UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

MÁSTER UNIVERSITARIO DE INVESTIGACIÓN EN INGENERÍA DE SOFTWARE Y SISTEMAS INFORMÁTICOS

ITINERARIO: INGENIERÍA DE SOFTWARE – 31105151

DESARROLLO DE UN SISTEMA DE GESTIÓN DE CONTENIDOS PARA APLICACIONES MÓVILES

Autor: Sergi Pedra Pagès
Director: Ismael Abad Cardiel
Curso: 2017 – 18
Convocatoria: Septiembre 2018

MÁSTER UNIVERSITARIO DE INVESTIGACIÓN EN INGENERÍA DE SOFTWARE Y SISTEMAS INFORMÁTICOS

ITINERARIO: INGENIERÍA DE SOFTWARE – 31105151

DESARROLLO DE UN SISTEMA DE GESTIÓN DE CONTENIDOS PARA APLICACIONES MÓVILES

Autor: Sergi Pedra Pagès
Director: Ismael Abad Cardiel

DECLARACIÓN JURADA DE AUTORÍA DEL TRABAJO CIENTÍFICO, PARA LA DEFENSA DEL TRABAJO FIN DE MASTER

Fecha: 07/09/2018

Quién suscribe:

Autor(a): Sergi Pedra Pagès
D.N.I/N.I.E/Pasaporte.: 43746509-A

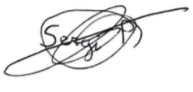
Hace constar que es la autor(a) del trabajo:

DESARROLLO DE UN SISTEMA DE GESTIÓN DE CONTENIDOS PARA APLICACIONES MÓVILES

En tal sentido, manifiesto la originalidad de la conceptualización del trabajo, interpretación de datos y la elaboración de las conclusiones, dejando establecido que aquellos aportes intelectuales de otros autores, se han referenciado debidamente en el texto de dicho trabajo.

DECLARACIÓN:

- ✓ Garantizo que el trabajo que remito es un documento original y no ha sido publicado, total ni parcialmente por otros autores, en soporte papel ni en formato digital.
- ✓ Certifico que he contribuido directamente al contenido intelectual de este manuscrito, a la génesis y análisis de sus datos, por lo cual estoy en condiciones de hacerme públicamente responsable de él.
- ✓ No he incurrido en fraude científico, plagio o vicios de autoría; en caso contrario, aceptaré las medidas disciplinarias sancionadoras que correspondan.

Fdo. 

IMPRESO TFDM05_AUTORPBL

AUTORIZACIÓN DE PUBLICACIÓN

CON FINES ACADÉMICOS

Impreso TFdM05_AutorPbl. Autorización de publicación y difusión del TFM para fines académicos

Autorización

Autorizo/amos a la Universidad Nacional de Educación a Distancia a difundir y utilizar, con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firma del/los Autor/es

Juan del Rosal, 16
28040, Madrid

Tel: 91 398 89 10
Fax: 91 398 89 09

www.issi.uned.es

RESUMEN

Hoy en día las empresas confían en soluciones informáticas basadas en aplicaciones móviles, como consecuencia de las ventajas ofrecidas por los dispositivos móviles. Existen distintas plataformas, la más utilizada es Android, la instalación de sus aplicaciones se realiza mediante ficheros con formato APK.

El problema del desarrollo de aplicaciones móviles es su alto coste, porque es necesario personal especializado y con altos conocimientos informáticos. Debido a este coste, es complicado para las empresas con menos recursos acceder a este tipo de soluciones informáticas.

En este trabajo se quiere plantear y desarrollar una solución que permita a las empresas desarrollar aplicaciones móviles Android a bajo coste. Esta solución está basada en un editor que permite gestionar los contenidos de la aplicación móvil y un modelo generativo que a partir de los contenidos crea la aplicación móvil y la empaqueta en un fichero APK.

Lista de palabras clave

Aplicación móvil, dispositivo móvil, Android, APK, gestor de contenidos y modelo generativo.

EXECUTIVE SUMMARY

Nowadays, the companies trust with computing solutions based on mobile applications, as a consequence for the advantages offered by mobile devices. There are different platforms, the most used is Android, the installation of their applications is done by files with APK format.

The problem of the mobile application development is its high cost, because it is necessary specialized people with high computer skills. Due to this cost, it is difficult for companies with fewer resources to access this kind of computer solutions.

This work wants to set out and develop a solution which allows companies to develop Android mobile applications at low cost. This solution is based on an editor which allows managing the mobile application content and a generative model which creates the mobile application from the managed contents and packages it into APK file.

Contenido

RESUMEN 5
Lista de palabras clave 5
EXECUTIVE SUMMARY 5
1. Introducción 10
2. Objetivos y ámbito del proyecto 11
   a. Objetivos 11
   b. Ámbito del proyecto 11
   c. Estructura del trabajo 12
3. Estado del arte 13
   a. Aplicación móvil 13
   b. Desarrollo de aplicaciones móviles 14
   c. Desarrollo de aplicaciones móviles multiplataforma 15
   d. Frameworks de desarrollo de aplicaciones móviles 16
      i. Android SDK 16
      ii. Xamarin 18
      iii. Apache Cordova 22
   e. Problemas en el desarrollo de aplicaciones móviles 23
4. Definición del dominio 25
5. Descripción del proceso de aplicación del dominio 27
6. Diseño de la solución 32
   a. Aplicación Web 32
      i. Parte Back 32
      ii. Parte Front 54
   b. Modelo generativo 63
      i. Analizador 64
      ii. Generador 67
7. Resultados de la solución 91
8. Conclusiones y trabajo futuro 98
   a. Conclusiones 98
   b. Trabajo futuro 98
9. Apéndices 100
   a. Preparación del entorno 100

i. Instalación de las herramientas 100
ii. Publicación de la solución 102
iii. Configuración de la solución 112
10. Bibliografía 114
11. Lista de acrónimos 117

Ilustración 1: Uso de las plataformas en España en Abril del 2017 13
Ilustración 2: Entornos de desarrollo de las distintas plataformas 14
Ilustración 3: Arquitectura de una aplicación Android 17
Ilustración 4: Arquitectura de compartición de código entre distintas plataformas 19
Ilustración 5: Arquitectura de Shared Project 19
Ilustración 6: Arquitectura de Portable Class Libraries 20
Ilustración 7: Estrategias de desarrollo de interfaz de usuario 20
Ilustración 8: Páginas y plantillas de Xamarin.Forms 21
Ilustración 9: Arquitectura de DependencyService 21
Ilustración 10: Arquitectura de aplicaciones Apache Cordova 22
Ilustración 11: Diagrama de características del dominio de las aplicaciones móviles 25
Ilustración 12: Diagrama BPMN del proceso generativo 27
Ilustración 13: Flujo del sistema de gestión de contenidos desde el punto de vista del usuario 28
Ilustración 14: Maqueta de las pantallas de edición de contenido de las aplicaciones móviles 29
Ilustración 15: Maqueta de las pantallas de la aplicación móvil generada 29
Ilustración 16: Flujo de trabajo interno del sistema de gestión de contenidos 30
Ilustración 17: Representación de la manipulación del dominio del modelo generativo 32
Ilustración 18: Arquitectura parte Back de la aplicación Web 33
Ilustración 19: Diagrama de clases del componente Repository 33
Ilustración 20: Diagrama de clases del componente Modelo 36
Ilustración 21: Diagrama de clases del componente Dominio 38
Ilustración 22: Diagrama de clases del componente Adaptadores 44
Ilustración 23: Diagrama de clases del componente Aplicación 46
Ilustración 24: Diagrama de clases del componente Web API 50
Ilustración 25: Arquitectura parte Front de la aplicación Web 54
Ilustración 26: Diagrama de clases de la capa Modelos 54
Ilustración 27: Diagrama de clases de la capa Cliente 55
Ilustración 28: Diagrama de clases de los modelos de vista de la capa presentación 58
Ilustración 29: Diagrama del modelo generativo 64
Ilustración 30: Diagrama de clases de la estructura de trabajo 64
Ilustración 31: Diagrama de clases del analizador 65
Ilustración 32: Diagrama de flujo del generador 67
Ilustración 33: Arquitectura aplicación móvil 68
Ilustración 34: Diagrama de clases del módulo generador de modelos y entidades 70
Ilustración 35: Diagrama de clases del módulo generador de vistas 74
Ilustración 36: Diagrama de clases del módulo generador de Web Api 81
Ilustración 37: Diagrama de clases del módulo generador de acceso a datos 83
Ilustración 38: Diagrama de clases del módulo generador de configuración 86
Ilustración 39: Diagrama de clases del módulo generador de la apk 89
Ilustración 40: Diagrama de clases del módulo de publicación de la parte servidora 90
Ilustración 41: Página de inicio del gestor de contenidos para la generación del primer ejemplo 91
Ilustración 42: Páginas del gestor de contenidos para la edición de las páginas del primer ejemplo 91

Ilustración 43: Páginas del gestor de contenidos para la edición de los controles del primer ejemplo 92
Ilustración 44: Página de inicio y menú de navegación del primer ejemplo 92
Ilustración 45: Página que lista los registros del primer ejemplo 93
Ilustración 46: Página de detalle de los registros del primer ejemplo 93
Ilustración 47: Página de inicio del gestor de contenidos para la generación del segundo ejemplo 94
Ilustración 48: Páginas del gestor de contenidos para la edición de las páginas listado del segundo ejemplo 94
Ilustración 49: Páginas del gestor de contenidos para la edición de las páginas detalle del segundo ejemplo 95
Ilustración 50: Menú de navegación del segundo ejemplo 96
Ilustración 51: Páginas que listan los registros del segundo ejemplo 96
Ilustración 52: Páginas que detallan los registros del segundo ejemplo 97

1. Introducción

Hoy en día es una realidad que los teléfonos móviles son una parte imprescindible de nuestras vidas, permitiendo a las personas realizar multitud de operaciones mediante las distintas aplicaciones existentes en el mercado.

Debido al auge del mercado de la movilidad las empresas, cada día están apostando más claramente por ellos, a pesar de que los costes de desarrollo de aplicaciones móviles son bastante altos. Son necesarios distintos perfiles especializados debido a que existen multitud de plataformas móviles, con sus distintos sistemas operativos, sus lenguajes de desarrollo y sus formas de gestionar las opciones nativas.

Además de ser necesarios perfiles especializados para el desarrollo de aplicaciones móviles, también son necesarios perfiles con unos altos conocimientos informáticos para poder trabajar con las tecnologías existentes en el mercado de desarrollo de estas aplicaciones.

Estos altos costes, provocan un impedimento a las pequeñas y medianas empresas a la hora de realizar una apuesta por este mercado. Analizando la situación actual del mercado de la movilidad, queda bastante claro que existe una necesidad real de abaratar los costes de producción de las aplicaciones de dispositivos móviles para que todas las empresas puedan acceder a él.

Este mercado tiene bastantes paralelismos con el mercado Web donde, al principio, solamente eran unas pocas empresas las que podían apostar por él, pero con el paso del tiempo ha ido apareciendo nuevas tecnologías que permiten desarrollar pequeñas aplicaciones Web o páginas Web de una forma mucho más simple y barata.

Algunos ejemplos de estas tecnologías son las siguientes: plataforma ASP.NET Web Forms, plataforma ASP.NET MVC, framework Ruby on Rails, plataforma JavaServer Pages, framework AngularJS, CMS Joomla o CMS WordPress.

También se está produciendo una evolución en las tecnologías de desarrollo de aplicaciones móviles, algunos ejemplos de ello son Xamarin o Apache Cordova, enfocadas al desarrollo multi-plataforma.

El presente proyecto se centra en la búsqueda de una solución a la necesidad de abaratar los costes de producción de las aplicaciones móviles. Para ello se realiza una propuesta, un diseño y una implementación de dicha solución, con la finalidad de determinar si realmente es posible abaratar los costes.

2. Objetivos y ámbito del proyecto

Este apartado define los objetivos marcados para el proyecto y detalla el ámbito en que se va a centrar.

a. Objetivos

El primer paso consistirá en realizar un estudio de algunas de las tecnologías existentes de desarrollo de aplicaciones móviles para la plataforma Android. También se investigará la posibilidad de generar automáticamente paquetes de instalación de aplicaciones móviles a partir de alguna de las tecnologías.

A continuación, a partir de las investigaciones y estudios realizados en el primer paso, se definirá una solución que permita abaratar los costes y simplificar la producción de las aplicaciones móviles.

Finalmente, se procederá al diseño e implementación de la solución planteada en el anterior paso y, con el resultado final, se extraerán una serie de conclusiones para comprobar si se ha logrado ofrecer una solución que simplifique la producción de aplicaciones móviles.

Los objetivos propuestos son los siguientes.

- Buscar y encontrar una solución que permita reducir y simplificar los desarrollos de las aplicaciones móviles.
- Generar de manera automática paquetes de instalación de aplicaciones móviles.
- Facilitar que los usuarios puedan crear aplicaciones móviles con la solución implementada, sin la necesidad de ser un perfil especializado ni tener altos conocimientos en informática.
- Proponer y diseñar un dominio para definir el contenido de una aplicación móvil.
- Diseñar e implementar el proceso de generación en el dominio definido.
- Diseñar e implementar, mediante tecnología Web, un editor que facilite al usuario la manipulación del dominio del modelo generativo.
- Diseñar e implementar un modelo generativo capaz de transformar la manipulación del dominio por parte del usuario al resultado deseado.

b. Ámbito del proyecto

El ámbito de este proyecto se centrará en la implementación de un sistema que permita abaratar los costes de producción de aplicaciones móviles. El uso de este sistema ha de estar enfocado a perfiles poco especializados, otorgando la posibilidad que personas con pocos conocimientos informáticos sean capaces de crear sus propias aplicaciones móviles.

Además, el sistema ha de estar abierto a poder desarrollar aplicaciones de distintas categorías (sociales, productivas, informativas, etc.) y que hagan uso de las distintas opciones nativas que ofrecen los dispositivos móviles (cámara, GPS, etc.).

El dominio que se definirá será aplicable a cualquier categoría de aplicación ya sea multiplataforma o de plataforma específica, pero la aplicación móvil resultado solamente se tendrá en consideración para la plataforma Android.

Las aplicaciones móviles resultado estarán basadas en una arquitectura cliente servidor. La parte cliente estará estructurada por la capa de presentación, la capa de cliente Rest-Api y la capa de modelos. En cambio, la parte servidora estará estructurada por la capa servidora, la capa de persistencia de datos y las entidades del dominio.

c. Estructura del trabajo

El trabajo está organizado en ocho apartados. El primer apartado es el responsable de realizar una introducción al problema. El siguiente apartado habla de los objetivos y el ámbito del proyecto.

El tercer apartado trata de algunas tecnologías que existen hoy en día para ayudar en el desarrollo de aplicaciones móviles.

En el cuarto apartado se define un dominio capaz de representar las aplicaciones móviles. En el siguiente apartado se describe el proceso planteado para la generación de aplicaciones móviles.

En el sexto apartado se realiza el diseño del proceso de generación planteado en el apartado anterior, y en el siguiente se muestra los resultados obtenidos con la implementación de la solución.

El octavo apartado detalla las conclusiones extraídas y enumera las futuras líneas de desarrollo en el caso de seguir trabajando con el proyecto.

Los dos últimos apartados contienen la bibliografía utilizada para la realización de este trabajo y los apéndices.

3. Estado del arte

En este apartado se realiza una breve introducción de qué es una aplicación móvil y de las distintas estrategias y frameworks de desarrollo existentes. También se detectan los problemas que representan a la hora de crear las aplicaciones.

a. Aplicación móvil

Una aplicación móvil es un programa de computación pensado para ejecutarse en un dispositivo móvil, tableta u otro tipo de dispositivo [9]. Se ha de tener en cuenta que estas aplicaciones solamente son compatibles para una determinada plataforma, las más conocidas son: Android, iOS y Windows. Como se puede ver en el siguiente diagrama sacado de [8], la plataforma Android es la más usada.

Ilustración 1: Uso de las plataformas en España en Abril del 2017

Muchas de estas aplicaciones ya vienen preinstaladas en el dispositivo, aunque también es posible adquirirlas a través de las tiendas de aplicaciones de los distintos distribuidores de los sistemas operativos [9]. Las tiendas de aplicaciones más conocidas son App Store, Google Play y Windows Store.

La instalación manual de una aplicación Android se realiza con la ejecución de ficheros con formato APK en los dispositivos que se desea instalar. Estos archivos se comportan como empaquetador de todas las partes que componen la aplicación Android. Antes de empaquetar sus partes es necesario compilarlo para Android [10].

Las aplicaciones móviles cada día son más comunes, existiendo una gran variedad de ellas enfocadas a objetivos muy distintos. Según [11], una posible categorización de las aplicaciones móviles puede ser la siguiente.

- Entretenimiento: Aplicaciones con la finalidad de entretener al usuario, los juegos son el ejemplo más claro de la categoría.
- Sociales: Aplicaciones con el objetivo de incrementar la comunicación de distintos tipos de usuarios, esta es posible mediante la construcción de redes. Facebook es la aplicación social más exitosa.
- Utilitarias y de productividad: Aplicaciones que se comportan como herramientas para ayudar en la resolución de problemas o la facilitación de actividades. Evernote es una aplicación que se puede ubicar dentro de la categoría.
- Educativas: Aplicaciones basadas en el aprendizaje de los usuarios, ABA English es un ejemplo de ellas.
- Informativas: Aplicaciones con la finalidad de divulgar cualquier tipo de noticia.
- Creación: Aplicaciones que ofrecen al usuario herramientas que les permiten crear nuevo contenido, algunos ejemplos de ello serían la edición de videos y el retoque de fotografías.
- Reproducción: Aplicaciones con la finalidad de reproducir cualquier tipo de contenido, como por ejemplo video o audio.

b. Desarrollo de aplicaciones móviles

Para el desarrollo de aplicaciones móviles son necesarios perfiles especializados, ya que se deben de tener en cuenta una serie de consideraciones. Este tipo de aplicaciones se ejecutan en dispositivos con unos procesadores menos potentes que los que incorporan los tradicionales equipos personales. Además, tienen funcionalidades que los otros no poseen, como por ejemplo cámara y GPS [9].

Otro aspecto muy importante a considerar en el desarrollo, es el diseño de la interfaz de usuario, ya que puede ser uno de los motivos por los cuales la aplicación no sea un éxito. La interfaz de usuario ha de ser lo más amigable posible, pensada para que se adapte en pantallas con un tamaño pequeño y de distintas resoluciones. Otro aspecto importante en su diseño, es cómo los usuarios realizan la introducción de datos y el tamaño de sus manos [9].

Una última consideración a tener en cuenta, es la plataforma en la que se va a ejecutar la aplicación, pues esto complica el desarrollo de aplicaciones nativas. Se entiende como aplicaciones nativas las que se han desarrollado con las herramientas y el lenguaje de programación propio del dispositivo [15].

Ilustración 2: Entornos de desarrollo de las distintas plataformas

Como se puede observar en la Ilustración 2: Entornos de desarrollo de las distintas plataformas, cada plataforma tiene su propio lenguaje de programación, framework y entorno de desarrollo integrado. Esto multiplica mucho el tiempo de desarrollo, ya que la aplicación se tiene que desarrollar específicamente para cada una de ellas, sin poder reutilizar ni una sola línea del código de implementación. Además es necesaria una gran preparación por parte de los desarrolladores [15].

	iOS	Windows	Android
Lenguaje de programación	Swift/Objective C	C#	Java
Framework	iOS Frameworks	.NET Library	Android SDK/Java SDK
Entrono de desarrollo integrado	Xcode	Visual Studio	Android Studio

Tabla 1: Detalle de los entornos de desarrollo de las distintas plataformas

Una de las soluciones para mitigar este problema es el diseño de una arquitectura cliente servidor: donde la parte cliente es la responsable de pintar los datos e interactuar con el usuario; y, la otra parte, es la responsable de implementar la lógica de negocio y aspectos relacionados con la seguridad y la trazabilidad. De esta forma, es posible reutilizar parte del código entre las distintas plataformas, asimismo esta reutilización no está limitada a los clientes móviles, ya que los clientes Web también se pueden beneficiar de ella.

Evidentemente, el desarrollo de aplicaciones nativas ofrece una serie de ventajas, ya que al desarrollarse teniendo en cuenta las particularidades de cada plataforma, se obtiene: una máxima flexibilidad, una interfaz de usuario mucho más adaptada a la plataforma y un máximo rendimiento [15].

c. Desarrollo de aplicaciones móviles multiplataforma

En informática, se entiende por multiplataforma un programa que es implementado para distintas plataformas. Este concepto trasladado al mundo del desarrollo de las aplicaciones móviles, consistiría en el desarrollo de una única aplicación que se ejecuta en cualquier plataforma.

Actualmente, hay distintas estrategias para el desarrollo de aplicaciones multiplataforma, estas son: aplicaciones HTML5, aplicaciones híbridas y aplicaciones pseudo-nativas [14].

Las aplicaciones HTML5 se ejecutan en navegadores Web, puesto que se desarrollan únicamente con HTML5, JavaScript y CSS. Con el uso de estos lenguajes se garantiza la compatibilidad con todas las plataformas, ya que cada una de ellas dispone de un navegador Web [14].

La idea de este tipo de aplicaciones es que sean responsivas (adaptación a cualquier tamaño de pantalla) y que toda la arquitectura se monte en el lado del cliente, dejando únicamente el lado servidor para la manipulación de datos.

La gran ventaja de las aplicaciones HTML5 es la gran reutilización de código, pues la misma aplicación funciona en todas las plataformas. Pero, al mismo tiempo, tienen grandes inconvenientes ya que, al ser ejecutadas por un navegador web, no es posible acceder a las funcionalidades nativas de cada plataforma (cámara y GPS), ni a funcionalidades a las que no se pueda acceder desde el navegador (notificaciones Push), además su rendimiento es mucho menor que el de las aplicaciones nativas. Tampoco tienen la capacidad de funcionar en modo offline porque necesitan de una conexión a Internet [14].

Las aplicaciones híbridas son aplicaciones HTML5 que se encapsulan en una aplicación nativa y se ejecutan mediante un componente WebView nativo de la plataforma. También ofrecen mecanismos que permiten la comunicación con la parte nativa, pudiendo de esta forma acceder a funcionalidades como las notificaciones Push y a características como la cámara y el GPS [14].

En el caso de necesitar una funcionalidad nativa sin existencia de Plugin para ello, se deberá desarrollar esta funcionalidad para cada plataforma. Igualmente, su rendimiento sigue siendo menor que el de las aplicaciones nativas, ya que se ejecuta dentro de un WebView, que sigue siendo una forma de navegación [14].

Las aplicaciones pseudo-nativas son aplicaciones desarrolladas con frameworks que generan código nativo a partir de un lenguaje de programación común. A efectos prácticos son aplicaciones nativas, puesto que trabajan con componentes nativos. Estos frameworks permiten que no existan limitaciones a la hora de acceder a cualquier librería o componente nativo, también se pueden acceder a API nativas directamente desde el código [14].

Las grandes ventajas de estas aplicaciones son: que hacen uso de componentes visuales nativos sin las restricciones de las aplicaciones híbridas y que su rendimiento es parecido al de las aplicaciones nativas. A pesar de ello, hay que considerar la penalización en el rendimiento debido a la existencia de una capa intermedia, que traduce el código común a código nativo. Además, las aplicaciones híbridas, reaprovechan mayor cantidad de código en la parte de la interfaz de usuario [14].

d. Frameworks de desarrollo de aplicaciones móviles

Una vez se ha hablado de las distintas estrategias existentes para el desarrollo de aplicaciones móviles, es el momento de realizar una breve introducción a algunos frameworks que se basan en ellas.

i. Android SDK

Android SDK es un framework que permite desarrollar aplicaciones móviles en nativo para la plataforma Android. Este es el responsable de compilar el código escrito con el lenguaje de programación Java, junto con los archivos de recursos y de datos, en una APK [29].

Android es un sistema Linux multiusuario, en el que se le asigna a cada aplicación un identificador de usuario, de esta forma es posible implementar el principio de mínimo

privilegio, garantizando que solamente se acceda a los componentes necesarios, evitando el acceso a otras partes del sistema para las que no se tenga permiso. De todas formas, Android ofrece mecanismos para que se puedan compartir datos entre aplicaciones y acceder a servicios del sistema. Cuando una aplicación solicita acceder a datos del dispositivo, es el usuario el responsable de darle los permisos necesarios. Las aplicaciones se ejecutan en un proceso de Linux propio, por lo que son independientes entre ellas [29].

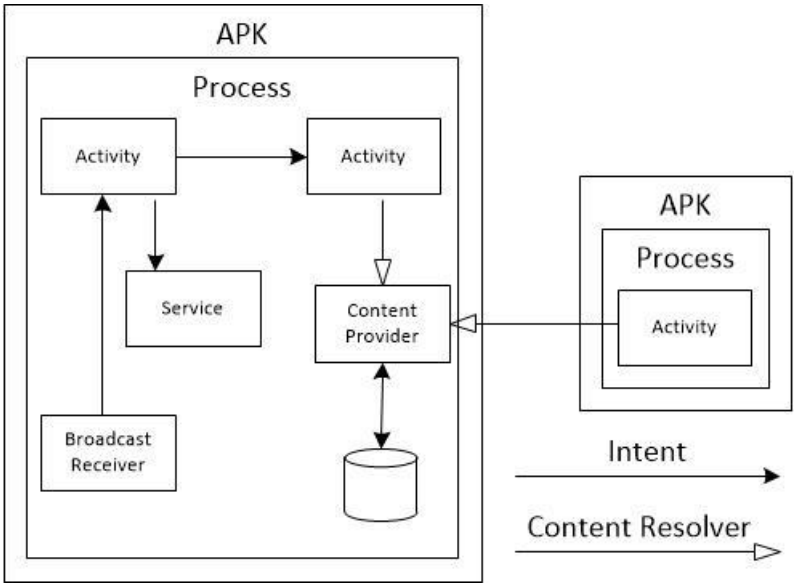


Ilustración 3: Arquitectura de una aplicación Android

En la Ilustración 3: Arquitectura de una aplicación Android, se puede observar que las aplicaciones Android están formadas por componentes, estos existen como una entidad individual y tienen un rol específico. Cada uno de ellos es un punto diferente a través del cual el sistema o el usuario pueden entrar en la aplicación [29]. Hay cuatro tipos diferentes de componentes en una aplicación y cualquier aplicación puede iniciar el componente de otra aplicación.

- Activity: Representa una pantalla con interfaz de usuario y, aunque trabajen juntas para proporcionar una experiencia de usuario, cada una es independiente de las otras. Se implementa como subclase de Activity [29].
- Service: Se ejecuta en segundo plano para realizar operaciones prolongadas o tareas para procesos remotos. Otro componente puede iniciarlo y permitir que se ejecute o interactúe con él. Se implementa como subclase de Service [29].
- Content Provider: Administra el conjunto de datos de la aplicación, estos pueden almacenarse en una base de datos SqLite o en cualquier ubicación de almacenamiento persistente a la que pueda acceder la aplicación. Se implementa como una subclase de ContentProvider [29].
- Broadcast Receiver: Responde a los anuncios de mensajes de todo el sistema, pudiendo crear una notificación a la barra de estado para alertar al usuario cuando se produce un evento. Se implementa como subclase de Broadcast Receiver [29].

Los componentes Activity, Service y Broadcast Receiver se activan a partir de mensajes asíncronos llamados Intent, estos permiten enlazar los componentes individuales en tiempo de

ejecución. Un Intent se crea con un objeto Intent y define un mensaje para activar un componente específico. En el caso de los componentes Activity y Service, un Intent define la acción a realizar y puede especificar la URI de los datos con los que debe actuar. En los Broadcast Receiver simplemente define el anuncio que está transmitiendo [29].

En cambio, el componente Content Provider se activa mediante solicitudes llamadas Content Resolver, éste gestiona todas las transacciones directas con el Content Provider, obligando al componente que desea activarlo a llamar a los métodos del objeto Content Resolver [29].

Para que el sistema Android pueda reconocer e iniciar los distintos componentes, éstos se deben declarar en el archivo de manifiesto. Este archivo puede realizar otras tareas además de registrar los componentes, como por ejemplo: identificar los permisos de usuario que requiere la aplicación; declarar el nivel API mínimo requerido por la aplicación; declarar las características del hardware y software que la aplicación usa, como una cámara; y declarar bibliotecas de la API a las que la aplicación necesita estar vinculada, como la de Android Framework API y la de Google Maps [29].

Las aplicaciones Android están formadas por otros recursos a parte del código: imágenes, archivos de audio y elementos relacionados con la presentación. El uso de estos recursos facilita el mantenimiento de algunas características sin necesidad de modificar el código y permite optimizarla en función de las distintas configuraciones de los dispositivos [29].

ii. Xamarin

Xamarin es un ejemplo de framework para el desarrollo de aplicaciones móviles multiplataforma pseudo-nativas, éste fue desarrollado por una compañía de software estadounidense que es propiedad de Microsoft. Permite escribir aplicaciones nativas para las plataformas Android, iOS y Microsoft, compartiendo gran cantidad de código escrito con el lenguaje de programación C# [17].

Tal como se muestra en la Ilustración 4: Arquitectura de compartición de código entre distintas plataformas, el objetivo de Xamarin es gestionar la reutilización de código, para ello ofrece diversas alternativas: Shared Projects, Portable Class Libraries y .NET Standard Libraries [30].

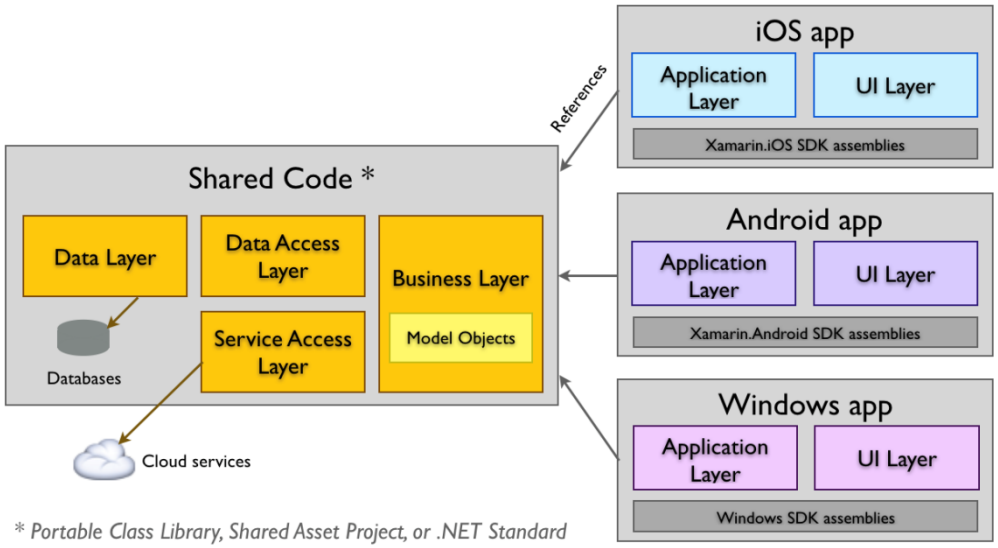


Ilustración 4: Arquitectura de compartición de código entre distintas plataformas

Shared Projects: Permiten ramificar el código fuente compartido según la plataforma utilizando las directivas de compilador (#if __ANDROID__) [30].

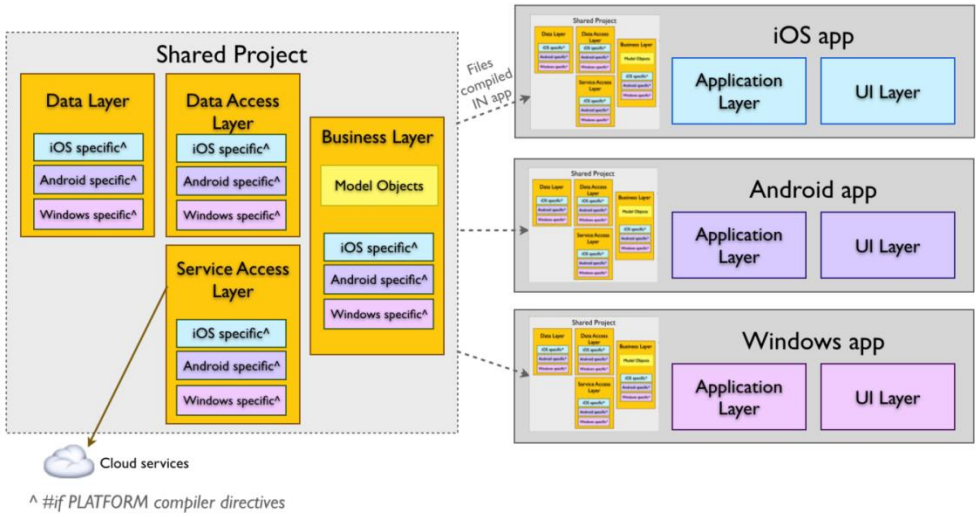


Ilustración 5: Arquitectura de Shared Project

Un Shared Project no tiene ensamblados de salida, el código fuente se compila en los ensamblajes de cada proyecto como si fuera parte de él. Esto provoca que las refactorizaciones que afectan el código dentro de las directivas del compilador inactivo no se actualicen [30].

Portable Class Libraries: Esta alternativa permite la creación de un proyecto donde el ensamblado resultante está restringido a trabajar con la plataforma específica para la que se ha creado. También es necesario el uso de interfaces que proporcionan la funcionalidad específica para cada plataforma [31].

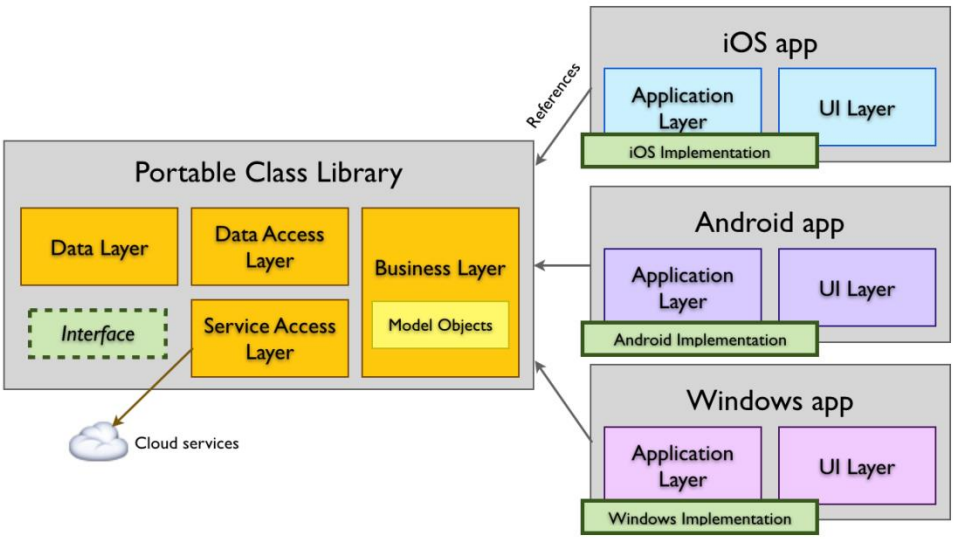


Ilustración 6: Arquitectura de Portable Class Libraries

El ensamblaje de salida de este tipo de proyectos puede ser referenciado fácilmente por otros proyectos. Esto permite centralizar el código compartido, escribiéndolo y testeándolo en un solo proyecto. Además, las operaciones de refactorización afectarán al proyecto Portable Class Libraries y a los proyectos específicos de la plataforma [31].

.NET Standard Libraries: Funcionan de forma similar a los Portable Class Libraries, requiriendo el uso de interfaces para inyectar la funcionalidad específica de la plataforma. Se puede ver cómo, la próxima generación de Portable Class Libraries, tiene un modelo más simple para el soporte de plataforma y un mayor número de clases del BCL (Base Class Library). La motivación que hay detrás del .NET Standard Libraries es establecer una mayor uniformidad en el ecosistema .NET [30].

Hasta ahora hemos visto las distintas alternativas que permiten compartir el código de la lógica de negocio, con ellas se garantiza una cobertura del 70% o más de código compartido. Xamarin también ofrece distintas estrategias para la implementación de la interfaz de usuario.

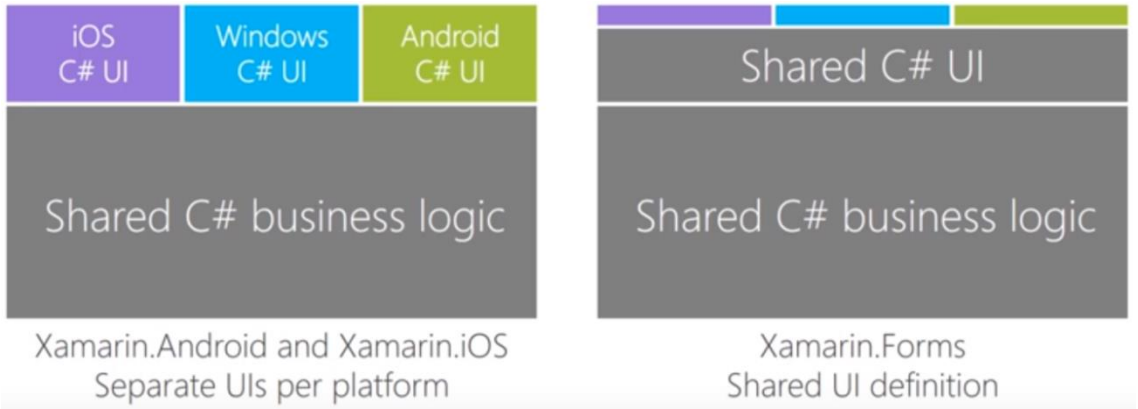


Ilustración 7: Estrategias de desarrollo de interfaz de usuario

Tal como se puede ver en la Ilustración 7: Estrategias de desarrollo de interfaz de usuario, una de ellas permite implementar la interfaz de usuario específicamente para cada plataforma, sin la oportunidad de reutilizar el código de esta parte. La otra estrategia permite reutilizar el

código de la interfaz de usuario entre todas las plataformas, en este caso el lenguaje de programación utilizado en la interfaz de usuario es el XAML.

Xamarin.Android y Xamarin.iOS son la mejor opción cuando se desea desarrollar una aplicación con interacciones que requieran un comportamiento nativo, cuando se usan muchas API específicas de la plataforma o cuando la personalización de la interfaz de usuario tiene más importancia que la reutilización de código.

En cambio, Xamarin.Forms es mejor opción cuando la aplicación requiere pocas funcionalidades específicas de la plataforma y cuando la reutilización del código es más importante que la personalización de la interfaz de usuario.

Ilustración 8: Páginas y plantillas de Xamarin.Forms

En la Ilustración 8: Páginas y plantillas de Xamarin.Forms, se puede observar que Xamarin.Forms ofrece un conjunto de páginas y plantillas a los desarrolladores con la finalidad de facilitarles el trabajo. Las páginas permiten encapsular el contenido de la pantalla, y las plantillas organizan este contenido.

Xamarin.Forms ofrece un mecanismo con la finalidad de que los desarrolladores puedan definir comportamiento específico de la plataforma, éste es el DependencyService y se comporta como un solucionador de dependencias [34].

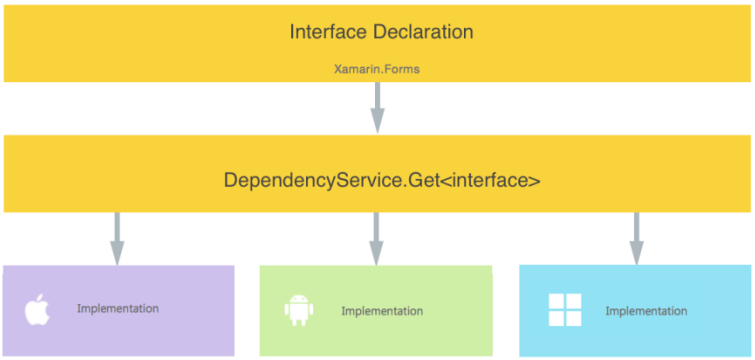


Ilustración 9: Arquitectura de DependencyService

El uso de DependencyService está dividido en cuatro partes [34].

- Interfaz: Define el contrato de la funcionalidad requerida.
- Implementación por plataforma: Clases que implementan la interfaz específicamente para cada plataforma.
- Registro: Cada una de las clases de implementación se registra mediante un atributo de metadatos. Éste permite a DependencyService encontrar la clase implementadora de la interfaz y utilizarla en tiempo de ejecución.
- Llamada: El código compartido necesita llamar explícitamente a DependencyService para solicitar la implementación de la interfaz.

iii. Apache Cordova

Apache Cordova es un framework de Adobe para el desarrollo de aplicaciones móviles híbridas, utilizando las tecnologías estándar Web de HTML5, JavaScript y CSS. Permite crear aplicaciones para las plataformas Android, iOS, Windows y Ubuntu. Aunque dependiendo de la plataforma no tiene soporte para todas las funcionalidades nativas del dispositivo [19].

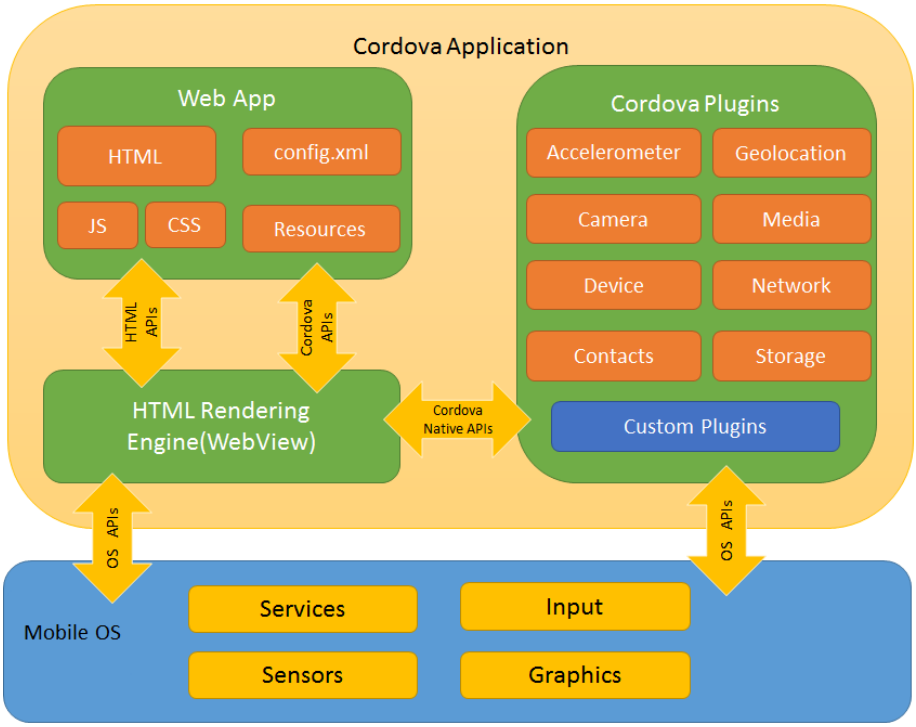


Ilustración 10: Arquitectura de aplicaciones Apache Cordova

La Ilustración 10: Arquitectura de aplicaciones Apache Cordova, muestra la existencia de distintos componentes en una aplicación móvil, estos se basan en un archivo común que proporciona información sobre la aplicación y permite configurar su funcionamiento mediante una serie de parámetros [19].

- WebView: Responsable de ejecutar el código de la aplicación, proporcionando la interfaz de usuario. En algunas plataformas también puede ser un componente de una aplicación híbrida más grande [19].

- Web App: En esta parte es donde reside el código de la aplicación, esta es implementada puramente como una página Web, incluso contiene un fichero index.html que referencia las CSS, el código JavaScript, las imágenes, los archivos multimedia y otros recursos de la aplicación [19].
- Plugins: Son una parte esencial del framework, proporcionando una interfaz que comunica Apache Cordova con las funcionalidades nativas, mediante enlaces a la API de los dispositivos. Esto permite invocar código nativo desde JavaScript [19].

  Apache Cordova mantiene un conjunto de Plugins llamados Core Plugins, estos permiten acceder a funcionalidades como la batería, la cámara, los contactos, etc. Además existen Plugins de terceras partes que proporcionan enlaces a funcionalidades que no son específicas de todas las plataformas [19].

Este framework ofrece dos flujos de desarrollo para la creación de aplicaciones móviles. Aunque normalmente se puedan utilizar cualquiera de los dos para realizar la misma tarea, cada uno de ellos ofrece distintas ventajas.

- Flujo de trabajo multiplataforma: Se centra alrededor del CLI, permitiendo crear proyectos para muchas plataformas a la vez que abstrae de las funcionalidades de bajo nivel. Para ello crea un conjunto de subdirectorios para cada plataforma y realiza los cambios de configuración necesarios para ellas [19].

  Es el flujo recomendable cuando se desee desarrollar una aplicación móvil para distintas plataformas, sin ser necesario realizar una gran cantidad de desarrollo específico para cada una de ellas [19].

- Flujo de trabajo centrado en plataforma: Éste permite modificar el proyecto dentro del SDK, ya que se basa en un conjunto de scripts de bajo nivel diseñados para cada plataforma. Desarrollar aplicaciones multiplataforma con este flujo es bastante difícil. Ello se debe a que no ofrece una herramienta de alto nivel como el CLI y, a que no es necesario realizar modificaciones en los Plugins por cada una de las plataformas. Aun así, se puede convertir en esencial cuando las aplicaciones multiplataforma son complejas, porque permite un mayor acceso a las opciones proporcionadas por cada SDK [19].

  Es el flujo recomendable cuando se desee desarrollar una aplicación móvil para una única plataforma y sea necesario modificarla a bajo nivel [19].

e. Problemas en el desarrollo de aplicaciones móviles

Todos los frameworks de desarrollo de aplicaciones móviles vistos en este apartado, son soluciones muy utilizadas actualmente para la creación de las mismas. Tal como se ha expuesto, cada uno de ellos ofrece unas ventajas respecto a los otros y, en función de ellas, puede adaptarse mejor como solución dependiendo del problema que se trata de resolver.

Estos frameworks han de ser utilizados por personal con unos altos conocimientos informáticos y con una larga experiencia en el mundo de la movilidad, ya que es necesario que

estén muy familiarizados con alguna de las distintas tecnologías existentes y con las características de cada una de las plataformas móviles.

Como consecuencia de la necesidad de personal altamente especializado, el proceso de creación de aplicaciones móviles se convierte en un proceso muy costoso económicamente, haciéndolo difícilmente accesible a las empresas con pocos recursos.

Otro de los inconvenientes, sobre todo con Android SDK y Xamarin (las mejores soluciones existentes en el mercado de desarrollo de aplicaciones móviles, ya que generan código nativo), requiere de equipos muy potentes para la ejecución de sus IDE, Android Studio y Visual Studio.

4. Definición del dominio

Antes de detallar la solución que se desea ofrecer, es necesario definir un dominio capaz de representar la mayor parte de las aplicaciones móviles. Este dominio ha de tener en consideración las características comunes de las aplicaciones móviles independientemente de su categoría y plataforma.

El dominio a definir ha de tener en consideración que una aplicación móvil está formada por distintas páginas, dónde cada una tendrá un comportamiento en concreto. Podrán listar un conjunto de registros o ver su detalle, permitiendo al usuario realizar la edición de ella. En el caso de las páginas que listarán los registros, también será necesario que se relacione con la página que permite ver el detalle de ella. Además de tener una finalidad, existirán varios tipos de páginas, ya que podrán mostrar contenido, podrán dividir el contenido por pestañas o mostrarlo como un carrusel.

Las páginas que mostrarán el detalle de la información estarán formadas por varios controles, que son los que permitirán al usuario editar la información. Existirán distintos tipos de controles, en función de su finalidad, ya que podrán permitir al usuario editar texto, introducir una imagen o realizar una fotografía. Estos controles se posicionarán en la pantalla mediante plantillas con distintos comportamientos, ya que se podrán posicionar como una pila o personalizar su distribución mediante tablas.

A continuación se puede ver una ilustración del diagrama de características del dominio definido, y la descripción de cada una de las características.

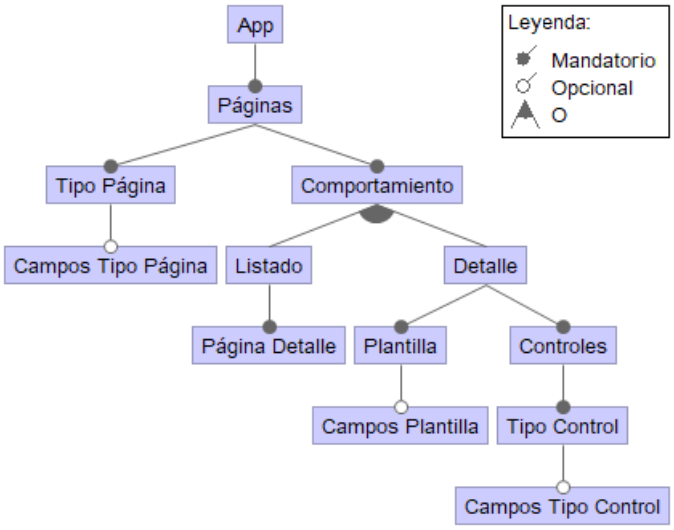


Ilustración 11: Diagrama de características del dominio de las aplicaciones móviles

- Páginas: Colección de contenedores que encapsularán el contenido de la pantalla.
    - Tipo Página: Indicará de qué tipo es la página; por ejemplo, si se tratará de una página de contenido, una con pestañas, etc.

        - Campos Tipo Página: Campos auxiliares que permitirán configurar el comportamiento de un tipo de página.
    - Comportamiento: Indicarán el comportamiento de una página.
        - Listado: Su comportamiento se basará en listar una colección de registros.
            - Página Detalle: Página que visualizará el detalle de uno de los registros listados.
        - Detalle: Su comportamiento se basará en visualizar el detalle de un registro.
            - Plantilla: Indicará cómo se organiza el contenido de una página.
                - Campos Tipo Plantilla: Campos auxiliares que permitirán configurar el comportamiento de una plantilla.
            - Controles: Controles que permitirán que el usuario interactúe con la app.
                - Tipo Control: Indicará el tipo de control; por ejemplo, si será un control calendario, texto, imagen, etc.
                    - Campos Tipo Control: Campos auxiliares que permitirán configurar el comportamiento de un control.

5. Descripción del proceso de aplicación del dominio

En apartados anteriores se ha confirmado la necesidad de personal especializado para trabajar con las tecnologías existentes para el desarrollo de aplicaciones móviles. El principal objetivo de este trabajo es ofrecer una solución que permita a los usuarios la creación de aplicaciones móviles abstrayéndose de los aspectos técnicos.

La solución estará basada en el proceso de generación mostrado en la siguiente ilustración. Este hará posible la creación de aplicaciones móviles por parte de un usuario sin la necesidad de conocimientos técnicos.

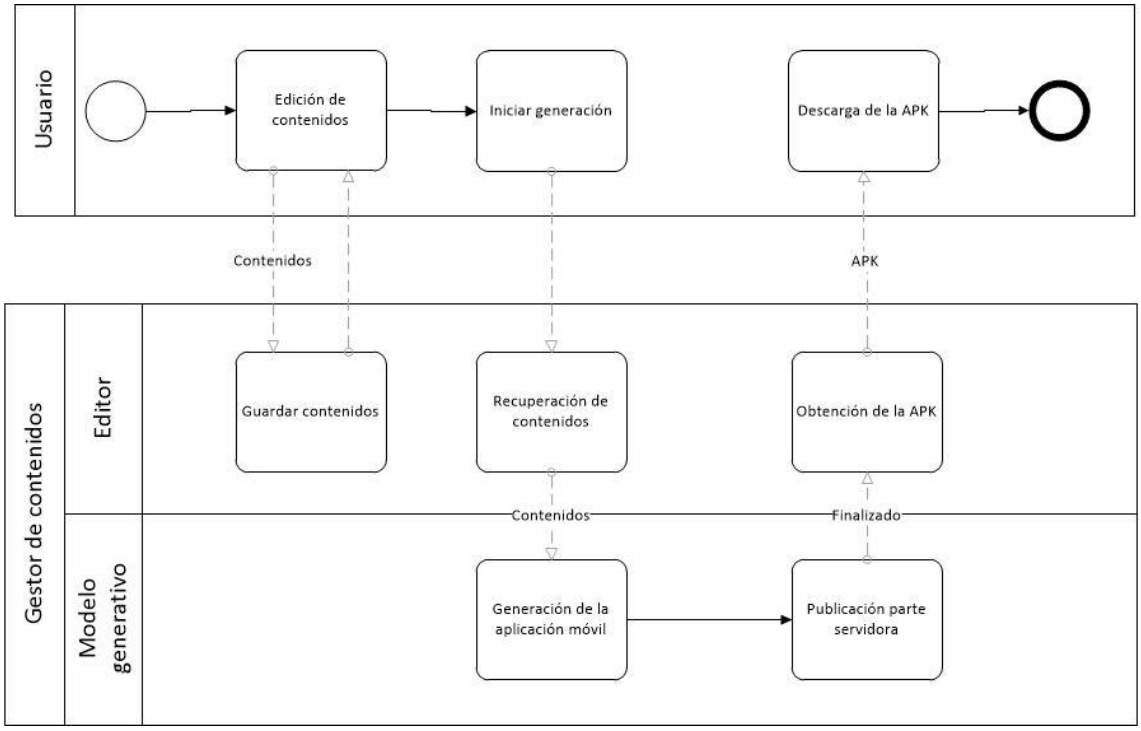


Ilustración 12: Diagrama BPMN del proceso generativo

El proceso generativo estará formado por dos participantes, el usuario y el gestor de contenidos, a su vez, el gestor de contenidos estará formado por dos componentes, el editor y el modelo generativo. El proceso generativo estará formado por las siguientes tareas.

- Edición de contenidos: El usuario editará los contenidos de la aplicación móvil que desea crear y los enviará al gestor de contenidos.
- Guardar contenidos: El editor del modelo generativo recibirá los contenidos editados por el usuario y los guardará.
- Iniciar generación: El usuario informará al gestor de contenidos que desea generar su aplicación móvil.
- Recuperación de contenidos: El editor recuperará los contenidos de la aplicación móvil editados por el usuario y los enviará al modelo generativo.

- Generación de la aplicación móvil: El modelo generativo recibirá los contenidos editados por el usuario, y a partir de ellos creará el fichero APK y la parte servidora de la aplicación móvil.
- Publicación parte servidora: El modelo generativo publicará la parte servidora creada en la tarea anterior.
- Obtención de la APK: El editor recuperará el fichero APK generado por el modelo generativo y lo enviará al usuario.
- Descarga de la APK: El usuario se descargará el fichero APK generado por el gestor de contenidos y lo instalará en su dispositivo móvil.

Una vez definido el dominio y el proceso de generación, es el momento de detallar la solución. Esta consistirá en el diseño e implementación de un sistema de gestión de contenidos, que permitirá a un usuario poder crear de forma sencilla, mediante formularios, su aplicación móvil deseada, editando las distintas páginas que la componen.

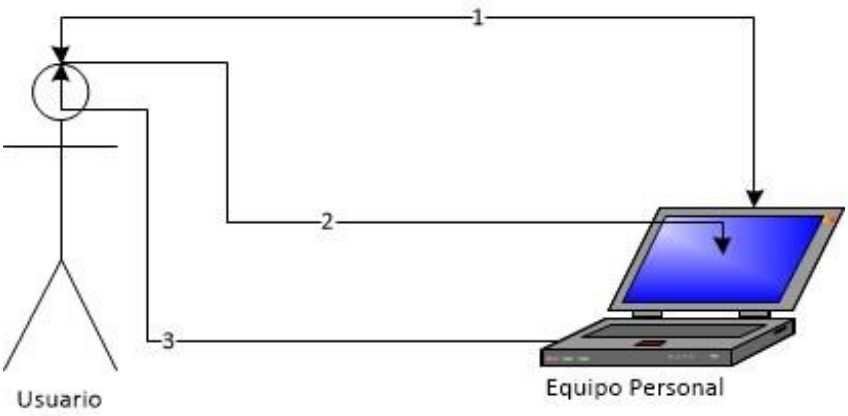


Ilustración 13: Flujo del sistema de gestión de contenidos desde el punto de vista del usuario

Tal como se puede observar en la Ilustración 13: Flujo del sistema de gestión de contenidos desde el punto de vista del usuario, el sistema permitirá la creación de aplicaciones móviles en tres pasos.

1. El usuario, con su equipo personal, accederá al sistema de creación de aplicaciones móviles y, mediante formularios, editará el contenido de las páginas que la componen.
2. Una vez el usuario ha editado todo el contenido de la aplicación móvil, informará al sistema que desea generarla.
3. El sistema generará la aplicación y la empaquetará en un fichero con formato APK, éste será descargado por parte del usuario para posteriormente proceder a su instalación en el dispositivo móvil deseado.

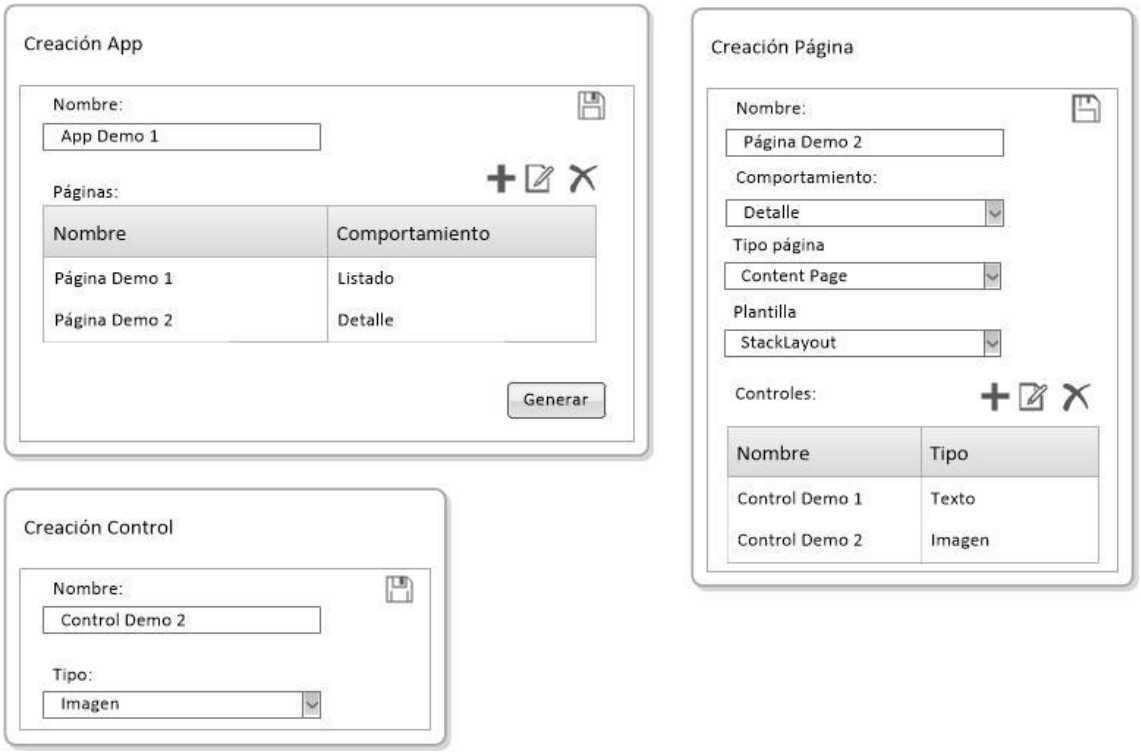


Ilustración 14: Maqueta de las pantallas de edición de contenido de las aplicaciones móviles

Los contenidos se gestionarán mediante tres tipos distintos de formularios de edición: el primero, será el responsable de trabajar con el contenedor de la aplicación, informando del nombre y administrando sus páginas; en el segundo formulario, se configurarán las características de la página, éstas variarán en función del comportamiento seleccionado; el último formulario, permitirá editar el comportamiento del control, dependiendo del tipo de control seleccionado, este formulario trabajará con unos campos u otros.

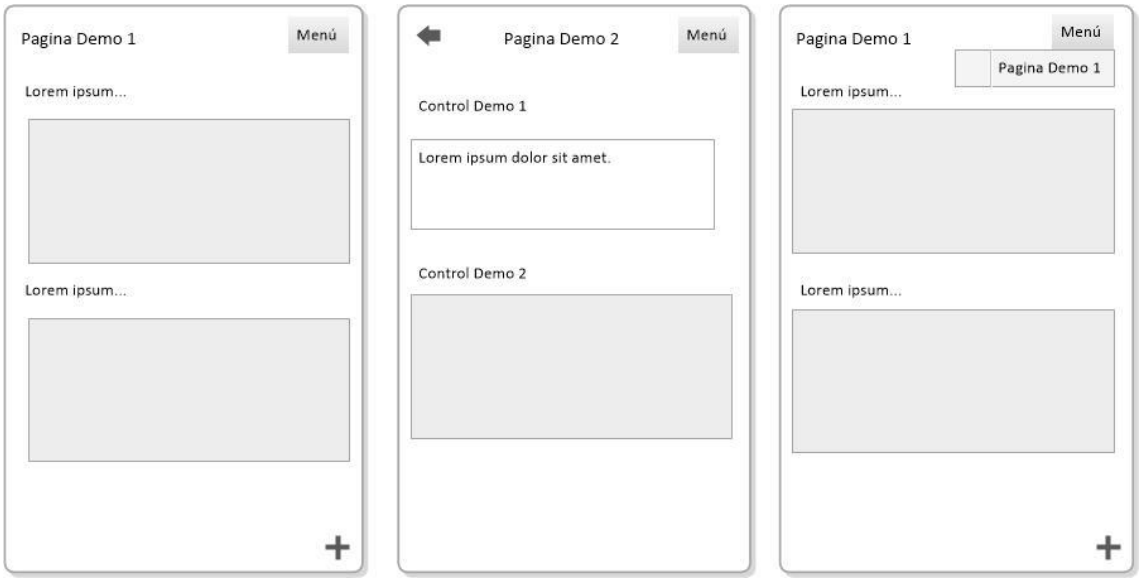


Ilustración 15: Maqueta de las pantallas de la aplicación móvil generada

A partir de los contenidos gestionados por el usuario, se generará una aplicación móvil. En la Ilustración 15: Maqueta de las pantallas de la aplicación móvil generada se puede ver el

resultado final del ejemplo mostrado en la Ilustración 14: Maqueta de las pantallas de edición de contenido de las aplicaciones móviles. La aplicación móvil generada estará formada por dos pantallas: una será la responsable de listar una colección de elementos, permitiendo añadir de nuevos y eliminando los existentes; la otra pantalla posibilitará la edición de estos elementos. Los elementos del menú de navegación de la aplicación móvil corresponderán a las páginas cuyo comportamiento será configurado como “Listado”.

El sistema de gestión de contenidos se comportará como una aplicación Web y se basará en una arquitectura cliente servidor, donde la parte cliente será la responsable de interactuar con el usuario. La parte servidora, se encargará de: guardar la configuración de la aplicación móvil a desarrollar y generarla, proporcionar el fichero APK al usuario, crear su esquema de base de datos y publicar su parte servidora en un IIS.

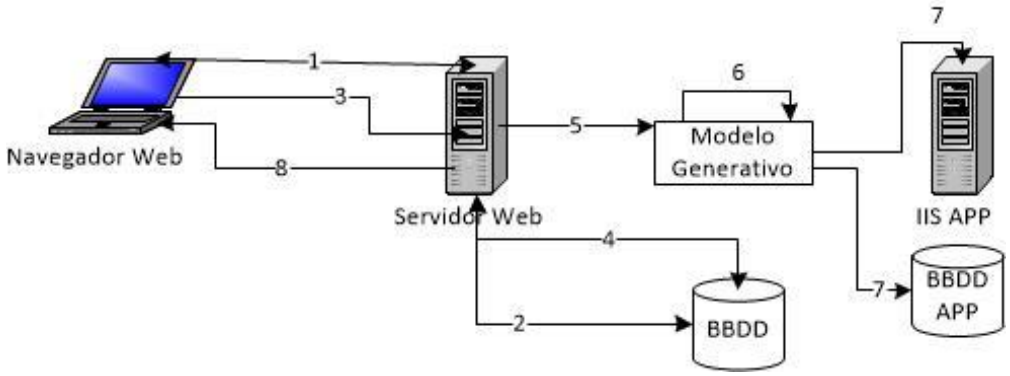


Ilustración 16: Flujo de trabajo interno del sistema de gestión de contenidos

En la Ilustración 16: Flujo de trabajo interno del sistema de gestión de contenidos, se puede ver que el flujo está formado por los siguientes ocho pasos.

1. Un navegador Web, que será la herramienta utilizada por el usuario para interactuar con el sistema de gestión de contenidos, se comunicará con el servidor Web, informándole de la gestión del contenido realizada por el usuario.
2. El servidor Web almacenará la gestión de contenidos realizada por el usuario en una base de datos.
3. El navegador Web, a petición del usuario, informará al servidor Web que se desea generar la aplicación móvil.
4. El servidor Web recuperará de la base de datos la gestión de contenidos realizada por el usuario.
5. El servidor Web ejecutará el modelo generativo interno, pasándole la gestión de contenidos recuperados en el paso anterior. Los contenidos editados por el usuario se pasarán a partir de un DSL definido con su sintaxis serializada en Json.
6. El modelo generativo convertirá la información contenida en la especificación del DSL en una estructura más adecuada para ser trabajada en el entorno Ruby, y a partir de ella se generará la aplicación móvil. El resultado de la generación serán los ficheros necesarios para la publicación en el IIS de la parte servidora y para la creación del esquema de la base de datos, además del fichero APK. Estos ficheros se almacenarán en una ruta interna del servidor Web.

7. El modelo generativo publicará la parte servidora de la aplicación móvil en el IIS y creará su esquema de base de datos.
8. El servidor Web enviará el fichero APK al navegador Web para que el usuario pueda realizar su descarga.

6. Diseño de la solución

El sistema de gestión de contenidos de la aplicación móvil está formado por dos subsistemas: el primer subsistema, es una aplicación Web y permite que interactúen el usuario y el sistema; el otro, es el modelo generativo y es el responsable de generar la aplicación móvil.

a. Aplicación Web

Subsistema del sistema de gestión de contenidos de la aplicación móvil que tiene la responsabilidad de interactuar con el usuario para que pueda editar los contenidos de la aplicación móvil. En realidad, este contenido se comporta como el dominio del modelo generativo.

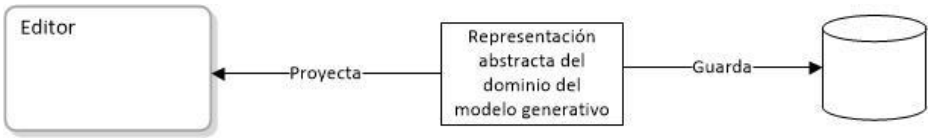


Ilustración 17: Representación de la manipulación del dominio del modelo generativo

Tal como se puede ver en la Ilustración 17: Representación de la manipulación del dominio del modelo generativo, la aplicación Web permite que el usuario manipule el dominio del modelo generativo de una forma más simple y amigable. Esto es debido a que se comporta como un editor dónde se proyecta la representación abstracta del dominio.

Este subsistema está formado por dos partes bien diferenciadas. Por un lado, existe la parte Back que es la encargada de proporcionar los datos y guardar la edición de los contenidos por parte del usuario. La otra parte, la Front, se comporta como el intermediario entre el usuario y la parte Back, el usuario interactúa directamente con esta parte.

i. Parte Back

Esta parte es la responsable de resolver las tareas de Guardar contenidos, Recuperación de contenidos y Obtención de la APK de la Ilustración 12: Diagrama BPMN del proceso generativo. Se ha desarrollado con la plataforma .NET y el lenguaje de programación C#, su arquitectura está basada en la arquitectura DDD [5]. En la Ilustración 18: Arquitectura parte Back de la aplicación Web puede verse cómo está formada por cuatro capas: la de servicio, la de aplicación, la de dominio y la de repositorio.

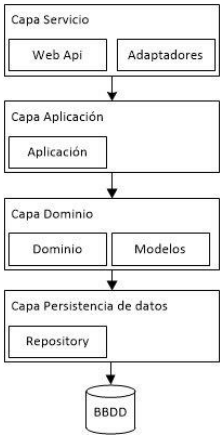


Ilustración 18: Arquitectura parte Back de la aplicación Web

1. Capa de repositorio

En esta capa solamente existe el módulo Repository, es el responsable de comunicarse con la base de datos y para ello se ha utilizado el ORM Entity Framework con enfoque Code First.

Entity Framework [22] es un ORM que permite el desarrollo de aplicaciones de software abstrayéndose de la base de datos, ya que los programadores trabajan con datos en forma de objetos y propiedades del dominio, estos objetos son llamados entidades. Las entidades pueden ser creadas con dos enfoques distintos; el enfoque DataBase First, dónde se crean a partir de una base de datos; o con Code First, dónde primero se implementan las entidades de dominio y luego se crea la base de datos a partir de ellas.

También se utilizan los patrones de diseño Repository y UnitOfWork [20]. El patrón Repository permite crear una capa de abstracción entre la capa de acceso a datos y la capa de dominio, actuando como un mediador. Su implementación ayuda en la separación de cambios entre la base de datos y la aplicación. El patrón UnitOfWork es el responsable de coordinar el trabajo de los distintos repositorios.

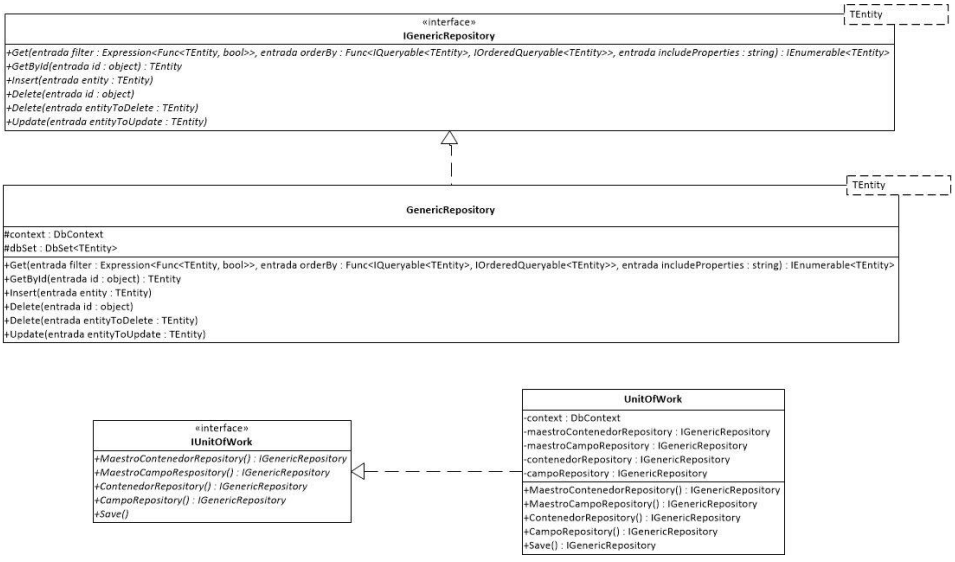


Ilustración 19: Diagrama de clases del componente Repository

Nombre:	IGenericRepository
Descripción: Interfaz que define el contrato de la clase que implementa los métodos CRUD (Create, Read, Update and Delete) de todas las entidades.	

Nombre:	GenericRepository
Descripción: Clase que implementa los métodos CRUD de todas las entidades.	
Miembros: context: Instancia de la clase DbContext. dbSet: Instancia de la clase DbSet.	
Métodos:	

Nombre:	Get
Descripción: Recupera una colección de entidades en función de los parámetros de entrada.	
Parámetros: filter: Filtro de la sentencia de consulta. orderBy: Ordenación a aplicar a las entidades recuperadas. includeProperties: Propiedades de navegación incluidas en las entidades recuperadas.	
Retorno: Colección de entidades recuperadas.	

Nombre:	GetById
Descripción: Recupera una entidad a partir de su identificador.	
Parámetros: id: Identificador de la entidad a recuperar	
Retorno: Entidad recuperada.	

Nombre:	Insert
Descripción: Añade una entidad.	
Parámetros: entity: Entidad a añadir.	

Nombre:	Delete
Descripción: Elimina una entidad a partir de su identificador.	
Parámetros: id: Identificador de la entidad a eliminar.	

Nombre:	Delete
Descripción: Elimina la entidad pasada como parámetro.	
Parámetros: entityToDelete: Entidad a eliminar.	

Nombre:	Update

Descripción: Actualiza una entidad.
Parámetros: entityToUpdate: Entidad a actualizar.

Nombre:	IUnitOfWork
Descripción: Interfaz que define el contrato de la clase que implementa los métodos de seguimiento de todas las acciones realizadas en una transacción de base de datos.	

Nombre:	UnitOfWork
Descripción: Clase que implementa los métodos de seguimiento de todas las acciones realizadas en una transacción de base de datos.	
Miembros: context: Instancia de la clase DbContext, permite hacer el seguimiento de las acciones realizadas en una transacción de base de datos. maestroContenedorRepository: Instancia de la clase GenericRepository para la entidad MaestroContenedor. maestroCampoRepository: Instancia de la clase GenericRepository para la entidad MaestroCampo. contenedorRepository: Instancia de la clase GenericRepository para la entidad Contenedor. campoRepository: Instancia de la clase GenericRepository para la entidad Campo.	

Métodos:

Nombre:	MaestroContenedorRepository
Descripción: Devuelve el miembro maestroContenedorRepository.	
Retorno: Miembro maestroContenedorRepository.	

Nombre:	MaestroCampoRepository
Descripción: Devuelve el miembro maestroCampoRepository.	
Retorno: Miembro maestroCampoRepository.	

Nombre:	ContenedorRepository
Descripción: Devuelve el miembro contenedorRepository.	
Retorno: Miembro contenedorRepository.	

Nombre:	CampoRepository
Descripción: Devuelve el miembro campoRepository.	
Retorno: Miembro campoRepository.	

Nombre:	Save

Descripción: Guarda en base de datos todas las acciones realizadas en el contexto y finaliza la transacción.

2. Capa de dominio

Esta capa es la responsable de representar los conceptos de negocio e implementar las reglas del dominio. Está formada por dos componentes: el de Modelos y el de Dominio. El componente de Modelos implementa las entidades del dominio, que son las que definen el DSL del modelo generativo.

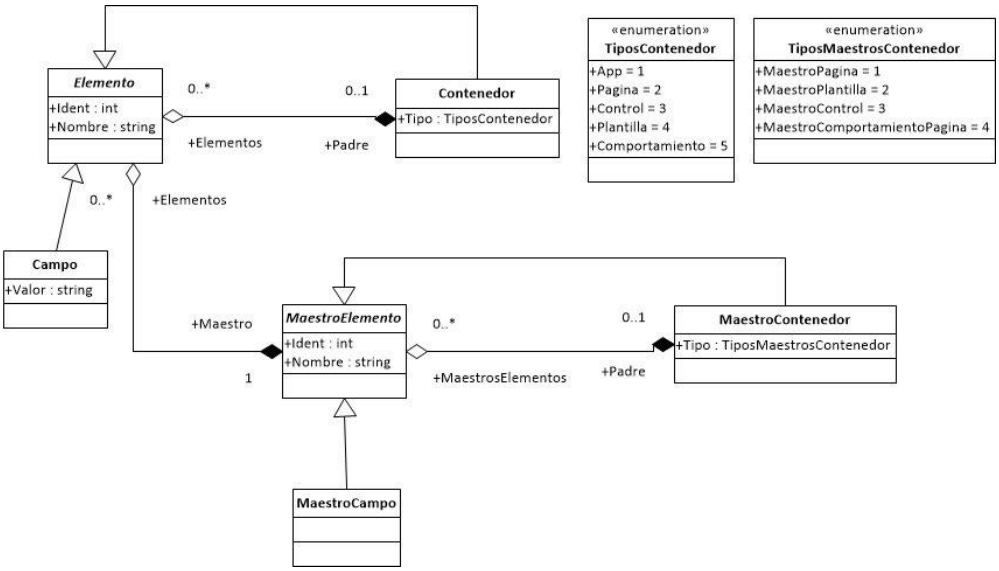


Ilustración 20: Diagrama de clases del componente Modelo

Para su implementación se ha utilizado el patrón de diseño Composite [35], tanto para representar los contenidos de la aplicación móvil como la de sus maestros.

Las clases MaestroElemento, MaestroContenedor y MaestroCampo son las responsables de representar los valores que pueden tomar los contenidos de la aplicación móvil como, por ejemplo, si la característica Comportamiento del diagrama de características es de tipo Listado o Detalle.

A partir de la clase MaestroContenedor es posible simbolizar los distintos valores de las características Tipo Página, Tipo Plantilla, Tipo Control y Tipo Comportamiento (Listado o Detalle). Todas ellas se pueden diferenciar con el miembro Tipo de la clase. La clase MaestroCampo simboliza todas las propiedades de configuración de su elemento padre.

Las clases Elemento, Contenedor y Campo son las responsables de representar los contenidos de la aplicación móvil. A partir de la clase Contenedor es posible simbolizar las características App, Páginas, Comportamiento, Plantilla y Controles del diagrama de características. Todas ellas se pueden diferenciar con el miembro Tipo de la clase.

La clase Campo simboliza todas las características Campos del diagrama de características, permitiendo establecer valores a las propiedades de configuración de su elemento padre.

La clase Elemento se relaciona con la clase MaestroElemento mediante su miembro Maestro, permitiendo trazar los valores asignados a los contenidos de la aplicación móvil.

Nombre:	MaestroElemento
Descripción: Clase abstracta que se comporta como el participante Componente del patrón de diseño Composite. Permite representar los distintos elementos de la parte de maestros.	
Miembros: Ident: Identificador del elemento. Nombre: Nombre del elemento.	

Nombre:	MaestroContenedor
Descripción: Clase que se comporta como el participante Composición del patrón de diseño Composite. Permite representar el maestro de páginas, plantillas, controles y comportamientos de la página.	
Miembros: Tipo: Informa de si el elemento es un maestro de páginas, plantillas, controles o comportamientos de la página.	

Nombre:	MaestroCampo
Descripción: Clase que se comporta como el participante Hoja del patrón de diseño Composite. Permite representar las distintas propiedades de configuración que puedan tener el maestro de páginas, plantillas, controles y comportamientos de la página.	

Nombre:	Elemento
Descripción: Clase abstracta que se comporta como el participante Componente del patrón de diseño Composite. Permite representar los distintos elementos de los contenidos de la aplicación móvil.	
Miembros: Ident: Identificador del elemento. Nombre: Nombre del elemento.	

Nombre:	Contenedor
Descripción: Clase que se comporta como el participante Composición del patrón de diseño Composite. Permite representar los elementos contenedores de los contenidos de la aplicación (app, páginas, plantillas, controles y comportamiento).	
Miembros: Tipo: Informa de si el elemento es un app, una página, una plantilla, un control o un comportamiento.	

Nombre:	Campo
Descripción:	

Clase que se comporta como el participante Hoja del patrón de diseño Composite. Permite representar las distintas propiedades de configuración que puedan tener los elementos de la aplicación móvil (páginas, plantillas, controles y comportamiento).
Miembros: Valor: Valor del campo.

En cambio, el componente de Dominio, implementa las reglas del dominio de la aplicación Web.

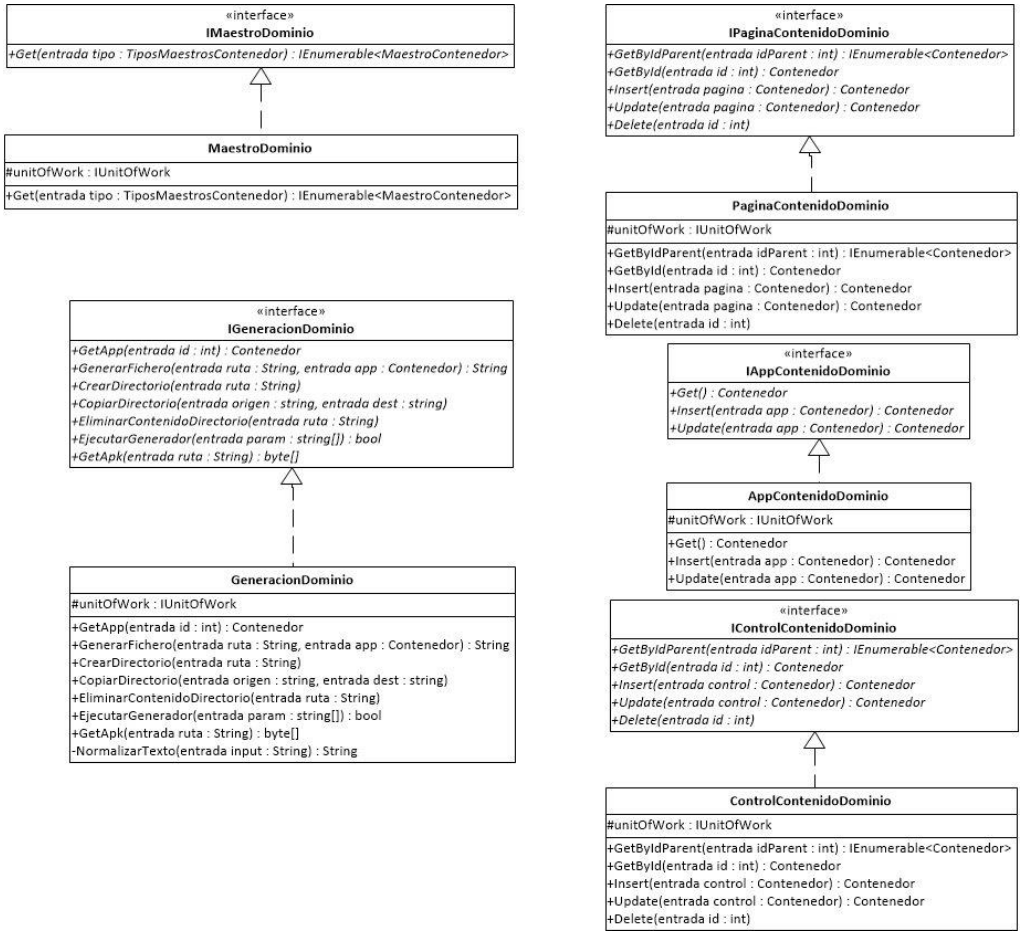


Ilustración 21: Diagrama de clases del componente Dominio

Nombre:	IMaestroDominio
Descripción: Interfaz que define el contrato de la clase que implementa los métodos de recuperación de los elementos maestro.	

Nombre:	MaestroDominio
Descripción: Clase que implementa los métodos de recuperación de los elementos maestro.	
Miembros: unitOfWork: Instancia de la clase que implementa la interfaz IUnitOfWork.	
Métodos:	
Nombre:	Get

Descripción: Recupera la colección de elementos maestros deseada.
Parámetros: tipo: Informa sobre si se desea recuperar el maestro de páginas, plantillas, controles o coportamientos.
Retorno: Colección de elementos maestros recuperada

Nombre:	IAppContenidoDominio
Descripción: Interfaz que define el contrato de la clase que implementa la gestión del contenido del elemento app.	

Nombre:	AppContenidoDominio
Descripción: Clase que implementa la gestión del contenido del elemento app.	
Miembros: unitOfWork: Instancia de la clase que implementa la interfaz IUnitOfWork.	
Métodos:	

Nombre:	Get
Descripción: Recupera el elemento app.	
Retorno: Elemento app recuperado.	

Nombre:	Insert
Descripción: Añade un elemento app.	
Parámetros: app: Elemento app a añadir.	
Retorno: Elemento app añadido.	

Nombre:	Update
Descripción: Actualiza un elemento app.	
Parámetros: app: Elemento app a actualizar.	
Retorno: Elemento app actualizado.	

Nombre:	IPaginaContenidoDominio
Descripción: Interfaz que define el contrato de la clase que implementa la gestión del contenido del elemento página.	

Nombre:	PaginaContenidoDominio

Descripción:
Clase que implementa la gestión del contenido del elemento página.

Miembros:
unitOfWork: Instancia de la clase que implementa la interfaz IUnitOfWork.

Métodos:

Nombre:	GetByIdParent
Descripción: Recupera una colección de elementos página a partir del identificador del elemento padre.	
Parámetros: idParent: Identificador del elemento padre.	
Retorno: Colección de elementos página recuperadas.	

Nombre:	GetById
Descripción: Recupera un elemento página a partir de su identificador.	
Parámetros: id: Identificador del elemento página.	
Retorno: Elemento página recuperada.	

Nombre:	Insert
Descripción: Añade un elemento página.	
Parámetros: pagina: Elemento página a añadir.	
Retorno: Elemento página añadida.	

Nombre:	Update
Descripción: Actualiza un elemento página.	
Parámetros: pagina: Elemento página a actualizar.	
Retorno: Elemento página actualizada.	

Nombre:	Delete
Descripción: Elimina un elemento página a partir de su identificador.	
Parámetros: id: Identificador del elemento página a eliminar.	

Nombre:	IControlContenidoDominio
Descripción: Interfaz que define el contrato de la clase que implementa la gestión del contenido del elemento control.	

Nombre:	ControlContenidoDominio

Descripción:
Clase que implementa la gestión del contenido del elemento control.

Miembros:
unitOfWork: Instancia de la clase que implementa la interfaz IUnitOfWork.

Métodos:

Nombre:	GetByIdParent
Descripción: Recupera una colección de elementos control a partir del identificador del elemento padre.	
Parámetros: idParent: Identificador del elemento padre.	
Retorno: Colección de elementos control recuperados.	

Nombre:	GetById
Descripción: Recupera un elemento control a partir de su identificador.	
Parámetros: id: Identificador del elemento control.	
Retorno: Elemento control recuperado.	

Nombre:	Insert
Descripción: Añade un elemento control.	
Parámetros: control: Elemento control a añadir.	
Retorno: Elemento control añadido.	

Nombre:	Update
Descripción: Actualiza un elemento página.	
Parámetros: control: Elemento control a actualizar.	
Retorno: Elemento control actualizado.	

Nombre:	Delete
Descripción: Elimina un elemento página a partir de su identificador.	
Parámetros: id: Identificador del elemento control a eliminar.	

Nombre:	IGeneracionDominio
Descripción: Interfaz que define el contrato de la clase que implementa la lógica de negocio correspondiente a la generación de la apk.	

Nombre:	GeneracionDominio

Descripción:
Clase que implementa la lógica de negocio correspondiente a la generación de la apk.

Miembros:
unitOfWork: Instancia de la clase que implementa la interfaz IUnitOfWork.

Métodos:

Nombre:	GetApp
Descripción: A partir del identificador de la app recupera todo su contenido estructurado de forma jerárquica.	
Parámetros: id: Identificador del elemento app.	
Retorno: Elemento app recuperado.	

Nombre:	GenerarFichero
Descripción: Genera un fichero con el contenido de la aplicación móvil serializada en json, el nombre del fichero es aleatorio.	
Parámetros: ruta: Ruta dónde se genera el fichero. app: Aplicación móvil a serializar en json y guardar en el fichero.	
Retorno: Nombre del fichero generado.	

Nombre:	CrearDirectorio
Descripción: Función responsable de crear un directorio.	
Parámetros: ruta: Directorio a crear.	

Nombre:	CopiarDirectorio
Descripción: Función recursiva responsable de copiar subdirectorios y ficheros de un directorio a una ruta destino.	
Parámetros: origen: Directorio de origen. dest: Directorio de destino.	

Nombre:	EliminarContenidoDirectorio
Descripción: Función recursiva responsable de eliminar todo el contenido de un directorio.	
Parámetros: ruta: Directorio del que se desea eliminar el contenido.	

Nombre:	EjecutarGenerador
Descripción: Responsable de ejecutar el Modelo generativo.	
Parámetros: param: Parámetros a pasar al modelo generativo.	

Retorno: Booleano informando si la generación del código se ha realizado correctamente.

Nombre:	GenerarApk
Descripción: Ejecuta un script de línea de comandos responsable de generar el paquete apk de la aplicación móvil.	
Parámetros: rutaSolucion: Ruta dónde se encuentra la solución de Visual Studio. rutaSalida: Ruta dónde se guarda el paquete apk de la aplicación móvil.	
Retorno: Booleano informando si la generación del paquete apk se ha realizado correctamente.	

Nombre:	GetApk
Descripción: Recupera el paquete apk.	
Parámetros: ruta: Ruta dónde se encuentra el paquete apk.	
Retorno: Cadena de bytes del paquete apk.	

Nombre:	NormalizarTexto
Descripción: Elimina espacios y acentos de un texto.	
Parámetros: input: Texto que se desea eliminar.	
Retorno: Texto normalizado.	

3. Capa de aplicación

Esta capa es la responsable de coordinar los distintos procesos de la aplicación, por ejemplo: gestión de transacciones; llamadas a métodos de las capas inferiores; conversión de datos; y en definitiva, realizar las llamadas a las tareas necesarias para la realización de las operaciones. Está formada por los componentes Adaptadores y Aplicación, el primero es el responsable de definir los DTOs, que son una representación minimizada de las entidades de dominio. El segundo componente, es el responsable de realizar las tareas de coordinación necesarias.

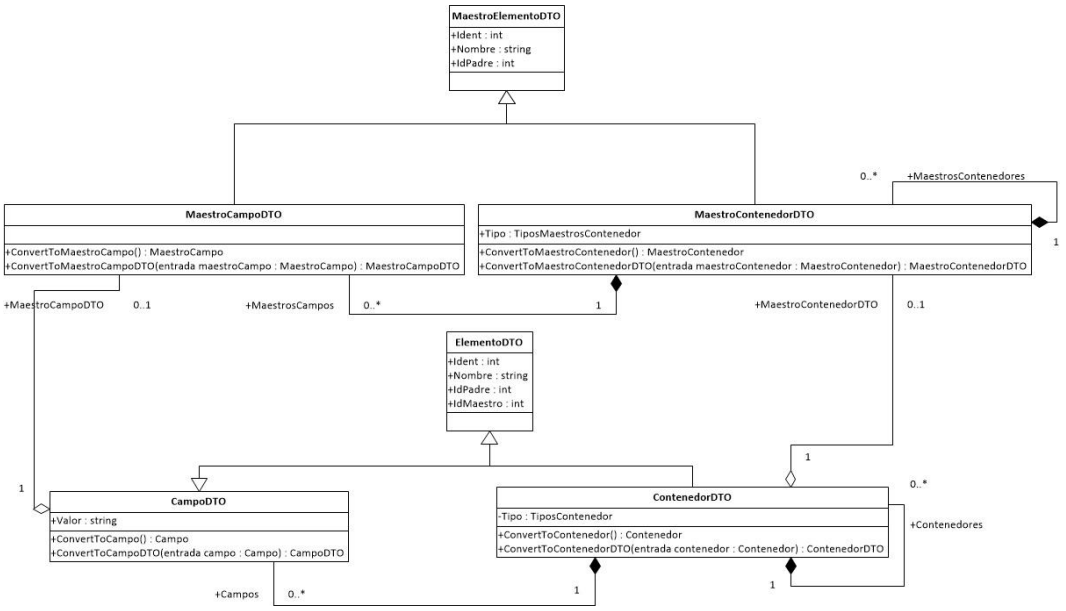


Ilustración 22: Diagrama de clases del componente Adaptadores

Nombre:	MaestroElementoDTO
Descripción: Clase base que representa los maestros de las entidades de dominio.	
Miembros: Ident: Identificador del elemento. Nombre: Nombre del elemento. IdPadre: Identificador del padre del elemento.	

Nombre:	MaestroCampoDTO
Descripción: Clase que representa la entidad de dominio MaestroCampo.	
Métodos:	

Nombre:	ConvertToMaestroCampo
Descripción: Convierte un elemento de tipo MaestroCampoDTO a MaestroCampo.	
Retorno: MaestroCampoDTO convertido a MaestroCampo.	

Nombre:	ConvertToMaestroCampoDTO
Descripción: Convierte un elemento de tipo MaestroCampo a MaestroCampoDTO.	
Parámetros: maestroCampo: Elemento a convertir a MaestroCampoDTO.	
Retorno: MaestroCampo convertido a MaestroCampoDTO.	

Nombre:	MaestroContenedorDTO
Descripción: Clase que representa la entidad de dominio MaestroContenedor.	
Miembros:	

Tipo: Informa de si el elemento es un maestro de páginas, plantillas o controles.

Métodos:

Nombre:	ConvertToMaestroContenedor
Descripción: Convierte un elemento de tipo MaestroContenedorDTO a MaestroContenedor.	
Retorno: MaestroContenedorDTO convertido a MaestroContenedor.	

Nombre:	ConvertToMaestroContenedorDTO
Descripción: Convierte un elemento de tipo MaestroContenedor a MaestroContenedorDTO.	
Parámetros: maestroContenedor: Elemento a convertir a MaestroContenedorDTO.	
Retorno: MaestroContenedor convertido a MaestroContenedorDTO.	

Nombre:	ElementoDTO
Descripción: Clase base que representa las entidades de dominio que se comportan como contenidos de la aplicación móvil.	
Miembros: Ident: Identificador del elemento. Nombre: Nombre del elemento. IdPadre: Identificador del padre del elemento. IdMaestro: Identificador del elemento maestro.	

Nombre:	CampoDTO
Descripción: Clase que representa la entidad de dominio Campo.	

Métodos:

Nombre:	ConvertToCampo
Descripción: Convierte un elemento de tipo CampoDTO a Campo.	
Retorno: CampoDTO convertido a Campo.	

Nombre:	ConvertToCampoDTO
Descripción: Convierte un elemento de tipo Campo a CampoDTO.	
Parámetros: campo: Elemento a convertir a CampoDTO.	
Retorno: Campo convertido a MaestroCampoDTO.	

Nombre:	ContenedorDTO
Descripción: Clase que representa la entidad de dominio Contenedor.	

Miembros:

Tipo: Informa de si el elemento es un app, una página, una plantilla, un control o un comportamiento.

Métodos:

Nombre:	ConvertToContenedor
Descripción: Convierte un elemento de tipo ContenedorDTO a Contenedor.	
Retorno: ContenedorDTO convertido a Contenedor.	

Nombre:	ConvertToContenedorDTO
Descripción: Convierte un elemento de tipo Contenedor a ContenedorDTO.	
Parámetros: contenedor: Elemento a convertir a ContenedorDTO.	
Retorno: Contenedor convertido a ContenedorDTO.	

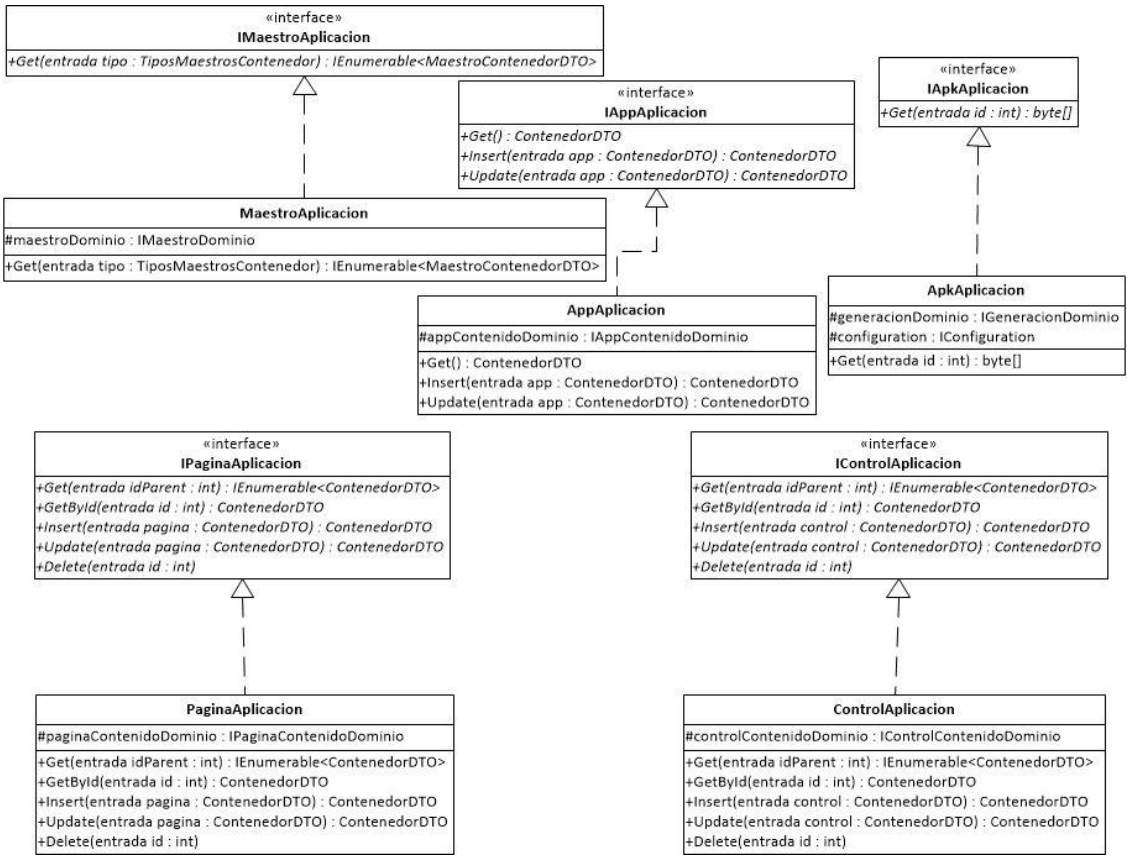


Ilustración 23: Diagrama de clases del componente Aplicación

Nombre:	IMaestroAplicacion
Descripción: Interfaz que define el contrato de la clase que implementa la coordinación de los métodos de dominio de los elementos maestros.	

Nombre:	MaestroAplicacion
Descripción: Clase que implementa la coordinación de los métodos de dominio de los elementos maestros.	
Miembros: maestroDominio: Instancia de la clase que implementa la interfaz IMaestroDominio.	
Métodos:	

Nombre:	Get
Descripción: Llama al método que recupera la colección de los elementos maestros.	
Parámetros: tipo: Informa si se desea recuperar el maestro de páginas, plantillas o controles.	
Retorno: Colección de elementos maestros recuperada.	

Nombre:	IAppAplicacion
Descripción: Interfaz que define el contrato de la clase que implementa la coordinación de los métodos del dominio del elemento app.	

Nombre:	AppAplicacion
Descripción: Clase que implementa la coordinación de los métodos de dominio del elemento app.	
Miembros: appContenidoDominio: Instancia de la clase que implementa la interfaz IAppContenidoDominio.	
Métodos:	

Nombre:	Get
Descripción: Llama al método que recupera un elemento app.	
Retorno: Elemento app recuperado.	

Nombre:	Insert
Descripción: Llama al método que añade un elemento app.	
Parámetros: app: Elemento app a añadir.	
Retorno: Elemento app añadido.	

Nombre:	Update
Descripción: Llama al método que actualiza un elemento app.	
Parámetros: app: Elemento app a actualizar.	
Retorno: Elemento app actualizado.	

Nombre:	IPaginaAplicacion
Descripción: Interfaz que define el contrato de la clase que implementa la coordinación de los métodos de dominio del elemento página.	

Nombre:	PaginaAplicacion
Descripción: Clase que implementa la coordinación de los métodos del dominio del elemento página.	
Miembros: paginaContenidoDominio: Instancia de la clase que implementa la interfaz IPaginaContenidoDominio.	
Métodos:	

Nombre:	Get
Descripción: Llama al método que recupera una colección de elementos página a partir del identificador del elemento padre.	
Parámetros: idParent: Identificador del elemento padre.	
Retorno: Colección de elementos página recuperados.	

Nombre:	GetById
Descripción: Llama al método que recupera un elemento página a partir de su identificador.	
Parámetros: id: Identificador del elemento página a recuperar.	
Retorno: Elemento página recuperado.	

Nombre:	Insert
Descripción: Llama al método que añade un elemento página.	
Parámetros: pagina: Elemento página a añadir.	
Retorno: Elemento página añadida.	

Nombre:	Update
Descripción: Llama al método que actualiza un elemento página.	
Parámetros: pagina: Elemento página a actualizar.	
Retorno: Elemento página actualizada.	

Nombre:	Delete
Descripción: Llama al método que elimina un elemento página.	
Parámetros:	

id: Identificador del elemento página a eliminar.

Nombre:	IControlAplicacion
Descripción: Interfaz que define el contrato de la clase que implementa la coordinación de los métodos de dominio del elemento control.	

Nombre:	ControlAplicacion
Descripción: Clase que implementa la coordinación de los métodos del dominio del elemento control.	
Miembros: controlContenidoDominio: Instancia de la clase que implementa la interfaz IControlContenidoDominio.	
Métodos:	

Nombre:	Get
Descripción: Llama al método que recupera una colección de elementos control a partir del identificador del elemento padre.	
Parámetros: idParent: Identificador del elemento padre.	
Retorno: Colección de elementos control recuperados.	

Nombre:	GetById
Descripción: Llama al método que recupera un elemento control a partir de su identificador.	
Parámetros: id: Identificador del elemento control a recuperar.	
Retorno: Elemento control recuperado.	

Nombre:	Insert
Descripción: Llama al método que añade un elemento control.	
Parámetros: control: Elemento control a añadir.	
Retorno: Elemento control añadido.	

Nombre:	Update
Descripción: Llama al método que actualiza un elemento control.	
Parámetros: control: Elemento control a actualizar.	
Retorno: Elemento control actualizado.	

Nombre:	Delete
Descripción:	

Llama al método que elimina un elemento control.
Parámetros: id: Identificador del elemento control a eliminar.

Nombre:	IApkAplicacion
Descripción: Interfaz que define el contrato de la clase que implementa la coordinación de los métodos de dominio de la generación de la aplicación móvil.	

Nombre:	ApkAplicacion
Descripción: Clase que implementa la coordinación de los métodos de dominio de la generación de la aplicación móvil.	
Miembros: generacionDominio: Instancia de la clase que implementa la interfaz IGeneracionDominio. configuration: Instancia de la clase con los parámetros de configuración.	
Métodos:	

Nombre:	Get
Descripción: Responsable de llamar a los métodos que permiten generar la aplicación móvil.	
Parámetros: id: Identificador del elemento app que se desea generar.	
Retorno: Cadena de bytes del paquete apk.	

4. Capa de servicio

Responsable de exponer puntos de entrada de la parte Back de la aplicación para que se pueda acceder a ella de forma remota, a partir de canales de comunicación y mensajes de datos. Para ello, se implementa un proyecto que utiliza la tecnología Web Api .NET. Sus clases no están relacionadas entre ellas, ya que su finalidad es realizar llamadas a los métodos de la capa de aplicación.

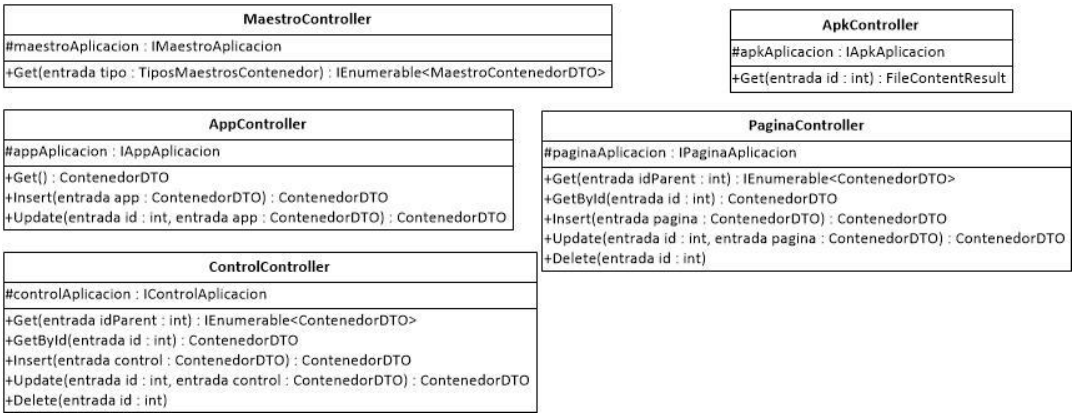


Ilustración 24: Diagrama de clases del componente Web API

Nombre:	MaestroController

Descripción: Clase que expone los métodos de la interfaz IMaestroAplicacion.
Miembros: maestroAplicacion: Instancia de la clase que implementa la interfaz IMaestroAplicacion.
Métodos:

Nombre:	Get
Descripción: Expone el método Get del miembro maestroAplicacion.	
Parámetros: tipo: Informa si se desea recuperar el maestro de páginas, plantillas o controles.	
Retorno: Colección de elementos maestros recuperada.	

Nombre:	AppController
Descripción: Clase que expone los métodos de la interfaz IAppAplicacion.	
Miembros: appAplicacion: Instancia de la clase que implementa la interfaz IAppAplicacion.	
Métodos:	

Nombre:	Get
Descripción: Expone el método Get del miembro appAplicacion.	
Retorno: Elemento app recuperado.	

Nombre:	Insert
Descripción: Expone el método Insert del miembro appAplicacion.	
Parámetros: app: Elemento app a añadir.	
Retorno: Elemento app añadido.	

Nombre:	Update
Descripción: Expone el método Update del miembro appAplicacion.	
Parámetros: id: Identificador de app a actualizar. app: Elemento app a actualizar.	
Retorno: Elemento app actualizado.	

Nombre:	PaginaController
Descripción: Clase que expone los métodos de la interfaz IPaginaAplicacion.	
Miembros: paginaAplicacion: Instancia de la clase que implementa la interfaz IAppAplicacion.	
Métodos:	

Nombre:	Get
Descripción: Expone el método Get del miembro paginaAplicacion.	
Parámetros: idParent: Identificador del elemento padre.	
Retorno: Colección de elementos página recuperados.	

Nombre:	GetById
Descripción: Expone el método GetById del miembro paginaAplicacion.	
Parámetros: id: Identificador del elemento página a recuperar.	
Retorno: Elemento página recuperado.	

Nombre:	Insert
Descripción: Expone el método Insert del miembro paginaAplicacion.	
Parámetros: pagina: Elemento página a añadir.	
Retorno: Elemento página añadida.	

Nombre:	Update
Descripción: Expone el método Update del miembro paginaAplicacion.	
Parámetros: id: Identificador de la página a actualizar. pagina: Elemento página a actualizar.	
Retorno: Elemento página actualizada.	

Nombre:	Delete
Descripción: Expone el método Delete del miembro paginaAplicacion.	
Parámetros: id: Identificador del elemento página a eliminar.	

Nombre:	ControlController
Descripción: Clase que expone los métodos de la interfaz IControlAplicacion.	
Miembros: controlAplicacion: Instancia de la clase que implementa la interfaz IControlAplicacion.	
Métodos:	

Nombre:	Get
Descripción: Expone el método Get del miembro controlAplicacion.	
Parámetros:	

idParent: Identificador del elemento padre.
Retorno: Colección de elementos control recuperados.

Nombre:	GetById
Descripción: Expone el método GetById del miembro controlAplicacion.	
Parámetros: id: Identificador del elemento control a recuperar.	
Retorno: Elemento control recuperado.	

Nombre:	Insert
Descripción: Expone el método Insert del miembro controlAplicacion.	
Parámetros: control: Elemento control a añadir.	
Retorno: Elemento control añadido.	

Nombre:	Update
Descripción: Expone el método Update del miembro controlAplicacion.	
Parámetros: id: Identificador del control a actualizar. control: Elemento control a actualizar.	
Retorno: Elemento control actualizado.	

Nombre:	Delete
Descripción: Expone el método Delete del miembro controlAplicacion.	
Parámetros: id: Identificador del elemento control a eliminar.	

Nombre:	ApkController
Descripción: Clase que expone los métodos de la interfaz IApkAplicacion.	
Miembros: apkAplicacion: Instancia de la clase que implementa la interfaz IApkAplicacion.	
Métodos:	

Nombre:	Get
Descripción: Expone el método Get del miembro apkAplicacion.	
Parámetros: id: Identificador del elemento app que se desea generar.	
Retorno: Fichero binario que representa la apk.	

ii. Parte Front

Esta parte es la que interactúa directamente con el usuario, permitiendo que el usuario pueda resolver las tareas de Edición de contenidos, Iniciar generación y Descarga de la APK de la Ilustración 12: Diagrama BPMN del proceso generativo. Se utiliza para su desarrollo el lenguaje de programación HTML, CSS, JavaScrip y el framework AngularJS. En la Ilustración 25: Arquitectura parte Front de la aplicación Web puede verse que la arquitectura está formada por tres capas: la capa de modelos, la capa de cliente Api y la capa de presentación.

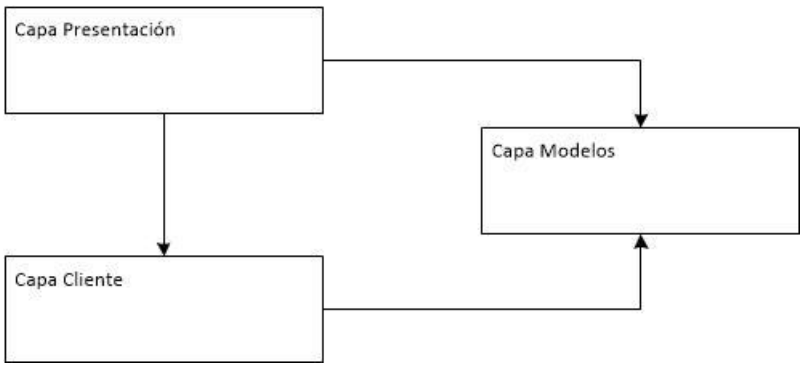


Ilustración 25: Arquitectura parte Front de la aplicación Web

1. *Capa de modelos*

Esta capa implementa los modelos de la parte Front, estos son una estructura que permite representar la información de las operaciones de la parte Front y de los datos a pintar en la capa de presentación. La capa de presentación no siempre usa estos modelos, también usa los DTOs que se reciben de la parte Back en las peticiones.

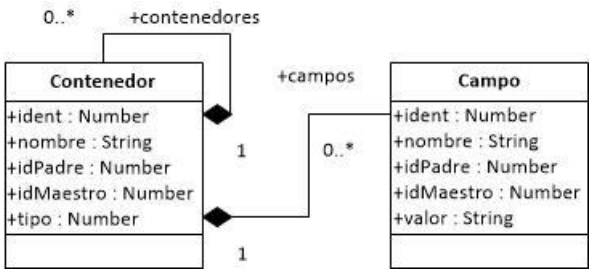


Ilustración 26: Diagrama de clases de la capa Modelos

Nombre:	Contenedor
Descripción: Permite representar los elementos contenedores de los contenidos de la aplicación (app, páginas, plantillas, controles y comportamiento).	
Miembros: ident: Identificador del contendor. nombre: Nombre del contenedor. idPadre: Identificador del padre. idMaestro: Identificador del maestro. tipo: Informa de si el contenedor es un app, una página, una plantilla, un control o un	

comportamiento.

Nombre:	Campo
Descripción: Permite representar los distintos campos que pueden tener los elementos de la aplicación móvil (app, páginas, plantillas, controles y comportamiento).	
Miembros: ident: Identificador del campo. nombre: Nombre del campo. idPadre: Identificador del padre. idMaestro: Identificador del maestro. valor: Valor del campo.	

2. *Capa de cliente Api*

Esta capa se comporta como el cliente de la parte Back de la aplicación Web, es la responsable de comunicar la parte Front con la Back mediante el envío peticiones y la recepción de las respuestas. Implementa las clases y métodos necesarios para las tareas de comunicación. Las clases de esta capa no se relacionan entre ellas, ya que solamente son instanciadas por las clases de capas superiores.

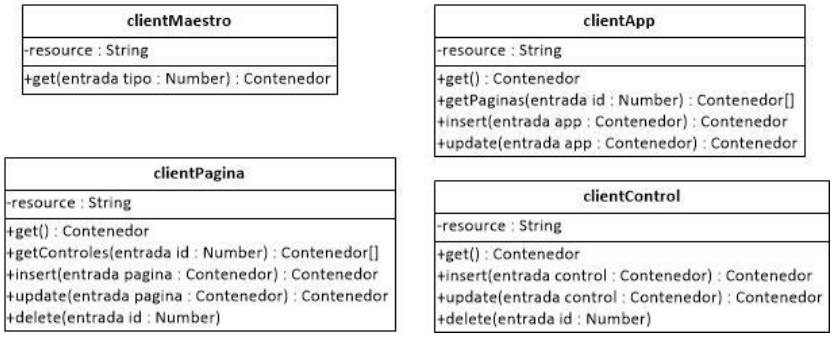


Ilustración 27: Diagrama de clases de la capa Cliente

Nombre:	clientMaestro
Descripción: Clase que se comunica con los recursos de la Web Api correspondientes a los elementos maestros.	
Miembros: resource: Informa de la ruta del recurso maestro de la Web Api.	
Métodos:	

Nombre:	get
Descripción: Llama al método Web Api responsable de recuperar una colección de elementos maestros.	
Parámetros: tipo: Informa del tipo de maestro a recuperar (páginas, plantillas o controles).	
Retorno: Colección de elementos maestros recuperados.	

Nombre:	clientApp

Descripción:
Clase que se comunica con los recursos de la Web Api correspondientes al elemento app.

Miembros:
resource: Informa de la ruta del recurso app de la Web Api.

Métodos:

Nombre:	get

Descripción:
Llama al método Web Api responsable de recuperar un elemento app.

Retorno:
Elemento app recuperado.

Nombre:	getPaginas

Descripción:
Llama al método Web Api responsable de recuperar una colección de elementos página hijos de un elemento app.

Parámetros:
id: Identificador del elemento app.

Retorno:
Colección de elementos página recuperados.

Nombre:	insert

Descripción:
Llama al método Web Api responsable de añadir un elemento app.

Parámetros:
app: Elemento app a añadir.

Retorno:
Elemento app añadido.

Nombre:	update

Descripción:
Llama al método Web Api responsable de actualizar un elemento app.

Parámetros:
app: Elemento app a actualizar.

Retorno:
Elemento app actualizado.

Nombre:	clientPagina

Descripción:
Clase que se comunica con los recursos de la Web Api correspondientes al elemento página.

Miembros:
resource: Informa de la ruta del recurso página de la Web Api.

Métodos:

Nombre:	get

Descripción:
Llama al método Web Api responsable de recuperar un elemento página.

Retorno:
Elemento página recuperado.

Nombre:	getControles
Descripción: Llama al método Web Api responsable de recuperar una colección de elementos control hijos de un elemento página.	
Parámetros: id: Identificador del elemento página.	
Retorno: Colección de elementos control recuperados.	

Nombre:	insert
Descripción: Llama al método Web Api responsable de añadir un elemento página.	
Parámetros: pagina: Elemento página a añadir.	
Retorno: Elemento página añadido.	

Nombre:	update
Descripción: Llama al método Web Api responsable de actualizar un elemento página.	
Parámetros: pagina: Elemento página a actualizar.	
Retorno: Elemento página actualizado.	

Nombre:	delete
Descripción: Llama al método Web Api responsable de eliminar un elemento página.	
Parámetros: id: Identificador del elemento página a eliminar.	

Nombre:	clientControl
Descripción: Clase que se comunica con los recursos de la Web Api correspondientes al elemento control.	
Miembros: resource: Informa de la ruta del recurso control de la Web Api.	
Métodos:	

Nombre:	get
Descripción: Llama al método Web Api responsable de recuperar un elemento control.	
Retorno: Elemento control recuperado.	

Nombre:	insert
Descripción: Llama al método Web Api responsable de añadir un elemento control.	
Parámetros: control: Elemento control a añadir.	
Retorno:	

Elemento control añadido.

Nombre:	update
Descripción: Llama al método Web Api responsable de actualizar un elemento control.	
Parámetros: control: Elemento control a actualizar.	
Retorno: Elemento control actualizado.	

Nombre:	delete
Descripción: Llama al método Web Api responsable de eliminar un elemento control.	
Parámetros: id: Identificador del elemento control a eliminar.	

3. Capa de presentación

Es la capa responsable de pintar los datos e interactuar con el usuario, para su implementación se utiliza el patrón de diseño MVVM, los lenguajes de programación HTML5, CSS, JavaScript y los frameworks AngularJS y Bootstrap. El patrón de diseño MVVM [21] permite desacoplar la interfaz de usuario de la lógica de la aplicación, está formado por tres partes: la vista; el modelo; y el modelo vista. La vista solamente pinta mediante los controles HTML los datos recibidos mediante los modelos. Tal como se ha visto en la capa de modelos, los modelos son una estructura que representan los datos a pasar a la vista. El modelo de vista es el mediador entre la vista y el modelo, contiene todo lo referente a la lógica de presentación [21]. A continuación se puede ver la documentación de los modelos de vista.

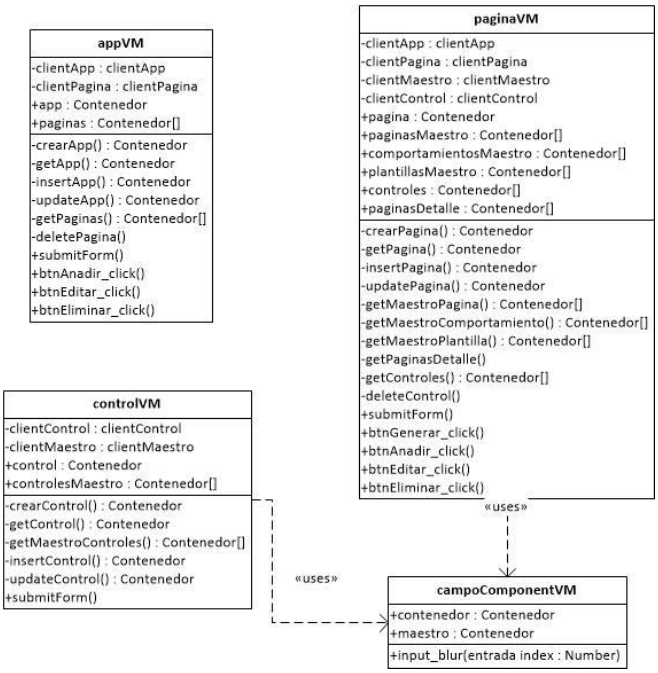


Ilustración 28: Diagrama de clases de los modelos de vista de la capa presentación

Nombre:	appVM

Descripción:
Modelo de la vista de creación de la app, gestiona la información de la aplicación móvil y de sus páginas.

Miembros:
clientApp: Instancia de la clase clientApp.
clientPagina: Instancia de la clase clientPagina.
app: Elemento app que contiene la información de los datos de la aplicación móvil.
paginas: Colección de elementos página de la aplicación móvil.

Métodos:

Nombre:	crearApp
Descripción:	En el caso que no exista, crea un elemento app.
Retorno:	Elemento app creado.

Nombre:	getApp
Descripción:	Llama al método de la capa cliente responsable de recuperar la información del elemento app.
Retorno:	Elemento app con la información de la aplicación móvil.

Nombre:	insertApp
Descripción:	Llama al método de la capa cliente responsable de añadir un elemento app.
Retorno:	Elemento app añadido.

Nombre:	updateApp
Descripción:	Llama al método de la capa cliente responsable de actualizar un elemento app.
Retorno:	Elemento app actualizado.

Nombre:	getPaginas
Descripción:	Llama al método de la capa cliente responsable de recuperar la colección de elementos página del elemento app.
Retorno:	Colección de elementos página recuperados.

Nombre:	deletePagina
Descripción:	Llama al método de la capa cliente responsable de eliminar un elemento página.

Nombre:	submitForm
Descripción:	Responsable de tratar el evento submit del formulario, llama al método insertApp o updateApp en función de si el elemento app ya existe.

Nombre:	btnGenerar_click
Descripción: Método responsable de tratar el evento click del botón de generar apk. Se descarga la apk.	

Nombre:	btnAnadir_click
Descripción: Método responsable de tratar el evento click del botón de añadir página. Redirige a la vista de creación de página.	

Nombre:	btnEditar_click
Descripción: Método responsable de tratar el evento click del botón de editar página. Redirige a la vista de creación de página, pasando el identificador del elemento app.	

Nombre:	btnEliminar_click
Descripción: Método responsable de tratar el evento click del botón de eliminar página. Llama al método deletePagina.	

Nombre:	paginaVM
Descripción: Modelo de la vista de creación de la página, gestiona la información de la página y de sus controles.	
Miembros: clientApp: Instancia de la clase clientApp. clientPagina: Instancia de la clase clientPagina. clientMaestro: Instancia de la clase clientMaestro. clientControl: Instancia de la clase clientControl. pagina: Elemento página que contiene la información de los datos de la página. paginasMaestro: Colección de elementos del maestro de páginas. comportamientosMaestro: Colección de elementos del maestro de comportamientos de la página. plantillasMaestro: Colección de elementos del maestro de plantillas de una página. controles: Colección de elementos control de la página. paginasDetalle: Colección de elementos página cuyo comportamiento se basa en la visualización del detalle de los registros.	
Métodos:	

Nombre:	crearPagina
Descripción: En el caso que no exista, crea un elemento página.	
Retorno: Elemento página creado.	

Nombre:	getPagina
Descripción: Llama al método de la capa cliente responsable de recuperar la información del elemento página.	
Retorno: Elemento página con la información de la página.	

Nombre:	insertPagina
Descripción: Llama al método de la capa cliente responsable de añadir un elemento página.	
Retorno: Elemento página añadido.	

Nombre:	updatePagina
Descripción: Llama al método de la capa cliente responsable de actualizar un elemento página.	
Retorno: Elemento página actualizado.	

Nombre:	getMaestroPagina
Descripción: Llama al método de la capa cliente responsable de recuperar los elementos del maestro de páginas.	
Retorno: Colección de elementos del maestro de páginas recuperados.	

Nombre:	getMaestroComportamiento
Descripción: Llama al método de la capa cliente responsable de recuperar los elementos del maestro de comportamientos.	
Retorno: Colección de elementos del maestro de comportamientos recuperados.	

Nombre:	getMaestroPlantilla
Descripción: Llama al método de la capa cliente responsable de recuperar los elementos del maestro de plantillas.	
Retorno: Colección de elementos del maestro de plantillas recuperados.	

Nombre:	getPaginasDetalle
Descripción: Llama al método de la capa cliente responsable de recuperar los elementos página cuyo comportamiento se basa en la visualización del detalle de los registros.	
Retorno: Colección de elementos del maestro de plantillas recuperados.	

Nombre:	getControles
Descripción: Llama al método de la capa cliente responsable de recuperar la colección de elementos control del elemento página.	
Retorno: Colección de elementos control recuperados.	

Nombre:	deleteControl
Descripción: Llama al método de la capa cliente responsable de eliminar un elemento control.	

Nombre:	submitForm
Descripción: Responsable de tratar el evento submit del formulario, llama al método insertPagina o updatePagina en función de si el elemento página ya existe.	

Nombre:	btnAnadir_click
Descripción: Método responsable de tratar el evento click del botón de añadir control. Redirige a la vista de creación de control.	

Nombre:	btnEditar_click
Descripción: Método responsable de tratar el evento click del botón de editar control. Redirige a la vista de creación de control, pasando el identificador del elemento página.	

Nombre:	btnEliminar_click
Descripción: Método responsable de tratar el evento click del botón de eliminar control. Llama al método deleteControl.	

Nombre:	controlVM
Descripción: Modelo de la vista de creación del control, gestiona la información del control.	
Miembros: clientControl: Instancia de la clase clientControl. clientMaestro: Instancia de la clase clientMaestro. control: Elemento control que contiene la información de los datos del control. controlesMaestro: Colección de elementos del maestro de controles.	
Métodos:	

Nombre:	crearControl
Descripción: En el caso que no exista, crea un elemento control.	
Retorno: Elemento control creado.	

Nombre:	getControl
Descripción: Llama al método de la capa cliente responsable de recuperar la información del elemento control.	
Retorno: Elemento control con la información del control.	

Nombre:	getMaestroControles
Descripción: Llama al método de la capa cliente responsable de recuperar los elementos del maestro de controles.	
Retorno: Colección de elementos del maestro de controles recuperados.	

Nombre:	insertControl
Descripción: Llama al método de la capa cliente responsable de añadir un elemento control.	
Retorno: Elemento control añadido.	

Nombre:	updateControl
Descripción: Llama al método de la capa cliente responsable de actualizar un elemento control.	
Retorno: Elemento control actualizado.	

Nombre:	submitForm
Descripción: Responsable de tratar el evento submit del formulario, llama al método insertApp o updateApp en función de si el elemento control ya existe.	

Nombre:	campoComponentVM
Descripción: Modelo de la vista campo, esta vista es un componente usado en otras vistas. Configura el comportamiento de los elementos representados con el modelo contenedor, a partir de los campos de su elemento maestro.	
Miembros: contenedor: Elemento del que se desea configurar su comportamiento. maestro: Elemento maestro del miembro contenedor, contiene los distintos campos de configuración.	
Métodos:	

Nombre:	Input_blur
Descripción: Responsable de tratar el evento blur de los distintos inputs de la vista, estos inputs son los controles HTML5 relacionados con los campos contenidos en el miembro maestro. Asigna al miembro contenedor el valor introducido en el input.	
Parámetros: index: Índice del control input, permite relacionar un input con su campo correspondiente.	

b. Modelo generativo

Subsistema que permite resolver las tareas de Generación de la aplicación móvil y Publicación de la parte servidora de la Ilustración 12: Diagrama BPMN del proceso generativo. Para su implementación se utiliza el lenguaje de programación Ruby y las plantillas ERb. Este es el responsable de generar el resultado final de la solución a partir de una entrada.

La entrada del modelo generativo es un fichero formado por un DSL que representa los contenidos editados por el usuario mediante la aplicación Web detallada en el apartado Aplicación Web.

El resultado final es una aplicación móvil basada en una arquitectura cliente servidor, donde la parte cliente se presenta como un fichero APK listo para ser instalado en el dispositivo móvil

del usuario, que va a permitir la interacción con este. La parte servidora es una Web Api que expone los métodos necesarios para realizar las operaciones CRUD en una base de datos.

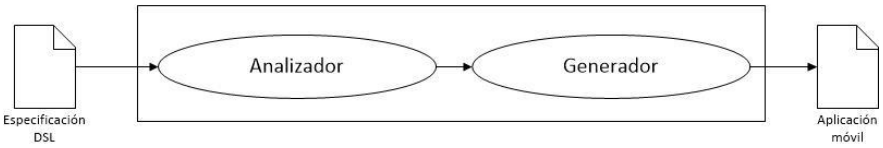


Ilustración 29: Diagrama del modelo generativo

En la Ilustración 29: Diagrama del modelo generativo [2], se pueden observar los distintos elementos participantes del modelo generativo.

- Especificación DSL: Un fichero de entrada especifica el DSL representado mediante su sintaxis serializada en Json. El DSL está definido por las entidades del dominio de la Ilustración 20: Diagrama de clases del componente Modelo.
- Analizador: Responsable de leer la información proporcionada por el DSL y, a partir de ella, generar una estructura de trabajo.
- Generador: Responsable de generar el resultado final del modelo generativo a partir de la estructura de trabajo proporcionada por parte del analizador.
- Aplicación móvil: Resultado final del modelo generativo.

i. Analizador

Este componente del modelo generativo se encarga de leer un fichero que contiene la especificación del DSL y convierte su información en una estructura que permita trabajar, de una forma más simple y menos abstracta, con el lenguaje de programación Ruby. Para la conversión de la información del DSL a la estructura deseada se utiliza el patrón de diseño Factory [36].

La estructura de trabajo creada por el analizador es muy parecida a las entidades de dominio definidas en el punto Parte Back, como se puede ver en el diagrama de clases de la Ilustración 20: Diagrama de clases del componente Modelo. La diferencia más significativa es la ausencia de una estructura que permita representar los maestros de los elementos que se comportan como contenidos de la aplicación móvil. También se puede observar que no existe el extremo Padre en la composición de Contenedor a Elemento.

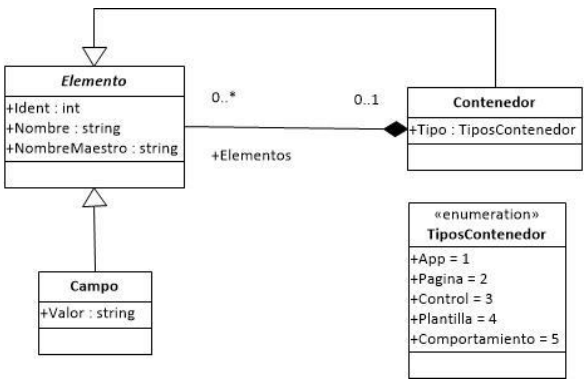


Ilustración 30: Diagrama de clases de la estructura de trabajo

Nombre:	Elemento
Descripción: Representa los distintos elementos de los contenidos de la aplicación móvil.	
Miembros: Ident: Identificador del elemento. Nombre: Nombre del elemento. NombreMaestro: Nombre del maestro del elemento	

Nombre:	Contenedor
Descripción: Representa los elementos contenedores de los contenidos de la aplicación (app, páginas, plantillas, controles y comportamiento).	
Miembros: Tipo: Informa de si el elemento es un app, una página, una plantilla, un control o un comportamiento.	

Nombre:	Campo
Descripción: Representa las distintas propiedades de configuración que pueden tener los elementos de la aplicación móvil (páginas, plantillas, controles y comportamiento).	
Miembros: Valor: Valor del campo.	

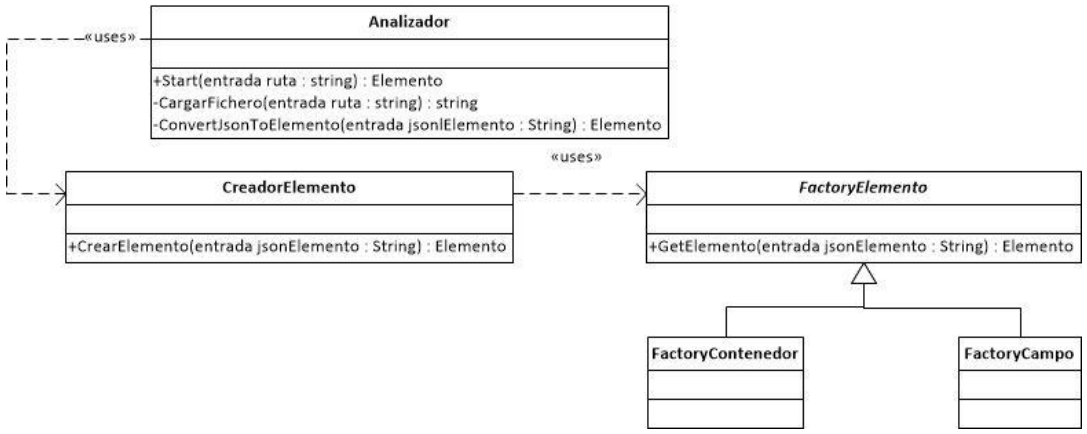


Ilustración 31: Diagrama de clases del analizador

Nombre:	Analizador
Descripción: Responsable de leer el fichero de entrada y convertir su contenido a un elemento app.	
Métodos:	

Nombre:	Start
Descripción: Inicia el proceso de análisis, para ello llama al método CargarFichero y ConvertJsonToElemento.	
Parámetros: ruta: Informa de la ruta del fichero de entrada.	
Retorno:	

Elemento app resultado de la conversión.

Nombre:	CargarFichero
Descripción: Recupera el contenido del fichero de entrada.	
Parámetros: ruta: Informa de la ruta del fichero de entrada.	
Retorno: Contenido del fichero de entrada.	

Nombre:	ConvertJsonToElemento
Descripción: Método recursivo que convierte los elementos serializados en Json a objetos de la estructura de trabajo definida. Para la conversión se usa la clase CreadorElemento.	
Parámetros: jsonElemento: Elemento serializado a Json.	
Retorno: Elemento de la estructura de trabajo definida.	

Nombre:	CreadorElemento
Descripción: Responsable de gestionar la creación de un elemento a partir de su representación serializada en Json.	

Métodos:

Nombre:	CrearElemento
Descripción: En función del parámetro de entrada usa la clase FactoryContenedor o FactoryCampo.	
Parámetros: jsonElemento: Elemento serializado a Json.	
Retorno: Elemento creado.	

Nombre:	FactoryElemento
Descripción: Clase abstracta responsable de instanciar un elemento a partir de su representación serializada.	

Métodos:

Nombre:	GetElemento
Descripción: Método abstracto responsable de instanciar un elemento a partir de su representación serializada.	
Parámetros: jsonElemento: Elemento serializado a Json.	
Retorno: Elemento instanciado.	

Nombre:	FactoryContenedor

Descripción:
Clase responsable de instanciar un elemento de tipo contenedor.

Nombre:	FactoryCampo
Descripción:	
Clase responsable de instanciar un elemento de tipo campo.	

ii. Generador

Este componente del modelo generativo, a partir de la estructura de trabajo resultante del analizador, es el responsable de generar automáticamente el código fuente de la aplicación móvil, crear la APK y publicar la parte servidora.

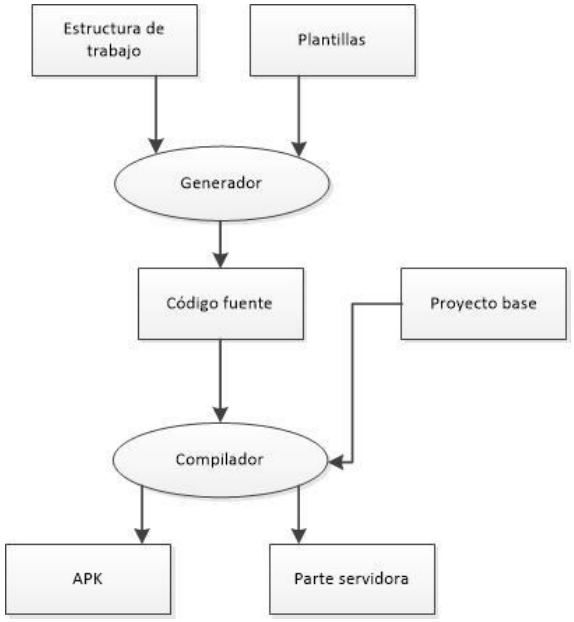


Ilustración 32: Diagrama de flujo del generador

Tal como se puede ver en la Ilustración 32: Diagrama de flujo del generador, este está formado por distintos elementos, a continuación se realiza una breve descripción de cada uno de ellos.

- Estructura de trabajo: Es la estructura de salida de la parte del analizador, en este caso se comporta como una estructura de entrada, informando al generador de los contenidos de la aplicación móvil.
- Plantillas: Conjunto de plantillas ERb que ayudan a la realización del proceso de generación del resultado de salida. A partir de la información de la estructura de trabajo recibida, el generador aplica unas plantillas u otras para transformar los contenidos de la aplicación móvil en código fuente.
- Generador: Parte responsable de generar el código fuente de la aplicación móvil a partir de la estructura de trabajo.
- Código fuente: Salida del generador, es una parte del código fuente de la aplicación móvil, concretamente son los activos reutilizables que contienen una variabilidad.

- Proyecto base: Es el código fuente restante de la aplicación móvil, éstos son activos reutilizables sin ninguna variabilidad.
- Compilador: Parte responsable de empaquetar la aplicación móvil en un fichero APK y de publicar su parte servidora.
- APK: Fichero donde se empaqueta la aplicación móvil.
- Parte servidora: Parte back de la aplicación móvil.

El resultado de salida del generador (APK y parte servidora) corresponden a los dos partes que se pueden ver en la Ilustración 33: Arquitectura aplicación móvil, está basada en una arquitectura cliente servidor. La APK empaqueta el código fuente compilado y los recursos de la Aplicación móvil de la ilustración. En cambio, la parte servidora contiene el código fuente compilado y los datos del Servidor de la ilustración.

Se puede observar que para llegar al resultado final del modelo generativo es necesaria la interacción de los dos subsistemas de la solución, ya que mediante la Aplicación Web el usuario es capaz de gestionar el contenido de la aplicación móvil, esta información es proporcionada al modelo generativo mediante una especificación DSL. El componente Analizador del modelo generativo convierte la información recibida mediante el DSL en una estructura más adecuada para trabajar en Ruby. Finalmente, el componente Generador recibe la nueva estructura y genera la aplicación móvil.

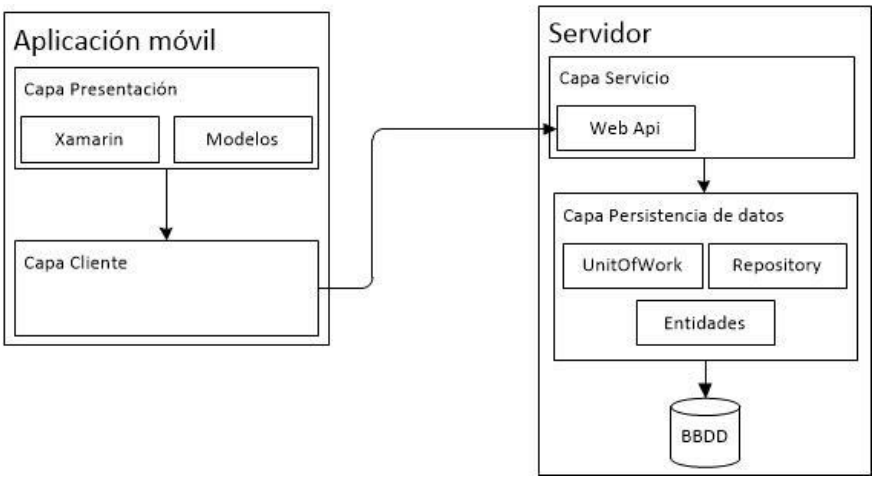


Ilustración 33: Arquitectura aplicación móvil

La parte cliente está formada por los siguientes capas.

- Capa de presentación: Responsable de interactuar con el usuario a partir de las vistas implementadas con la tecnología Xamarin, para su implementación se utiliza el patrón de diseño MVVM, así que también es necesario implementar la parte de modelos. Tanto las vistas como los modelos se comportan como activos reutilizables con variabilidad.
- Capa cliente: Responsable de realizar la comunicación con la parte servidora de la aplicación móvil, se comporta como un activo reutilizable sin variabilidad.

La parte servidora está formada por las siguientes capas.

- Capa de servicio: Mediante la tecnología Web Api expone los métodos necesarios para que los clientes puedan realizar las operaciones CRUD, las clases de la Web Api se comportan como activos reutilizables con variabilidad.
- Capa persistencia de datos: Responsable de comunicarse con la base de datos, para ello se utiliza el ORM Entity Framework y los patrones de diseño Repository y UnitOfWork. Tanto las entidades como la clase UnitOfWork se comportan como activos reutilizables con variabilidad.

En la Ilustración 32: Diagrama de flujo del generador se puede observar que hay tres elementos que se comportan como entradas al sistema, estos son: la estructura de trabajo; las plantillas y un proyecto base. El generador a partir de la estructura de trabajo y las plantillas es capaz de realizar las transformaciones necesarias para generar los activos reutilizables con variabilidad. En la siguiente tabla se detallan las partes de la aplicación móvil que se generan a partir de la estructura de trabajo.

Destino transformación	Origen transformación	Descripción
Vistas Xamarin	Página tipo listado Página tipo detalle	Cada una de las páginas se trasforma en una vista y en un modelo de vista distinto. Las páginas de tipo listado dan lugar a las vistas que listan todos los registros y las de tipo detalle dan lugar a las vistas que muestran el detalle de un registro.
Vista menú de navegación	Páginas tipo listado	Cada una de las páginas se transforma en un elemento del menú de navegación.
Entidad	Página tipo detalle	La página se transforma en la entidad, y los controles que contiene en los atributos de la entidad.
Modelo	Página tipo detalle	La página se transforma en el modelo, y los controles que contiene en los atributos del modelo.
Clases de la capa de servicio	Página tipo detalle	La página se transforma en una clase formada por métodos que se comportan como puntos de entrada para la realización de las operaciones CRUD.
UnitOfWork	Página tipo detalle	Cada una de las páginas se transforma como un atributo de la clase UnitOfWork.

Hay otras partes de la aplicación móvil que vienen del proyecto base, estas se han identificado como elementos reutilizables sin variabilidad, estos son: estilos de las vistas; modelo de vista y modelo del menú de navegación; comunicación con las partes nativas de los dispositivos móviles; capa cliente de la aplicación móvil y el Repository del servidor.

Con la finalidad de reducir la complejidad en el desarrollo del generador, se utiliza un diseño modular, donde cada uno de los distintos módulos se responsabiliza de unas transformaciones en concreto. A continuación, se enumeran los distintos módulos del generador.

1. Módulo de generación de modelos y entidades
2. Módulo de generación de vistas
3. Módulo de generación de Web Api
4. Módulo de generación de acceso a datos
5. Módulo de generación de configuración
6. Módulo de generación del fichero apk
7. Módulo de publicación de la parte servidora

1. Módulo de generación de modelos y entidades

Modulo responsable de generar el código fuente de los modelos de la aplicación móvil y de las entidades de la parte servidora.

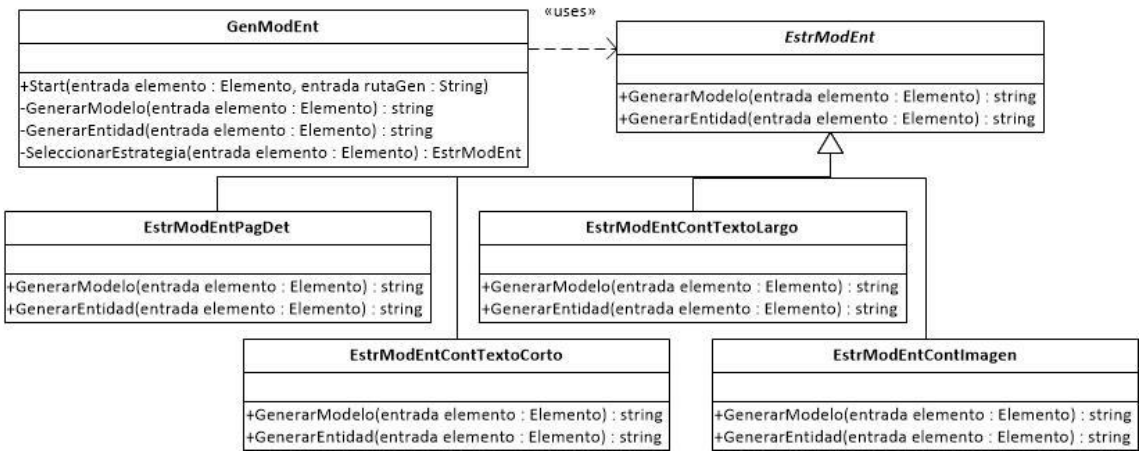


Ilustración 34: Diagrama de clases del módulo generador de modelos y entidades

Nombre:	GenModEnt
Descripción: Clase responsable de gestionar la generación de código automático de los modelos y las entidades.	
Métodos:	
Nombre:	Start
Descripción: Método público responsable de iniciar el proceso de generación de código automático de los modelos y las entidades.	
Parámetros: elemento: Elemento del que se desea realizar la transformación. rutaGen: Directorio dónde se guarda el código fuente generado.	
Nombre:	GenerarModelo
Descripción: Método responsable de realizar la generación de código del modelo a partir de la estrategia de transformación obtenida con el método SeleccionarEstrategia. Es un	

método recursivo que es llamado por cada uno de los hijos del elemento.
Parámetros: elemento: Elemento del que se desea realizar la transformación.
Retorno: Código fuente generado.

Nombre:	GenerarEntidad
Descripción: Método responsable de realizar la generación de código de la entidad a partir de la estrategia de transformación obtenida con el método SeleccionarEstrategia. Es un método recursivo que es llamado por cada uno de los hijos del elemento.	
Parámetros: elemento: Elemento del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	SeleccionarEstrategia
Descripción: Método responsable de determinar la estrategia de transformación a utilizar.	
Parámetros: elemento: Elemento que determina la estrategia a utilizar.	
Retorno: Estrategia de transformación a utilizar.	

Nombre:	EstrModEnt
Descripción: Clase abstracta que define un contrato común a las distintas estrategias de transformación.	

Nombre:	EstrModEntPagDet
Descripción: Clase responsable de implementar la estrategia de transformación de un elemento página con comportamiento detalle.	

Métodos:

Nombre:	GenerarModelo
Descripción: Método responsable de implementar la estrategia de transformación de un modelo para un elemento página con comportamiento detalle.	
Parámetros: elemento: Elemento página con comportamiento detalle del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarEntidad
Descripción: Método responsable de implementar la estrategia de transformación de una entidad para un elemento página con comportamiento detalle.	
Parámetros: elemento: Elemento página con comportamiento detalle del que se desea realizar la	

transformación.
Retorno: Código fuente generado.

Nombre:	EstrModEntContTextoCorto

Descripción:
Clase responsable de implementar la estrategia de transformación de un elemento control texto corto.

Métodos:

Nombre:	GenerarModelo
Descripción: Método responsable de implementar la estrategia de transformación de un modelo para un elemento control texto corto.	
Parámetros: elemento: Elemento control texto corto del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarEntidad
Descripción: Método responsable de implementar la estrategia de transformación de una entidad para un elemento control texto corto.	
Parámetros: elemento: Elemento control texto corto del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	EstrModEntContTextoLargo

Descripción:
Clase responsable de implementar la estrategia de transformación de un elemento control texto largo.

Métodos:

Nombre:	GenerarModelo
Descripción: Método responsable de implementar la estrategia de transformación de un modelo para un elemento control texto largo.	
Parámetros: elemento: Elemento control texto largo del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarEntidad
Descripción: Método responsable de implementar la estrategia de transformación de una entidad para un elemento control texto largo.	
Parámetros: elemento: Elemento control texto largo del que se desea realizar la transformación.	
Retorno:	

Código fuente generado.

Nombre:	EstrModEntContImagen
Descripción: Clase responsable de implementar la estrategia de transformación de un elemento control imagen.	

Métodos:

Nombre:	GenerarModelo
Descripción: Método responsable de implementar la estrategia de transformación de un modelo para un elemento control imagen.	
Parámetros: elemento: Elemento control imagen del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarEntidad
Descripción: Método responsable de implementar la estrategia de transformación de una entidad para un elemento control imagen.	
Parámetros: elemento: Elemento control imagen del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

2. Módulo de generación de vistas

Módulo responsable de generar el código fuente de las vistas y los modelos de vistas de la parte de Xamarin, para ella se utiliza el patrón de diseño MVVM.

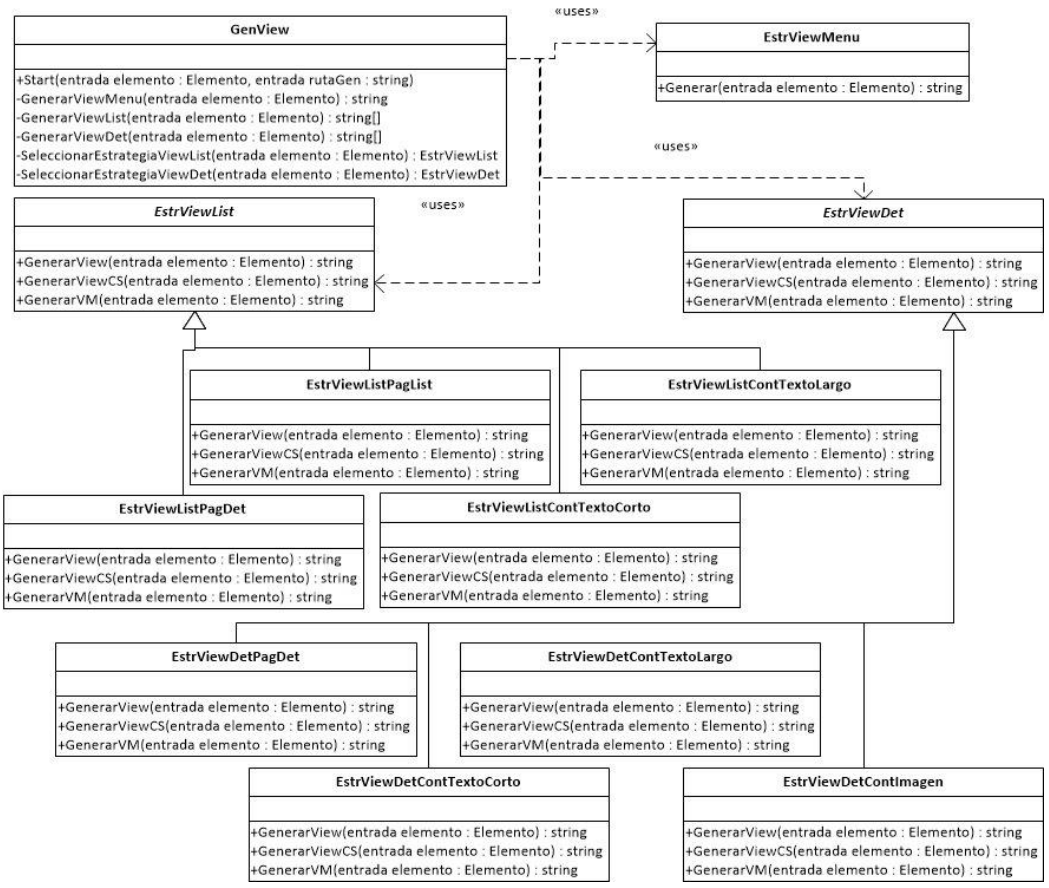


Ilustración 35: Diagrama de clases del módulo generador de vistas

Nombre:	GenView

Descripción:
Clase responsable de gestionar la generación de código automático de las vistas.

Métodos:

Nombre:	Start
Descripción: Método público responsable de iniciar el proceso de generación de código automático de las vistas.	
Parámetros: elemento: Elemento del que se desea realizar la transformación. rutaGen: Directorio dónde se guarda el código fuente generado.	

Nombre:	GenerarViewList
Descripción: Método responsable de realizar la generación de código de la vista de tipo listado, a partir de las estrategias de transformación obtenidas con el método SeleccionarEstrategiaViewList. Es un método recursivo que es llamado por cada uno de los hijos del elemento.	
Parámetros: elemento: Elemento del que se desea realizar la transformación.	
Retorno: Colección con los códigos fuente generados de las vistas de tipo listado (View, ViewCS y VM).	

Nombre:	GenerarViewDet
Descripción: Método responsable de realizar la generación de código de la vista de tipo detalle, a partir de las estrategias de transformación obtenidas con el método SeleccionarEstrategiaViewDet. Es un método recursivo que es llamado por cada uno de los hijos del elemento.	
Parámetros: elemento: Elemento del que se desea realizar la transformación.	
Retorno: Colección con los códigos fuente generados de las vistas de tipo detalle (View, ViewCS y VM).	

Nombre:	SeleccionarEstrategiaViewList
Descripción: Método responsable de determinar la estrategia de transformación a utilizar para las vistas de tipo listado.	
Parámetros: elemento: Elemento que determina la estrategia a utilizar.	
Retorno: Estrategia de transformación a utilizar.	

Nombre:	SeleccionarEstrategiaViewDet
Descripción: Método responsable de determinar la estrategia de transformación a utilizar para las vistas de tipo detalle.	
Parámetros: elemento: Elemento que determina la estrategia a utilizar.	
Retorno: Estrategia de transformación a utilizar.	

Nombre:	GenerarViewMenu
Descripción: Método responsable de realizar la generación de código a partir de la estrategia de transformación EstrViewMenu.	
Parámetros: elemento: Elemento del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	EstrViewMenu
Descripción: Clase responsable de implementar la estrategia de transformación del menú de la aplicación	

Métodos:

Nombre:	Generar
Descripción: Método responsable de implementar la estrategia de transformación del menú de la aplicación.	
Parámetros: elemento: Elemento app.	

Retorno:
Código fuente generado.

Nombre:	EstrViewList
Descripción:	
Clase abstracta que define un contrato común a las distintas estrategias de transformación de la vista de tipo listado.	

Nombre:	EstrViewListPagList
Descripción:	
Clase responsable de implementar la estrategia de transformación de la vista de tipo listado para un elemento página con comportamiento listado.	

Métodos:

Nombre:	GenerarView
Descripción:	
Método responsable de implementar la estrategia de transformación de la vista de tipo listado para un elemento página con comportamiento listado.	
Parámetros:	
elemento: Elemento página con comportamiento listado del que se desea realizar la transformación.	
Retorno:	
Código fuente generado.	

Nombre:	GenerarViewCS
Descripción:	
Método responsable de implementar la estrategia de transformación del code behind de la vista de tipo listado para un elemento página con comportamiento listado.	
Parámetros:	
elemento: Elemento página con comportamiento listado del que se desea realizar la transformación.	
Retorno:	
Código fuente generado.	

Nombre:	GenerarVM
Descripción:	
Método responsable de implementar la estrategia de transformación del modelo de la vista de la vista de tipo listado para un elemento página con comportamiento listado.	
Parámetros:	
elemento: Elemento página con comportamiento listado del que se desea realizar la transformación.	
Retorno:	
Código fuente generado.	

Nombre:	EstrViewListPagDet
Descripción:	
Clase responsable de implementar la estrategia de transformación de la vista de tipo listado para un elemento página con comportamiento detalle.	
Métodos:	

Nombre:	GenerarView
Descripción: Método responsable de implementar la estrategia de transformación de la vista de tipo listado para un elemento página con comportamiento detalle.	
Parámetros: elemento: Elemento página con comportamiento detalle del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarViewCS
Descripción: Método responsable de implementar la estrategia de transformación del code behind de la vista de tipo listado para un elemento página con comportamiento detalle.	
Parámetros: elemento: Elemento página con comportamiento detalle del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarVM
Descripción: Método responsable de implementar la estrategia de transformación del modelo de vista de la vista de tipo listado para un elemento página con comportamiento detalle.	
Parámetros: elemento: Elemento página con comportamiento detalle del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	EstrViewListContTextoCorto
Descripción: Clase responsable de implementar la estrategia de transformación de la vista de tipo listado para un elemento control texto corto.	

Métodos:

Nombre:	GenerarView
Descripción: Método responsable de implementar la estrategia de transformación de la vista de tipo listado para un elemento control texto corto.	
Parámetros: elemento: Elemento control texto corto del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarViewCS
Descripción: Método responsable de implementar la estrategia de transformación del code behind de la vista de tipo listado para un elemento control texto corto.	

Parámetros: elemento: Elemento control texto corto del que se desea realizar la transformación.
Retorno: Código fuente generado.

Nombre:	GenerarVM
Descripción: Método responsable de implementar la estrategia de transformación del modelo de vista de la vista de tipo listado para un elemento control texto corto.	
Parámetros: elemento: Elemento control texto corto del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	EstrViewListContTextoLargo
Descripción: Clase responsable de implementar la estrategia de transformación de la vista de tipo listado para un elemento control texto largo.	

Métodos:

Nombre:	GenerarView
Descripción: Método responsable de implementar la estrategia de transformación de la vista de tipo listado para un elemento control texto largo.	
Parámetros: elemento: Elemento control texto largo del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarViewCS
Descripción: Método responsable de implementar la estrategia de transformación del code behind de la vista de tipo listado para un elemento control texto largo.	
Parámetros: elemento: Elemento control texto largo del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarVM
Descripción: Método responsable de implementar la estrategia de transformación del modelo de vista de la vista de tipo listado para un elemento control texto largo.	
Parámetros: elemento: Elemento control texto largo del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	EstrViewDet
Descripción: Clase abstracta que define un contrato común a las distintas estrategias de transformación de	

la vista de tipo detalle.

Nombre:	EstrViewDetPagDet

Descripción:
Clase responsable de implementar la estrategia de transformación de la vista de tipo detalle para un elemento página con comportamiento detalle.

Métodos:

Nombre:	GenerarView
Descripción: Método responsable de implementar la estrategia de transformación de la vista de tipo detalle para un elemento página con comportamiento detalle.	
Parámetros: elemento: Elemento página con comportamiento detalle del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarViewCS
Descripción: Método responsable de implementar la estrategia de transformación del code behind de la vista de tipo detalle para un elemento página con comportamiento detalle.	
Parámetros: elemento: Elemento página con comportamiento detalle del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarVM
Descripción: Método responsable de implementar la estrategia de transformación del modelo de vista de la vista de tipo detalle para un elemento página con comportamiento detalle.	
Parámetros: elemento: Elemento página con comportamiento detalle del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	EstrViewDetContTextoCorto

Descripción:
Clase responsable de implementar la estrategia de transformación de la vista de tipo detalle para un elemento control texto corto.

Métodos:

Nombre:	GenerarView
Descripción: Método responsable de implementar la estrategia de transformación de la vista de tipo detalle para un elemento control texto corto.	
Parámetros: elemento: Elemento control texto corto del que se desea realizar la transformación.	
Retorno:	

Código fuente generado.

Nombre:	GenerarViewCS
Descripción: Método responsable de implementar la estrategia de transformación del code behind de la vista de tipo detalle para un elemento control texto corto.	
Parámetros: elemento: Elemento control texto corto del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarVM
Descripción: Método responsable de implementar la estrategia de transformación del modelo de vista de la vista de tipo detalle para un elemento control texto corto.	
Parámetros: elemento: Elemento control texto corto del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	EstrViewDetContTextoLargo
Descripción: Clase responsable de implementar la estrategia de transformación de la vista de tipo detalle para un elemento control texto largo.	

Métodos:

Nombre:	GenerarView
Descripción: Método responsable de implementar la estrategia de transformación de la vista de tipo detalle para un elemento control texto largo.	
Parámetros: elemento: Elemento control texto largo del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarViewCS
Descripción: Método responsable de implementar la estrategia de transformación del code behind de la vista de tipo detalle para un elemento control texto largo.	
Parámetros: elemento: Elemento control texto largo del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarVM
Descripción: Método responsable de implementar la estrategia de transformación del modelo de vista de la vista de tipo detalle para un elemento control texto largo.	
Parámetros: elemento: Elemento control texto largo del que se desea realizar la transformación.	

Retorno:
Código fuente generado.

Nombre:	EstrViewDetContImagen

Descripción:
Clase responsable de implementar la estrategia de transformación de la vista de tipo detalle para un elemento control imagen.

Métodos:

Nombre:	GenerarView
Descripción: Método responsable de implementar la estrategia de transformación de la vista de tipo detalle para un elemento control imagen.	
Parámetros: elemento: Elemento control imagen del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarViewCS
Descripción: Método responsable de implementar la estrategia de transformación del code behind de la vista de tipo detalle para un elemento control imagen.	
Parámetros: elemento: Elemento control imagen del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarVM
Descripción: Método responsable de implementar la estrategia de transformación del modelo de vista de la vista de tipo detalle para un elemento control imagen.	
Parámetros: elemento: Elemento control imagen del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

3. Módulo de generación de Web Api

Módulo responsable de generar el código fuente de los métodos que se comportan como puntos de entrada en la parte servidora.

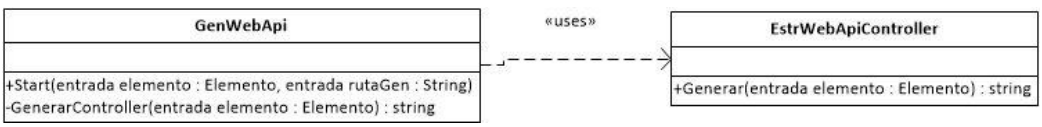


Ilustración 36: Diagrama de clases del módulo generador de Web Api

Nombre:	GenWebApi

Descripción:
Clase responsable de gestionar la generación de código automático de la Web Api.

Métodos:

Nombre:	Start
Descripción: Método público responsable de iniciar el proceso de generación de código automático de la Web Api. Llama al método GenerarController para cada uno de los elementos página con comportamiento detalle.	
Parámetros: elemento: Elemento del que se desea realizar la transformación. rutaGen: Directorio dónde se guarda el código fuente generado.	

Nombre:	GenerarController
Descripción: Método responsable de realizar la generación de código de los controladores de la Web Api a partir de la estrategia de transformación EstrWebApiController.	
Parámetros: elemento: Elemento del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	EstrWebApiController
Descripción: Clase responsable de implementar la estrategia de transformación de los controladores de la Web Api.	

Métodos:

Nombre:	Generar
Descripción: Método responsable de implementar la estrategia de transformación de los controladores de la Web Api.	
Parámetros: elemento: Elemento página con comportamiento detalle del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

4. Módulo de generación de acceso a datos

Módulo responsable de generar el código fuente para la comunicación con la base de datos, codifica los patrones de diseño Repositoty, UnitOfWork y el contexto del Entity Framework.

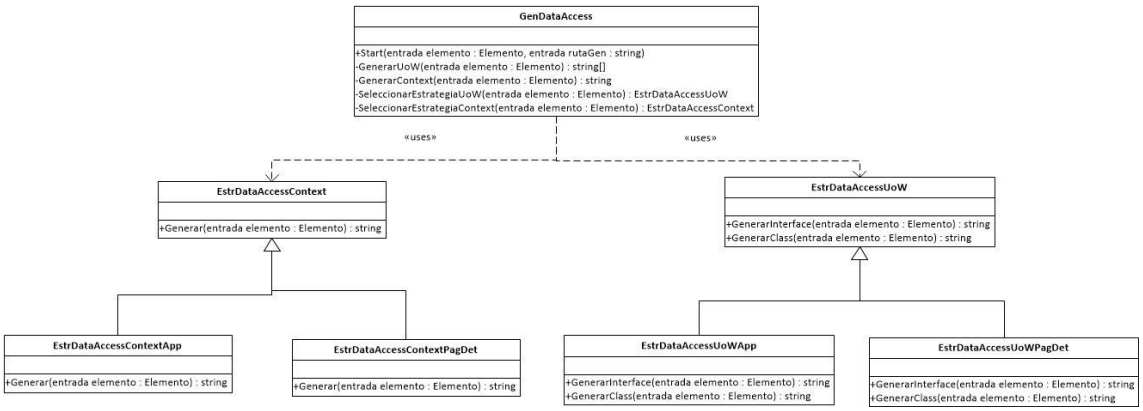


Ilustración 37: Diagrama de clases del módulo generador de acceso a datos

Nombre:	GenDataAccess

Descripción:
Clase responsable de gestionar la generación de código automático del acceso a datos.

Métodos:

Nombre:	Start

Descripción:
Método público responsable de iniciar el proceso de generación de código automático del acceso a datos.

Parámetros:
elemento: Elemento del que se desea realizar la transformación.
rutaGen: Directorio dónde se guarda el código fuente generado.

Nombre:	GenerarUoW

Descripción:
Método responsable de realizar la generación de código del UnitOfWork a partir de las estrategias de transformación obtenidas con el método SeleccionaEstrategiaUoW. Es un método recursivo que es llamado para cada uno de los hijos elemento.

Parámetros:
elemento: Elemento del que se desea realizar la transformación.

Retorno:
Colección con los códigos fuente generados de la interfaz y de la clase del Unit Of Work.

Nombre:	GenerarContext

Descripción:
Método responsable de realizar la generación de código del contexto a partir de las estrategias de transformación obtenidas con el método SeleccionaEstrategiaContext. Es un método recursivo que es llamado para cada uno de los hijos elemento.

Parámetros:
elemento: Elemento del que se desea realizar la transformación.

Retorno:
Código fuente generado del contexto.

Nombre:	SeleccionarEstrategiaUoW

Descripción:
Método responsable de determinar la estrategia de transformación a utilizar para el UnitOfWork.

Parámetros:

elemento: Elemento que determina la estrategia a utilizar.
Retorno: Estrategia de transformación a utilizar.

Nombre:	SeleccionarEstrategiaContext
Descripción: Método responsable de determinar la estrategia de transformación a utilizar para el contexto.	
Parámetros: elemento: Elemento que determina la estrategia a utilizar.	
Retorno: Estrategia de transformación a utilizar.	

Nombre:	EstrDataAccessUoW
Descripción: Clase abstracta que define un contrato común a las distintas estrategias de transformación del UnitOfWork.	

Nombre:	EstrDataAccesUoWApp
Descripción: Clase responsable de implementar la estrategia de transformación del Unit Of Work para un elemento app.	

Métodos:

Nombre:	GenerarInterface
Descripción: Método responsable de implementar la estrategia de transformación de la interfaz del UnitOfWork para un elemento app.	
Parámetros: elemento: Elemento app del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarClass
Descripción: Método responsable de implementar la estrategia de transformación de la clase del UnitOfWork para un elemento app.	
Parámetros: elemento: Elemento app del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	EstrDataAccesUoWPagDet
Descripción: Clase responsable de implementar la estrategia de transformación del UnitOfWork para un elemento página con comportamiento detalle.	

Métodos:

Nombre:	GenerarInterface

Descripción:
Método responsable de implementar la estrategia de transformación de la interfaz del UnitOfWork para un elemento página con comportamiento detalle.

Parámetros:
elemento: Elemento página con comportamiento detalle del que se desea realizar la transformación.

Retorno:
Código fuente generado.

Nombre:	GenerarClass

Descripción:
Método responsable de implementar la estrategia de transformación de la clase del Unit Of Work para un elemento página con comportamiento detalle.

Parámetros:
elemento: Elemento página con comportamiento detalle del que se desea realizar la transformación.

Retorno:
Código fuente generado.

Nombre:	EstrDataAccessContext

Descripción:
Clase abstracta que define un contrato común a las distintas estrategias de transformación del contexto.

Nombre:	EstrDataAccesContextApp

Descripción:
Clase responsable de implementar la estrategia de transformación del contexto para un elemento app.

Métodos:

Nombre:	Generar

Descripción:
Método responsable de implementar la estrategia de transformación del contexto para un elemento app.

Parámetros:
elemento: Elemento app del que se desea realizar la transformación.

Retorno:
Código fuente generado.

Nombre:	EstrDataAccesContextPagDet

Descripción:
Clase responsable de implementar la estrategia de transformación del contexto para un elemento página con comportamiento detalle.

Métodos:

Nombre:	Generar

Descripción:
Método responsable de implementar la estrategia de transformación del contexto para un elemento página con comportamiento detalle.

Parámetros:

elemento: Elemento página con comportamiento detalle del que se desea realizar la transformación.
Retorno: Código fuente generado.

5. *Módulo de generación de configuración*

Módulo responsable de generar el código fuente de los ficheros de configuración de la aplicación móvil y de la parte servidora. También genera el código del fichero CSPROJ de la aplicación móvil, este contiene la información de los ficheros y ensamblados que se usa en el proyecto Xamarin.

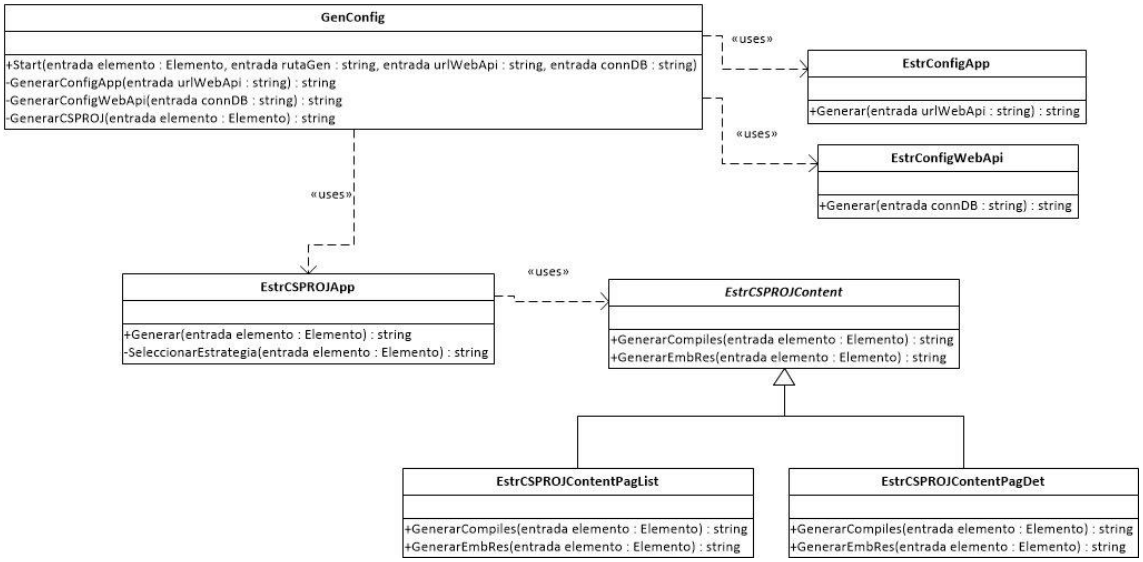


Ilustración 38: Diagrama de clases del módulo generador de configuración

Nombre:	GenConfig
Descripción: Clase responsable de gestionar la generación de código automático de los ficheros de configuración del proyecto.	

Métodos:

Nombre:	Start
Descripción: Método público responsable de iniciar el proceso de generación de los ficheros de configuración del proyecto.	
Parámetros: elemento: Elemento del que se desea realizar la transformación. rutaGen: Directorio dónde se guarda el código fuente generado. urlWebApi: Ruta HTTP de la Web Api. connDB: Cadena de conexión de la base de datos.	

Nombre:	GenerarConfigApp
Descripción: Método responsable de realizar la generación de código del fichero de configuración de la aplicación móvil a partir de la estrategia de transformación EstrConfigApp.	
Parámetros:	

urlWebApi: Ruta HTTP de la Web Api.
Retorno: Código fuente generado del fichero de configuración de la aplicación móvil.

Nombre:	GenerarConfigWebApi
Descripción: Método responsable de realizar la generación de código del fichero de configuración de la Web Api a partir de les estrategia EstrConfigWebApi.	
Parámetros: connDB: Cadena de conexión de la base de datos	
Retorno: Código fuente generado del fichero de configuración de la Web Api.	

Nombre:	GenerarCSPROJ
Descripción: Método responsable de realizar la generación de código del fichero CSPROJ a partir de la estrategia de transformación EstrCSPROJ.	
Parámetros: elemento: Elemento del que se desea realizar la transformación.	
Retorno: Código fuente generado del fichero CSPROJ.	

Nombre:	EstrConfigApp
Descripción: Clase responsable de implementar la estrategia de transformación del fichero de configuración de la aplicación móvil.	

Métodos:

Nombre:	Generar
Descripción: Método responsable de implementar la estrategia de transformación del fichero de configuración de la aplicación móvil.	
Parámetros: urlWebApi: Ruta HTTP de la Web Api.	
Retorno: Código fuente generado.	

Nombre:	EstrConfigWebApi
Descripción: Clase responsable de implementar la estrategia de transformación del fichero de configuración de la Web Api.	

Métodos:

Nombre:	Generar
Descripción: Método responsable de implementar la estrategia de transformación del fichero de configuración de la Web Api.	
Parámetros: connDB: Cadena de conexión de la base de datos	
Retorno:	

Código fuente generado.

Nombre:	EstrCSPROJApp

Descripción:
Clase responsable de implementar la estrategia de transformación del fichero de CSPROJ para un elemento app.

Métodos:

Nombre:	Generar

Descripción:
Método responsable de implementar la estrategia de transformación del fichero CSPROJ para un elemento app. Para todos sus elementos hijos página llama a la estrategia de transformación obtenida con el método SeleccionarEstrategia.

Parámetros:
elemento: Elemento del que se desea realizar la transformación.

Retorno:
Código fuente generado.

Nombre:	SeleccionarEstrategia

Descripción:
Método responsable de determinar la estrategia de transformación a utilizar para el contenido del fichero CSPROJ.

Parámetros:
elemento: Elemento que determina la estrategia a utilizar.

Retorno:
Estrategia de transformación a utilizar.

Nombre:	EstrCSPROJContent

Descripción:
Clase abstracta que define un contrato común a las distintas estrategias de transformación del contenido del fichero CSPROJ.

Nombre:	EstrCSPROJContentPagList

Descripción:
Clase responsable de implementar la estrategia de transformación del contenido del fichero de CSPROJ que depende del elemento página con comportamiento listado.

Métodos:

Nombre:	GenerarCompiles

Descripción:
Método responsable de implementar la estrategia de transformación del contenido compila del fichero CSPROJ para un elemento página con comportamiento listado.

Parámetros:
elemento: Elemento del que se desea realizar la transformación.

Retorno:
Código fuente generado.

Nombre:	GenerarEmbRes

Descripción:
Método responsable de implementar la estrategia de transformación del contenido de

recursos embebidos del fichero CSPROJ para un elemento página con comportamiento listado.
Parámetros: elemento: Elemento del que se desea realizar la transformación.
Retorno: Código fuente generado.

Nombre:	EstrCSPROJContentPagDet

Descripción:
Clase responsable de implementar la estrategia de transformación del contenido del fichero de CSPROJ que depende del elemento página con comportamiento detalle.

Métodos:

Nombre:	GenerarCompiles
Descripción: Método responsable de implementar la estrategia de transformación del contenido compila del fichero CSPROJ para un elemento página con comportamiento detalle.	
Parámetros: elemento: Elemento del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

Nombre:	GenerarEmbRes
Descripción: Método responsable de implementar la estrategia de transformación del contenido de recursos embebidos del fichero CSPROJ para un elemento página con comportamiento detalle.	
Parámetros: elemento: Elemento del que se desea realizar la transformación.	
Retorno: Código fuente generado.	

6. *Módulo de generación del fichero apk*

Módulo responsable de generar el fichero apk a partir del código fuente de la aplicación móvil, para su generación se usa un script .bat.

Ilustración 39: Diagrama de clases del módulo generador de la apk

Nombre:	GenApk

Descripción:
Clase responsable de gestionar la generación del fichero apk.

Métodos:

Nombre:	Start
Descripción: Método público responsable de llamar al script .bat que genera la apk a partir del	

proyecto base del generador automático de código y el código fuente generado.
Parámetros: rutaGen: Directorio dónde se encuentra el código fuente generado y el proyecto base. rutaOutput: Directorio dónde se guarda el fichero apk generado. rutaAndroidSDK: Directorio del sistema dónde está instalado el SDK de Android.

7. Módulo de publicación de la parte servidora

Módulo responsable de publicar en el servidor Web los ficheros necesarios del código fuente de la parte servidora. Para la publicación se usa un script .bat.

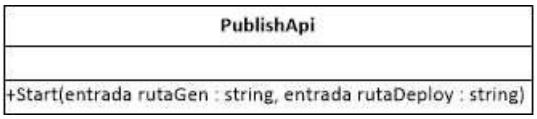


Ilustración 40: Diagrama de clases del módulo de publicación de la parte servidora

Nombre:	PublishApi
Descripción: Clase responsable de gestionar la publicación de la parte servidora.	
Métodos:	
Nombre:	Start
Descripción: Método público responsable de llamar al script .bat que publica la parte servidora a partir del proyecto base del generador automático de código y el código fuente generado.	
Parámetros: rutaGen: Directorio dónde se encuentra el código fuente generado y el proyecto base. rutaDeploy: Directorio dónde se publica la parte servidora.	

7. Resultados de la solución

Después de la implementación de la solución es el momento de mostrar los resultados obtenidos mediante un par de ejemplos. El primer ejemplo consiste en la generación de una aplicación móvil que permite realizar fotos con el dispositivo y catalogarlas.

A partir de la interfaz de usuario del gestor de contenidos se ha definido como se deseaba que fuese la aplicación móvil. En la pantalla de inicio se puede observar que hay un formulario para dar un nombre a la aplicación móvil y el listado de las páginas, pudiendo añadir de nuevas, editarlas o eliminarlas. También contiene el botón que permite iniciar su generación.

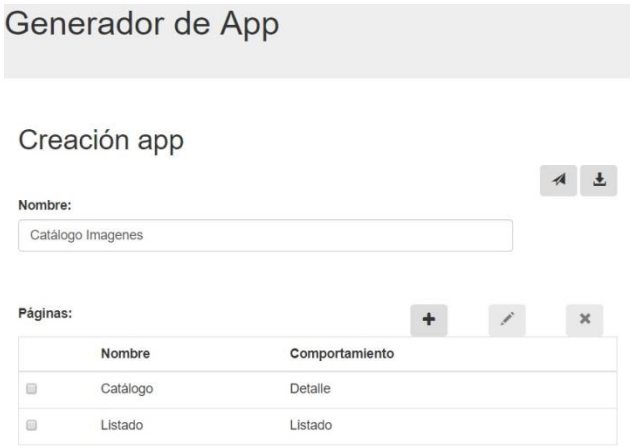


Ilustración 41: Página de inicio del gestor de contenidos para la generación del primer ejemplo

Para esta aplicación se han definido dos páginas, una que lista los registros y la otra que muestra el detalle del registro. Las páginas de edición de las páginas de la aplicación móvil contienen un formulario que permite definir el nombre, el comportamiento, el tipo de página, la página de detalle y la plantilla. Estos campos del formulario se muestran en función del comportamiento seleccionado. Además, en el caso de seleccionar el comportamiento de página detalle, se muestran los controles que contiene, permitiendo su inserción, edición o eliminación.

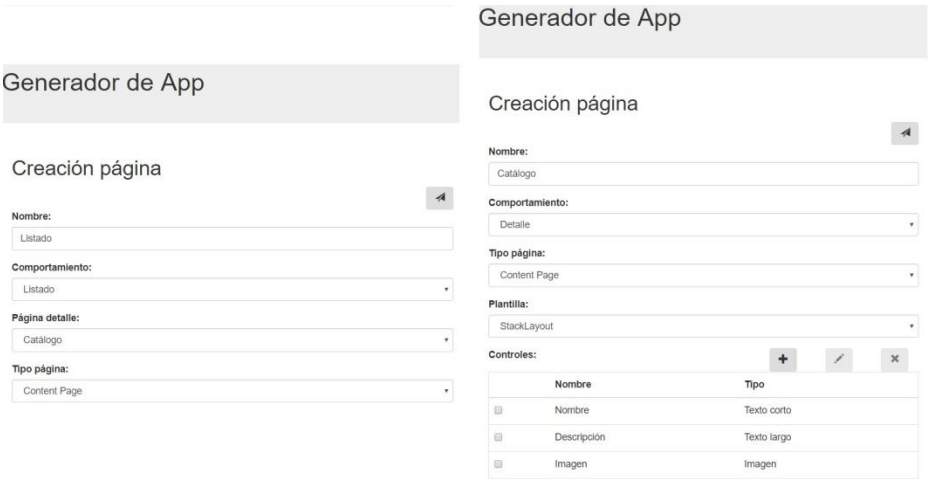


Ilustración 42: Páginas del gestor de contenidos para la edición de las páginas del primer ejemplo

Se puede ver que la página con comportamiento detalle tiene tres controles, uno va a permitir al usuario introducir el nombre, el otra la descripción y el último capturar la imagen. Las páginas de edición de controles contienen un formulario para definir el nombre y el tipo de control.

Ilustración 43: Páginas del gestor de contenidos para la edición de los controles del primer ejemplo

Después de la finalización de la definición de la aplicación móvil se generó la aplicación a partir del control habilitado para ello en la página de inicio del gestor de contenidos. La aplicación resultante está formada por un menú que permite al usuario navegar por las distintas secciones de la aplicación, para este caso solamente hay una entrada. Este elemento dirige al usuario a una página que muestra un listado de todas las fotos realizadas, y a partir de cada uno de los distintos registros, se puede consultar o editar su detalle mediante una nueva página.

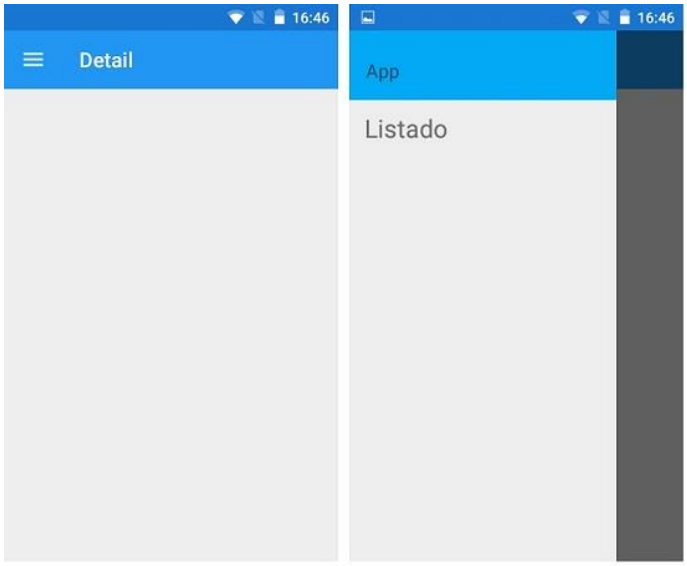


Ilustración 44: Página de inicio y menú de navegación del primer ejemplo

Desde la página que lista los distintos registros, es posible realizar la inserción, edición y eliminación de los registros.

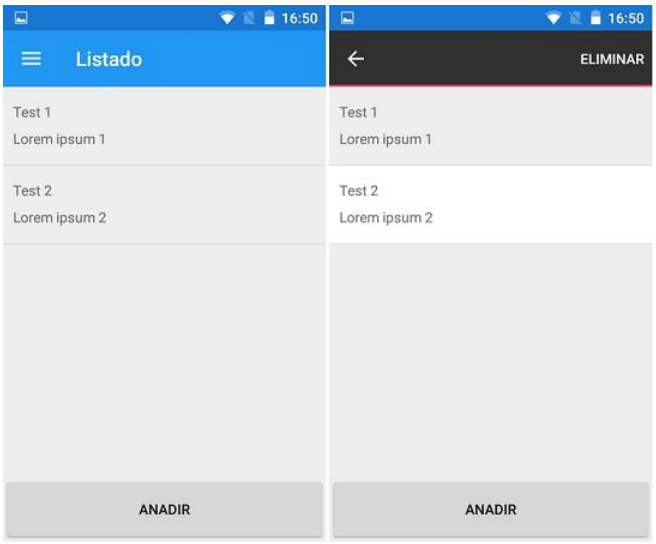


Ilustración 45: Página que lista los registros del primer ejemplo

La página que de detalle de los registros de la aplicación, permite al usuario editar el nombre, la descripción y realizar la fotografía.

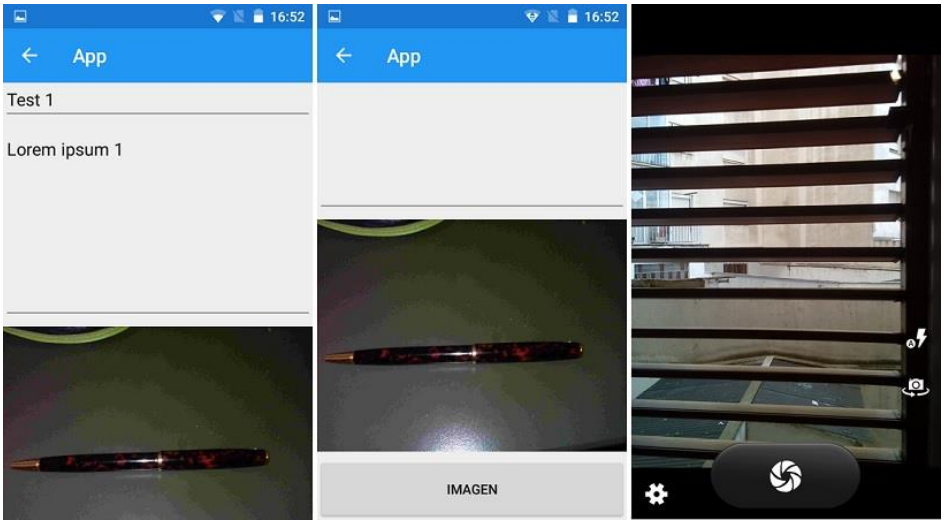


Ilustración 46: Página de detalle de los registros del primer ejemplo

El segundo ejemplo consiste en la generación de una aplicación móvil pensada para los comerciantes, para que pueda llevar un control de sus proveedores, de sus clientes y de las compras y ventas realizadas. Mediante el gestor de contenidos se ha definido una aplicación móvil con ocho páginas, cuatro con comportamiento detalle y cuatro más con comportamiento listado.

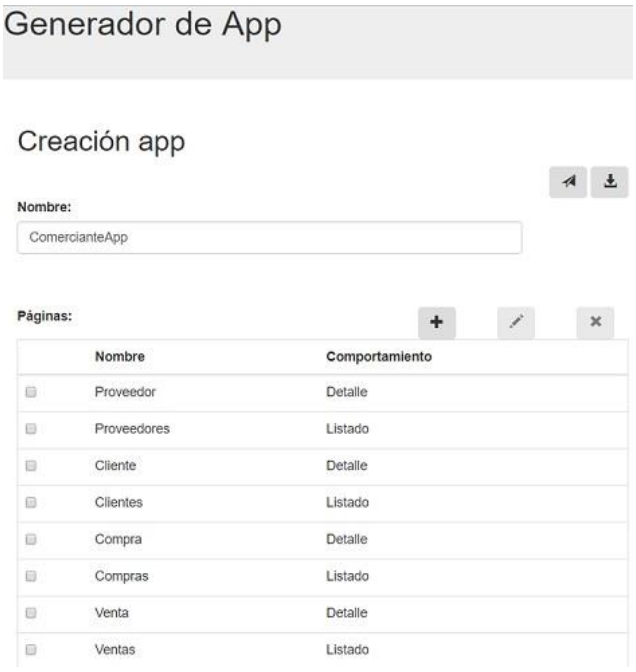


Ilustración 47: Página de inicio del gestor de contenidos para la generación del segundo ejemplo

Cada una de las cuatro páginas con comportamiento listado crea un elemento del menú de navegación de la aplicación resultado. Las páginas con comportamiento listado creadas son: proveedores; clientes; compras y ventas.

Ilustración 48: Páginas del gestor de contenidos para la edición de las páginas listado del segundo ejemplo

Las cuatro páginas con comportamiento detalle permiten consultar y editar la información de los proveedores, de los clientes, de las compras y de las ventas. Para su edición se han definido distintos controles que permitan informar del nombre, de la dirección, del email y del teléfono de los proveedores y de los clientes. También se han creado controles para informar del producto comprado, del producto vendido, del precio de compra y del precio de venta.

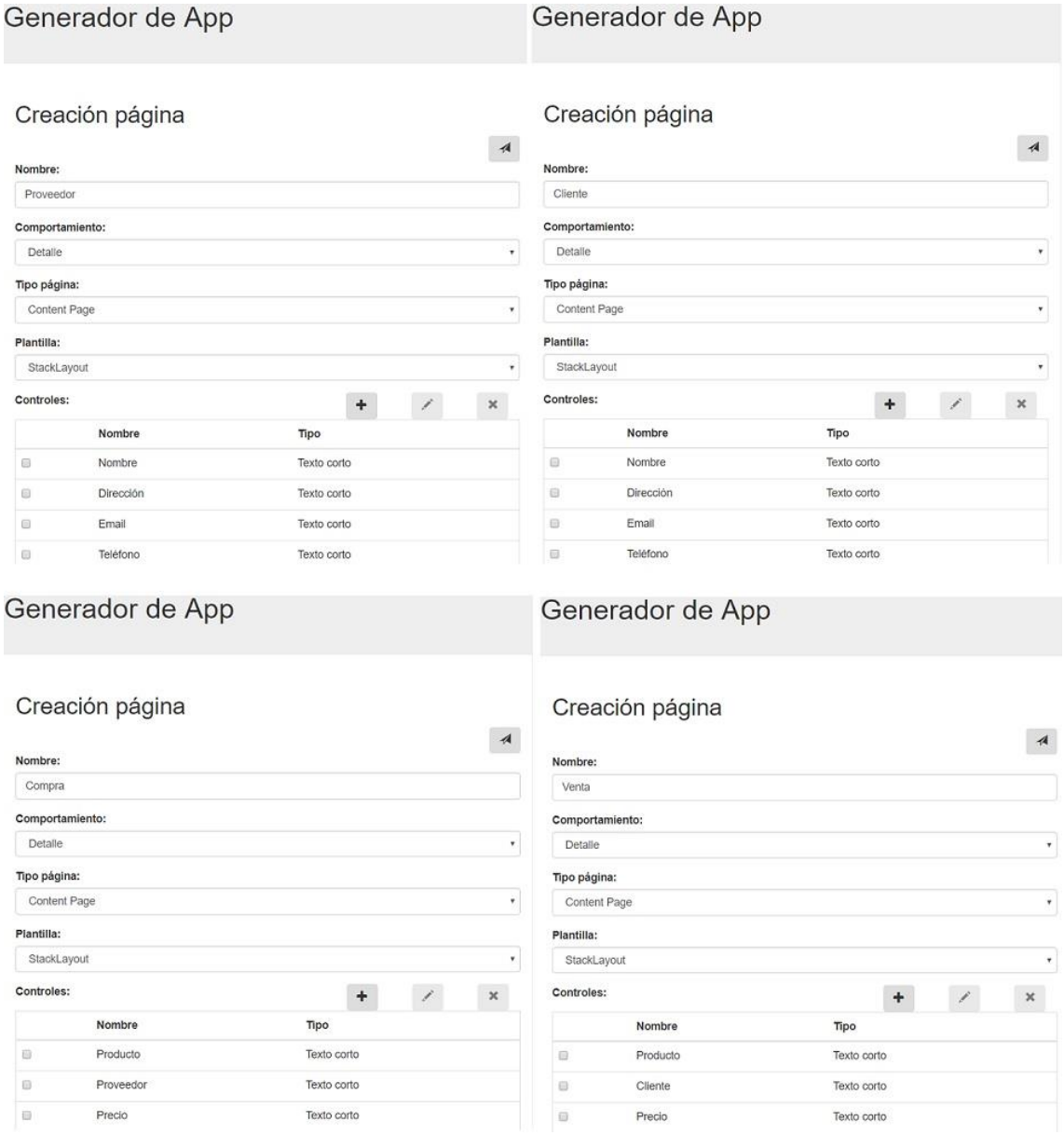


Ilustración 49: Páginas del gestor de contenidos para la edición de las páginas detalle del segundo ejemplo

El resultado de la generación de la aplicación móvil definida en el gestor de contenidos, da lugar a una aplicación móvil con un menú de navegación de cuatro elementos (proveedores, clientes, compras y ventas).

Ilustración 50: Menú de navegación del segundo ejemplo

Cada uno de los elementos del menú de navegación dirige al usuario a su página correspondiente de listado de los distintos registros. Como en el ejemplo anterior, desde las páginas de listado se puede insertar, editar y eliminar registros.

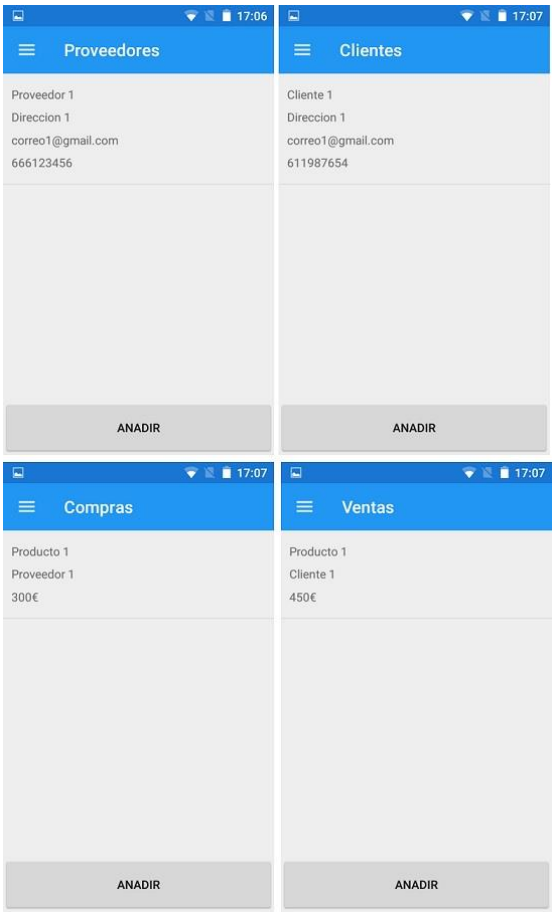


Ilustración 51: Páginas que listan los registros del segundo ejemplo

Desde cada uno de los registros de las páginas de listado se pude ir a la página de detalle del registro, permitiendo consultar o editar la información.

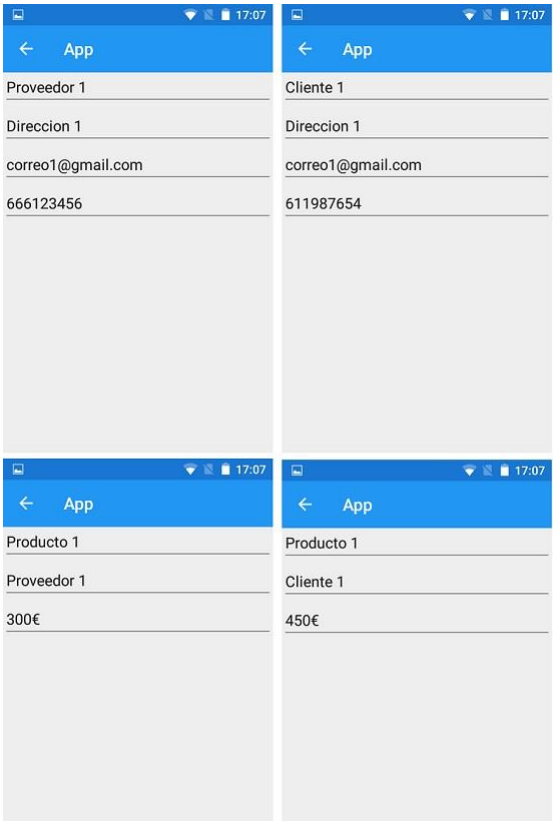


Ilustración 52: Páginas que detallan los registros del segundo ejemplo

8. Conclusiones y trabajo futuro

a. Conclusiones

Tal y como se ha demostrado en el apartado Resultados de la solución un usuario sin conocimientos de programación puede crear sus propias aplicaciones móviles de una forma rápida y sencilla, ya que solamente es necesario que interactúe con la interfaz visual del gestor de contenidos para diseñar las distintas páginas, siendo capaz de crear toda la aplicación móvil a partir de ellas. Las empresas con pocos recursos son los que puedan sacar un mayor beneficio de ello, ya que les permite desarrollar sus propias aplicaciones móviles a unos costes muy inferiores en comparación con la necesidad de disponer de un equipo especializado en el desarrollo de aplicaciones móviles.

El gestor de contenidos de las aplicaciones móviles solamente permite crear aplicaciones móviles basadas en CRUD y con acceso a las funciones nativas de los dispositivos. En el caso de desear una aplicación móvil con la implementación de reglas de negocio o con cualquier tipo de personalización que no se encuentren dentro del ámbito de alcance del generador, es necesario decantarse por otra opción.

Otro de los aspectos a destacar gestor de contenidos, es como se ha realizado su diseño, ya que se ha pensado teniendo en cuenta la necesidad de crear nuevas plantillas y controles. En el caso de querer ampliar el catálogo de controles y plantillas, solamente sería necesario añadirlos en la colección de maestros e implementar las nuevas transformaciones en el modelo generativo.

Finalmente, ha de destacarse el acierto en la elección de las tecnologías para el desarrollo del gestor de contenidos de las aplicaciones móviles. Xamarin ha facilitado las transformaciones en la generación automática de código, ya que al compartir código entre distintas plataformas, no ha sido necesario crear varias transformaciones para lo mismo, esto facilita la ampliación del gestor de contenidos para que también tenga en cuenta la generación para otras plataformas. El uso de Ruby con las plantillas ERb ha facilitado el desarrollo del modelo generativo, ya que permite tener centralizada toda la parte de generación del código resultado de una forma simple, eficaz, fácil de mantener y reusable. Entity Framework con enfoque Code First ha facilitado la creación del esquema de base de datos en el momento de la generación de la aplicación móvil, ya que las herramientas que ofrece han permitido la abstracción de esta tarea.

b. Trabajo futuro

El gestor de contenidos de las aplicaciones móviles no es una solución finalizada, ya que es posible realizar ampliaciones, estas serán implementadas como trabajo futuro.

- Automatización del proceso de obtención e instalación del fichero APK, ya que actualmente se realiza de forma manual. Cuando el usuario desea generar la aplicación móvil, ha de esperar a la finalización del proceso de generación y descargarla en su

equipo informático. Una vez descargada ha de copiar el fichero APK en el dispositivo donde se desea realizar la instalación. Si el gestor publicase la aplicación móvil en el Play Store, sería más cómodo para el usuario, además se podría realizar la generación de la aplicación móvil en segundo plano, sin la necesidad de hacer esperar al usuario a que finalice todo el proceso de generación.

- Creación de nuevas plantilla y controles.
- Implementación de un sistema de seguridad basada en la autenticación y la autorización.
- Implementación de un sistema de validación de los datos introducidos en los formularios.
- Ampliar el gestor de contenidos de las aplicaciones móviles para que también tenga en cuenta otras plataformas, como UWP y IOS.
- Modificación del gestor de contenidos de las aplicaciones móviles para que al finalizar la generación, publique la aplicación en las tiendas de las distintas plataformas.

9. Apéndices

a. Preparación del entorno

Para la preparación del entorno es necesario instalar un conjunto de herramientas, configurar la solución desarrollada correctamente y publicarla.

i. Instalación de las herramientas

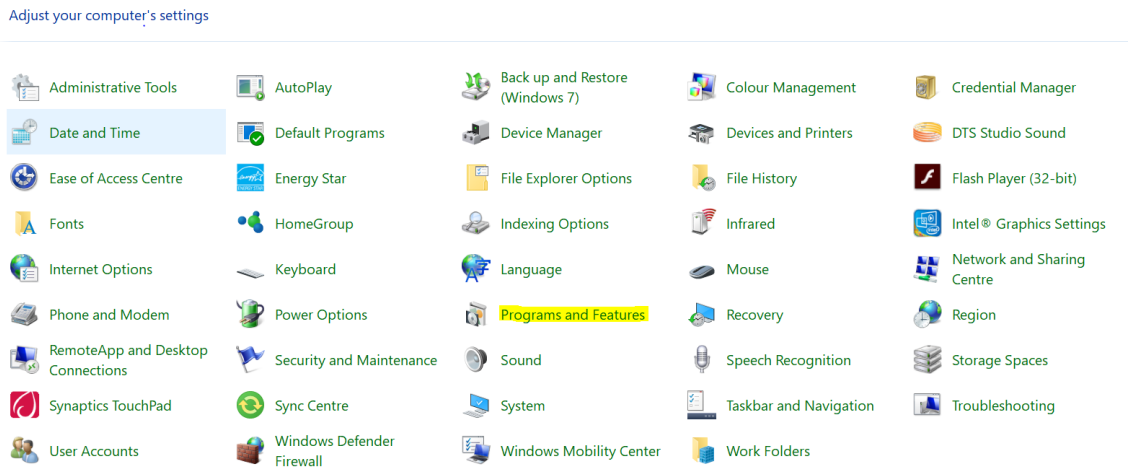
Las herramientas utilizadas en el entorno son las siguientes:

- Internet Information Service con el conjunto de hospedaje de .NET Core
- Microsoft SQL Server 2017
- Microsoft Visual Studio Community 2017 y Xamarin
- Ruby 2.4.4 o superior

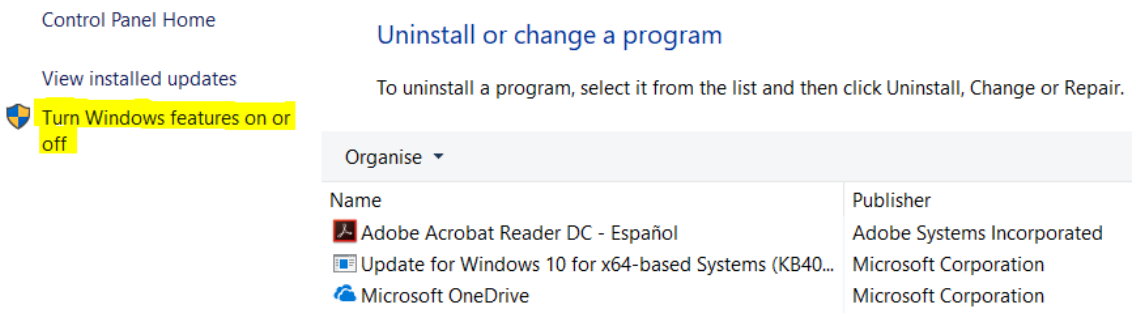
1. Instalación de Internet Information Service

Viene como una característica de Windows, solamente es necesario activarla. Su activación se realiza desde la ventana de Programas y Características de Windows.

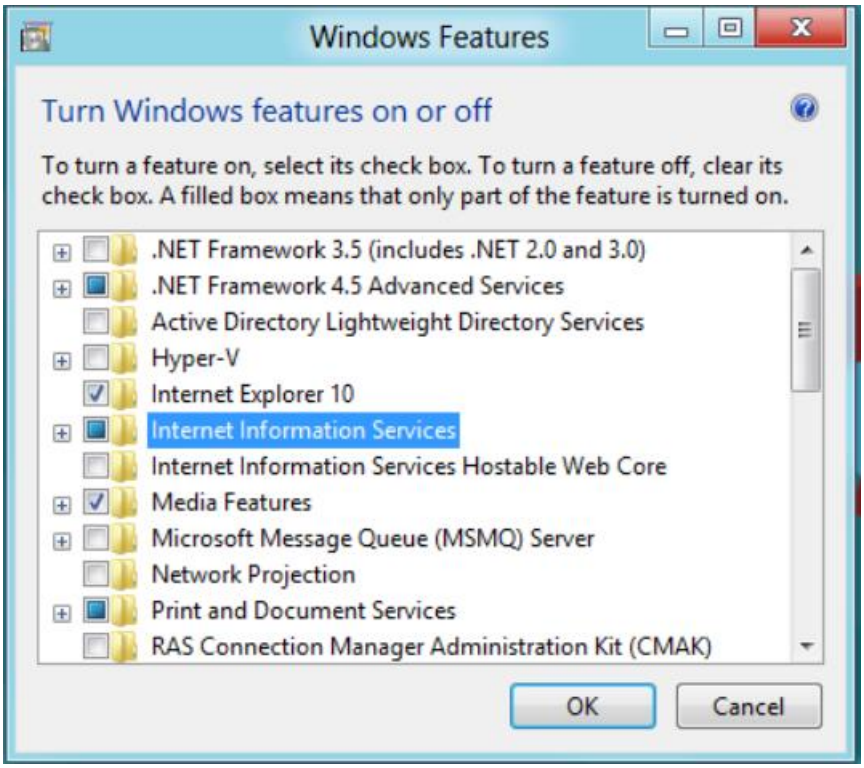
Se puede acceder a la ventana de Programas y Características de Windows desde el Panel de Control.



Una vez dentro de la ventana de Programas y Características de Windows se ha de seleccionar la opción de Habilitar/Deshabilitar características de Windows del menú lateral.



En la ventana de Características de Windows se ha de seleccionar la característica Internet Information Service y pulsar el botón OK.



A partir de los pasos descritos ya se encuentra activada la característica Internet Information Service de Windows, pero aún falta instalar el conjunto de hospedaje de .NET Core. Para ello se ha de descargar el .NET Core Runtime de la página de descargas de .NET y ejecutar el instalador descargado. Una vez finalizada la instalación solamente es necesario reiniciar el Internet Information Service.

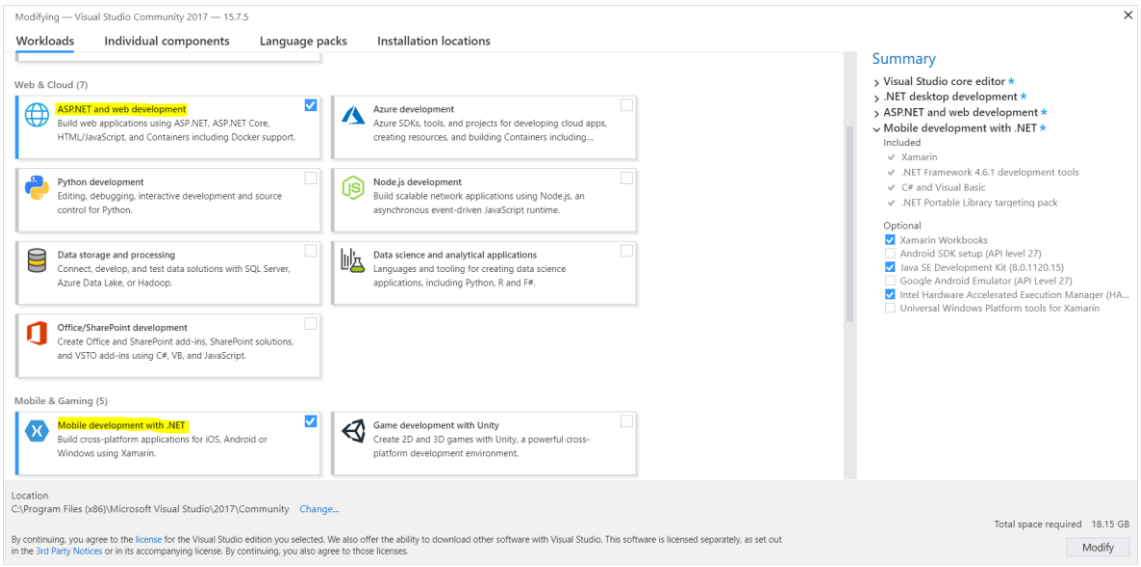
2. *Instalación de Microsoft SQL Server 2017*

Su instalación se realiza a partir del instalador Developer Edition descargado de la página de descargas de Sql Server. También es recomendable instalar SQL Server Management Studio para facilitar la administración de la base de datos.

3. *Instalación de Microsoft Visual Studio 2017 Community Edition y Xamarin*

Como en el caso anterior, su instalación también se realiza a partir de un instalador, este se obtiene de la página de descargas de Visual Studio. Al ejecutar el instalador obtenido se ha de seleccionar los siguientes componentes:

- ASP.NET and web development
- Mobile development with .NET



Una vez finalizada la instalación se ha de añadir a la variable de entorno PATH del sistema el valor: *C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\MSBuild\15.0\Bin*. El valor puede variar en función del directorio de instalación de Visual Studio.

4. Instalación de Ruby 2.4.4 o superior

Su instalación se realiza a partir del instalador de Windows obtenido de la página RubyInstaller. Una vez finaliza la instalación se ha de añadir a la variable de entorno PATH del sistema el siguiente valor: *C:\Ruby24-x64\bin*. El valor puede variar en función del directorio de instalación de Ruby.

Para finalizar la instalación de Ruby se ha de instalar la gema Json mediante el comando *gem install json* introducido en la consola de Windows.

ii. Publicación de la solución

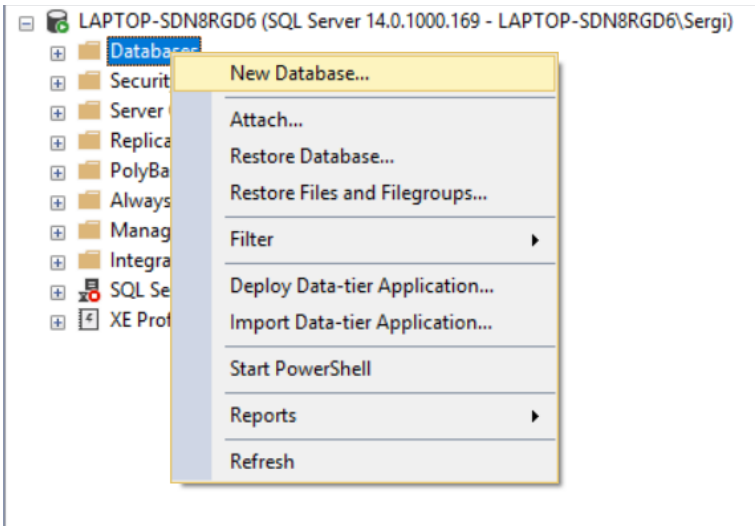
Una vez instaladas las herramientas necesarias, es el momento de realizar las publicaciones de la parte Back y Front del gestor de contenidos. También es necesario preparar el entorno de la parte Back de la aplicación móvil, solamente es necesaria su preparación, ya que la publicación la realiza automáticamente el modelo generativo.

1. Publicación de la parte Back del editor de contenidos

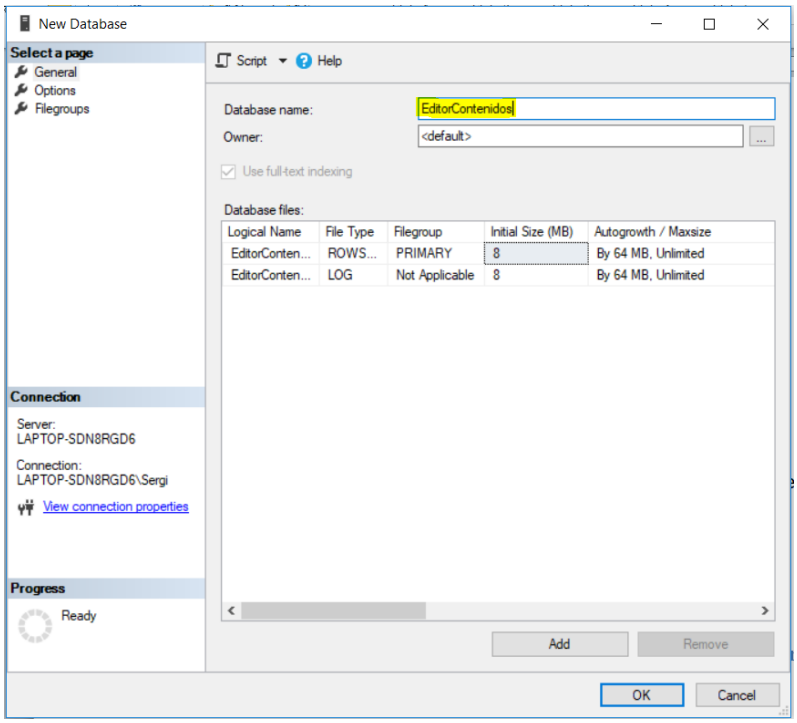
Para su publicación se ha de crear su base de datos y publicar los ficheros binarios en un sitio Web. No es necesario crear el esquema de base de datos, ya que su creación se realiza automáticamente la primera vez que se ejecuta el editor de contenidos.

a. Creación de la base de datos

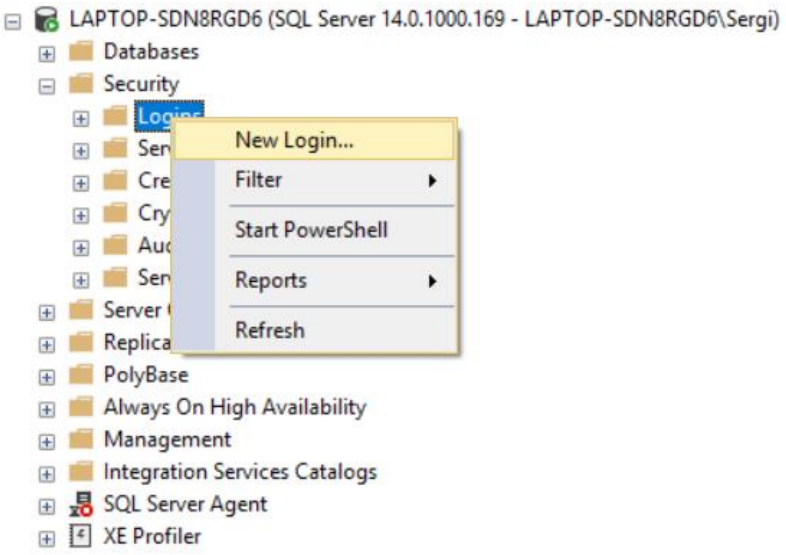
Para ello es necesario conectarse a la instancia creada en la instalación de Microsoft SQL Server, la conexión se puede realizar con la herramienta SQL Server Management Studio. La opción de Nueva base de datos del menú lateral permite crear una nueva base de datos.



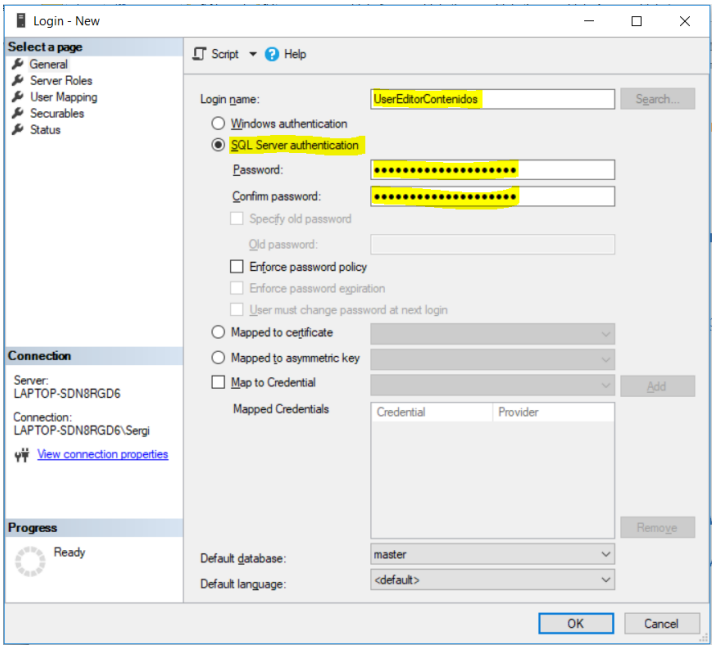
En la siguiente ventana solamente es necesario asignarle el nombre de la base de datos que se desea crear y pulsar el botón OK.



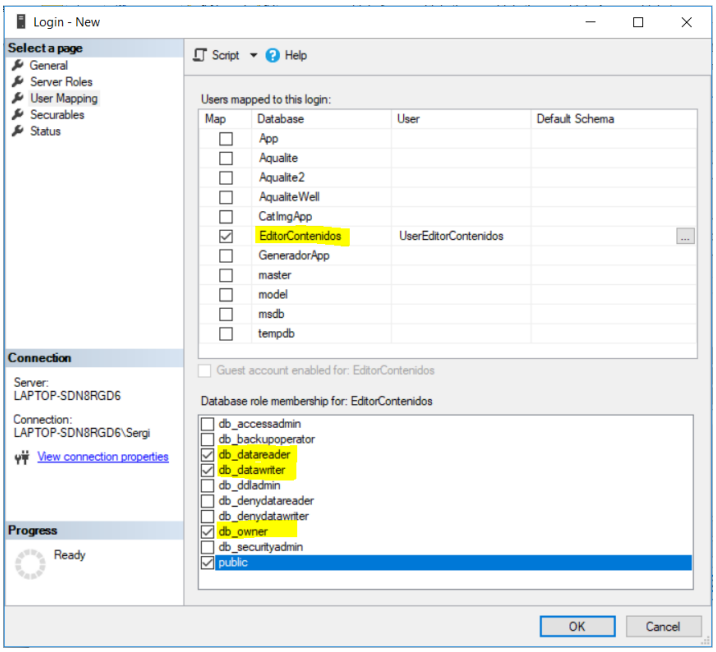
Ahora solamente falta añadir un usuario a la nueva base de datos, para ello se ha seleccionar la opción Nuevo Login del menú lateral.



En la nueva ventana abierta se ha de seleccionar la opción Autenticación SQL Server e introducir un nombre de usuario y una contraseña.



A continuación, en la pestaña Mapeo de usuario se ha de relacionar el nuevo usuario con la base de datos creada. Para realizar la relación se ha de seleccionar la base de datos creada y asignarle los permisos db_datareader, db_datawriter y db_owner. Finalmente se ha de pulsar el botón OK.

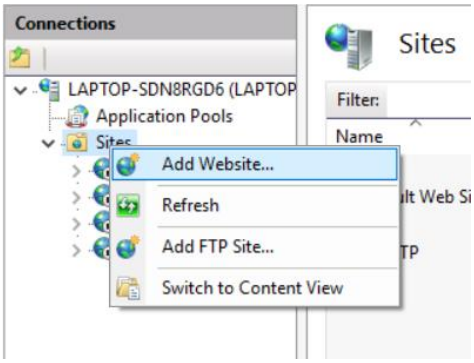


b. Creación y publicación del sitio Web

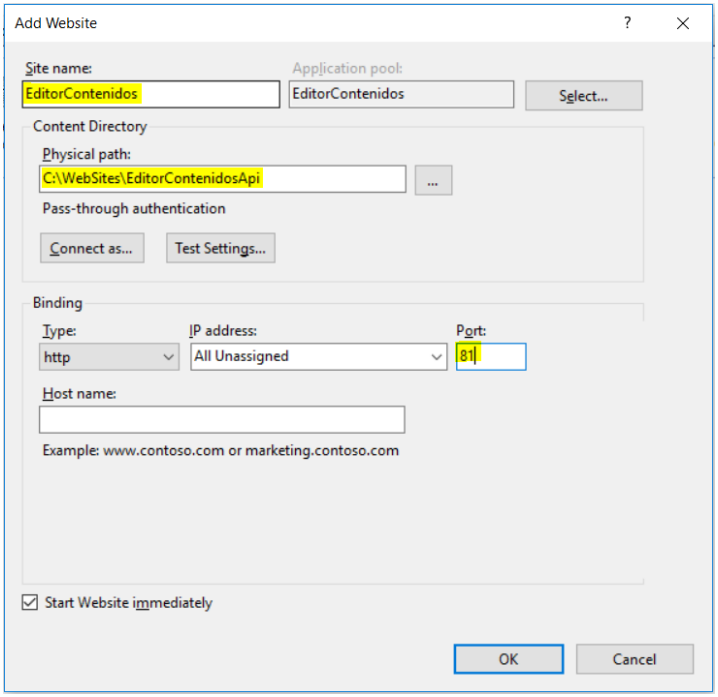
Este paso está dividido en tres pasos: creación del directorio de hospedaje; creación del sitio Web y publicar los ficheros binarios en el directorio de hospedaje.

El directorio creado es *C:\WebSites\EditorContenidosApi*, que alojará los ficheros binarios de la parte Back del editor de contenidos.

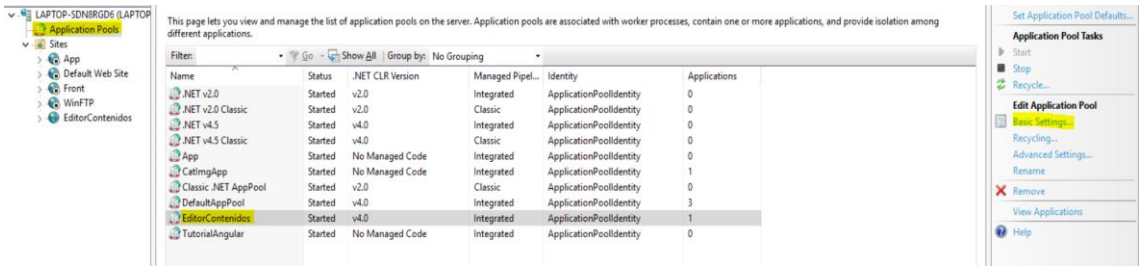
Una vez creado el directorio es el momento de crear el sitio Web, este paso se realiza a partir del Administrador del Internet Information Service. La opción Añadir sitio Web del menú lateral permite crear un nuevo sitio Web.



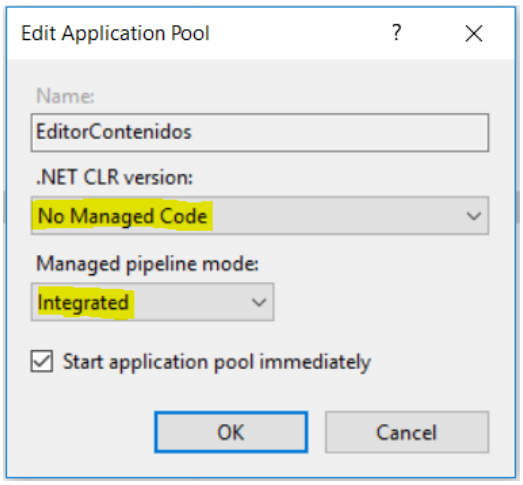
En la nueva ventana abierta se ha de asignar un nombre del nuevo sitio Web, el directorio creado en el paso anterior y un puerto. Para su confirmación se ha de pulsar el botón OK.



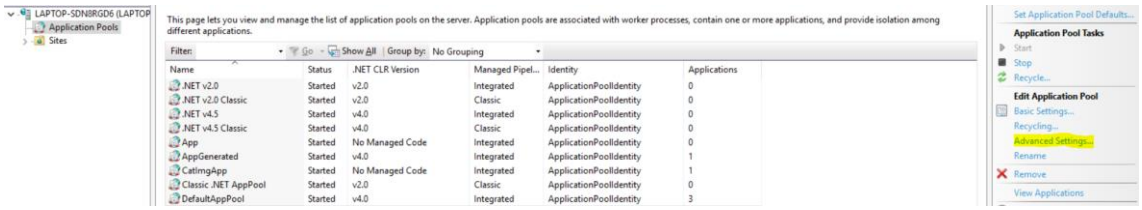
A continuación se ha de configurar el Application pool EditorContenidos creado, mediante la ventana Basic Settings de la opción Application Pools del menú lateral.



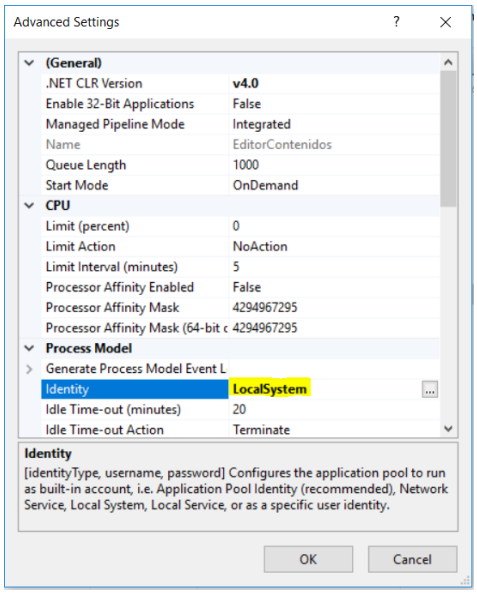
En la ventana abierta se ha de configurar el Application Pool como Código no administrado e Integrado.



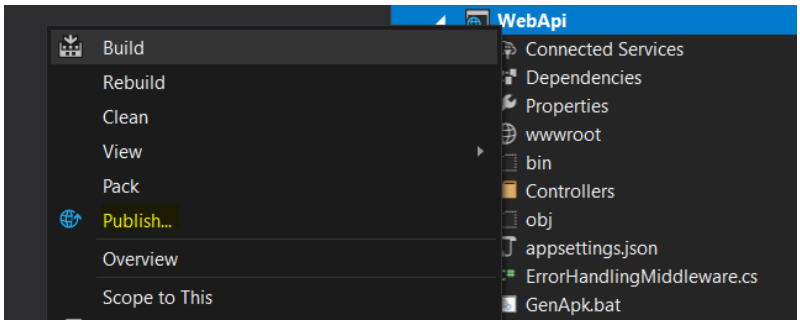
También es necesario configurar la identidad del Aplication Pool como sistema local, mediante la ventana Advanced Settings de la opción Aplication Pools del menú lateral.



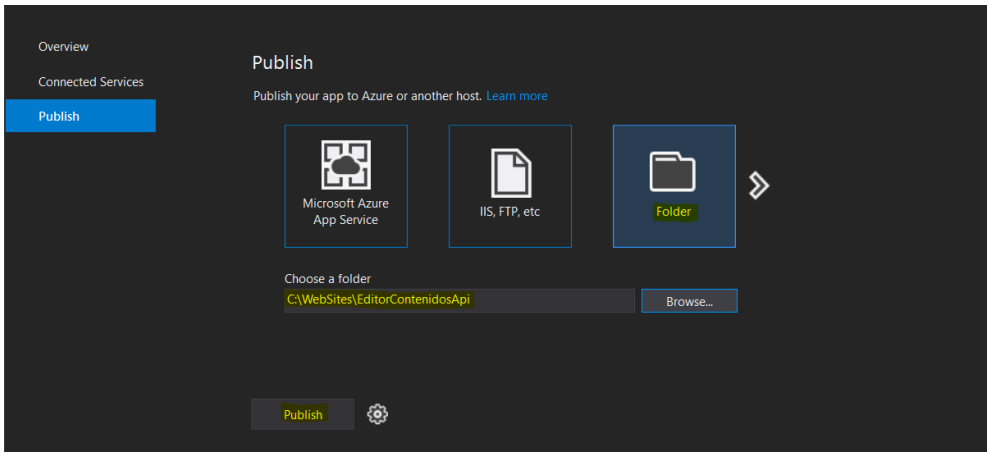
En la ventana abierta se ha de configurar el parámetro identity como LocalSystem.



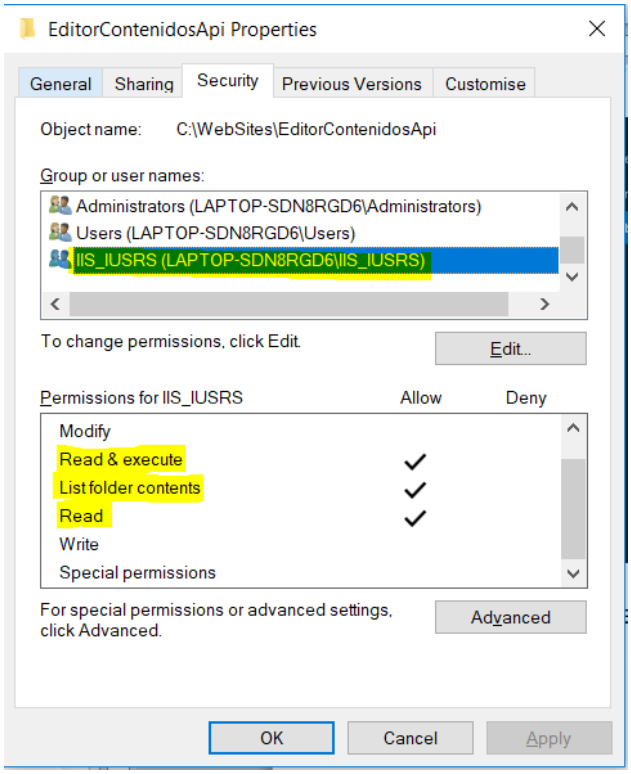
La publicación de los ficheros binarios se realiza con la herramienta Microsoft Visual Studio 2017, para ello es necesario abrir la solución GenApp. Una vez abierta la solución se ha de publicar el proyecto WebApi de la capa de servicio.



En la nueva ventana se ha de seleccionar la opción Fichero y el directorio creado. Finalmente, para completar la publicación se ha de pulsar el botón Publicar.



Finalmente, solo es necesario asignarle los permisos de lectura, ejecución y listado del contenido al usuario IIS_IUSRS para el directorio creado.



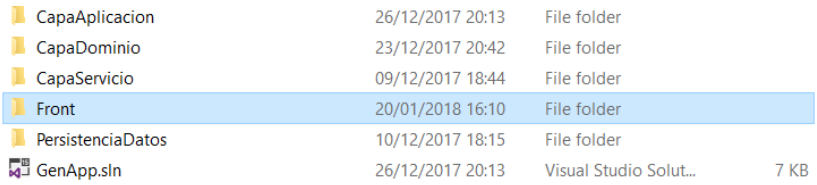
2. *Publicación de la parte Front del editor de contenidos*

La publicación de la parte Front del editor de contenidos es más simple que la de la parte Back, ya que solamente es necesario publicar los ficheros HTML, JavaScript y CSS.

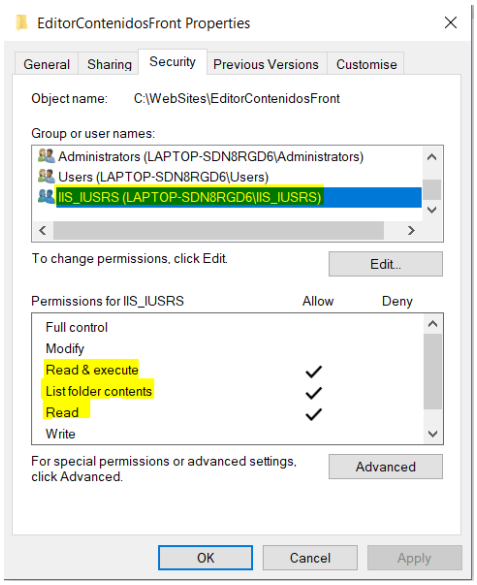
a. Creación y publicación del sitio Web

Como en el caso de la creación y publicación del sitio Web de la parte Back del editor de contenidos, se ha de crear un directorio de hospedaje, el sitio Web y copiar los ficheros necesarios en él.

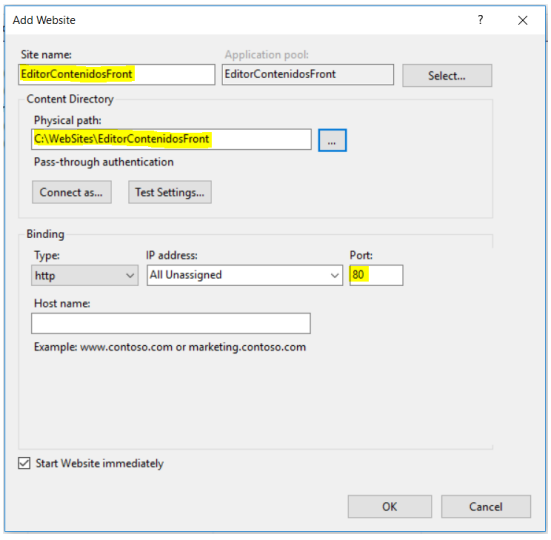
El directorio creado es *C:\WebSites\EditorContenidosFront* y los ficheros a copiar en él son los de la carpeta Front de la solución GenApp.



También es necesario asignarle los permisos de lectura, ejecución y listado del contenido al usuario IIS_IUSRS para el directorio creado.



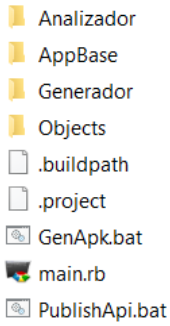
La creación del sitio Web se realiza prácticamente igual que en la creación y publicación del sitio Web de la parte Back del editor de contenidos, la diferencia se encuentra en el nombre del sitio Web, el directorio asignado, el puerto y que no es necesario configurar la identidad del Application Pool, ya que para este sitio la configuración por defecto (ApplicationPoolIdentity) ya nos sirve.



3. Creación de los directorios del modelo generativo y de trabajo

Es importante ubicar el código del modelo generativo y de la solución AppBase en un directorio, ya que la parte Back ha de ser conocedor de este directorio para que funcione correctamente el sistema de gestión de contenidos para aplicaciones móviles. También es importante crear un directorio de trabajo para que el editor de contenidos y el modelo generativo trabajen sobre el mismo directorio.

El directorio del modelo generativo creado es *C:\ModeloGenerativo* y contiene los ficheros de código del modelo generativo y los de la solución AppBase.



El directorio de trabajo creado es C:\Workspace, su contenido se crea automáticamente cuando se genera una aplicación móvil.

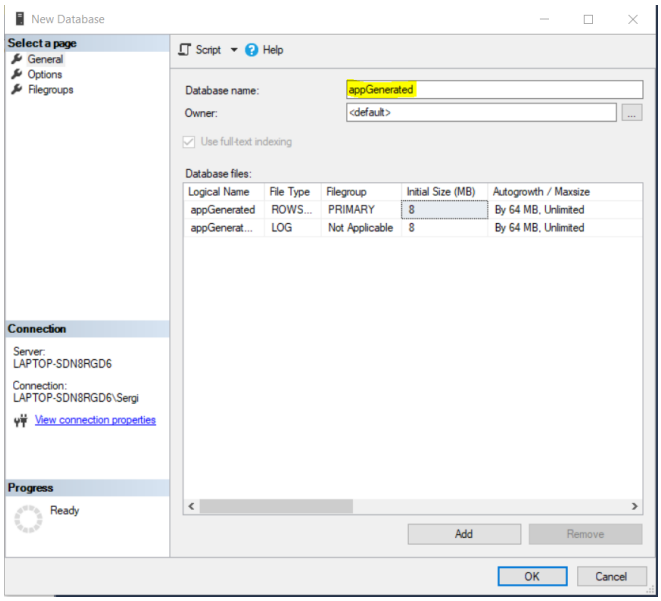
4. Preparación del entorno de la parte Back de la aplicación móvil generada

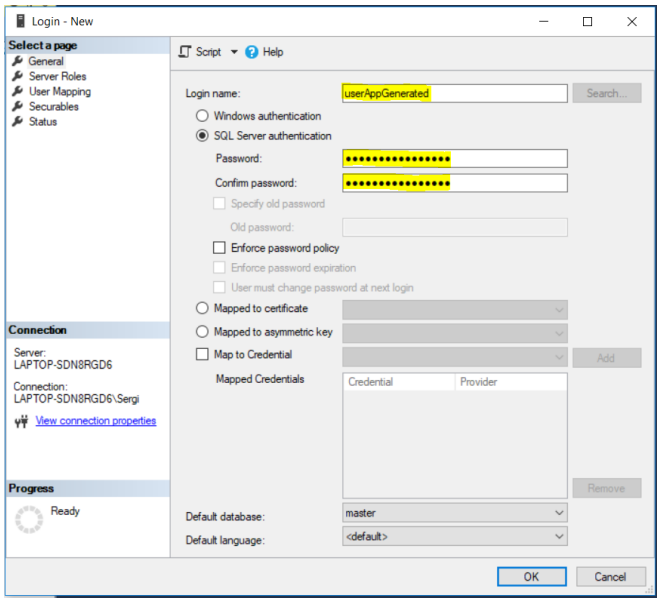
Como en el punto publicación de la parte Back del editor de contenidos, también es necesario crear la base de datos y el sitio Web de la parte Back de la aplicación móvil.

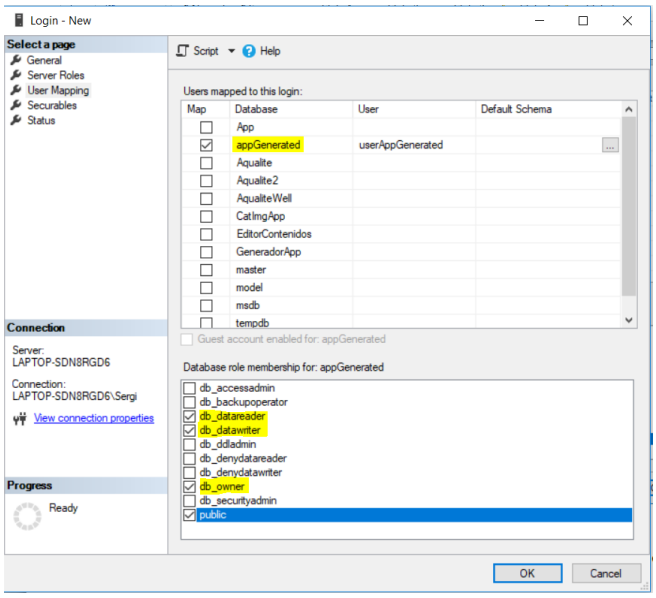
a. Creación de la base de datos

La creación de la base de datos se realiza prácticamente igual que como se describe en el punto de creación de la base de datos de la publicación de la parte Back del editor de contenidos.

Las diferencias están en el nombre de la nueva base de datos creada, el usuario añadido y la relación del usuario con la nueva base de datos.


New Database
Select a page
General
Options
Filegroups
Script
Help
Database name:
appGenerated
Owner:
<default>
Use full-text indexing
Database files:
Logical Name
File Type
Filegroup
Initial Size (MB)
Autogrowth / Maxsize
appGenerated
ROWS...
PRIMARY
8
By 64 MB, Unlimited
appGenerat...
LOG
Not Applicable
8
By 64 MB, Unlimited
Connection
Server:
LAPTOP-SDN8RGD6
Connection:
LAPTOP-SDN8RGD6\Sergi
View connection properties
Progress
Ready
Add
Remove
OK
Cancel

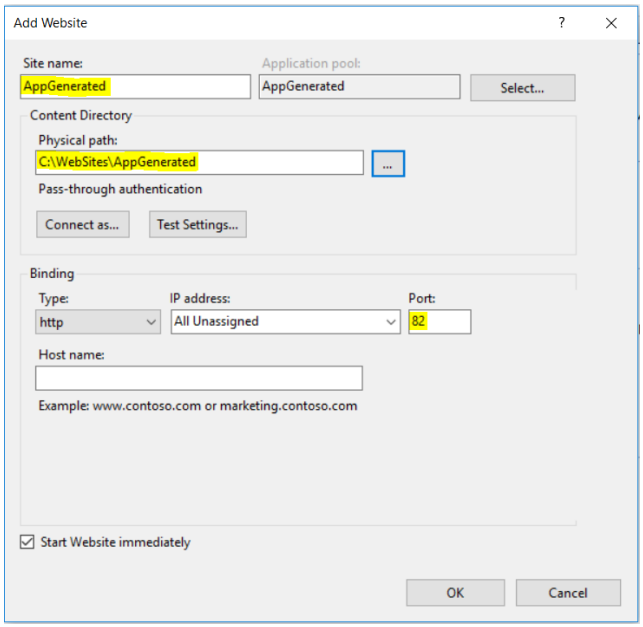

Login - New
Select a page
General
Server Roles
User Mapping
Securables
Status
Script
Help
Login name:
userAppGenerated
Search...
Windows authentication
SQL Server authentication
Password:
Confirm password:
Specify old password
Old password:
Enforce password policy
Enforce password expiration
User must change password at next login
Mapped to certificate
Mapped to asymmetric key
Map to Credential
Add
Mapped Credentials
Credential
Provider
Remove
Default database:
master
Default language:
<default>
Connection
Server:
LAPTOP-SDN8RGD6
Connection:
LAPTOP-SDN8RGD6\Sergi
View connection properties
Progress
Ready
OK
Cancel


Login - New
Select a page
General
Server Roles
User Mapping
Securables
Status
Script
Help
Users mapped to this login:
Map
Database
User
Default Schema
App
appGenerated
userAppGenerated
Aqualite
Aqualite2
AqualiteWell
CatImgApp
EditorContenidos
GeneradorApp
master
model
msdb
tempdb
Guest account enabled for: appGenerated
Database role membership for: appGenerated
db_accessadmin
db_backupoperator
db_datareader
db_datawriter
db_ddladmin
db_denydatareader
db_denydatawriter
db_owner
db_securityadmin
public
Connection
Server:
LAPTOP-SDN8RGD6
Connection:
LAPTOP-SDN8RGD6\Sergi
View connection properties
Progress
Ready
OK
Cancel

b. Creación del sitio Web

Los pasos a seguir son similares a los descritos en la creación y publicación del sitio Web de la parte Back del editor de contenidos. Pero en este caso no es necesario realizar la publicación ni asignar los permisos necesarios del directorio de hospedaje al usuario IIS_IUSRS, ya que se realiza automáticamente en la ejecución del modelo generativo. Solamente es necesario crear el directorio de hospedaje y el sitio Web.

El directorio creado es *C:\WebSites\AppGenerated* y el nuevo sitio Web es AppGenerated, su creación se diferencia de la descrita en pasos anteriores: en el nombre; el puerto; el directorio de hospedaje y la configuración de la identidad del Application Pool, que para este caso es ApplicationPoolIdentity.



iii. Configuración de la solución

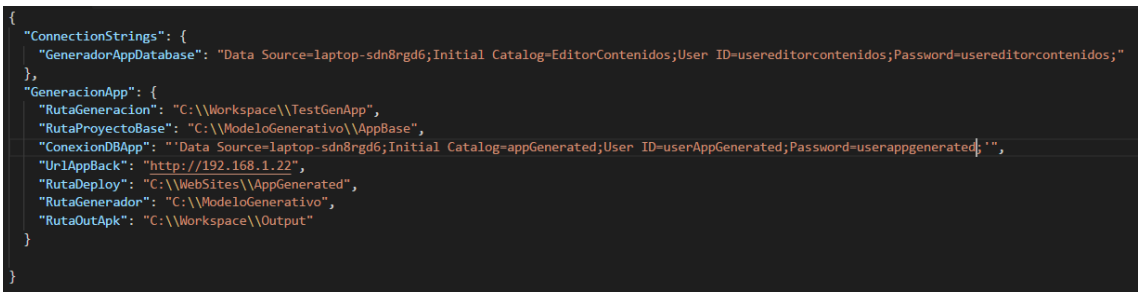
El último paso a realizar para finalizar la preparación del entorno, consiste en configurar correctamente la solución desarrollada.

1. Configuración de la parte Back del editor de contenidos

En esta parte existen todos los parámetros de configuración relacionados con la configuración del mismo editor de contenidos, del modelo generativo y de la aplicación móvil generada. Estos parámetros se encuentran en el fichero *appsettings.json* del directorio de hospedaje de la parte Back del editor de contenidos.

Parámetro	Descripción
GeneradorAppDatabase	Cadena de conexión de la base de datos del editor de contenidos
RutaGeneracion	Directorio de trabajo

RutaProyectoBase	Directorio dónde se encuentra la solución AppBase
ConexionDBApp	Cadena de conexión de la base de datos de la aplicación móvil generada
UrlAppBack	URL de la parte Back de la aplicación móvil generada
RutaDeploy	Directorio de hospedaje de la parte Back de la aplicación móvil generada
RutaGenerador	Directorio dónde se encuentra el código fuente del modelo generativo
RutaOutApk	Directorio dónde se guarda el fichero apk de la aplicación móvil generada
RutaAndroidSDK	Directorio dónde está instalado la SDK de Android



2. *Configuración de la parte Front del editor de contenidos*

La única configuración a realizar en la parte Front del editor de contenidos es la URL de la parte Back dónde hacer las peticiones. Para su realización se ha de editar el parámetro url del fichero app.js de la carpeta Front de la solución GenApp. En el caso visto el valor es *http://localhost:81/api*.

```
app.constant('url', 'http://localhost:81/api');
```

10. Bibliografía

[1] Herrington, Jack. Code Generation in action. Manning, 2003.

[2] Heradio Gil, R. *Metodología de desarrollo de software basada en el paradigma generativo. Realización mediante la transformación de ejemplares*. Ph. D. Thesis, Departamento de Ingeniería de Software y Sistemas Informáticos de la UNED, España, Abril 2007.

[3] Czarnecki, K. *Generative Programming Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models*. Ph. D. Thesis, Department of Computer Science and Automation, Technical University of Ilmenau, October 1998.

[4] *Language Workbenches: The Killer-App for Domain Specific Languages?*. MartinFowler.com. Biblioteca: En línea. Actualizada: 12-06-2005. Disponible en: https://www.martinfowler.com/articles/languageWorkbench.html

[5] de la Torre Llorente, César – Zorrila Castro Unai – Calvarro Nelson, Javier – Ramos Barroso, Miguel Ángel. *Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0*. Microsoft.

[6] Bruce Eckel. *Thinking in Patterns Problem-Solving Techniques with Java*.

[7] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Design Patterns Elements of Reusable Object-Oriented Software*. Addison Wesley, 1994

[8] *iOS vs Android, tendencias y cuota de Mercado*. Digital55. Biblioteca: En línea. Actualizada: 07-10-2017. Disponible en: https://www.digital55.com/ios-vs-android-tendencias-y-cuota-de-mercado/

[9] *Mobile App*. Wikipedia. Biblioteca: En línea. Actualizada: 29-05-2018. Disponible en: https://en.wikipedia.org/wiki/Mobile_app

[10] APK (formato). Wikipedia. Biblioteca: En línea. Actualizada: 25-05-2018. Disponible en: https://es.wikipedia.org/wiki/APK_(formato)

[11] *Las Categorías Más Comunes En El Desarrollo De Aplicaciones Móviles*. Danthop. Biblioteca: En línea. Disponible en: http://info.danthop.com/las-categorias-mas-comunes-en-el-desarrollo-de-aplicaciones-m%C3%B3viles

[12] *Xamarin*. Microsoft. Biblioteca: En línea. Disponible en: https://www.xamarin.com/

[13] *Cross-platform*. Wikipedia. Biblioteca: En línea. Actualizada: 20-06-2018. Disponible en: https://en.wikipedia.org/wiki/Cross-platform

[14] *Alternativas para desarrollar aplicaciones móviles multiplataforma*. Alfonso Marin. Biblioteca: En línea. Actualizada: 22-08-2016. Disponible en: https://alfonsomarin.com/desarrollo-movil/articulos/alternativas-para-desarrollar-aplicaciones-moviles-multiplataforma

[15] *¿Qué es una App Nativa?*. qodeBlog. Biblioteca: En línea. Actualizada: 03-08-2014. Disponible en: http://qode.pro/blog/que-es-una-app-nativa/

[16] *Documentación de Xamarin*. Microsoft. Biblioteca: En línea. Disponible en: https://developer.xamarin.com/

[17] *Xamarin*. Wikipedia. Biblioteca: En línea. Actualizada: 27-05-2018. Disponible en: https://es.wikipedia.org/wiki/Xamarin

[18] *Android Developers*. Android Developers. Biblioteca: En línea. Disponible en: https://developer.android.com

[19] *Apache Cordova*. Apache Cordova. Biblioteca: En línea. Disponible en: https://cordova.apache.org/

[20] *Implementar el repositorio y unidad de patrones de trabajo en una aplicación de ASP.NET MVC*. Biblioteca: En línea. Disponible en: https://docs.microsoft.com/es-es/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application

[21] *El patrón MVVM en Xamarin Forms*. Miguel A. Gómez. Biblioteca: En línea. Disponible en: https://miguelgomez.io/xamarin/patron-mvvm-xamarin-forms/

[22] *Información general sobre Entity Framework*. Microsoft. Biblioteca: En línea. Disponible en: https://docs.microsoft.com/es-es/dotnet/framework/data/adonet/ef/overview

[23] *Migraciones EF-Core*. Microsoft. Biblioteca: En línea. Disponible en: https://docs.microsoft.com/es-es/ef/core/managing-schemas/migrations/

[24] *MSBuild1*. Microsoft. Biblioteca: En línea. Disponible en: https://msdn.microsoft.com/es-es/library/dd393574.aspx

[25] *Referencia de la línea de comandos de MSBuild*. Microsoft. Biblioteca: En línea. Disponible en: https://msdn.microsoft.com/es-es/library/ms164311.aspx

[26] *Proceso de compilación*. Microsoft. Biblioteca: En línea. Disponible en: https://docs.microsoft.com/es-es/xamarin/android/deploy-test/building-apps/build-process

[27] *Dotnet publish*. Microsoft. Biblioteca: En línea. Disponible en: https://docs.microsoft.com/es-es/dotnet/core/tools/dotnet-publish?tabs=netcore21

[28] *Dotnet-build*. Microsoft. Biblioteca: En línea. Disponible en: https://docs.microsoft.com/es-es/dotnet/core/tools/dotnet-build?tabs=netcore2x

[29] *Aspectos fundamentales de la aplicación*. Android Developers. Biblioteca: En línea. Disponible en: https://developer.android.com/guide/components/fundamentals?hl=es-419

[30] *Sharing code overview*. Microsoft. Biblioteca: En línea. Disponible en: https://docs.microsoft.com/en-us/xamarin/cross-platform/app-fundamentals/code-sharing

[31] *Bibliotecas de clases portables (PCL)*. Microsoft. Biblioteca: En línea. Disponible en: https://docs.microsoft.com/es-es/xamarin/cross-platform/app-fundamentals/pcl?tabs=windows

[32] *Páginas de Xamarin.Forms*. Microsoft. Biblioteca: En línea. Disponible en: https://docs.microsoft.com/es-es/xamarin/xamarin-forms/user-interface/controls/pages

[33] *Diseños de Xamarin.Forms*. Microsoft. Biblioteca: En línea. Disponible en: https://docs.microsoft.com/es-es/xamarin/xamarin-forms/user-interface/controls/layouts

[34] *Introducción a DependencyService*. Microsoft. Biblioteca: En línea. Disponible en: https://docs.microsoft.com/es-es/xamarin/xamarin-forms/app-fundamentals/dependency-service/introduction

[34] *Tutorial de BPMN y BPMN 2.0*. Lucidchart. Biblioteca: En línea. Disponible en: https://www.lucidchart.com/pages/es/bpmn-bpmn-20-tutorial

[35] *Composite*. Wikipedia. Biblioteca: En línea. Actualizada: 10-07-2018 Disponible en: https://en.wikipedia.org/wiki/Composite_pattern

[36] *Factory method pattern*. Wikipedia. Biblioteca: En línea. Actualizada: 16-10-2017 Disponible en: https://en.wikipedia.org/wiki/Factory_method_pattern

11. Lista de acrónimos

API Application Programming Interface

APK Android Application Package

CLI Command-Line Interface

CMS Content Management System

CRUD Create, Read, Update and Delete

CSS Cascading Style Sheets

DDD Domain Driven Design

DSL Domain-Specific Language

DTO Data Transfer Object

GPS Global Positioning System

HTML HyperText Markup Language

IDE Integrated Development Environment

IIS Internet Information Service

MVVM Model-View-ViewModel

ORM Object-Relational mapping

SDK Software Development Kit

URI Uniform Resource Identifier

URL Uniform Resource Locator

XAML eXtensible Application Markup Language