technology **not seen in class** that **reinforces** and improves the performance and quality of the project.

Keywords

Suite, Office Suite, Tools, Application

Índice

Análisis del proyecto

A continuación, se adjuntan una serie de esquemas/bocetos/borradores utilizados para la realización del proyecto:

Planeación/Planteamiento de la pantalla/view principal:

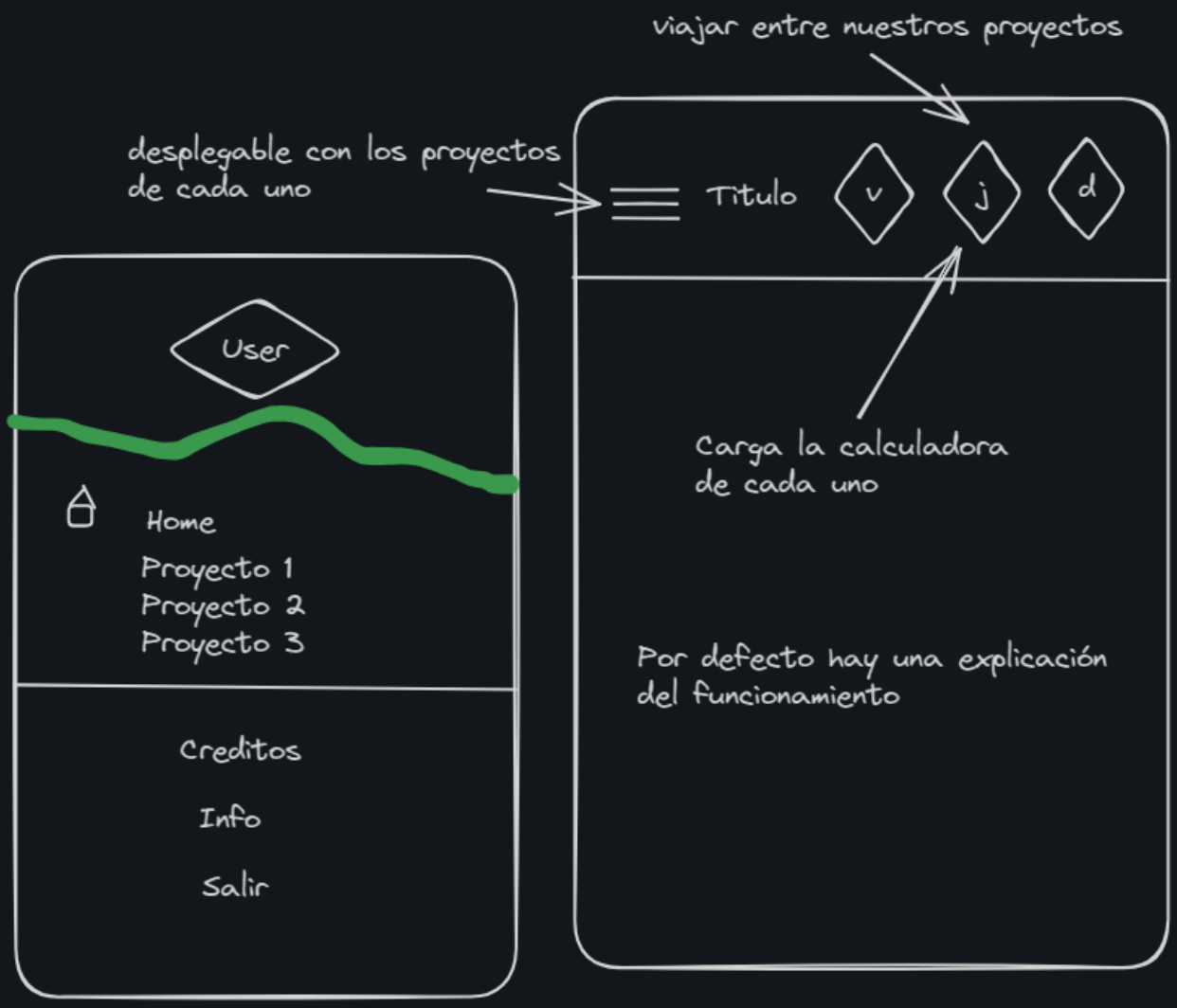
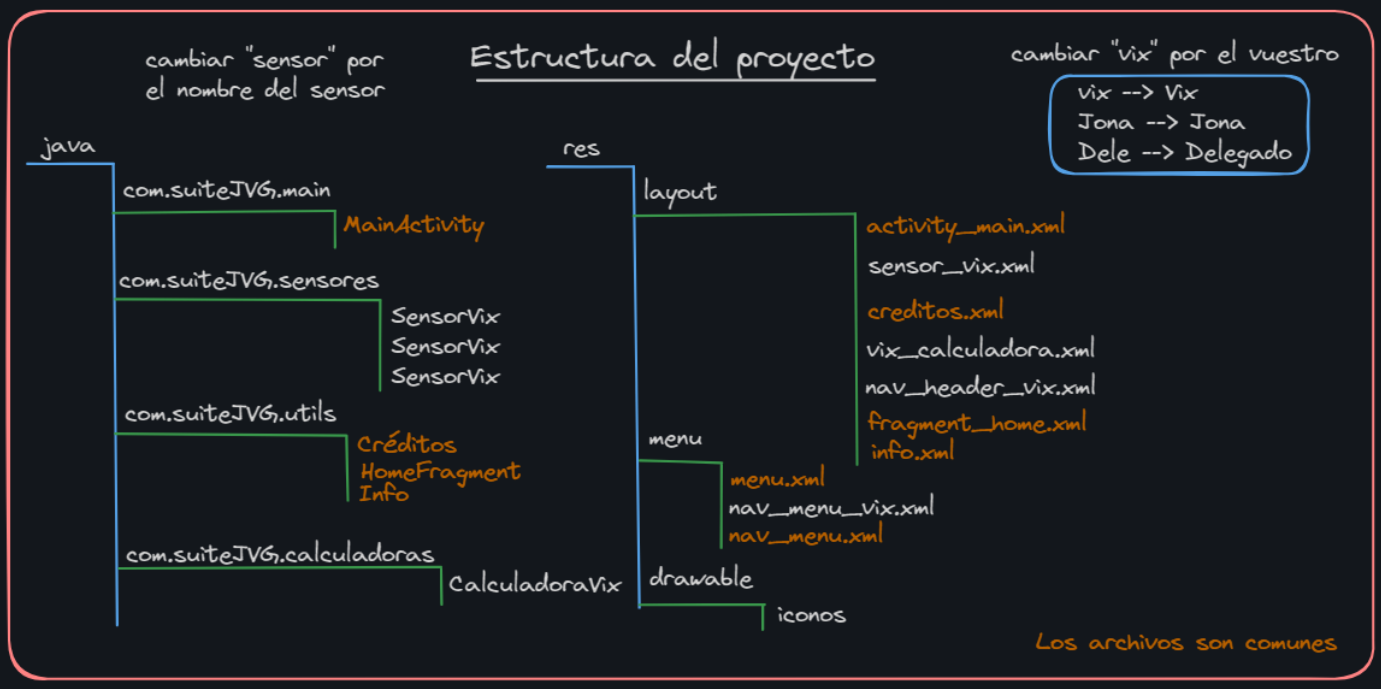


Diagrama independiente sobre la estructura del proyecto:



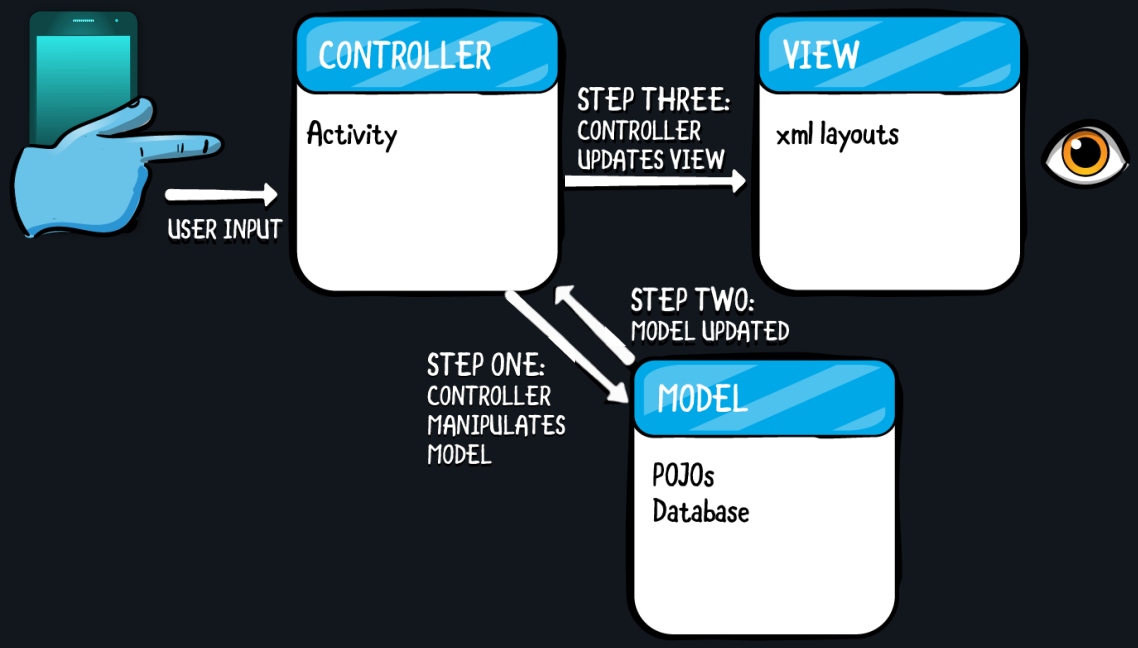
Arquitectura del proyecto:

La arquitectura utilizada para el desarrollo de este proyecto ha sido la denominada Modelo-Vista-Controlador (Model-View-Controller), MVC, ya que esta estructura es una de las muchas que se pueden utilizar para el desarrollo de aplicaciones Android.

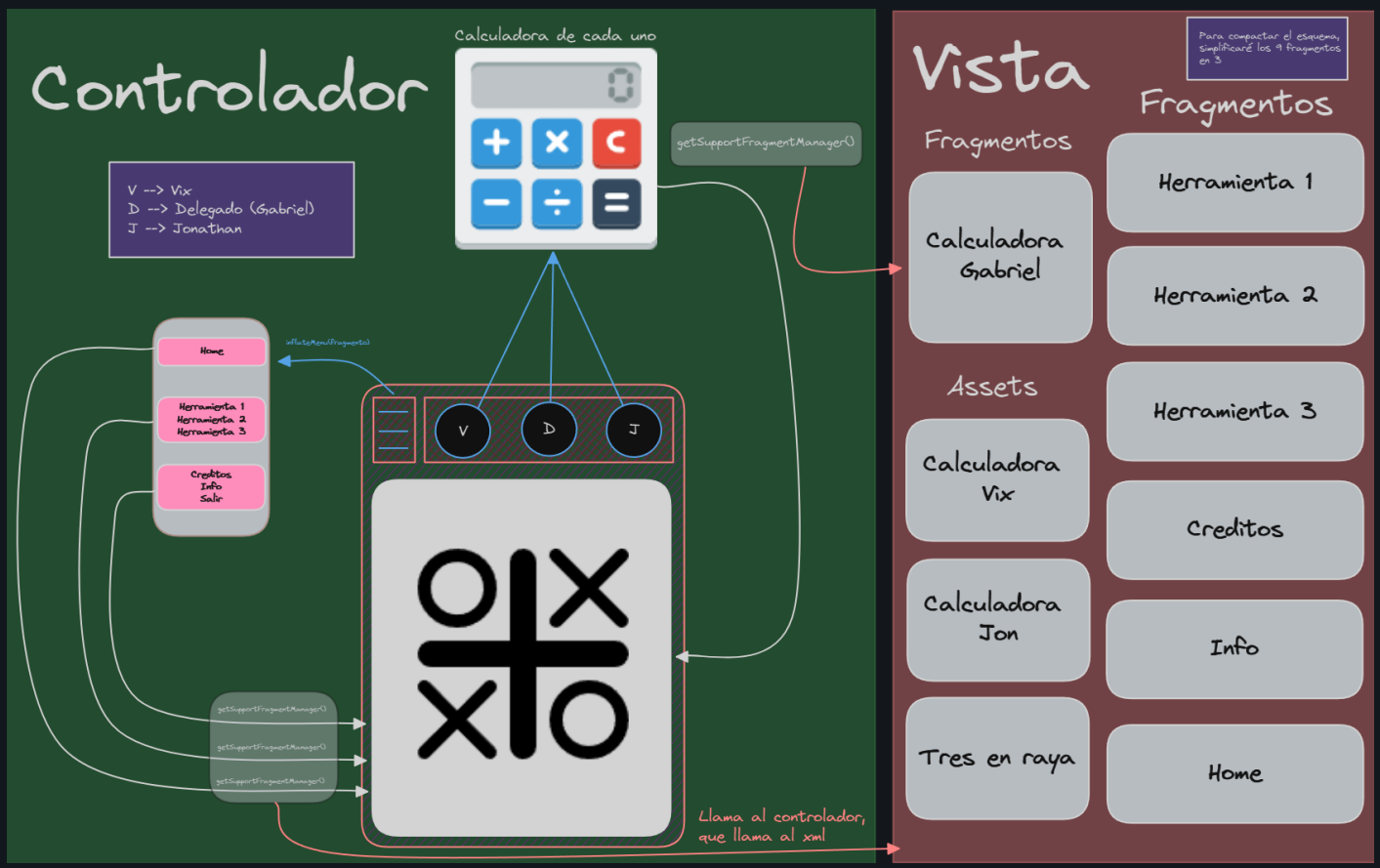
La arquitectura MVC se compone de tres elementos principales:

- **Modelo**: representa los datos y la lógica de negocio de la aplicación. No está vinculado a la vista ni al controlador, y gracias a esto, es reutilizable en muchos contextos.
- **Vista**: es la interfaz de usuario (UI) de la aplicación, es decir, lo que el usuario ve y con lo que interactúa. También se encarga de mostrar los datos al usuario y de capturar las acciones del usuario.
- **Controlador**: es el intermediario entre el modelo y la vista. Recibe las interacciones del usuario desde la vista, procesa esas interacciones y realiza cambios en el modelo. Luego, actualiza la vista para reflejar los cambios en el modelo. En el caso de una aplicación Android, el controlador casi siempre está representado por una Activity o un Fragment.

Funcionamiento MVC de Android



MVC de nuestro Proyecto



En la imagen anterior podemos observar cuál es el funcionamiento de este tipo de arquitectura en nuestra aplicación:

- Al seleccionar a un integrante del equipo, el menú hamburguesa cambia al de dicho integrante al “inflar” el menú.
- Al acceder al menú hamburguesa, podemos ver las diferentes opciones que podemos elegir. En cualquier caso, para regresar al menú principal, se deberá dar a “Home” en la hamburguesa.
- Al cambiar de actividad mientras ejecutamos otra, hará que la anterior pare, para ahorrar recursos del dispositivo, y evitar fallos. También, para eso último, hemos establecido la orientación de la aplicación en vertical siempre.
- La llamada de las pantallas que se ven, son fragmentos. Tenemos un fragmento “padre” que engloba al hijo cuando se llama a su controlador. Más información en los requisitos.

Requisitos

Requisitos funcionales:

- La pantalla inicial cargará junto al menú hamburguesa y la toolbar o navbar. La navbar cargará mediante un **setSupportActionBar** apuntando al xml deseado. El menú hamburguesa será llamado mediante un **inflateMenu**, al que se le aplicará un **setNavigationItemSelectedListener** para saber si se pulsa algún botón. Además, el logo del menú hamburguesa cambiará con **inflateHeaderView**.
- Menciono la inicialización del fragmento que se ve por separado, ya que para todas las llamadas, independientemente de que sean fragmentos o assets, se hace de la misma forma, que es haciendo un **getSupportFragmentManager** que reemplaza el fragmento “padre” por el que haga referencia el controlador al que llamamos a continuación. De esta forma, no “apilamos” los fragmentos, si no que sustituimos el original.
- Al seleccionar el usuario de la navbar, hará el mismo proceso que el punto uno, con la diferencia de que se llamanrán a los XML de cada uno.
- Para salir de la actividad se usa el método **finishAffinity**. Se usa este método porque es buena práctica su uso con **navigationView**, que es nuestro caso.
- Para casi todos los sensores, se inicia con la clase **SensorManager**, y se implementa en la clase que se pretende usar el **SensorEventListener.**

Requisitos no funcionales:

Los requisitos no funcionales necesarios para ejecutar la suite son mínimos:

- Se necesita un teléfono móvil con un S.O. de Android del año 2016 o superior, o, como alternativa, un emulador de Android compatible con la versión necesaria para la suite.
- Todos los dispositivos desde los cuales se quiera acceder a la suite deberán tener una versión de Android igual o superior a **9**.
- Conexión a internet para la carga de links, pantallas de calculadoras, y juego “Tic Tac Toe”

Despliegue

El proyecto está pensado para ejecutarse en un dispositivo móvil con un **Sistema Operativo Android**, sin embargo, también es posible ejecutarlo con máquinas virtuales.

Para la realización de esta aplicación, hemos optado por usar **GitHub** como sistema de control de versiones. GitHub nos ha servido para poder desarrollar, de manera independiente el proyecto asignado a cada uno de los integrantes, dentro de su propia **rama**. Dichas ramas están relacionadas, pero no se comunican inicialmente entre sí.

El despliegue, actualización, y control de versiones se han realizado con **Git**. Git es una herramienta que hemos usado con el propósito de conectar con el repositorio de GitHub de forma local y en consola. Esto nos ha permitido un manejo y control de directorios y ficheros de forma **rápida** y **segura**.

Una vez los proyectos han sido comprobados, testeados, y aceptados por todos los integrantes del equipo, se han implementado en el proyecto real, ubicado en la rama Unido.

Conclusiones personales

Este proyecto nos ha parecido muy útil. Cada uno de nosotros ha aprendido algo nuevo y lo ha compartido con el resto, como por ejemplo la **implementación de un proyecto web en Android** para aumentar el número de tecnologías usadas, o el uso del **FragmentManager para poder controlar los fragmentos** de una manera más sencilla y ordenada.

Por otro lado, el uso de la metodología **SCRUM** en un inicio nos pareció difícil y poco útil para el proyecto, pero cuando nos pusimos serios para hacer el proyecto de forma ordenada y correcta, vimos un **incremento en el rendimiento del equipo** y una claridad mayor a la hora de representar nuestro progreso en el Daily Scrum.

Todos estamos de acuerdo que este trabajo y examen en equipo ha supuesto una **mejora tanto en la lógica como socialmente**, ya que nos ha permitido unir ideas innovadoras y aplicarlas en el resultado final.

Pruebas y resultados obtenidos

Caja Blanca.

Las pruebas de caja blanca de un software son aquellas que implican examinar y evaluar internamente la lógica interna, la estructura y el código fuente del software. Su cometido principal es identificar errores en la lógica del programa, asegurar la ejecución de todas las instrucciones y rutas de código, y evaluar la calidad y eficiencia del código fuente.

Estas pruebas revisan las funciones, las estructuras de datos utilizadas y cómo interactúan los diferentes componentes, para así evaluar el correcto funcionamiento del programa.

Durante el desarrollo del proyecto se han hecho multitud de pruebas para verificar que el proyecto seguía el flujo de ejecución correcto. Todas las partes, finalmente, han obtenido un resultado positivo.

Caja Negra.

Las pruebas de caja negra son aquellas que se pueden realizar sin necesidad de conocer el código fuente del programa. Esto es debido a que su objetivo principal es comprobar el flujo de entradas y salidas del sistema y en cómo responde a diferentes situaciones para así identificar errores en la interfaz del usuario, comportamientos incorrectos o inesperados del sistema y asegurar que cumpla con los requisitos especificados.

Estas pruebas se han realizado durante todo el desarrollo del proyecto, aunque se han realizado más en profundidad al finalizar el mismo. Algunas de estas pruebas han sido comprobar el correcto enlace entre las diferentes actividades, giro vertical-horizontal en fragmentos que no se debería, links funcionales y que lleven al lugar esperado, y acciones inesperadas por parte del usuario. Todas las partes, finalmente, han obtenido un resultado positivo.

Bibliografía

Para llevar a cabo la elaboración de este proyecto, ha sido necesario consultar diversas fuentes, entre las que destacan:

- Documentación propia de la asignatura
- https://www.kaggle.com/
- https://openwebinars.net/
- https://www.youtube.com/watch?v=ahNruIZX130
- https://chat.openai.com/
- https://www.youtube.com/@SarthiTechnology
- https://developer.android.com/docs?hl=es-419