4GL DEL SIGLO XXI

De pantallas verdes a teléfonos inteligentes

Genero®

En el apogeo de su gloria, IBM® Informix® 4GL (4GL) dominó los sectores: minorista, financiero, manufacturero y gubernamental debido a su capacidad para crear aplicaciones comerciales confiables y de clase mundial de forma rápida y rentable.

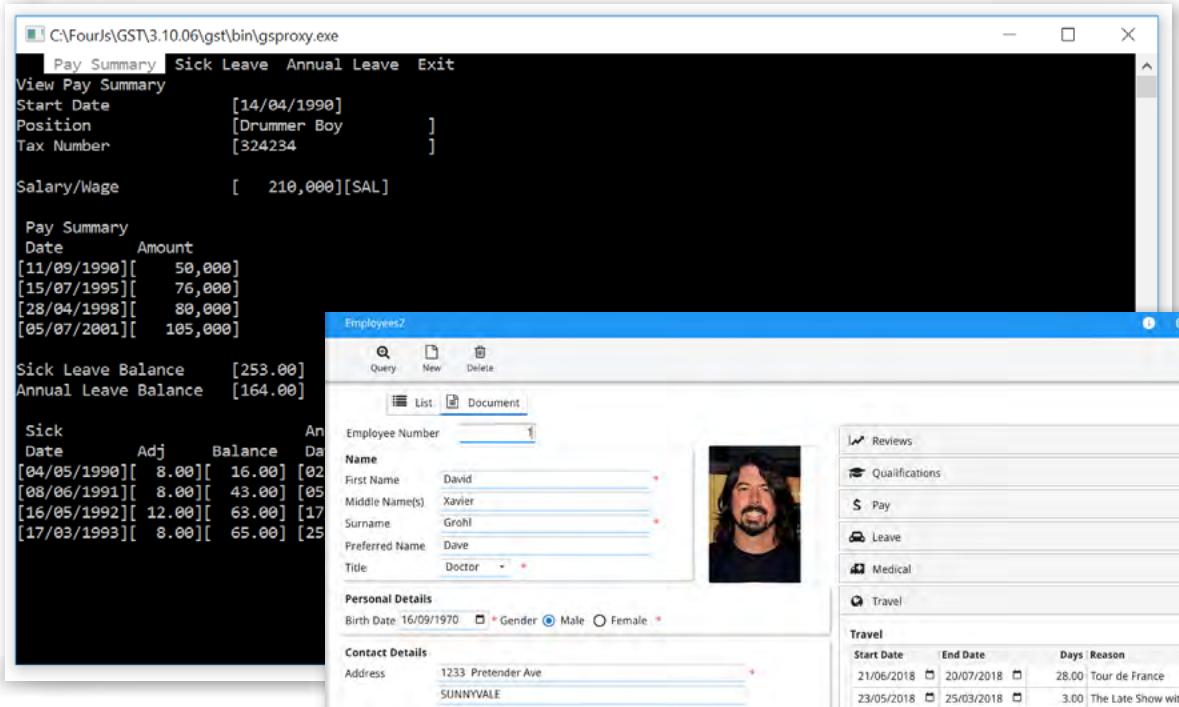
Sin embargo, a mediados de los años noventa, el advenimiento de la computación cliente-servidor y los lenguajes de programación orientados a objetos acabaron con esa gloria, incluso cuando esas tendencias nunca podrían igualar 4GL en la productividad del desarrollador.

Sin embargo, en un mundo donde los presupuestos de TI están constantemente bajo presión, 4GL sigue siendo relevante y dominante hasta el día de hoy. Todavía es la forma más productiva de desarrollar aplicaciones comerciales.

En los últimos 30 años, los avances tecnológicos han revolucionado la forma en que trabajamos. Durante ese tiempo, 4GL luchó por mantener el ritmo, razón por la cual nació Genero®. Genero se escribió desde cero utilizando las últimas técnicas y tecnologías, sin dejar de ser 100% compatible. Genero respeta la inversión tan importante en código y habilidades de las últimas décadas. Es uno de los entornos de desarrollo más portables, ampliables y escalables que permite que las aplicaciones evolucionen sin esfuerzo con las tendencias tecnológicas.

En su versión Genero, las aplicaciones 4GL se ejecutan en cualquier lugar, independientemente del cliente, la base de datos o el sistema operativo, y con poca o ninguna modificación. Ya sea en una pantalla verde, en una computadora portátil MacOS® o MS® Windows, en un navegador o en un teléfono inteligente, las aplicaciones se adaptan en tiempo de ejecución desde un solo árbol de código fuente. Esto explica por qué Genero sigue promoviendo la causa 4GL y prosperando como la pieza clave de tantos sistemas de información de misión crítica que manejan grandes empresas e instituciones gubernamentales.

GENERO protege décadas de inversión en programación y conocimiento



4GL

Genero

4GL nunca evolucionó de sus raíces de 'pantalla verde' de los 80 y nunca ha proporcionado una experiencia de usuario contemporánea a sus clientes. Como consecuencia, no es compatible con ninguna de las funciones ergonómicas que ofrecen los clientes actuales: MS Windows, macOS, navegadores o dispositivos móviles.

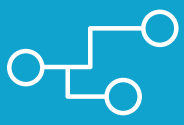
En ausencia de pestañas, widgets y barras de herramientas, las aplicaciones desarrolladas con 4GL tienden a apiñar tanta información como sea posible en un solo formulario dividido en ventanas del tipo 'maestro' y 'detalle'.

Genero despeja el desorden, revoluciona la experiencia del usuario y hace que las aplicaciones sean indistinguibles del resto.

> Genero es único en su capacidad para combinar código escrito hace 25 años con un nuevo código que explota los paradigmas informáticos de vanguardia.

ERGONOMÍA CONTEMPORÁNEA

En 4GL, las ventanas de diálogo emergentes proporcionan un mecanismo para acceder a más información, pero los usuarios deben salir de uno antes de ingresar a otro. Los usuarios pueden 'saltar' de campo a campo, pero solo secuencialmente y en un diálogo a la vez. Las aplicaciones 4GL tampoco son compatibles con el navegador, donde es fácil perder el contexto a través del botón 'regresar'. Como resultado, son pesados y frustrantes de usar para aquellos con experiencia en sistemas modernos. Genero resuelve estos problemas con una tecnología de presentación en el cliente de última generación.

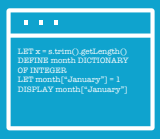


INTEROPERABILIDAD

4GL también es notoriamente introvertido. Se habla muy bien pero ignora a los demás. No es que no quiera comunicarse, simplemente no sabe cómo hacerlo. Las aplicaciones modernas, independientemente de cómo se escribieron, interactúan a través de SOAP y servicios web RESTful utilizando estructuras de datos XML y JSON. Estas características se han incorporado a Genero para hacer con que las aplicaciones sean sociables, capaces de comunicarse sin esfuerzo con el mundo exterior y de ser utilizadas sin un conocimiento profundo de los protocolos subyacentes.

SEGURIDAD

Sin embargo, las aplicaciones expuestas al mundo exterior son extremadamente vulnerables a ataques. Por lo tanto, es fundamental desarrollar aplicaciones seguras que se adapten perfectamente a este complejo mundo. Genero proporciona clases y métodos que realizan criptografía, algoritmos de resumen, autenticación, certificación e inicio de sesión único para que las aplicaciones puedan interactuar con las redes sociales y otros sistemas orientados a Internet.



MEJORAS DE LENGUAJE

El lenguaje ha sido mejorado para incorporar características tales como parámetros nombrados, pasar parámetros de funciones por referencia, manejo de eventos, internacionalización e interconexión con Java®. Una clase de métodos abstractos del sistema operativo cubre la administración del sistema de archivos lo que mejora en gran medida la portabilidad de la aplicación.

UN ENTORNO DE DESARROLLO INTEGRADO

Cada lenguaje de desarrollo moderno necesita un entorno rico para poder crear efectivamente. Genero Studio, Genero Report Writer y Genero Ghost Client cumplen esa función: para mejorar la productividad de los desarrolladores y probar las aplicaciones en cuanto a validez, confiabilidad y escalabilidad.

ARQUITECTURA

Genero libera al desarrollador de los grilletes de 4GL, en particular de su cultura monolítica, mono-base de datos, mono-sistema operativo y de 'pantalla verde'. Al liberar al desarrollador de estas limitaciones, Genero da un nuevo soplo de vida a las aplicaciones heredadas, preparándolas para los días de hoy y mañana. Lo hace desde un único entorno de desarrollo, creando aplicaciones que son portátiles, seguras y que escalan a miles de usuarios simultáneos sin la necesidad de un costoso middleware.

ERGONOMÍA CONTEMPORÁNEA

Es posible recompilar su aplicación 4GL con Genero 'tal cual' utilizando una interfaz de usuario de texto (TUI) y seguir desarrollándose como de costumbre en el modo 'ASCII'. Alternativamente, recompilar en 'modo tradicional' presenta una interfaz gráfica de usuario, teclas de funciones suaves, barras de desplazamiento y el uso del mouse 'automágicamente' sin necesitar cambios en el código. Este enfoque es rápido, pero la interfaz de usuario conserva las restricciones de su geometría ASCII. Sería una lástima, porque los beneficios de las extensiones de la Interfaz gráfica de usuario (GUI) de Genero no se emplearían.

Quizás el cambio más importante en el lenguaje realizado en Genero, es la introducción de la palabra clave LAYOUT para los archivos .per. Cuando se utiliza en lugar de SCREEN, el formulario se despliega mediante la activación de una sintaxis con un superconjunto que comprende verbos gráficos y tokens.

En muchos casos, el mayor impacto visual se puede lograr simplemente modificando los archivos .per más importantes. Genero amplía la sintaxis del formulario .per con los siguientes elementos gráficos abstractos que se despliegan de forma nativa en Windows, Linux®, macOS, navegador y API móviles:

Los widgets gráficos definen cómo se muestran y editan los valores en un formulario GUI:

- BUTTON – presenta una zona activable o seleccionable en la pantalla que desencadena una acción,
- BUTTONEDIT – una edición asociada a un botón que se usa para activar una acción para editar o proporcionar más detalles sobre un valor,
- CHECKBOX – el usuario puede alternar entre dos valores (o tres, incluyendo NULL),
- COMBOBOX – el usuario puede seleccionar de una lista desplazable de valores,
- DATEEDIT – el usuario puede seleccionar una fecha de un widget de calendario,
- DATETIMEEDIT – el usuario puede seleccionar una fecha y hora desde un widget especializado,
- EDIT – despliegue predeterminado y edición de un valor,
- IMAGE – desplegar una imagen,
- LABEL – muestra texto que no se puede editar, por ejemplo títulos o descripciones,
- PROGRESSBAR - muestra un entero como una barra que aumenta hasta el valor máximo,
- RADIOGROUP – el usuario puede seleccionar un valor de múltiples opciones,
- SLIDER – el usuario puede aumentar o disminuir un valor entero deslizando un widget,
- SPINEDIT – el usuario puede ajustar un valor entero seleccionando los botones arriba / abajo,
- TEXTEDIT – el usuario puede editar varias líneas de texto,
- TIMEEDIT – el usuario puede seleccionar una hora desde un widget especializado.

Los contenedores de diseño definen cómo se organizan los widgets GUI uno con respecto a otro.

- GRID – posiciona widgets "hijos" con un parámetro de X, Y, ancho y alto determinado,
- SCROLLGRID: organiza varias instancias de un GRID,
- TABLE - organiza los widgets "hijos" en una o más filas idénticas de una estructura de tabla,
- TREE - similar a TABLE pero con un estructura de árbol plegable y jerárquica,
- STACK - organice los widgets "hijos" con un estilo que sea nativo de la interfaz de usuario,
- GROUP - coloca un borde alrededor de varios contenidos "hijos" y opcionalmente le da un título,
- FOLDER, ACCORDIONS + PAGE: organiza los contenedores "hijos" en varias páginas donde solo se vea el contenido de una página, y los demás aparecerán como pestañas o acordeones plegables,
- VBOX: apila los contenedores "hijos" verticalmente,
- HBOX: organiza los contenedores "hijos" horizontalmente.

Componentes web – permite el uso de tecnologías HTML de terceros. (HTML / JS / CSS) y la creación de componentes personalizados de interfaz de usuario. Ejemplos de usos incluyen: incrustación de videos, documentos HTML, , gráficas, edición de texto enriquecido, calendario, captura de firmas, tablas enriquecidas, mapas de imágenes mapas / pantallas táctiles de punto de venta, autenticación SSO. Los componentes web predefinidos proporcionan edición de texto enriquecido, selección de imágenes múltiples y despliegue SGV.

Valores predeterminados de las acciones que centralizan la definición de texto, comentarios, imágenes, aceleradores y otras propiedades de las acciones,

Menús superiores (barra de menús) – aun arreglo estructurado de opciones de menú en la parte superior de una ventana,

Barras de herramientas – un arreglo de botones en la parte superior, inferior, izquierda o derecha del formulario,

Diálogos paralelos – habilita una pantalla dividida con cada panel ejecutando código independiente,

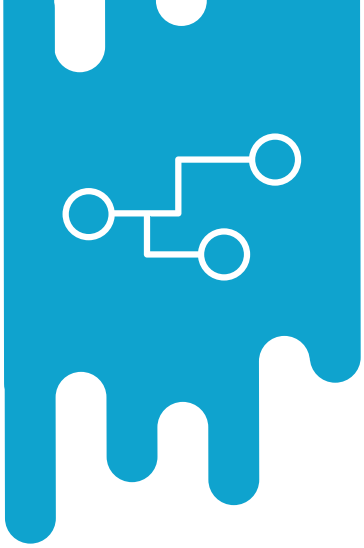
Múltiples diálogos: fusiona múltiples instrucciones de interacción del usuario (MENÚ, INPUT, INOUT ARRAY, DISPLAY ARRAY, CONSTRUCT) en un diálogo único activo que facilita la navegación del usuario,

Diálogos dinámicos: una instrucción de interacción del usuario (INPUT, CONSTRUCT, DISPLAY ARRAY, INPUT ARRAY, DIALOG, ...) que se define en tiempo de ejecución, en lugar de en tiempo de compilación. Esto permite el uso de código genérico con una estructura de datos variable. Se usa mejor junto con la creación de formularios dinámicamente en tiempo de ejecución y el uso de cursores dinámicos de la base de datos utilizando base.Sqlhandle class,

Multiple selects - la capacidad de seleccionar varias filas dentro de una matriz, con el fin de realizar una acción como eliminarlas o imprimirlas,

Enfoque en el campo: la capacidad de seleccionar una celda desde un DISPLAY ARRAY, no solamente toda una fila,

Drag and drop – permite arrastrar y soltar ciertos objetos uno encima del otro para realizar una acción determinada. dentro o entre las aplicaciones.



INTEROPERABILIDAD

Genero ha evolucionado en los últimos 25 años para incluir las últimas API y estructuras de datos para la interacción con otros sistemas. Esto incluye las estructuras de datos más populares: XML y JSON junto con los métodos más populares: servicios web, SOAP y RESTful.

Data Structures

XML

Genero proporciona un conjunto completo de clases y métodos para trabajar con documentos XML. Esto incluye compatibilidad con los modelos DOM (Document Object Modelling) y StAX (Streaming API for XML), así como serialización y transformaciones XSLT.

JSON

Genero proporciona clases para trabajar con JSON (JavaScript Object Notation) - un formato ligero muy conocido de intercambio de datos.

Methods

Servicios web – Genero proporciona bibliotecas para crear servicios web, tanto como servidor o proveedor del servicio Web o como cliente del servicio web.

SOAP – parte de la clave es el archivo WSDL que proporciona los detalles de la interfaz. Toma una función 4GL existente con parámetros de entrada y salida definidos, y crea o publica una operación como servicio web basado en esa función.

RESTful – más ligero que SOAP. Pasa mensajes (JSON, XML normalmente) en el formato de su elección.

File – utiliza todas estas clases, pero elimina la necesidad de bibliotecas 'C'.

Base.Channel – antes de los servicios web, usted podría haber interactuado con otros sistemas leyendo / escribiendo en archivos. Como la API de llamada al sistema de 4GL era tan pobre, se tenía que escribir bibliotecas en C y / o scripts de shell. La clase base.Channel es un conjunto de métodos utilizados para leer y escribir en archivos y se puede usar para reemplazar los scripts de shell y C.

OS – esta clase permite enumerar los archivos en los directorios y probar la existencia, así como los permisos. Esto elimina las dependencias de SO y hace que el código sea portable.

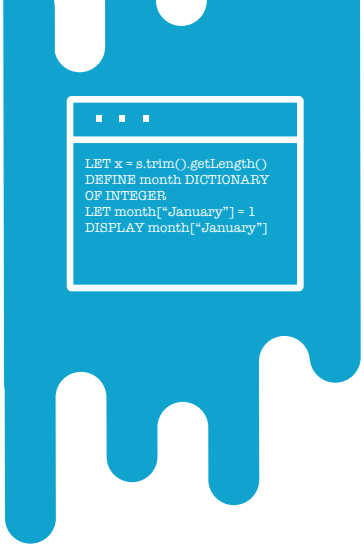
Sockets – se agregaron métodos a la clase base.Channel para facilitar la comunicación del socket, tanto como cliente o servidor a/desde un servidor y puerto.

Pipes – se agregó el método openPipe a la clase base.Channel para facilitar la comunicación con un subproceso a través de un canal.

SEGURIDAD

Clase de seguridad – el paquete de seguridad proporciona clases y métodos para realizar criptografía básica. Estos incluyen clases PBKDF2 y BCrypt para el almacenamiento y la verificación de contraseñas encriptadas. También se proporcionan clases para implementar algoritmos de resumen como SHA512, MD5 (etc.) juto con clases para los algoritmos HexBinary y Base64.

Comunicaciones Seguras – si una aplicación desea intercambiar mensajes con una aplicación financiera, comercial o personal en la web , debe ser capaz de autenticar el origen del mensaje, garantizar que ninguna aplicación maliciosa haya alterado el mensaje original y asegurarse de que ninguna aplicación de terceros pueda interceptar el mensaje. Esto involucra certificados, autoridades de certificación, claves privadas, etc. todo provisto por Genero.



MEJORAS DEL LENGUAJE

La sintaxis del lenguaje 4GL se ha mejorado mucho para eliminar sus limitaciones y ajustarse a las prácticas de codificación actuales. Aquí hay una lista no exhaustiva de mejoras que hemos realizado:

Syntaxis

Palabras clave – STRING, CONSTANT, BOOLEAN, TYPE, BIGINT, TINYINT, PUBLIC, PRIVATE, NVL, IFF, SFMT, DICTIONARY, DYNAMIC ARRAY, TRY/CATCH

ON ACTION – el bloque ON ACTION ejecuta código cuando ocurre un evento de usuario:

```
ON ACTION zoom
```

en lugar de:

```
ON KEY(F5)
```

La acción 'zoom' está especificada por un formulario .per o un archivo predeterminado de acción, y se ejecuta independientemente de cómo se activó.

Pasar los parámetros de una función por referencia – esta característica es útil cuando se trata de estructuras complejas, registros y matrices. Las estructuras complejas se pasan a la función sin la necesidad de tratar con el resultado de retorno. Solo una instancia de este reside en la memoria, evitando dos copias (una en la función de llamada y otra en la función llamada), consumiendo menos memoria.

```
DEFINE arr DYNAMIC ARRAY OF complexDataType

# arr contains untransformed data
CALL transform_array(arr)
# arr contains transformed data
```

Inicialización de Literales – las Literales se pueden inicializar simplemente:

```
TYPE colorType RECORD
   red, green, blue INTEGER
END RECORD

DEFINE purple colorType = (255,0,255)
```

Sensible a Mayúsculas (opcional) - si se selecciona, se convierte en una sintaxis inválida:

```
DEFINE foo, Foo, FOO STRING
```

Y:

```
DEFINE accountCode STRING
DEFINE accountCodeLength INTEGER
        LET accountCodeLength = accountCode.getlength()
```

Parámetros Nombrados – los Parámetros Nombrados eliminan la necesidad de especificar cada argumento y obtener su orden correcto al llamar a las funciones. Los parámetros no especificados se establecen con valores predeterminados. Mejora la legibilidad del código.

```
CALL fgl_report_configurePDFDevice(fromPage = 1, toPage = 1)
```

Características varias del código – estos son ejemplos de sintaxis que se ven en otros lenguajes, pero que no se ven en 4GL. Esto facilita la migración para desarrolladores que desconocen las limitaciones de 4GL.

Se permiten declaraciones de tipo en los parámetros de llamada y retorno de tipos de funciones: una forma abreviada de mejorar la seguridad del tipo de función:

```
FUNCTION add(x INTEGER, y INTEGER) RETURNS INTEGER
```

Método abreviado del método de encadenamiento de objeto encadenamiento:

```
LET x = s.trim().getLength()
```

Un nuevo tipo de datos DICTIONARY (también conocido como hash table) usa una cadena como llave para indexar tablas. Puede almacenar todos los tipos de datos Genero:

```
DEFINE month DICTIONARY OF INTEGER
        LET month["January"] = 1
        DISPLAY month["January"]
```

ANYRECORD – tipo de registro genérico
En la compilación, solo el tipo de una función (parámetros, valores devueltos) es conocido cuando se usan referencias en funciones. El tipo de registro genérico ANYRECORD se utiliza cuando una función toma como entrada un registro cuya estructura es desconocida.

IMPORT – si hay una biblioteca externa que le gustaría usar dentro de su código, la declaración IMPORT se puede usar de la siguiente manera:

- `IMPORT FGL` - para importar una biblioteca 4GL
- `IMPORT` - para importar una biblioteca 'C'
- `IMPORT JAVA` - para importar una biblioteca de Java.

Nuestra comunidad de desarrolladores usa, por ejemplo, las instrucciones IMPORT JAVA para los siguientes propósitos:
- Creación de archivos MS® Word y Excel utilizando la biblioteca POI de Apache
- Cargar videos a You-Tube®

Internacionalización – las aplicaciones Genero están diseñadas para trabajar en todo el mundo a través de soporte para UTF-8 y código independiente del juego de caracteres.

Una *Cadena Localizada* es texto de la aplicación que se define en catálogos de cadenas fuera del código fuente estándar. Esto facilita el mantenimiento del texto traducido y el uso de una traducción particular en tiempo de ejecución sin la necesidad de recompilar.

Una *configuración regional* de la aplicación (Application Locale) determina qué catálogo regional está en uso para un usuario determinado. Por lo tanto, la misma instancia de aplicación puede admitir múltiples idiomas simultáneamente. El seguimiento de derecha a izquierda también es compatible con las regiones árabes y tanto la semántica de longitud de BYTE como la de CHARACTER son compatibles con Asia.

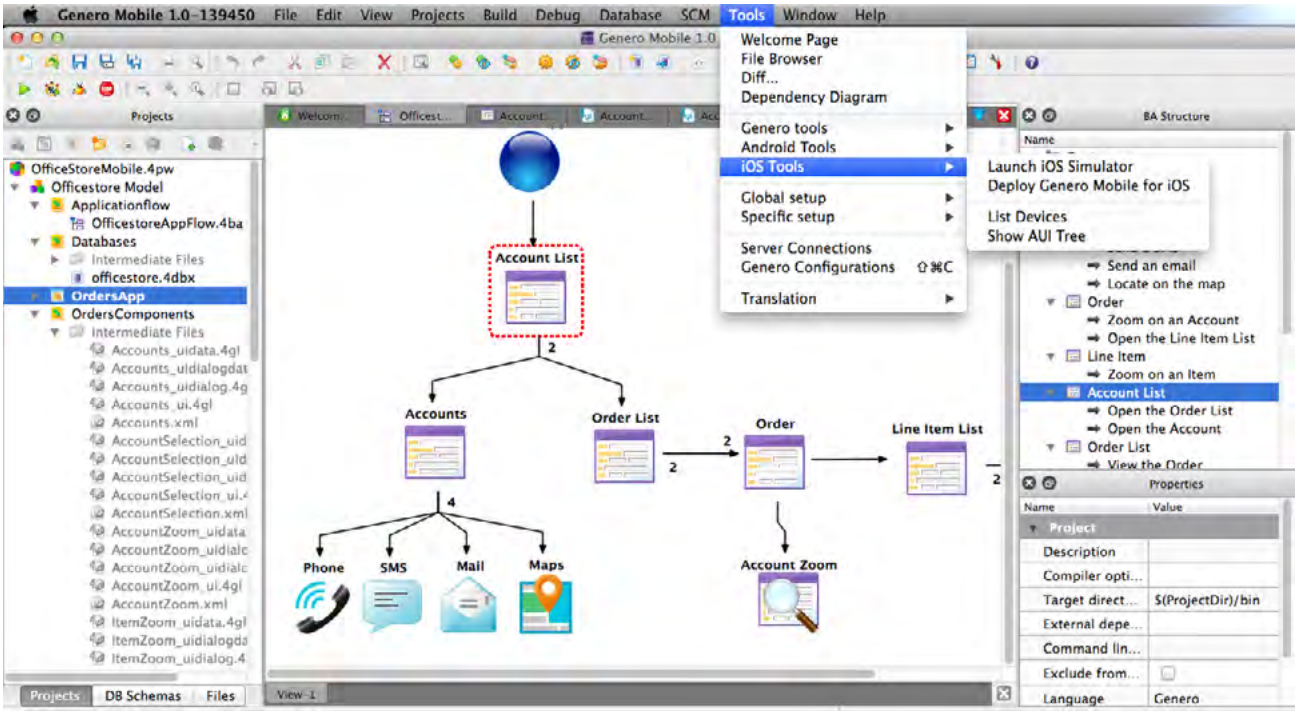
UN ENTORNO DE DESARROLLO INTEGRADO

Genero Studio es un entorno de desarrollo integrado (IDE) diseñado para mejorar la productividad del desarrollador. Permite la depuración simultánea de aplicaciones desarrolladas para computadoras de escritorio, navegadores y dispositivos móviles.

Incluye un editor de código sensible al idioma que admite la finalización de código, la verificación de sintaxis en tiempo real, el resaltado de palabras clave de 4GL y otros formatos de archivo. También incluye un depurador gráfico, un diseñador de formularios gráficos, un escritor de informes gráficos, un administrador de proyectos para 'crear' aplicaciones complejas, un generador de esquemas de bases de datos, un meta-esquema, un generador de código, una herramienta de modelado visual, herramientas de análisis y perfiles.

UNA PLATAFORMA DE DESARROLLO DE BAJO CÓDIGO (LCDP)

El Business Application Modeller (BAM) de Genero Studio es una plataforma de desarrollo de bajo código (LCDP) destinada a desarrolladores que buscan un ciclo de vida de desarrollo incremental, continuo y multiplataforma usando diagramas en lugar de codificación manual.



BAM - integrando apps nativas para teléfonos móviles en un módulo de Genero Mobile

El código para aplicaciones y servicios web se define en plantillas y se modela visualmente, lo que garantiza la coherencia y código estructurado organizado en capas:

- Almacenamiento de datos
- Servicios de datos
- Publicación de servicios web (RESTful, SOAP)
- Formularios e informes

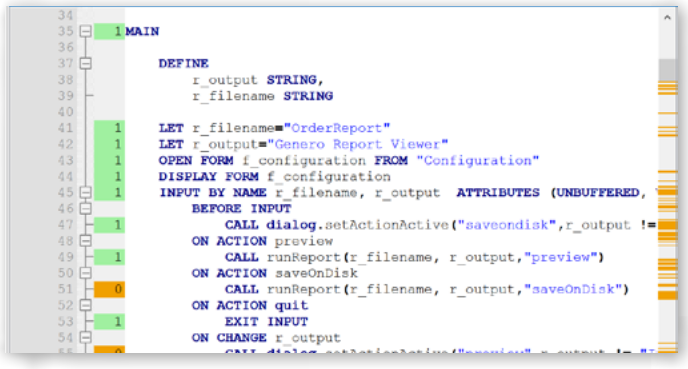
La mayoría de los patrones de codificación orientados a negocios están cubiertos en las plantillas predeterminadas como formularios 'CRUD': listas, detalles, filtrado, navegación, administración de simultaneidad e informes. Las aplicaciones móviles aprovechan las herramientas nativas, como los contactos, el calendario y el teléfono que pueden integrarse estrechamente en su desarrollo. Esto permite que gran parte de la aplicación se describa en forma de diagramas sin codificación alguna. El desarrollador modifica los diagramas a lo largo del tiempo para mejorar o modificar el comportamiento de la aplicación. Las reglas de negocio pueden codificarse en eventos.

PODEROSAS AYUDAS PARA EL DESARROLLO

Estas herramientas se pueden activar durante la compilación o durante la ejecución desde Genero Studio o desde la línea de comandos.

Profiler – profiler identifica los cuellos de botella del programa. Resalta el porcentaje de tiempo dedicado a las funciones y el número de veces que se invocan funciones desde otras funciones.

Code coverage – esta herramienta de cobertura del código fuente, analiza si un proceso de prueba está probando cada línea de código en una aplicación. Ejecute un programa muchas veces con Cobertura de código habilitada y se mantendrá un contador para cada línea de código a medida que se ejecuta.



```
MAIN

    DEFINE
        r_output STRING,
        r_filename STRING

    LET r_filename="OrderReport"
    LET r_output="Genero Report Viewer"
    OPEN FORM f_configuration FROM "Configuration"
    DISPLAY FORM f_configuration
    INPUT BY NAME r_filename, r_output  ATTRIBUTES (UNBUFFERED,
        BEFORE INPUT
            CALL dialog.setActionActive("saveondisk",r_output !=
        ON ACTION preview
            CALL runReport(r_filename, r_output,"preview")
        ON ACTION saveOnDisk
            CALL runReport(r_filename, r_output,"saveOnDisk")
        ON ACTION quit
            EXIT INPUT
        ON CHANGE r_output
```

Trace – trace es una función de depuración útil que crea un registro completo o filtrado de las llamadas a funciones realizadas por un programa , cuáles fueron los argumentos que se enviaron y qué se devolvió

Bibliotecas compartidas – la arquitectura de implementación se basa en bibliotecas compartidas, lo que elimina la necesidad de compilar 'runners'.

Herramientas para sangrías y refactorización

PRUEBAS AUTOMÁTICAS (GENERO GHOST CLIENT)

El Genero Ghost Client proporciona las siguientes capacidades:

- Realiza pruebas unitarias utilizando escenarios que validan la lógica del negocio.
- Realiza pruebas de carga para determinar cuántos usuarios puede admitir la infraestructura.
- Permite analizar tiempos de respuesta con cargas altas (benchmarks).

GENERADOR DE INFORMES (GENERO REPORT WRITER)

La declaración REPORT en 4GL se utiliza para producir informes de gestión, que según los estándares actuales se ven muy primitivos.

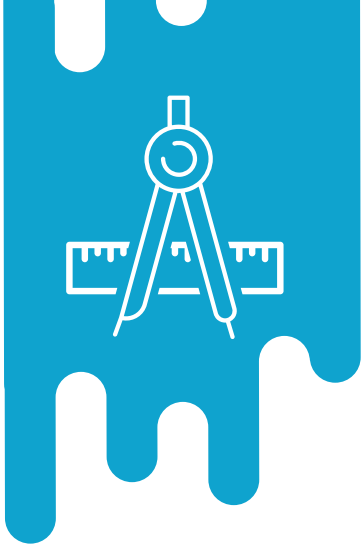
Genero Report Writer proporciona un sofisticado sistema de publicación de informes para producir informes de nivel ejecutivo o comerciales, facturas y etiquetas de envío, por nombrar solo algunos usos. Su particularidad radica en su capacidad para transmitir documentos de gran volumen a impresoras, pantallas o, de hecho, a muchos formatos de archivos diferentes, como: pdf, svg, xlsx, docx, html, ps, jpg y png. La transmisión permite la impresión inmediata de la primera página aún cuando hubiera cientos o incluso miles de páginas para imprimir. Ahorra recursos valiosos del sistema para documentos voluminosos al eliminar la necesidad de crear grandes archivos temporales.

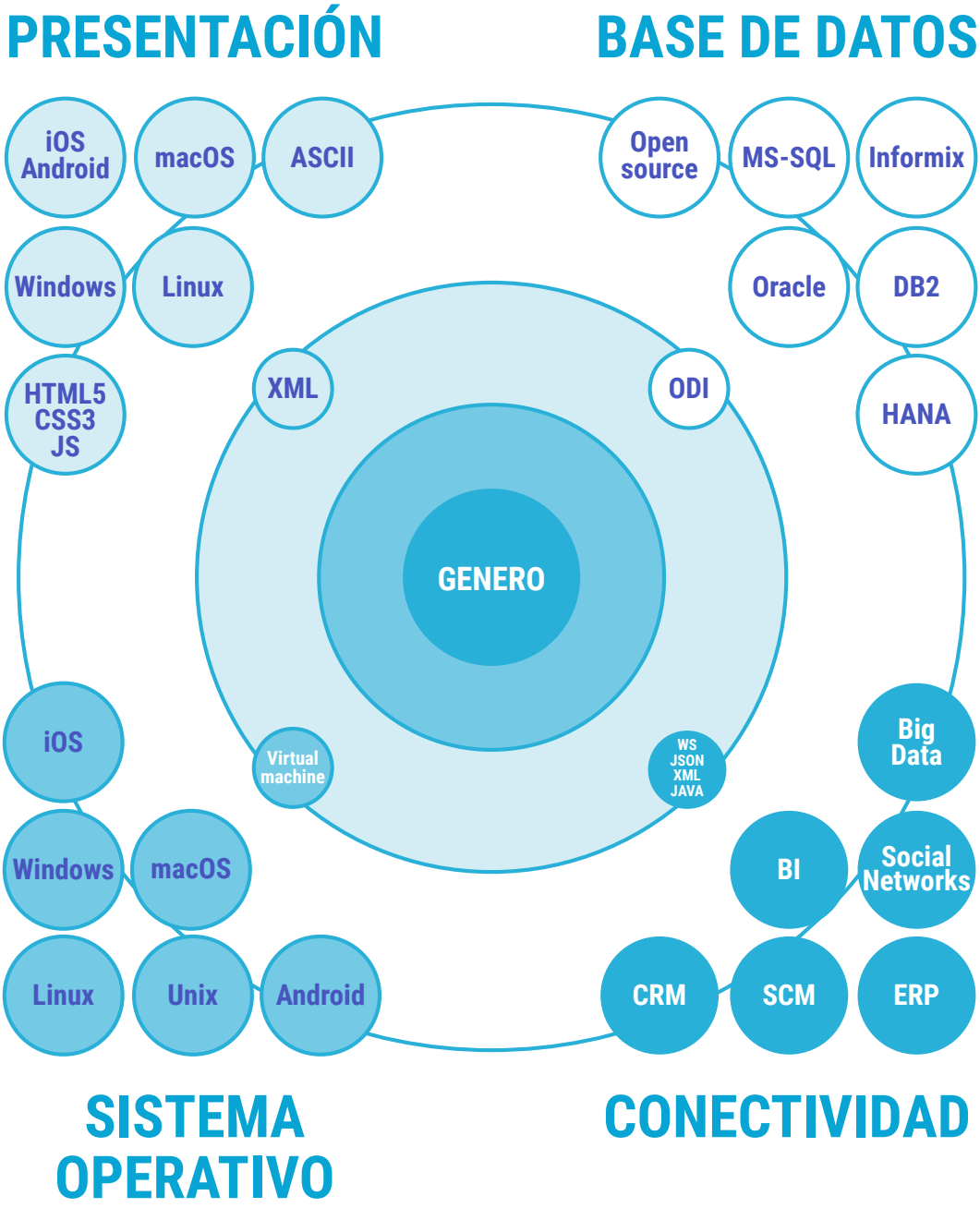
Genero Report Writer es compatible con los objetos gráficos más populares, como imágenes, tablas, códigos de barras, códigos QR y gráficas, así como con el manejo de diferentes tipos de letra, tamaños de puntos y colores. Los diseños son dinámicos para facilitar la transformación de documentos de una forma a otra, sin necesidad de modificar el diseño.

Los informes se crean con Graphical Report Designer, un diseñador casi WYSIWYG que permite modificar los formatos de página de manera simple sin necesidad de recurrir al diseño general. Genero Report Writer se puede usar con otros lenguajes además de Genero, como Java, 'C' y PHP.

Modo de compatibilidad – 'modo de compatibilidad' permite emplear los informes 4GL existentes usando cualquiera de los métodos compatibles con Genero Report Writer.

ARQUITECTURA





Segura, portable y escalable para miles de usuarios sin la necesidad de costoso middleware

OFICINAS EN EL MUNDO

USA & CANADA
Four Js Development Tools Inc.

251 O'Connor Ridge Blvd. Suite 125
Irving, Texas, 75038
United States

Toll free phone: 866 314 7300 (Free)
Phone: (972) 893-7300
Fax: (972) 893-7304

LATIN AMERICA
Four Js Development Tools Latin America.

Insurgentes Sur 1602
4o piso Credito Constructor
CDMX 03940
Mexico

Telefono +52 55 1000 9160
Web: latin.4js.com
Correo: ventas@4js.com

EUROPE
Four Js Development Tools Europe Ltd.

Suite 5, Rineanna House
Shannon Free Zone
Shannon V14 CA36
Co Clare, Ireland

Phone: +353 61 708 314
Fax: +353 61 708 315

ASIA
Four J's Development Tools Asia

Suite 210 29 Kiora Rd,
Miranda NSW, 2228 Sydney
Australia

Phone: +612-8004-5890

FRANCE
Four Js Development Tools SARL.

28 Quai Gallieni
92 150 Suresnes
France

Phone: +33 1 41 38 86 30
Fax: +33 1 41 38 84 46

UNITED KINGDOM
Four Js Development Tools UK LTD.

Regus House, Victory Way Admirals Park
Dartford, DA2 6QD
United Kingdom

Phone: +44 (0) 370 111 5140
Fax: +44 (0) 370 111 5154

CENTRAL EUROPE
Four Js Development Tools Software Vertriebs GmbH

Leonhardsweg 2
82008 Unterhaching
Germany

Phone: +49 89 60815400
Fax: +49 89 60815402

IBERICA
Four Js Development Tools Iberica.

Martinez Villergas 49 1ªPlanta
28027
Madrid

Teléfo: (+34) 910 477 715
Correo: informacion@4js.con

ITALY
Four Js Development Tools Italia

Via Cipriani, 2
42124 Reggio Emilia
Italy

Phone: +39 (0)522 420786
Fax: +39 (0)522 420768

www.4js.com

FOUR Js
The Power of Simplicity