**Entrega de corte 1**

**Cristian Fernando Lozada Padilla**

**Ingeniería de sistemas**

**Docente**

**German Morales**

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

**Institución Universitaria Antonio José Camacho**

**Facultad de ingeniería**

**Ingeniería de sistemas**

**2022**

1. ¿Qué es XML?

Es un lenguaje de marcado extensible (Extensible Markup Language) que sirve para almacenar, envolver y transmitir información. XML almacena los datos en formato de texto plano, lo que garantiza compatibilidad en cualquier sistema que intercambie información con otro.

1. La principal diferencia de HTML es que fue diseñado con el enfoque de cómo luce la información, XML hace énfasis en cual información.

* XML no utiliza etiquetas predefinidas como lo hace HTML (como <p> o <header>), las etiquetas que utilice XML deben ser definidas o ‘’inventadas’’ por el autor del documento.
* HTML sirve para construir aplicaciones y paginas web y es utilizado para **mostrar** la información y no para transportarla, define el documento de texto en etiquetas que definen la estructura de la web.
* XML también es utilizado para construir aplicaciones y páginas web, pero es dinámico porque se utiliza para **transportar** la información y no para mostrarla, está diseñado para la simplicidad, generalidad y usabilidad a través de internet.
* HTML puede ignorar pequeños errores mientras XML no.
* XML es sensible a mayúsculas y minúsculas mientas HTML no.
* Las etiquetas HTML son utilizadas para mostrar la información y las XML para describir la información, no para mostrarla.

1. Glosario: como mencionamos anteriormente, XML no tiene etiquetas predefinidas, sin embargo, se van a ejemplificar con algunas etiquetas comunes de Android.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Etiqueta | Atributos | Definición |
| Inicio | xml | versión, encoding | Comúnmente llamado prologo, se usa para especificar el tipo de codificación, versión de xml, etc |
| Cuerpo | LinearLayout | Layout\_width, layout\_height, orientation | Define la posición de los elementos de la interfaz de usuario de manera estrictamente lineal (vertical u horizontal) |
| Constraint Layout | “ ” | Permite trabajar con elementos de interfaz de manera anidada |
| Relative Layout | “ ” | Permite manipular de manera manual la ubicación de los elementos de la interfaz en el modo de diseño del IDE |
| Formulario | Button | onClick, text | Permite al usuario ejecutar funcionalidades especificadas en el atributo onClick |
| ImageView | src, contentDescription, maxHeight, maxWidth | Funcionan como contenedores de imágenes que se muestran en la interfaz de usuario |
| TextView | text, hint, inputType | Muestran en la interfaz textos de manera dinámica o estática |
| EditText | inputType, hint | Muestra un campo de texto editable y modificable por el usuario |
| CkeckBox | Text, enabled | Es una casilla modificable por el usuario que puede estar chequeada o no |

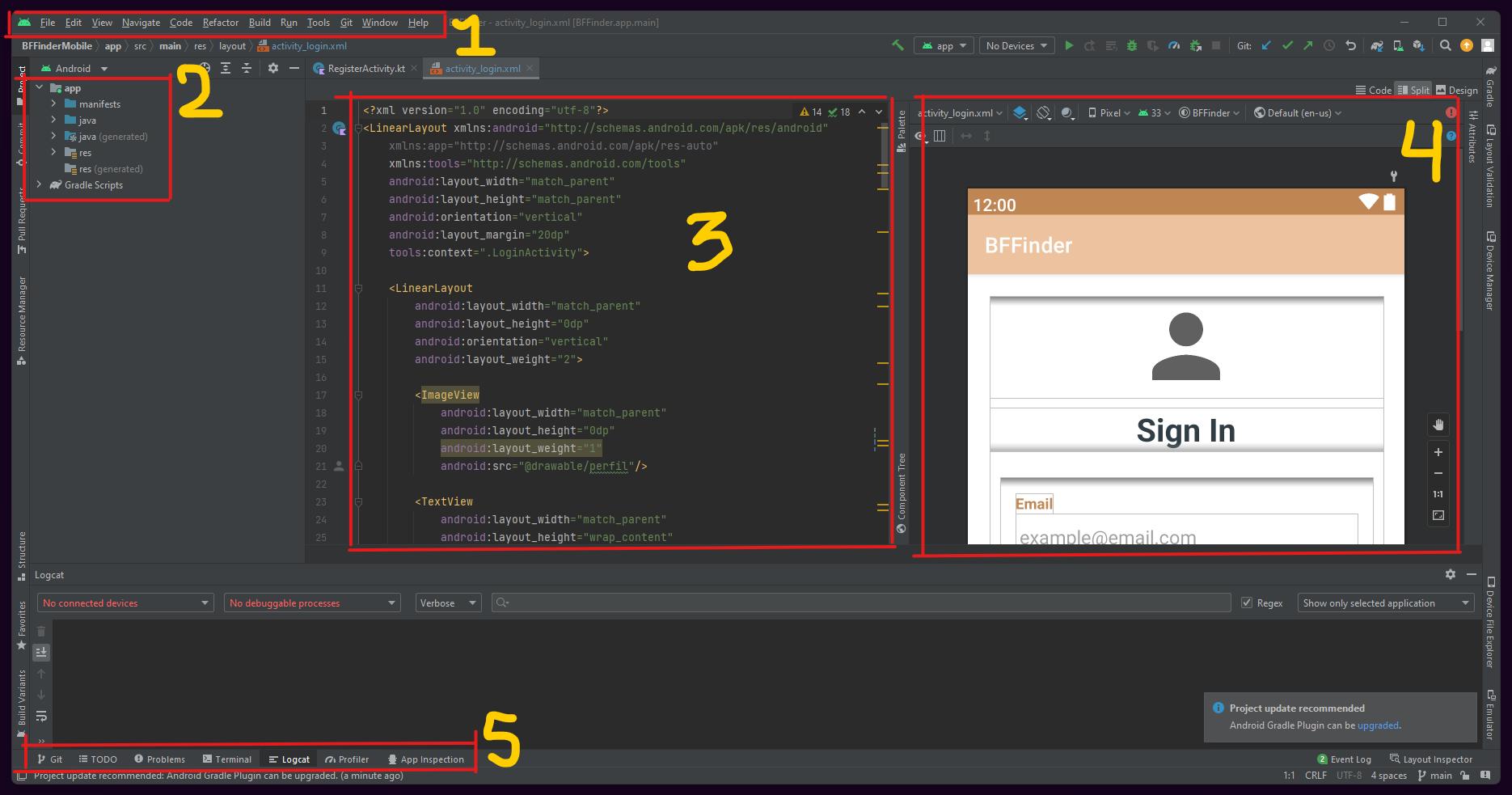
1. Instalación de Android Windows y Mac:

|  |  |  |
| --- | --- | --- |
| Paso | Windows | Mac |
| 1 opcion1 | Descargar archivo .exe y ejecutarlo o descomprimir el .zip en la carpeta **Android-studio** en la carpera **Program files** y luego ejecutar el archivo studio.exe dentro de **Android studio > bin.** | Ejecutar el archivo DMG de Android studio |
| 2 | Después de ejecutar el .exe, seguir los pasos de instalación según indica el sistema | Arrastrar Android studio hasta la carpeta de aplicaciones y luego iniciarlo |
| 3 |  | Seguir los pasos de configuración de Android que incluyen la descarga del SDK. |

1. Versiones importantes de Android y sus características principales:

|  |  |
| --- | --- |
| Android 5.0 Lollipop | Lanzado en el 2014, con el se mostró por primera vez el estilo de diseño **Material Design.** |
| Android 7.0-7.1 Nougat | En 2016, trajo por primera vez algunas características a Android como pantalla dividida y algunas otras características de optimización de batería, además de optimizar y mejorar Material Design. |
| Android 2.0-2.1 Eclair | La característica principal que trajo esta versión fue el panel de ajustes rápidos. |
| Android 10 | La característica mas notable que trajo Android 10 es la navegación por gestos en la pantalla, así que los dispositivos ya no necesitaban de botones **home** o **back** |

1. Interfaz de usuario del IDE Android Studio: Elementos principales



* 1. **Barra de herramientas**: permite realizar diferentes acciones tales como ejecutar la aplicación compilar, mostrar u ocultar elementos de la interfaz, acceder a la configuración, etc.
* 2. **Estructura del proyecto**: permite ver, navegar y editar la estructura de carpetas y archivos del proyecto. Hay varios modos de visualización, sin embargo, el modo por defecto es Android, cuya utilidad es permitir al usuario acceder de manera rápida a los módulos y archivos relevantes dentro del proyecto.
* 3. **Ventana del editor**: es la ventana sobre la cual el usuario puede modificar los archivos del proyecto (código). Esta ventana cambia su diseño en función del tipo de archivo que estemos modificando.
* 4. **Editor de diseño**: En realizad esta ventana o sub ventana hacer parte de la **ventana del editor**, se abre cuando el archivo que estamos modificando o visualizando es de diseño, y permite modificar el archivo mediante arrastrando los elementos de UI.
* 5. **Barra de ventana de herramientas**: está ubicada fuera de la ventana principal del IDE y su funcionalidad variada incluye desplegar ventanas de herramientas individuales como el Logcat (herramienta que permite visualizar la traza en ejecución) o la terminal (consola para ejecutar comandos).

1. Layouts:

**Definición y uso**: los Layouts son elementos contenedores que definen la estructura de una interfaz de usuario en una aplicación de manera vectorial, tales como una actividad.

**Tipos y su definición**:

* Constraint: son útiles para elaborar interfaces de grandes agrupaciones de elementos jerarquizados dinámicamente.
* Frame: Alinea los elementos de la interfaz en la parte izquierda de la pantalla, por lo que hay que manejar las márgenes para ubicar los componentes de la interfaz de manera visualmente agradable.
* Linear: layout por excelencia para alinear elementos de manera vertical u horizontal como formularios o botones.
* Coordinator: Indicado para elementos de la interfaz animados o altamente interactivos con el usuario.
* Relative: Ideal para utilizar en el modo de diseño de UI del IDE arrastrando los elementos, se recomienda solamente para interfaces sencillas o minimalistas debido a que pueden generar código desordenado.

1. Activity:

**Definición y uso**: Una actividad es una acción especifica que el usuario puede hacer, por lo general (pero no siempre) interactúa con el usuario. En este orden de ideas, la actividad de encarga de crear una ventana para que el usuario interactúe con la interfaz y pueda llevar a cabo la actividad.

**Tipos de activity:** son plantillas por defecto que provee Android studio para agilizar el trabajo al desarrollador dependiendo del modelo de actividad de desea desarrollar, algunos ejemplos son:

* No Activity
* Empty Activity
* Bottom Navigation Activity
* Scrolling Activity
* Login Activity

1. Componentes y sus atributos en Android:

* Cajas de texto: text, textSize, textColor, textStyle
* Etiquetas: text, autoText, digits, editable
* Spinner: textAlignament, gravity, prompt, spinnerMode,
* List View: divider, entries, dividerHeight
* View: es una clase que representa bloques de construcción básicos para los componentes.
* ImageButton: baseline, src,
* EditText: inputType, hint
* RadioGroup: onClick, visiblity, background

1. **Token**: En General, un token suele representar ‘otra’ cosa u ‘otro’ objeto. En programación un token es un símbolo que representa **algo**, a menudo un **hecho**. Por ejemplo, cuando un usuario hace login en alguna app, un token es creado para validar que el usuario esté logeado en el sistema, en este caso ese **algo** o **hecho** es el hecho de que el usuario está logeado.
2. **Gradle:** Gradle es una herramienta de automatización de compilación de código abierto diseñada para ser lo suficientemente flexible como para compilar cualquier tipo de software. Esta basado en Apache ant y apache Maven, pero a diferencia de Maven que utiliza XML, Gradle hace uso de un lenguaje especifico de dominio (Domain Specific Language) basado en Groovy para especificar la configuración del proyecto.

## Cuestionario

1. R/ **Un entorno de ejecución** ya que se ejecuta en el servidor y compila y ejecuta programas javascript, permitiendo establecer múltiples conexiones simultaneas.
2. R/ **RECIBE PETICIONES Y ENVIA RESPUESTAS** porque el cliente es ejecutado en el navegador y muestra la información al usuario, haciendo peticiones de información al servidor.
3. R/ **HACE PETICIONES, RECICE RESPUESTAS, MUESTRA LAS COSAS E INTERACTUA A TRAVES DEL NAVEGADOR.**
4. R/ **ORIENTADO A EVENTOS** ya que, para gestionar las peticiones en lugar de asignar un hilo para cada cliente, lo cual es bastante ineficiente por el uso de memoria, nodejs crea un evento por cada peticion que se le hace, este evento se gestiona de manera independiente evitando que todo el servidor se bloquee en caso de errores en tiempo de ejecución.
5. R/ **NADA** porque no existe el atribuyo -Y del comando npm start
6. R/ **REQ, RES** porque representan la peticion y la respuesta que se esta haciendo al servidor de nodejs. La solicitud que el usuario http envía se captura en un objeto Request (req) y la respuesta en un objeto Response (res).
7. R/ **UN OBJETO QUE REPRESENTA EL VALOR DE UNA OPERACIÓN QUE SERÁ RESUELTA EN EL FUTURO,** es decir, representa una operación que aun no se ha completado se espera saber si será resuelta o no.
8. R/ **VERDADERO** npm es el registro de software mas grande en el mundo y permite gestionar dependencias, compartir tu software en su registro y reutilizarlos en el futuro descargándolos.
9. R/ **FALSO** npm necesita conexión a internet porque descarga los módulos y paquetes de su repositorio en la nube
10. R/ **NODE\_MODULES.** El comando npm install <paquete> creará la carpeta node\_modules si no existe y guardará la descarga en esta carpeta.
11. R/ **NPM INSTALL -G EXPRESS** es el comando correcto ya que -g especifica que se instale de manera global
12. R/ **PACKAGE.JSON** es el archivo que utiliza npm para definir todos los módulos necesarios en un proyecto con sus versiones, metadatos, y comandos que el desarrollador puede especificar para la automatización.
13. R/ **NODEJS** es un sistema del lado del servidor ya que permite la compilación y ejecución de javascript que por defecto es un lenguaje interpretado por el navegador.
14. R/ esta basado en el lenguaje **JAVASCRIPT** según su documentación oficial.
15. R/ **MOTOR V8 DE CHROME** es el motor que ejecuta a Google Chrome y según la documentación de nodejs, este es el motor que se decidió para darle funcionamiento.
16. R/  **NODE –-VERSION Y NODE -V**  ambos sirven para verificar la versión.
17. R/ **NODE MIAPPNODE.JS** es el comando correcto para ejecutar archivos javascript.
18. R/ **ASINCRONO Y CONCURRENTE** por defecto javascript es asíncrono porque ejecuta tareas y mientras espera su resolución continua con la ejecución de la app y concurrente en nodejs ya que en este entorno recolecta eventos de procesamiento y ejecuta subtareas de manera independiente (no paralela).
19. R/ **RESOLVE, REJECT**  son los parámetros que recibe una promesa y sirven para indicar si una promesa se resolvió o fue rechazada, si se resuelve se utiliza el método resolve() en la función de la promesa y si fue rechazada se utiliza reject y se captura el error.
20. R/ **MANEJAR PROMESAS DE FORMA ASINCRONA** y **SIMPLIFICAR AUN MÁS LA FORMA EN QUE TRABAJAMOS CON LAS PROMESAS,** porque async es una instrucción que puede manejar cualquier función de manera asíncrona y además implementándola en las promesas hace que nuestro código sea mucho más robusto y fácil de construir.
21. R/ **Javascript** es un lenguaje de programación de tipado débil utilizado mayormente en el desarrollo web siendo por lo general interpretado por el navegador (no compilado) en tiempo de ejecución. **Node.js** es, en vagas palabras, un paquete de software construido en javascript que instalamos en una computadora, y permite “compilar” en un entorno de servidor evitando asi la necesidad de utilizar del lado del servidor otro lenguaje que sea compilado nativamente.
22. R/ El primer gran beneficio que logro identificar de utilizar node.js es que no es necesario aprender un nuevo lenguaje de programación para el lado del servidor, pudiendo asi construir toda una web completamente desacoplada con un solo lenguaje base.

Por otro lado, otra gran ventaja es que, al ser **javascript** el lenguaje predominante en el desarrollo web y uno de los mas utilizados, cuenta con una comunidad muy grande, lo cual pude garantizar un mejor soporte o una mayor cantidad de herramientas y librerías que podamos utilizar en nuestros proyectos.

1. R/ **ANGULAR** es un framework para el lenguaje javascript, al igual que node.js, sin embargo, son muy distintos. Empandando porque **node.js**, como ya lo mencioné anteriormente, es utilizado para construir aplicaciones web con javascript del lado del servidor (back-end) para proveer servicios a diferentes clientes. **Angular**, por otro lado y a pesar de estar construido con javascript (más específicamente typescript), es utilizado para construir aplicaciones web del lado del cliente (front-end), permitiéndole al usuario interactuar con una interfaz grafica y conectarse al servidor.
2. Un solo subproceso significa que, si tenemos varias instrucciones, estas serán ejecutadas en un solo hilo y de manera secuencial. En este orden de ideas, node.js es un framework de un solo subproceso (single thread) porque está enfocado en la velocidad y en la escalabilidad, por ende, aunque suene contradictorio, las tareas multihilo pueden llegar a ser menos eficientes porque tenemos varios núcleos ejecutando procesos, y alguno de esos procesos puede permanecer bloqueados mucho tiempo, bloqueando con él a la aplicación o al hilo principal.
3. R/
4. R/ Node.js puede ser ejecutado en cualquier computadora, no obstante, por lo general es implementado en servidores web para ejecutar aplicaciones back-end escritas en javascript.
5. R/ No, node.js no tiene las características de una API, porque no es una API, mas bien sirve para construir API’s en javascript gracias a que es un entorno de ejecución que permite compilar los servicios escritos en javascript (la API).
6. R/ Aunque se trata se dos conceptos distintos van de la mano, Una **operación asíncrona** implica que se ejecuta de manera independiente del hilo principal y que opera durante un tiempo indefinido hasta que concluye mientras el software continúa con sus tareas. Esto nos lleva al concepto de **no-bloqueante** que significa que una **tarea asíncrona** no detendrá la ejecución del programa (no lo bloquea) esperando hasta que se complete como en una operación **síncrona**, sino que el programa continúa con su ejecución normalmente e independiente de la tarea **asíncrona** que esta esperando a ser concluida. En conclusión, un evento **no bloqueante** sucede en una tarea asíncrona.
7. R/ El patrón de arquitectura **orientado a eventos** permite que las tareas que lleva a cabo el software o el flujo del mismo, dependen de los **eventos** o sucesos que el usuario desencadena, estos eventos transportan información sobre el estado o sobre el evento hacia el servidor o plataforma y posteriormente es ejecutada la lógica necesaria. Los programas que implementan este modelo son altamente útiles cuando queremos generar mucha interactividad con el usuario y que el programa responda en consecuencia
8. R/ Como mencioné anteriormente, node.js es un framework orientado a la velocidad, por esta razón se utiliza el motor v8 de Google Chrome, porque este es el motor más rápido para javascript creado hasta la actualidad, escrito en el lenguaje de bajo nivel C++, además el motor v8 es independiente del navegador en el que esté corriendo.
9. R/ repetida
10. R/
11. R/ **module.exports**  es un objeto especial incluido por defecto en cada archivo javascript en una aplicación node.js. el modulo es una variable que representa el modulo actual, y el exports es el objeto que será expuesto como modulo.
12. R/ este módulo exporta como un módulo cualquier cosa que se le asigne , un objeto, una cadena, etc.