**Capítulo I**

**Marco Teórico**

**Capitulo I. Marco Teórico**

Dentro del presente capítulo, se abordaron los aspectos referenciales, teóricos y conceptuales en los cuales se basó dicho proyecto para el referente desarrollo de una aplicación móvil que permita diferenciar a la empresa dentro de una de las mejores en el Estado de México, Hidalgo, Querétaro y Puebla.

* 1. ***Industria en el sector del acero***

Los metales y las aleaciones empleados en la industria y en la construcción pueden dividirse en dos grupos principales: materiales ferrosos y no ferrosos. Ferroso viene de la palabra *ferrum* que los romanos empleaban para el fierro o hierro; los materiales ferrosos son aquellos que contienen hierro como componente principal, mientras que los no ferrosos no contienen hierro. El acero es una aleación de hierro y de carbono, y su fabricación comienza con la reducción de hierro (producción de arrabio) para su posterior conversión en acero. A la planta industrial dedicada al proceso completo de producir acero a partir del mineral de hierro, se le denomina siderurgia, mientras que se le nombra acería a una planta industrial dedicada exclusivamente a la producción y elaboración de acero partiendo de otro acero o de hierro. (Economia, 2012)

* 1. ***Tecnología Móvil***

La tecnología móvil es la tecnología que va a dónde va el usuario. Consiste en dispositivos portátiles de comunicaciones bidireccionales, dispositivos de computación y la tecnología de red que los conecta.

Actualmente, la tecnología móvil se caracteriza por dispositivos habilitados para Internet como smartphones, tabletas y relojes. Estos son los últimos en una progresión que incluye buscapersonas bidireccionales, computadoras portátiles, teléfonos celulares (teléfonos plegables), dispositivos de navegación GPS y más.

Las redes de comunicaciones que conectan estos dispositivos se denominan en términos generales tecnologías inalámbricas. Permiten que los dispositivos móviles compartan voz, datos y aplicaciones (aplicaciones móviles).

La tecnología móvil es omnipresente y está creciendo. El número de usuarios de teléfonos inteligentes ha superado los 3 mil millones. (IBM)

* 1. ***Aplicación Móvil***

Una aplicación móvil consta esencialmente de dos partes: las aplicaciones nativas y la web móviles. Sin importar el tipo de aplicación que se decida usar, ambas deben proporcionar la misma calidad de información a los usuarios que las utilizan. Sin embargo, es importante considerar algunos aspectos a la hora de evaluar la calidad en el funcionamiento de las aplicaciones, como los espacios restringidos de navegación y el elevado costo de su desarrollo.

Cuando se desea desarrollar aplicaciones se debe tomar en cuenta que los lenguajes de programación para el desarrollo de aplicaciones varían de un sistema operativo a otro; esto conlleva que los diseñadores deban plantear varias veces la misma programación para ser utilizada en distintos dispositivos. Las aplicaciones nativas son aquellas creadas o desarrolladas, en este caso en particular, por las bibliotecas y que permiten acceder a los servicios y productos tradicionales y novedosos; este tipo de aplicaciones se encuentran habilitadas en las tiendas de aplicaciones, como, por ejemplo, iTunes Store y Google Play. (informacion, 2013)

* 1. ***Sistemas operativos móviles***

Es el software que se sitúa entre la máquina y los programas. Básicamente su función es administrar los recursos del sistema.

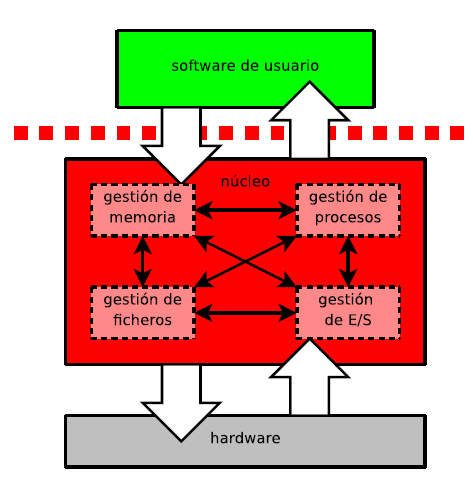


Ilustración 1 Sistemas Operativos (LH, 2020)

La parte más importante de un sistema operativo es el kernel o núcleo, que se encarga de facilitar a las distintas aplicaciones acceso seguro al hardware del sistema informático. Puesto que hay muchas aplicaciones y el acceso al hardware es limitado, el kernel decide también qué aplicación podrá hacer uso de un dispositivo de hardware y durante cuánto tiempo. (informacionales, 2020)

Los núcleos tienen como funciones básicas:

* Garantizar la carga y la ejecución de los procesos.
* Controlar las entradas/salidas.
* Proponer una interfaz entre el espacio núcleo y los programas del espacio del usuario.

En cuanto a las funciones principales de los sistemas operativos, tenemos las siguientes:

* Gestionar las transferencias de información internas.
* Proporcionar la comunicación de la máquina con los operadores.
* Controlar la ejecución de los programas con la detección de los errores.
* Encadenar automáticamente las tareas.
* Optimizar los recursos (memoria, unidad aritmética, etc.).
* Cargar y descargar automáticamente los programas en función del espacio de memoria y de los diferentes periféricos.
  1. ***Comparativa sistemas operativos***

A continuación, se presenta una comparación entre los dos grandes sistemas operativos móviles como son Android y iOS en la que se analizan sus principales características.

|  |  |  |
| --- | --- | --- |
| Sistema Operativo | Android | iOS |
| Interfaz | Material You | iOS 14: novedades, características y fecha de lanzamiento |
| Kernel | Linux | XNU |
| Tipo de SO | Abierto | Cerrado |
| Lenguaje de Programación nativo | Java | Swift |
| Seguridad | Susceptible a malware | Muy buena |
| Adaptabilidad | Excelente | Excelente |
| Multitarea | Si | Si |
| Standares Soportados | GSM, CDMA | GSM, CDMA |
| Hardware Soportado | Amplia gama de dispositivos | iPhone, iPad, iPod touch |
| Programa de productividad | Google Docs | iWork |
| Tienda de software | Google Play | App store |
| Apps | 500000+ | 650000+ |
| Soporte para Tablet | Si | Si |
| Expansión de almacenamiento | Almacenamiento en la nube | Almacenamiento en la nube |
| Soporte en la nube | Google Sync, Google Drive | iCloud |
| Interfaz de usuario | Más técnico | Fácil |
| Asistente de voz | Google | Siri |
| Personalización | Profunda | Limitada |
| Vida de la batería | Mayor duración | Media duración |

Tabla 1 Comparativa SO

* 1. ***Lenguaje unificado de modelado (UML)***

El Lenguaje Unificado de Modelado (*Unified Modeling Language, UML*) es un lenguaje estándar para escribir planos de software. UML puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucre una gran cantidad de software. (Grady Booch, 2006)

UML es apropiado para modelar desde sistemas de información empresariales hasta aplicaciones distribuidas basadas en la web, e incluso para sistemas embebidos de tiempo real muy exigentes. Es un lenguaje muy expresivo, que cubre todas las vistas necesarias para desarrollar y luego desplegar tales sistemas. Aunque sea expresivo, UML no es difícil de aprender ni de utilizar. Aprender a aplicar UML de modo eficaz comienza por crearse un modelo conceptual del lenguaje, lo cual requiere aprender tres elementos principales: los bloques básicos de construcción de UML, las reglas que dictan cómo pueden combinarse esos bloques y algunos mecanismos comunes que se aplican a lo largo de todo el lenguaje.

UML es sólo un lenguaje y, por tanto, es tan sólo una parte de un método de desarrollo de software. UML es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental. (Grady Booch, 2006)

***1.6.1 Visión general de UML***

UML es un lenguaje para

* Visualizar
* Especificar
* Construir
* Documentar

los artefactos de un sistema con gran cantidad de software. (Grady Booch, 2006)

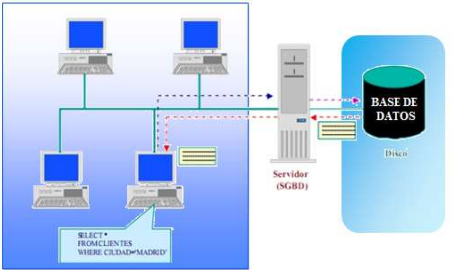
* 1. ***Arquitectura cliente – servidor***

¿Qué es una arquitectura? Una arquitectura es un entramado de componentes funcionales que, aprovechando diferentes estándares, convenciones, reglas y procesos, permite integrar una amplia gama de productos y servicios informáticos, de manera que pueden ser utilizados eficazmente dentro de la organización. Se debe señalar que, para seleccionar el modelo de una arquitectura, hay que partir del contexto tecnológico y organizativo del momento y, que la arquitectura Cliente/Servidor requiere una determinada especialización de cada uno de los diferentes componentes que la integran. (Orfali, 1998)

¿Qué es un cliente? Es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente. (Orfali, 1998)

¿Qué es un servidor? Es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LAN o WAN, para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos fax, procesamiento de imágenes, etc. (Orfali, 1998)

Ilustración 2 Arquitectura Cliente-Servidor (Orfali, 1998)



* 1. ***Servidor web***

Los servidores web son los encargados de recibir las peticiones referidas a páginas o elementos de la web a través del protocolo http. Normalmente es un software alojado en un ordenador servidor. Normalmente es el navegador el que pide al servidor web el recurso que desea el usuario, para finalmente recibir dicho recurso (si fue válida la petición) y traducirle si es necesario a su forma legible por el usuario (es decir la traducción de HTML la hace el navegador). (Asenjo, 2011)

* + 1. ***Servidores de aplicaciones web***

Podemos entender que es una ampliación de los anteriores. Es decir, son servidores web, pero que tienen capacidad de almacenar y gestionar aplicaciones web. Entendiendo que una aplicación web es un servicio al que los usuarios acceden a través de la web. Este tipo de servidores no sólo sirven para atender peticiones http, sino que además son capaces de entender instrucciones de lenguajes avanzados de la web y traducirlas o bien son capaces de acceder a recursos de otros servidores. Ese proceso se hace de forma transparente al usuario, es decir el usuario pide el servicio a través, normalmente, de su navegador y el servidor de aplicaciones atiende la petición, e interpreta el código de la aplicación a fin de traducirle y mostrar al usuario el resultado de forma entendible por su navegador (es decir en formato HTML). (Asenjo, 2011)

A la forma de trabajar de un servidor de aplicaciones, se le conoce normalmente como arquitectura de tres capas (a veces se habla de más capas). Una primera capa es la del navegador que es capaz de traducir código del lado del cliente (HTML, JavaScript, CSS, Flash). Para ello esa capa debe de disponer de todos los componentes necesarios para hacer esa labor en el ordenador del usuario. La segunda capa la forma el servidor de aplicaciones en su labor de traducir código en el lado del servidor (JSP, PHP, Ruby on Rails, Cold Fussion) y convertirlo al formato entendible por el navegador. La tercera capa son todos los servicios a los que accede el servidor de aplicaciones para poder realizar la tarea encomendada a la aplicación (por ejemplo, el acceso a la base de datos).

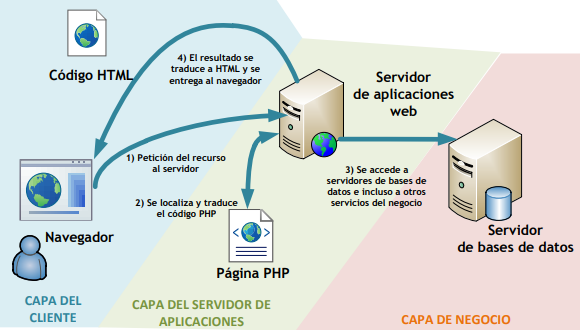
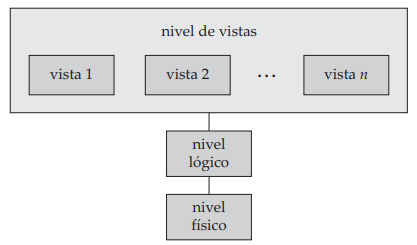


Ilustración 3 Funcionamiento de un servidor (Asenjo, 2011)

* 1. ***Base de datos***

Son repositorios de datos estructurados, organizados, relacionados. Uno de los propósitos de base de datos es proporcionar a los usuarios una visión abstracta de los datos. Conjunto de datos comunes que se almacenan sin redundancia para ser útiles en diferentes aplicaciones. (Silberschatz, 2006)

Ilustración 4 Los tres niveles de la abstracción de datos (Silberschatz, 2006)



* 1. ***Web app nativa***

Las Apps Nativas son aplicaciones desarrolladas específicamente para una plataforma o dispositivo en particular, lo que les permite funcionar sin necesidad de programas externos. Están estrechamente asociadas a dispositivos móviles, y son ideales para aprovechar al máximo las características y funcionalidades de cada dispositivo. Sin embargo, el principal inconveniente de las Apps Nativas es que, para estar disponible en diferentes sistemas, se debe duplicar el proyecto por cada plataforma, lo que aumenta significativamente los tiempos y costos de desarrollo. (GSOFT, s.f.)

* + 1. **Web app**

Las Web apps son aplicaciones accesibles a través de un navegador web, lo que las hace más económicas y fáciles de desarrollar. Además, con el diseño responsive pueden adaptarse a cualquier dispositivo. Sin embargo, su principal desventaja es que no pueden acceder a todas las características específicas del dispositivo, lo que limita la experiencia de usuario, y su nivel de seguridad puede depender del navegador utilizado. (GSOFT, s.f.)

* + 1. **Web app hibrida**

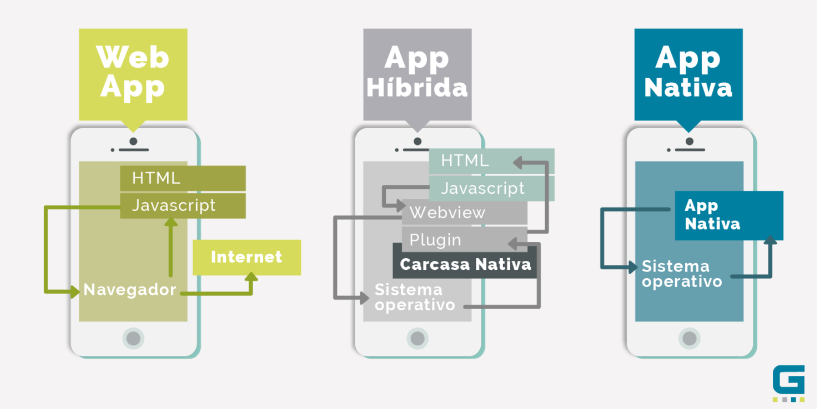
Las aplicaciones híbridas combinan las ventajas de las aplicaciones web y nativas, permitiendo el uso de algunas funciones hardware específicas mediante plugins en un WebView. Son independientes del sistema operativo, lo que reduce la inversión inicial en el proyecto. Sin embargo, la experiencia de usuario no es tan buena como en las aplicaciones nativas y su rendimiento y velocidad de carga son más lentos. (GSOFT, s.f.)

Ilustración 5 Web App, Hibrida y Nativa (GSOFT, s.f.)

* 1. ***Modelo vista controlador***

Es un patrón de arquitectura de las aplicaciones software

* Separa la lógica de negocio de la interfaz de usuario
* Facilita la evolución por separado de ambos aspectos
* Incrementa reutilización y flexibilidad

Utilizado en múltiples frameworks

* Java Swing
* Java Enterprise Edition (J2EE)
* XForms (Formato XML estándar del W3C para la especificación de un modelo de proceso de datos XML e interfaces de usuario como formularios web)
* GTK+ (escrito en C, toolkit creado por Gnome para construir aplicaciones gráficas, inicialmente para el sistema X Window)
* ASP.NET MVC Framework (Microsoft)
* Google Web Toolkit (GWT, para crear aplicaciones Ajax con Java)
* Apache Struts (framework para aplicaciones web J2EE)
* Ruby on Rails (framework para aplicaciones web con Ruby)
  1. ***Variables***

Las variables son elementos que se utilizan para almacenar información temporalmente en la memoria de la aplicación. Estas variables pueden contener diferentes tipos de datos, como números, cadenas de texto, valores booleanos, objetos y otros tipos de datos.

* + 1. ***Independiente***

Aplicación móvil para la cotización, orden de pedidos de materiales para la construcción de infraestructura de metal, materiales ferreteros y de herrajes, facturación, consulta de catálogo, entregas, entre otros.

* + 1. ***Dependiente***

Facilitar a los clientes sus cotizaciones, ordenes de entrega, pedidos a domicilio, facturaciones, en las diferentes sucursales de la empresa.

* + 1. ***Matriz de variables***

|  |  |  |  |
| --- | --- | --- | --- |
| Variables independientes | Palabras clave | Indicadores | Instrumentos |
| Aplicación móvil para la cotización, orden de pedidos, de materiales para la construcción de infraestructura de metal, materiales ferreteros y de herrajes, facturación, consultas de catálogos, entregas, entre otros. | * Aplicación móvil. * Cotizaciones * Pedidos * Facturación * Entregas * Consultas | * Nivel de satisfacción de los gerentes de las sucursales. * Empleo de medios tecnológicos. | * Observaciones * Calificaciones mediante app |

Tabla 2 Variables Independientes

|  |  |  |  |
| --- | --- | --- | --- |
| Variables Dependientes | Palabras clave | Indicadores | Instrumentos |
| Facilitar a los clientes sus cotizaciones, ordenes de entrega, pedidos a domicilio, facturaciones, en las diferentes sucursales de la empresa. | * Facilitar a los clientes sus pedidos | * Nivel de satisfacción del cliente * Cantidad de pedidos por el medio tecnológico | * Calificaciones mediante la app. |

Tabla 3 Variables Dependientes

* 1. ***Determinación y descripción de herramientas para el desarrollo de la aplicación móvil***

Se determino realizar este proyecto para los sistemas operativos Android y iOS, ya que representan sistemas de software libre para teléfonos y dispositivos móviles, lo cual permite a bajo costo desarrollar una aplicación como la que se requiere sin la necesidad de un gran presupuesto.

Por otra parte, el desarrollo del proyecto se describe en dos partes:

* Aplicación móvil
* Web Administrador

Por la parte de aplicación móvil se desarrolló con el sistema operativo Android y iOS, utilizando sus librerías y un framework compatible con la tecnología que utiliza.

El sistema operativo Android es una excelente plataforma para desarrollar aplicaciones, por ser de software libre y estar apadrinado por Google. Para un desarrollador Android las ventajas son obvias: no hay formularios tediosos para darse de alta, tampoco diferentes certificados que se debe obtener para probar y distribuir las aplicaciones como el casi de Apple, en Android solo hay que pagar una sola vez para poder subir aplicaciones a Google Play, hay libertad total para publicar, no hay que esperar a que la aplicación sea revisada, sino que, al ser subida, inmediatamente estarán disponible, que en caso contrario a lo que es Apple.

Ilustración 6 iOS Y Apple

* 1. ***Android***

Android es una plataforma de software basada en el núcleo de Linux. Permite controlar dispositivos por medio de bibliotecas desarrolladas o adaptadas por Google mediante el lenguaje de programación java.

Android es de código abierto que cualquier desarrollador puede crear y desarrollar aplicaciones escritas en lenguaje C u otros lenguajes y compilarlas a código nativo de ARM (API de Android).

Inicialmente, Android fue desarrollada por Google Inc. aunque poco después de unió *Open Handset Alliance*, un consorcio de 48 compañías de hardware, software y telecomunicaciones, las cuales llegaron a un acuerdo para proporcionar los estándares de código abiertos para dispositivos móviles.

Google, sin embargo, ha sido quien ha publicado la mayoría de código fuente de Android bajo la licencia de Software Apache, una licencia de software libre y de código abierto a cualquier desarrollador. (Computing, 2020)

* 1. ***Arquitectura de Android***

Android es una pila de software de código abierto basado en Linux creada para una variedad amplia de dispositivos y factores de forma. En la siguiente figura se muestran los componentes principales de la plataforma Android. (Studio, 2020)

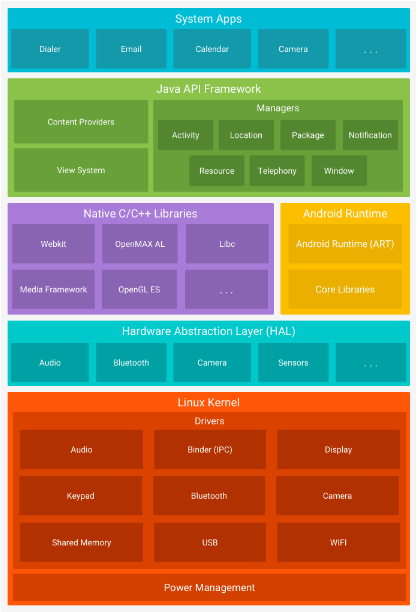


Ilustración 7 Arquitectura Android (Studio, 2020)

* + 1. ***Kernel de Linux***

La base de la plataforma Android es el kernel de Linux. El tiempo de ejecución de Android (ART), por ejemplo, se basa en el kernel de Linux para sus funcionalidades subyacentes, como la generación de subprocesos y la administración de memoria de bajo nivel. El uso del kernel de Linux permite que Android aproveche funciones clave de seguridad y, al mismo tiempo, permite a los fabricantes de dispositivos desarrollar controladores de hardware para un kernel conocido. (Studio, 2020)

* + 1. ***Capa de abstracción de hardware***

La capa de abstracción de hardware (HAL) brinda interfaces estándares que exponen las capacidades de hardware del dispositivo al marco de trabajo de la API de Java de nivel más alto. Cada módulo de biblioteca en la HAL implementa una interfaz para un tipo específico de componente de hardware, como el módulo de la cámara o de Bluetooth. Cuando el marco de trabajo de una API realiza una llamada para acceder a hardware del dispositivo, el sistema Android carga el módulo de biblioteca correspondiente al componente de hardware en cuestión. (Studio, 2020)

* + 1. ***Tiempo de ejecución de Android***

En los dispositivos con Android 5.0 (nivel de API 21) o versiones posteriores, cada aplicación ejecuta sus propios procesos con sus propias instancias del tiempo de ejecución de Android (ART). El ART está diseñado para ejecutar varias máquinas virtuales en dispositivos con poca memoria, ejecutando archivos DEX, un formato de código de bytes diseñado especialmente para Android y optimizado para ocupar un espacio mínimo de memoria. El ART crea cadenas de herramientas, como Jack, y compila fuentes de Java en código de bytes DEX que se pueden ejecutar en la plataforma Android. (Studio, 2020)

Entre las funciones principales del ART se encuentran:

* Compilación ahead-off-time (AOT) y just-in-time (JIT)
* Recolección optimizada de elementos no utilizados (GC)
* En Android 9 (nivel de API 28) y versiones posteriores, se convierten los archivos de formato ejecutable (DEX) de un paquete de aplicaciones a un código de máquina más compacto.
* Esto mejora la compatibilidad con la depuración, el generador de perfiles de muestras dedicado, las excepciones de diagnóstico detalladas, los informes de fallos y la capacidad de establecer puntos de control para supervisar campos específicos.

Antes de Android 5.0 (nivel de API 21), *Dalvik* era el entorno de ejecución del sistema operativo. Si una aplicación se ejecuta bien en el ART, también debería funcionar en *Dalvik*, pero no necesariamente al revés.

En Android, también se incluye un conjunto de bibliotecas de entorno de ejecución centrales que proporcionan la mayor parte de la funcionalidad del lenguaje de programación Java. Algunas funciones del lenguaje Java 8 están incluidas en el marco de trabajo de la API de Java. (Studio, 2020)

* + 1. ***Bibliotecas C/C++ nativas***

Muchos de los componentes y servicios centrales del sistema Android, como el ART y la HAL, se basan en código nativo que requiere bibliotecas nativas escritas en C y C++. La plataforma Android proporciona API del marco de trabajo de Java para exponer la funcionalidad de algunas de estas bibliotecas nativas a las aplicaciones. Por ejemplo, se puede acceder a OpenGL ES a través de la API de OpenGL de Java del marco de trabajo de Android para agregar a una aplicación compatibilidad con dibujos y la manipulación de gráficos 2D y 3D. (Studio, 2020)

En caso de que se necesite desarrollar una aplicación que requiera C o C++, se puede utilizar el NDK de Android para acceder a algunas de estas bibliotecas de plataformas nativas directamente desde el código nativo. (Studio, 2020)

* + 1. ***Java API framework***

Todas las funciones del sistema operativo de Android están disponibles mediante API escritas en el lenguaje Java. Estas API proporcionan los cimientos necesarios para crear aplicaciones de Android, simplificando la reutilización de componentes del sistema y servicios centrales y modulares, como los siguientes:

* Un sistema de vista enriquecido y extensible que se puede utilizar para construir la interfaz de usuario de una aplicación, incluyendo listas, cuadrículas, cuadros de texto, botones e incluso un navegador web integrable.
* Un administrador de recursos que proporciona acceso a recursos sin código, como cadenas localizadas, gráficos y archivos de diseño.
* Un administrador de notificaciones que permite que todas las aplicaciones muestren alertas personalizadas en la barra de estado.
* Un administrador de actividad que gestiona el ciclo de vida de las aplicaciones y proporciona una pila de retroceso de navegación común.
* Proveedores de contenido que permiten que las aplicaciones accedan a datos desde otras aplicaciones, como la aplicación de Contactos, o compartan sus propios datos.

Los desarrolladores tienen acceso total a las mismas API del marco de trabajo que utilizan las aplicaciones del sistema Android. (Studio, 2020)

* + 1. ***Apps del sistema***

En la plataforma Android, se incluye un conjunto de aplicaciones centrales para correo electrónico, mensajería SMS, calendarios, navegación en Internet y contactos, entre otros elementos. Estas aplicaciones incluidas en la plataforma no tienen un estado especial entre las aplicaciones que el usuario elige instalar. Por lo tanto, una aplicación externa se puede convertir en el navegador web, el sistema de mensajería SMS o incluso el teclado predeterminado del usuario (con algunas excepciones, como la aplicación *Settings* del sistema). (Studio, 2020)

Las aplicaciones del sistema funcionan como aplicaciones para los usuarios y brindan capacidades claves a las cuales los desarrolladores pueden acceder desde sus propias aplicaciones. Por ejemplo, si una aplicación intenta entregar un mensaje SMS, no es necesario que compile esa funcionalidad por su cuenta. En su lugar, puede invocar la aplicación de SMS que ya está instalada para entregar un mensaje al receptor especificado. (Studio, 2020)

* 1. ***Ventajas y desventajas Android***

Android es un sistema operativo móvil desarrollado por Google, que se utiliza en una amplia variedad de dispositivos móviles de diferentes fabricantes. Se caracteriza por su flexibilidad y personalización, y por estar estrechamente integrado con los servicios de Google. Sin embargo, puede presentar problemas de fragmentación, seguridad y fluidez en algunos dispositivos.

***1.16.1 Ventajas***

El código en Android es abierto ya que Google libero Android bajo licencia Apache. Cualquier persona puede realizar una aplicación para Android.

Hoy día hay más de 650.000 aplicaciones disponibles para teléfonos Android, aproximadamente 2/3 son gratis. Además, la libertad de código permite adaptar Android a bastantes otros dispositivos además de teléfonos celulares. Está implantado en tabletas, GPS, relojes, microondas, incluso hay por internet una versión de Android para PC.

El sistema Android es capaz de hacer funcionar a la vez varias aplicaciones y además se encarga de gestionarlas, dejarlas en modo suspensión si no se utilizan e incluso cerrarlas si llevan un periodo determinado de inactividad. (Computing, 2020)

***1.16.2 Desventajas***

Es muy vulnerable debido a que es de código abierto. El hecho de que sea de código abierto trae esta gran desventaja, ya que el sistema operativo es utilizado por muchos usuarios aprovechando las fallas del sistema.

Necesidad de descargar aplicaciones adicionales desde Google Play para optimizar el sistema operativo. Por ejemplo, Android no gestiona bien el uso de la multitarea. Esto quiere decir que esta procesando aplicaciones que no se han cerrado por completo en segundo plano. También, se puede decir que puede descargar una aplicación para el ahorro de energía, ya que si no se gastaría mucha batería.

La configuración del celular mediante Android no es tan sencilla de realizar. Tiene funcionalidades que pueden resultar difícil de configurar si no se sabe mucho de tecnología.

No tiene un soporte de actualización como en el caso de Apple con su sistema operativo iOS. En esta compañía, todos los iPhone, iPads, iPod, etcétera, se pueden actualizar a la última versión de software brindando por la empresa de una forma sencilla sin importar el hardware. (Computing, 2020)

* 1. ***iOS***

iOS es un derivado de OS X que, como a su vez está basado en Darwin BSD, es un sistema operativo UNIX (lo cual, por su estructura, implica robustez y fiabilidad).



Ilustración 8 SO iOS (informacionales, 2020)

En junio de 2010 pasó a llamarse iOS, tras la presentación a principios de año del segundo gran producto disruptivo de Apple, el iPad, que crearía el segmento de las tabletas y daría el pistoletazo de salida a la denominada por Steve Jobs "era Post-PC" (para espanto y desesperación de su gran rival Microsoft, totalmente descolocado en dispositivos móviles y dependiente, en lo fundamental, de sus productos informáticos centrados en el PC).

Aunque iOS va por detrás de Android en número de usuarios, en porcentaje de tráfico de datos de dispositivos móviles, iOS bate ampliamente a Android. (informacionales, 2020)

* 1. ***Ventajas y desventajas iOS***

iOS es un sistema operativo móvil desarrollado por Apple, que se utiliza en dispositivos móviles como el iPhone y el iPad. Se caracteriza por su estabilidad y fluidez, así como por su seguridad, gracias al control que ejerce Apple sobre el ecosistema de aplicaciones. Sin embargo, puede presentar limitaciones en cuanto a la personalización del sistema operativo y de las aplicaciones, y los dispositivos iOS suelen ser más caros que otros dispositivos móviles.

* + 1. ***Ventajas***

Interfaz de usuario. Algo que distingue a Apple es el haber logrado una interfaz de usuario altamente intuitiva y elegante, que hasta el usuario menos experimentado puede aprender a utilizar en cuestión de minutos.

Actualizaciones. Los usuarios de iOS tienen garantizadas prácticamente todas las actualizaciones de sus equipos, minutos después de que la compañía las anuncia, que a diferencia de Android en ocasiones tardan mucho tiempo y en otros casos nunca llegan.

Sincronización. La sincronización de tu multimedia en iOS puede resultar muy sencillo, gracias a que también puedes hacerlo sin depender de un cable USB. (Computing, 2020)

* + 1. ***Desventajas***

Personalización. Desafortunadamente la extrema sencillez de iOS es también su talón de Aquiles, ya que cuando un usuario intenta realizar un cambio o algún tipo de personalización puede ver muy complicada su existencia gracias a que Apple no habilito la opción.

Mapas. El rey indiscutible de los mapas el Google, y aunque Apple ha avanzado bastante en este tema, todavía le falta mucho camino por recorrer.

Pocas opciones. Adquirir un iDevices no es tarea difícil, ya que el catalogo de opciones se limita a tres modelos de smartphones y dos de tabletas (gama media y alta, colores y espacio de almacenamiento) además que si hoy se opta por una versión básica no se podrá escalar en ningún momento. (Computing, 2020)

* 1. ***C#***

C# (pronunciado *"See Sharp"*) es un lenguaje de programación moderno, orientado a objetos y con seguridad de tipos. C# permite crear muchos tipos de aplicaciones sólidas y seguras que se ejecutan en .NET. C# tiene sus raíces en la familia de lenguajes C, C++, Java y JavaScript.

C# proporciona construcciones de lenguaje para respaldar directamente los conceptos de programación orientada a objetos, lo que lo convierte en un lenguaje natural para crear y utilizar componentes de software. Desde su origen, se han agregado funciones a C# para admitir nuevas cargas de trabajo y prácticas de diseño de software emergentes. En esencia, C# es considerado un lenguaje orientado a objetos. (Microsoft, 2023)

En cuanto al control de versiones, C# hace hincapié en garantizar que los programas y las bibliotecas puedan evolucionar de manera compatible a lo largo del tiempo. En el diseño de C#, se tuvieron en cuenta las consideraciones de versiones, lo que incluye los modificadores separados virtual y override, las reglas de resolución de sobrecarga de métodos y la compatibilidad con declaraciones explícitas de miembros de interfaz. (Microsoft, 2023)

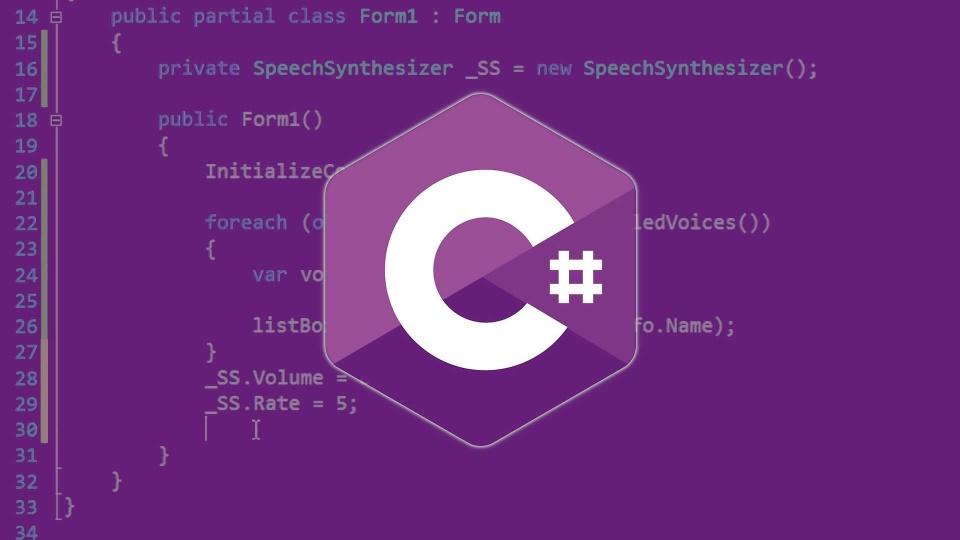


Ilustración 9 C# Lenguaje de Programación (S.AS, 2023)

* 1. ***Xamarin Forms***

Xamarin.Forms es un marco de interfaz de usuario de código abierto que permite a los desarrolladores crear aplicaciones de Xamarin.Android, Xamarin.iOS y Windows a partir de una única base de código compartida. Con Xamarin.Forms, los desarrolladores pueden crear interfaces de usuario en XAML con el código subyacente en C#. Dichas interfaces se representan como controles nativos de alto rendimiento en cada plataforma. (Microsoft, Microsoft Learn, 2023)

Xamarin.Forms proporciona una API coherente para crear elementos de interfaz de usuario en todas las plataformas, que se puede implementar en XAML o C#, y admite el enlace de datos para patrones como Model-View-ViewModel (MVVM).

Durante la ejecución, Xamarin.Forms utiliza renderizadores de plataforma para convertir los elementos de la interfaz de usuario multiplataforma en controles nativos en Xamarin.Android, Xamarin.iOS y UWP. Esto permite a los desarrolladores obtener el aspecto, la sensación y el rendimiento nativos mientras se benefician de compartir código entre plataformas.

En general, las aplicaciones de Xamarin.Forms constan de una biblioteca .NET Standard compartida y proyectos de plataforma individuales. La biblioteca compartida contiene las vistas XAML o C# y cualquier lógica de negocio, como servicios, modelos u otro código. Los proyectos de la plataforma contienen cualquier lógica o paquete específico de la plataforma que requiera la aplicación. (Microsoft, Microsoft Learn, 2023)

* 1. ***Arquitectura de Xamarin.Forms***

Uno de los principios clave en la creación de aplicaciones multiplataforma es diseñar una arquitectura que permita maximizar el código compartido entre plataformas. Para ello, es importante cumplir con los siguientes principios de Programación Orientada a Objetos:

* Encapsulación: asegurar que las clases y capas arquitectónicas sólo expongan una API mínima que realice las funciones requeridas y oculte los detalles de implementación. A nivel de clase, esto significa que los objetos se comporten como "cajas negras" y que el código consumidor no necesite saber cómo realizan sus tareas. A nivel arquitectónico, significa implementar patrones como *Façade* que fomenten una API simplificada que organice las interacciones más complejas en nombre del código en capas más abstractas. Esto significa que el código de la interfaz de usuario, por ejemplo, sólo debe ser responsable de mostrar pantallas y aceptar la entrada del usuario, sin interactuar directamente con la base de datos. De manera similar, el código de acceso a datos sólo debe leer y escribir en la base de datos, pero nunca interactuar directamente con botones o etiquetas.
* Separación de responsabilidades: asegurarse de que cada componente, tanto a nivel arquitectónico como de clase, tenga un propósito claro y bien definido. Cada componente debe realizar sólo sus tareas definidas y exponer esa funcionalidad a través de una API accesible para otras clases que necesiten usarla.
* Polimorfismo: la programación en una interfaz (o clase abstracta) que admite múltiples implementaciones permite que el código central se pueda escribir y compartir entre plataformas, mientras interactúa con las características específicas de la plataforma.

El resultado natural es una aplicación modelada a partir del mundo real o entidades abstractas con capas lógicas separadas. Separar el código en capas hace que las aplicaciones sean más fáciles de entender, probar y mantener. Se recomienda que el código de cada capa esté físicamente separado, ya sea en directorios o incluso en proyectos separados para aplicaciones muy grandes, así como lógicamente separado mediante el uso de espacios de nombres. (Microsoft, Microsoft Learn, 2021)

* + 1. ***Capas de aplicación típicas***

En la arquitectura de una aplicación, se pueden identificar varias capas, cada una con una función específica:

* Capa de datos: encargada de la persistencia de datos no volátiles, generalmente implementada mediante una base de datos SQLite, aunque también podría utilizarse archivos XML u otro mecanismo adecuado.
* Capa de acceso a datos: es un envoltorio alrededor de la capa de datos que proporciona acceso de creación, lectura, actualización y eliminación (CRUD) a los datos sin exponer los detalles de implementación al código que la llama. Por ejemplo, el DAL puede contener instrucciones SQL para consultar o actualizar los datos, pero el código de referencia no necesita saber esto.
* Capa empresarial: también conocida como capa lógica empresarial o BLL, contiene definiciones de entidades empresariales (el modelo) y lógica empresarial. Se recomienda utilizar el patrón de Fachada Comercial en esta capa.
* Capa de acceso al servicio: se utiliza para acceder a los servicios en la nube, desde servicios web complejos (REST, JSON, WCF) hasta la recuperación simple de datos e imágenes de servidores remotos. Encapsula el comportamiento de red y proporciona una API simple para ser consumida por las capas de aplicación e interfaz de usuario.
* Capa de aplicación: este código suele ser específico de la plataforma (generalmente no compartido entre plataformas) o específico de la aplicación (generalmente no reutilizable). Una buena prueba para determinar si colocar código en la capa de aplicación o en la capa de interfaz de usuario es (a) determinar si la clase tiene controles de visualización reales o (b) si se puede compartir entre varias pantallas o dispositivos (por ejemplo, iPhone y iPad).
* Capa de interfaz de usuario (UI): esta capa está orientada al usuario y contiene pantallas, widgets y los controladores que los administran. (Microsoft, Microsoft Learn, 2021)

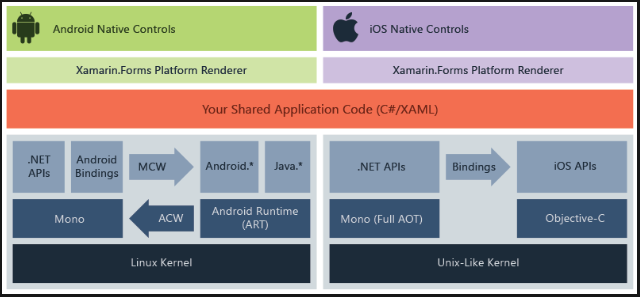


Ilustración 10 Arquitectura (Microsoft, Microsoft Learn, 2023)

* 1. ***Diseño de interfaz de usuario***

La interfaz de usuario de una aplicación de Xamarin.Forms está construida con objetos que se asignan a los controles nativos de cada plataforma de destino. Esto permite que las aplicaciones específicas de la plataforma para iOS, Android y la plataforma universal de Windows usen el código de Xamarin. Forms contenido en una biblioteca de .NET Standard. (Microsoft, Microsoft Learn, 2023)

Los cuatro grupos de control principales que se usan para crear la interfaz de usuario de una aplicación de Xamarin.Forms son los siguientes:

* Pages: Las páginas de Xamarin.Forms representan pantallas de aplicaciones móviles multiplataforma.

Estos elementos visuales ocupan toda o la mayor parte de la pantalla. Un *Page* objeto representa un *ViewController* en iOS y un *Page* en la plataforma universal de Windows. En Android, cada página ocupa la pantalla como un *Activity*, pero las páginas de Xamarin.Forms no *Activity* son objetos.

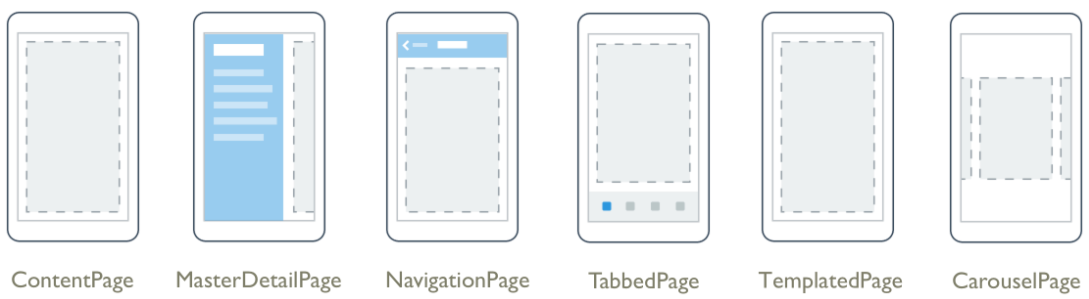


Ilustración 11 Páginas Xamarin.Forms (Microsoft, Microsoft Learn, 2023)

* Layouts: Los diseños de Xamarin.Forms se usan para componer controles de interfaz de usuario en estructuras visuales.

Las clases *Layout* y *Layout<T>* en Xamarin.Forms son subtipos especializados de vistas que actúan como contenedores para vistas y otros diseños. La *Layout* clase en sí se deriva de *View*. Un *Layout* derivado normalmente contiene lógica para establecer la posición y el tamaño de los elementos secundarios en las aplicaciones de Xamarin.Forms.

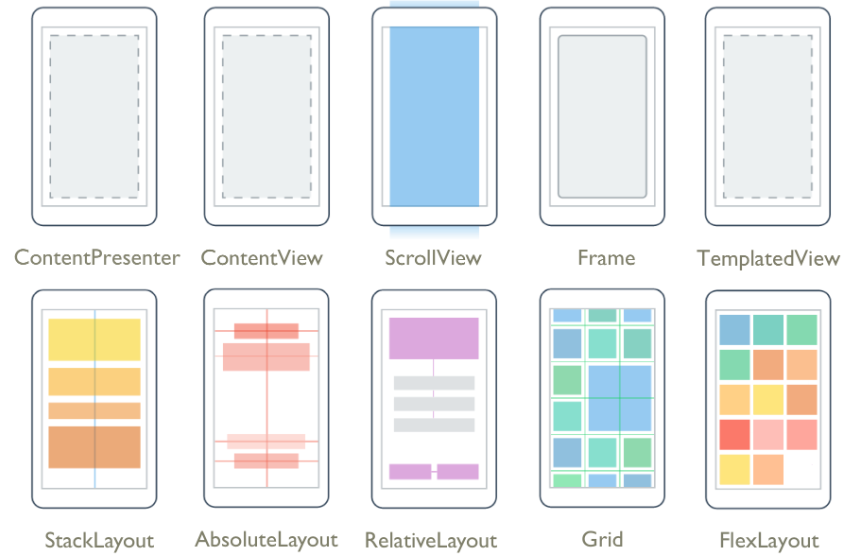


Ilustración 12 Diseños Xamarin.Forms. (Microsoft, Microsoft Learn, 2023)

* Views: Las vistas de Xamarin.Forms son los componentes básicos de las interfaces de usuario móviles multiplataforma.

Las vistas son objetos de la interfaz de usuario, como etiquetas, botones y controles deslizantes, que se conocen comúnmente como controles o widgets en otros entornos de programación gráfica. Todas las vistas admitidas por Xamarin.Forms se derivan de la *View* clase. Se pueden dividir en varias categorías: vistas para presentación, vistas que inician comandos, vistas para establecer valores, vistas para editar texto, vistas para indicar actividad, vistas que muestran colecciones.

* Cells: Las celdas de Xamarin.Forms se pueden agregar a *ListViews* y *TableViews*.

Una celda es un elemento especializado que se utiliza para los elementos de una tabla y describe cómo debe representarse cada elemento de una lista. La *Cell* clase deriva de *Element*, de la cual *VisualElement* también deriva. Una celda no es en sí misma un elemento visual; en cambio, es una plantilla para crear un elemento visual.

*Cell* se utiliza exclusivamente con controles *ListView* y *TableView*. (Microsoft, Microsoft Learn, 2023)

* 1. ***Ciclo de vida de la aplicación***

El ciclo de vida de desarrollo móvil es, en gran medida, parecido al SDLC para aplicaciones web o de escritorio. Al igual que con estos, normalmente hay 5 partes importantes del proceso:

* Inicio: todas las aplicaciones se inician con una idea. Normalmente, la idea se perfecciona en una base sólida para una aplicación.
* Diseño: la fase de diseño consiste en definir la experiencia del usuario (UX) de la aplicación (como cuál es el diseño general, cómo funciona, etc.), así como convertir esa experiencia del usuario en un diseño de interfaz de usuario (UI) adecuado, normalmente con la ayuda de un diseñador gráfico.
* Desarrollo: normalmente, es la fase con un uso más intensivo de recursos, esta es la creación real de la aplicación.
* Estabilización: cuando el desarrollo ha avanzado lo suficiente, normalmente el control de calidad empieza a probar la aplicación y se corrigen los errores. A veces, una aplicación pasará a una fase beta limitada en la que una audiencia de usuarios más amplia tiene la oportunidad de usarla, enviar comentarios y notificar cambios.
* Implementación

Muchas de estas partes se superponen, por ejemplo, es común que el desarrollo siga mientras se finaliza la interfaz de usuario e incluso puede afectar al diseño de la interfaz de usuario. Además, una aplicación puede estar en una fase de estabilización al mismo tiempo que se agregan nuevas características a una nueva versión.

Estas fases se pueden usar en varias metodologías de SDLC como Agile, Spiral, Waterfall, etc. (Microsoft, Microsoft Learn, 2023)

* 1. ***Integración de servicios web***

Las aplicaciones de Xamarin.Android a menudo necesitan acceso a los datos (ya sea desde una base de datos local o desde la nube), y muchas de estas aplicaciones consumen servicios web implementados mediante una amplia variedad de tecnologías.

La mayoría de las aplicaciones tienen algún requisito para guardar datos en el dispositivo localmente. A menos que la cantidad de datos sea trivialmente pequeña, esto generalmente requiere una base de datos y una capa de datos en la aplicación para administrar el acceso a la base de datos. Android tiene el motor de base de datos SQLite "incorporado" y el acceso a los datos se simplifica con la plataforma de Xamarin que viene con el proveedor de datos SQLite. (Microsoft, Microsoft Learn, 2021)

Xamarin.Android admite API de acceso a la base de datos como:

* Marco ADO.NET.
* Biblioteca de terceros SQLite-NET.

La transferencia de estado representacional (REST) es un estilo arquitectónico para crear servicios web. Las solicitudes REST se realizan a través de HTTP utilizando los mismos verbos HTTP que utilizan los navegadores web para recuperar páginas web y enviar datos a los servidores. Los verbos son:

* + 1. ***Reset***
* GET: esta operación se utiliza para recuperar datos del servicio web.
* POST: esta operación se utiliza para crear un nuevo elemento de datos en el servicio web.
* PUT: esta operación se utiliza para actualizar un elemento de datos en el servicio web.
* PATCH: esta operación se utiliza para actualizar un elemento de datos en el servicio web describiendo un conjunto de instrucciones sobre cómo se debe modificar el elemento. Este verbo no se usa en la aplicación de ejemplo.
* DELETE: esta operación se utiliza para eliminar un elemento de datos en el servicio web.

Las API de servicios web que se adhieren a *REST* se denominan API *RESTfull* y se definen mediante:

* URI básico.
* Métodos HTTP, como GET, POST, PUT, PATCH o DELETE.
* Tipo de medio para los datos, como la notación de objetos de JavaScript (JSON).

La simplicidad de REST ha ayudado a convertirlo en el método principal para acceder a servicios web en aplicaciones móviles. (Microsoft, Microsoft Learn, 2021)

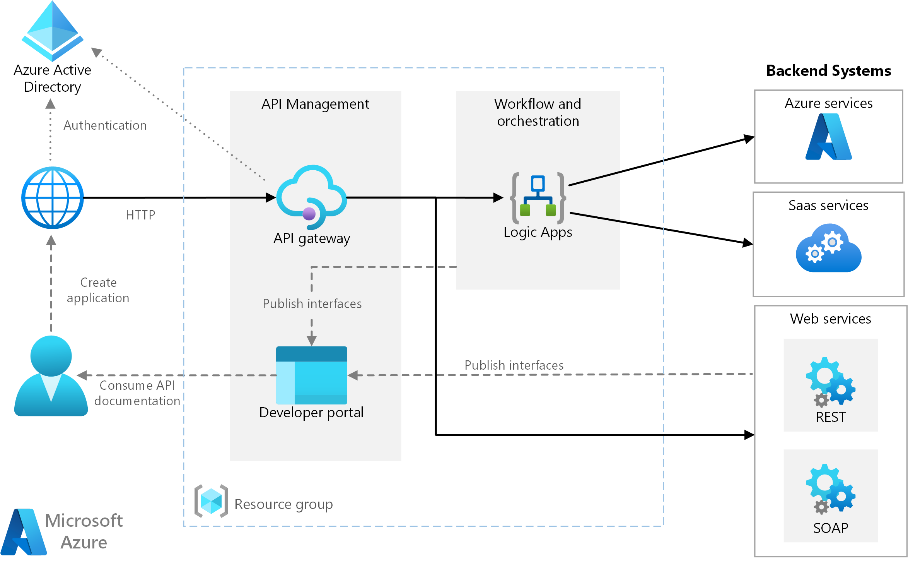


Ilustración 13 Integración de Servicios Web (Microsoft, Microsoft Learn, 2021)

* 1. ***Procesamiento de pagos***

Una de las claves de esta nueva era digital en la que estamos inmersos es la captación y análisis de datos. Para ello, la interconexión y las integraciones están siendo de gran ayuda ya que permiten conectar diferentes herramientas y sistemas con el fin de centralizar y organizar la información que estas proporcionan a los negocios.

Uno de los principales objetivos de las integraciones es la mejora en la productividad empresarial gracias a poder desarrollar soluciones totalmente personalizadas. Hoy en día, se desarrollan todo tipo de aplicaciones móviles que pueden ser integradas con los sistemas y softwares de una empresa como un ERP, CRM o SAP.

* + 1. ***Stripe y Redsys***

La primera es una empresa estadounidense que cuenta con una amplia documentación y permite realizar diversos tipos de operaciones relacionadas con pagos, tales como pagos únicos, recurrentes, manejo de suscripciones, transferencias entre cuentas, entre otros. Stripe ofrece dos modalidades de tarifas: la integrada y la personalizada, en función de diferentes parámetros de la empresa, como el volumen de ventas. La tarifa integrada tiene una comisión por transacción del 1.4% + 0.25€ (sujeta a variaciones).

Por su parte, Redsys es una opción comúnmente ofrecida por entidades bancarias para la integración de un TPV Virtual en aplicaciones móviles. La documentación de Redsys es adecuada y la integración de las diferentes formas de pago de la aplicación queda en manos del desarrollador. Las comisiones en este caso varían según la entidad bancaria con la que se contrate el servicio. Se realizará un análisis comparativo de estas dos opciones en términos de facilidad de uso, costos y funcionalidades para determinar cuál es la opción más adecuada para las necesidades de una aplicación de procesamiento de pagos móviles. (Technologies, 2021)

* 1. **Análisis de datos**

Xamarin Forms es una plataforma de desarrollo de aplicaciones móviles multiplataforma que permite crear aplicaciones para iOS, Android y Windows Phone utilizando un único código base. Esta plataforma es una opción popular para el desarrollo de aplicaciones de ventas debido a su flexibilidad y capacidad para integrar herramientas de análisis de datos.

Para integrar herramientas de análisis de datos en una aplicación móvil de ventas utilizando Xamarin Forms, se deben seguir los siguientes pasos:

Identificar las herramientas de análisis de datos adecuadas: Es importante elegir la herramienta de análisis de datos más adecuada para la aplicación, considerando las opciones disponibles en el mercado, como Google *Analytics, Firebase Analytics* y *Mixpanel*, entre otras.

Integrar la herramienta de análisis de datos en la aplicación: Una vez seleccionada la herramienta, se debe integrar en la aplicación mediante la adición de un paquete NuGet o un SDK específico. La documentación proporcionada por la herramienta puede ofrecer instrucciones detalladas sobre cómo realizar esta integración.

Configurar la herramienta de análisis de datos: Para recopilar los datos relevantes para el negocio, como el seguimiento de las ventas, el seguimiento de los clientes y los productos más populares, entre otros, se debe configurar la herramienta de análisis de datos.

Analizar los datos recopilados: Los datos recopilados deben ser analizados para obtener información valiosa sobre el comportamiento de los clientes y mejorar la toma de decisiones de negocio. Por ejemplo, se pueden evaluar patrones de compra, productos más populares, frecuencia de compras, entre otros.

Utilizar los datos para mejorar la experiencia del usuario y aumentar las ventas: Los datos recopilados y analizados pueden ser utilizados para mejorar la experiencia del usuario y aumentar las ventas. Por ejemplo, se pueden personalizar ofertas basadas en patrones de compra de los clientes, mejorar las recomendaciones de productos o servicios, ajustar precios, entre otros. (Microsoft, Microsoft Learn, 2021)

* 1. ***Integración con sistemas de ERP***

*Enterprise Resource Planning* es un software de gestión empresarial que ayuda a automatizar las funciones empresariales esenciales de una empresa. Para muchas organizaciones, ERP es el sistema de registro que recopila información de los departamentos de toda la organización, incluidos contabilidad, fabricación, cadena de suministro, ventas, marketing y más, para garantizar que haya una única fuente de información veraz para sus datos.

La integración de ERP conecta y sincroniza el software de ERP con otras aplicaciones y fuentes de datos. La integración del software ERP le brinda una vista unificada de la información de diferentes sistemas en tiempo real, ya sea que los datos se originen en el ERP o en otros sistemas. La importancia de la integración con ERP es que aumenta la eficiencia de los procesos y flujos de trabajo comerciales y aumenta la colaboración entre equipos. (Jitterbit, 2023)

Para integrar una aplicación móvil de venta de productos con un sistema de planificación de recursos empresariales (ERP) utilizando Xamarin Forms, se deben identificar el sistema ERP adecuado, configurarlo, desarrollar una API, integrar la API en la aplicación móvil de ventas y construir una interfaz de usuario que permita a los usuarios acceder a información crítica del sistema ERP.

Identificar el sistema ERP adecuado: Existen diversos sistemas ERP disponibles en el mercado, tales como SAP, Oracle, Microsoft Dynamics, entre otros. Es importante elegir el sistema ERP más adecuado para el negocio.

Configurar el sistema ERP: El sistema ERP debe ser configurado para permitir el acceso a la información relevante para la aplicación móvil de ventas, como los datos de inventario, precios, clientes, pedidos, entre otros.

Desarrollar una API para el sistema ERP: Es necesario desarrollar una API (Interfaz de Programación de Aplicaciones) que permita la comunicación entre el sistema ERP y la aplicación móvil de ventas. Esta API debe proporcionar acceso a los datos relevantes del sistema ERP.

Integrar la API en la aplicación móvil de ventas: Una vez desarrollada la API, se debe integrar en la aplicación móvil de ventas mediante la adición de una referencia o un paquete NuGet. En general, la documentación de la API proporciona instrucciones detalladas sobre cómo realizar esta integración.

Construir una interfaz de usuario en Xamarin Forms: Utilizando Xamarin Forms, se debe construir una interfaz de usuario que permita a los usuarios acceder a la información crítica del sistema ERP. La interfaz de usuario debe ser fácil de usar y proporcionar acceso rápido a la información relevante.

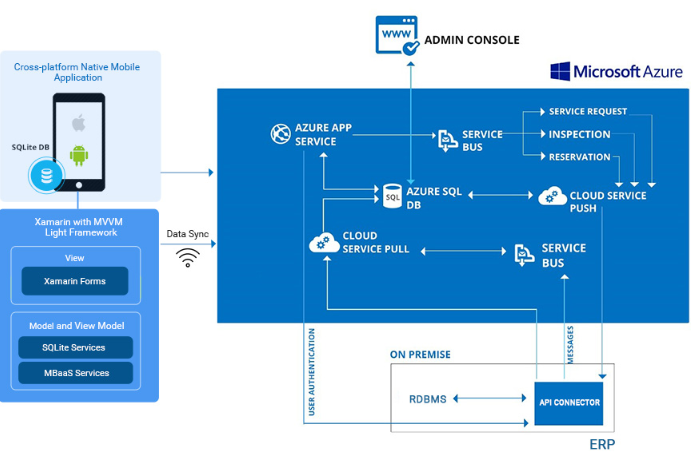


Ilustración 14 ERP APP Móvil (Consulting, 2023)

En el Capítulo 1, se ha realizado un exhaustivo marco teórico para el desarrollo de una aplicación móvil destinada a una empresa de metal y productos ferreteros. A través de la investigación y análisis de las necesidades de la empresa y el mercado, se ha establecido una base sólida para crear una solución tecnológica que impulse el crecimiento y mejore la experiencia del cliente.

La elección de Xamarin Forms en Visual Studio como plataforma de desarrollo para la aplicación se justifica por su capacidad para ofrecer una experiencia de usuario uniforme y eficiente en múltiples plataformas, asegurando así una mayor cobertura y accesibilidad para los clientes potenciales.

En el Capítulo 2, se abordará la metodología SCRUM como enfoque para la gestión del desarrollo del proyecto. La aplicación de SCRUM permitirá un enfoque ágil y colaborativo, donde se trabajará de manera iterativa y transparente para alcanzar los objetivos establecidos en el marco teórico. A través de sprints, la aplicación avanzará de manera progresiva y se mantendrá alineada con las necesidades y expectativas de la empresa y los usuarios. Con esta conjunción entre la sólida base teórica y la metodología ágil, se prevé un desarrollo exitoso de la aplicación móvil para la empresa de metal y productos ferreteros, lo que permitirá optimizar sus operaciones, fortalecer su presencia en el mercado y brindar una experiencia de usuario satisfactoria y diferenciada.