ISSN 0798 1015

REVISTA ESPACIOS

HOME Revista ESPACIOS ÍNDICES / Index A LOS AUTORES / To the AUTORS

Vol. 40 (Nº 28) Año 2019. Pág. 18

Modelo para la actualización tecnológica de los procesos de gestión a través de técnicas de ingeniería impulsada por modelos

Model for the technological upgrading of management processes through techniques of model-driven engineering

CABRERA, Alexis 1 y RAMIREZ, Manuel O. 2

Recibido: 10/05/2019 • Aprobado: 27/07/2019 • Publicado 26/08/2019

Contenido

1. Introducción
2. Enfoque y trabajos relacionados
3. Metodología
4. Propuesta de modelo general
5. Conclusiones
Referencias bibliográficas

RESUMEN:
Muchas empresas están migrando de las aplicaciones de escritorio tradicionales a las aplicaciones web. Estas aplicaciones incluyen hojas de cálculo y sistemas de bases de datos de oficina. Sin embargo, estos sistemas no son extensibles ni escalables y eventualmente se convierten en una gran cantidad de datos difíciles de mantener. Presentamos un modelo de la estructura de aplicaciones de oficina para transformarlo en modelos que describen procesos de negocios. Los modelos obtenidos se utilizarán para la creación futura de varios lenguajes específicos de dominio (DSL), para facilitar la generación automática de aplicaciones web.
Palabras clave: Aplicaciones de oficina, ingeniería dirigida por modelos, aplicaciones web, transformaciones de modelos.

ABSTRACT:
Many companies are migrating from traditional desktop applications to web applications. These applications, include spreadsheets and office database systems, However, these systems are not extensibles neither scalables and eventually become a large store of data difficult to maintain. We present a model from the structure of office applications to transform it in models describing business processes. The models obtained will be used for future creation of several domain specific languages (DSL), to facilitate the automatic generation of web applications.
Keywords: Office applications, model driven engineering, , web applications, model transformations.

1. Introducción

El vertiginoso desarrollo de las nuevas tecnologías ha demandado una innovación continua de las empresas, sobre todo en los procesos de gestión, con el fin de mantener un nivel competitivo. Es evidente que el uso de los avances en nuevas tecnologías ha contribuido a

una mayor eficiencia en todos los procesos de gestión empresarial, jugando un papel importante en la inserción de las organizaciones en la economía global.

Destaca, dentro de los avances más importantes experimentados con el uso de las TIC, el desarrollo de aplicaciones web, con un crecimiento exponencial en los últimos años. Inicialmente solo eran usadas para difundir información, sin embargo, su complejidad ha crecido sustancialmente, éstas están presentes en numerosos dominios siendo usadas con diferentes fines como comercio electrónico, aplicaciones educativas, financieras, etcétera.

Muchas empresas están migrando de las aplicaciones tradicionales de escritorio a aplicaciones web con vistas a aprovechar las ventajas que éstas ofrecen. Desde la perspectiva de Barbier (2011), el proceso de creación de una aplicación web a partir de aplicaciones existentes es un proceso complejo que requiere de mucho tiempo y esfuerzo.

Entre estas aplicaciones están las hojas de cálculo, así como sistemas de bases de datos de oficina, que actúan como simples aplicaciones de gestión de información. Sin embargo, estos sistemas no son extensibles ni escalables y con el tiempo se convierten en un gran almacén de datos difíciles de mantener. Muchas personas desean convertir estas aplicaciones en aplicaciones web dinámicas.

Durante el desarrollo de proyectos de software la especificación de requerimientos y aspectos en la solución prevista es una tarea que consume tiempo, muchas veces por problemas de comunicación entre los programadores y expertos del negocio (United State Patente nº 11/777,588., 2012), es por ello que la modernización de software hacia aplicaciones web debe aprovechar la información que estos sistemas poseen con vistas a agilizar el proceso de desarrollo.

En años recientes, el creciente impacto del enfoque de la ingeniería dirigida por modelos (*MDE*, *Model Driven Engineering*) ha producido un cambio en la perspectiva con que estas propuestas consideran los requerimientos. En tal sentido Molina (2009) sostiene que mediante este enfoque, el proceso de desarrollo es dirigido por modelos y, además, los requerimientos también son representados como modelos, a través del uso de un metamodelo de requerimientos que formalmente define los conceptos y relaciones involucradas en el proceso.

Por otro lado, ha crecido la necesidad de nuevas herramientas que permitan disminuir el tiempo de desarrollo, garantizando la calidad de los mismos. Los programadores necesitan, cada vez más, herramientas que proporcionen un prototipo rápido de aplicaciones. Las modernas técnicas MDE, en especial los lenguajes de dominio específico (DSL, *Domain Specific Languages*), pueden contribuir notablemente a este fin al permitir generar una aplicación básica con los formularios estandarizados para la administración del sistema.

El proceso de evolución del software requiere de la extracción de modelos de los anteriores sistemas con vistas a ser usados en el desarrollo de la nueva aplicación. Muchos enfoques sofisticados han surgido orientados hacia una descripción sistemática y formal de aspectos de una solución. Han evolucionado, en especial para el desarrollo de aplicaciones web, una gran cantidad de investigaciones en el campo de la ingeniería de requerimientos y gestión, así como para el modelado conceptual.

La ingeniería inversa dirigida por modelos (MDRE, Model Driven Reverse *Engineering*) es una metodología basada en MDE cuya clave consiste en producir modelos como salida al procesamiento de código de aplicaciones existentes. Es muy utilizado en procesos de modernización y migración de software. Los modelos obtenidos, generalmente modelos PIM (*Platform Independent Model*), están libres de restricciones tecnológicas para eliminar, en lo mayormente posible, las restricciones del sistema anterior. Los modelos PIM constituyen las entradas para los procesos de generación de la nueva aplicación (Bruneliere, 2010).

La necesidad de brindar a los programadores una herramienta para agilizar la creación de sistemas, y que muchas veces éstas surgen a partir de la automatización de procesos que han sido administrados a través de hojas de cálculo, bases de datos, etc., abre un nuevo campo en el área de la ingeniería dirigida por modelos, surgiendo la necesidad de crear nuevas herramientas que permitan crear aplicaciones de manera rápida y eficiente, a partir de dichas aplicaciones.

En la presente investigación se propone definir un modelo principal a partir de la estructura de las aplicaciones de oficina para su transformación en modelos que describan los procesos del negocio. Los modelos obtenidos servirán para la futura creación de varios lenguajes de dominio específicos (*DSL, Domain Specific languages*), que faciliten la generación de aplicaciones web.

---

2. Enfoque y trabajos relacionados.

Varios autores han abordado el modelado de software aplicado a hojas de cálculo, Pozle (2011) describe un método que provee una interfaz de tipo hoja de cálculo para bases de datos relacionales. El método parte de extraer meta-datos de la estructura de la hoja de cálculo, estableciendo una conexión bidireccional con una base de datos.

El proyecto 2LT J. Visser. Comput. Sci. (2008) utiliza una estrategia de dos niveles para reescribir varias transformaciones de esquemas de datos, instancias, consultas y restricciones, en particular, orientada a mapeos de datos con relaciones jerárquicas.

Por su parte Cunha , Fernandes, Mendes, Pacheco, & Saraiva. (2012) exponen técnicas basadas en ingeniería dirigida por modelos para hojas de cálculo donde emplea transformaciones bidireccionales para mantener los modelos de hojas de cálculo y sus instancias sincronizadas. La lógica del negocio de las hojas de cálculo se define a través de modelos de *ClassSheet* a los cuales se ajustan los datos de las hojas de cálculo, y los usuarios pueden evolucionar tanto el modelo como las instancias de datos. Usa técnicas basadas en el *framework MDSheet*, una extensión de los sistemas de hoja de cálculo tradicionales.

Adicionalmente Fernandes, cunha, mendes, & Pereira. (2015) propone un lenguaje de consulta para hojas de cálculo dirigidas por modelos. Se presenta en detalles el diseño y la implementación del lenguaje, desde la de normalización de datos y traducción del lenguaje, hasta una consulta más eficiente.

En otro orden de ideas Paige, Dimitrios S, & Matragkas. (2014) se enfocan en el tratamiento de las hojas de cálculo como modelos, con el fin de apoyar el ciclo de vida del sistema, y para proporcionar una ruta de migración a seguir adecuada para las organizaciones en la maduración del uso de técnicas de modelado.

Cunha, Saraiva, & Visser, From spreadsheets to relational databases and back. (2009) definen reglas de refinamiento de datos para la conversión de los tipos de datos de Excel a tipos de datos de bases de datos relacionales.

---

3. Metodología

A partir de los criterios antes mencionados, la investigación propone un análisis a partir de los datos almacenados en la hoja de cálculo, y su presentación a través de una herramienta visual donde puede ser cambiado si fuese necesario. El enfoque va más allá del tipo de datos y pretende predefinir aspectos para validación y presentación en formularios como el tipo de control, ayuda, entre otros aspectos.

Se ha implementado dos modelos que definen la información a extraer de las hojas de cálculo y aplicaciones de base de datos. Estos modelos deben permitir obtener la información que se necesita para obtener los modelos de negocio y la definición del diagrama entidad-relación. Se han desarrollado dos herramientas para leer, optimizar y consultar hojas de cálculo y aplicaciones de bases de datos.

Las herramientas consultan las hojas de cálculo y bases de datos Access y transforman la información en un formato JSON o XML. Los objetos como tablas y campos son la base fundamental para poder definir las entidades y sus atributos, así como los tipos de datos de estos últimos. A través de una interfaz visual será posible detectar ambigüedades y poder aplicar técnicas como normalización para crear nuevas entidades y sus relaciones entre sí.

3.1. Modelo de análisis de hojas de cálculo.

El uso de técnicas de modelado (MDE) está orientado principalmente a definir aspectos de la aplicación usando un lenguaje de alto nivel y generar la aplicación web corriendo de manera automática, agregando agilidad y productividad al proceso de desarrollo. En esto se basa la ingeniería dirigida por modelos que permite a los desarrolladores especificar semánticamente aspectos relevantes en lugar de codificar. Rivero, Grigera, & Rossi (2014). En este trabajo se definen inicialmente cuáles son los aspectos que se necesitan extraer de aplicaciones anteriores para definir el modelo a crear.

En el artículo se propone un modelo para hojas de cálculo, Figura. 1, que comienza por detectar las hojas presentes en el archivo, su estructura, posibles entidades y sus relaciones. Así como identificar la integridad referencial y cardinalidad de las mismas.

También es necesario analizar la información de las posibles entidades detectadas dentro de una hoja, y aplicar la normalización cuando sea necesario. La normalización ayuda a perfeccionar las definiciones de datos para eliminar grupos repetidos y dependencias innecesarias. Cada tabla debe describir sólo un tipo de entidad (como una persona, un lugar, un pedido de cliente o un producto) (Figura. 1.1).

Los campos deben ser analizados para detectar sus atributos generales como son: si es clave primaria, si es único, si tiene un valor por defecto, su tipo de codificación y si es auto numérico. Los atributos más importantes se han colocado aparte porque determinan la validación, estos son: el tipo de dato que almacena el campo, el tamaño del campo y si es requerido.

El tipo de dato no solo es clave para la validación, sino también para el futuro diseño de la interfaz de formularios. Estos determinan el tipo de control de formulario, o posible componente a utilizar para el ingreso de los datos. También los tipos de datos *Enum* o listas determinan si es necesario crear nuevas entidades o tablas, y también el tipo de control asociado a éste.

Figura 1
Modelo de análisis
de una hoja de cálculo

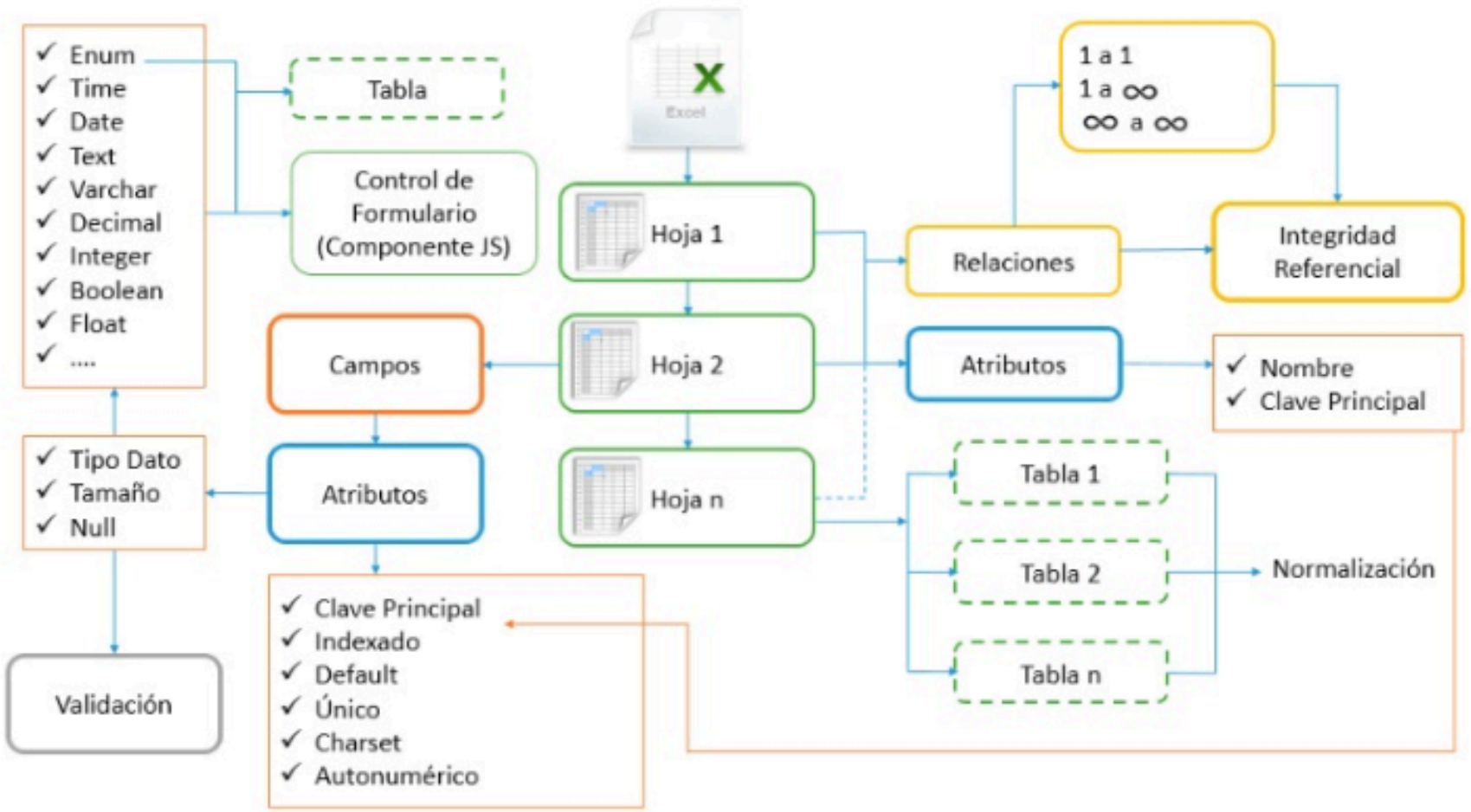


Figura 1.1
Ejemplo de Aplicación en
MS Excel, tabla productos

	A	B	C	D	E	F	G	H	I	J	K
1					TALLAS						
2	CODIGO	PRODUCTO	Color	STOCK	XS	S	M	L	XL	XXL	3XL
3	2040815	BASICA CLASSIC	SURTIDOS	309		85	117	77	30		
4	1052925	BATOLA DARLENE	NEGRO	2					2		
5	1062622	BATOLA LADY	VERDE LIMON	90		21	35	11	23		
6	1062623	BATOLA LAYLA	NEGRO	92		18	37	13	24		
7	1042915	BATOLA YESLY	AGUA MARINA	3		2			1		
8	2060006	BERMUDA PABLO	VERDE ACEITUN	57		7	18	12	16	4	
9	1032703	BICICLETERO NAYELY	BEIGE	70		25	27	14	4		
10	1032703	BICICLETERO NAYELY	NEGRO	68		23	20	17	8		
11	1050147	BLUSA AGAR	BLANCO	3		3					
12	1050170	BLUSA ADELINE	BLANCO	2		1	1				
13	1060178	BLUSA ADRIANA	ROSADO	34		1	13	7	13		
14	1060177	BLUSA ALESSIA	FUCSIA	14		4	4	2	4		

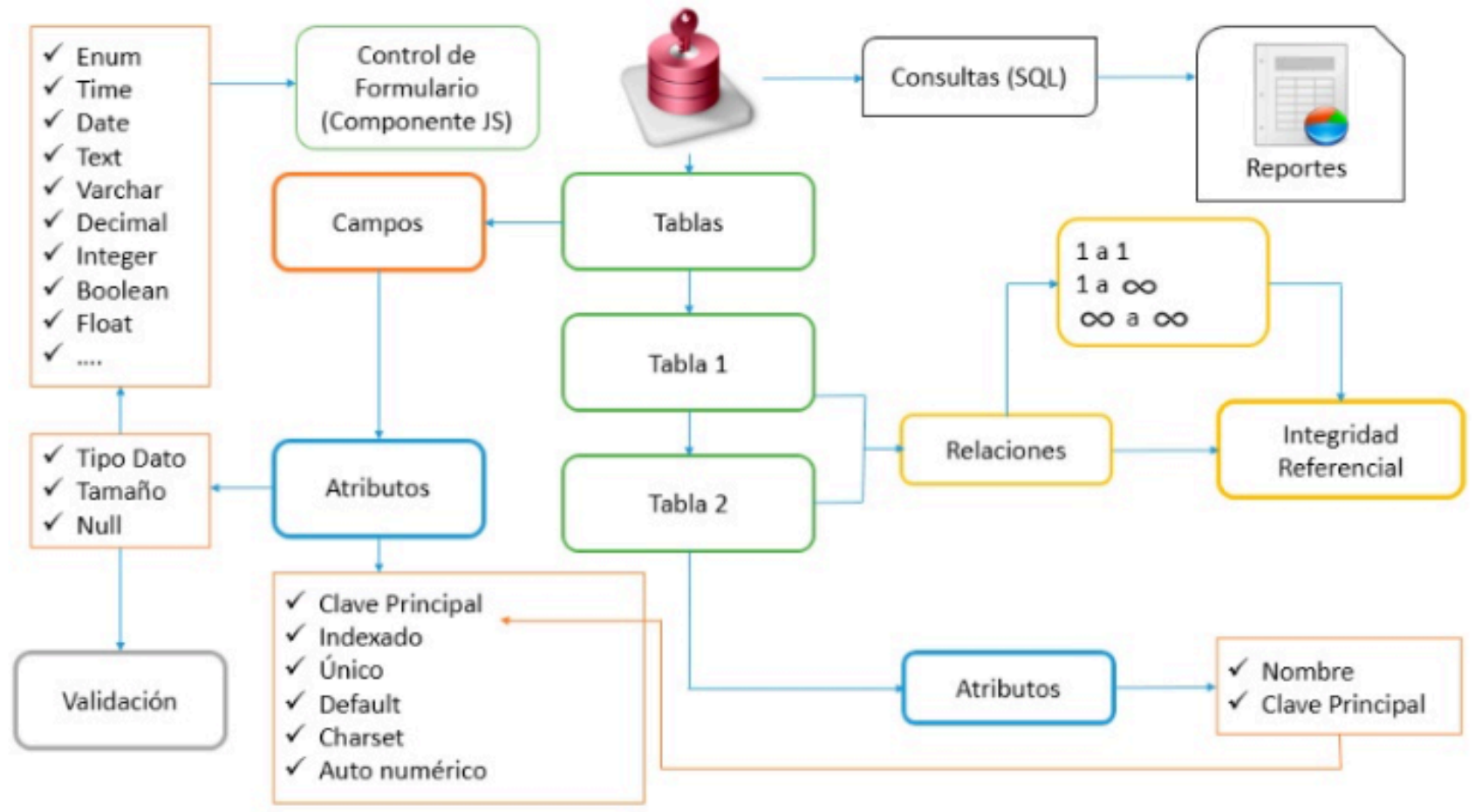
3.2. Modelo de análisis de aplicaciones MS Access.

El análisis de aplicaciones de bases de datos como MS Access se acerca un poco más al concepto de aplicación. Las bases de datos bien diseñadas proporcionan la información básica necesaria como son las tablas, sus relaciones, campos y sus atributos, validaciones, etcétera. No obstante, es necesario analizar las mismas para extraer toda la información necesaria, e incluso tratar de mejorar su diseño.

La Fig. 2 muestra el modelo de análisis planteado para bases de datos. Inicialmente se identifican las tablas, sus relaciones y la integridad referencial de las mismas. Así mismo, se identifican los campos, sus atributos como: si es clave primaria, si es único, si es indexado, si tiene valor por defecto, auto numérico y codificación. Al igual que en el modelo anterior, los atributos más importantes, determinan la validación; y los tipos de datos, los controles de formulario para el ingreso de los datos

Figura 2

Modelo de análisis de una aplicación
de base de datos MS Access.



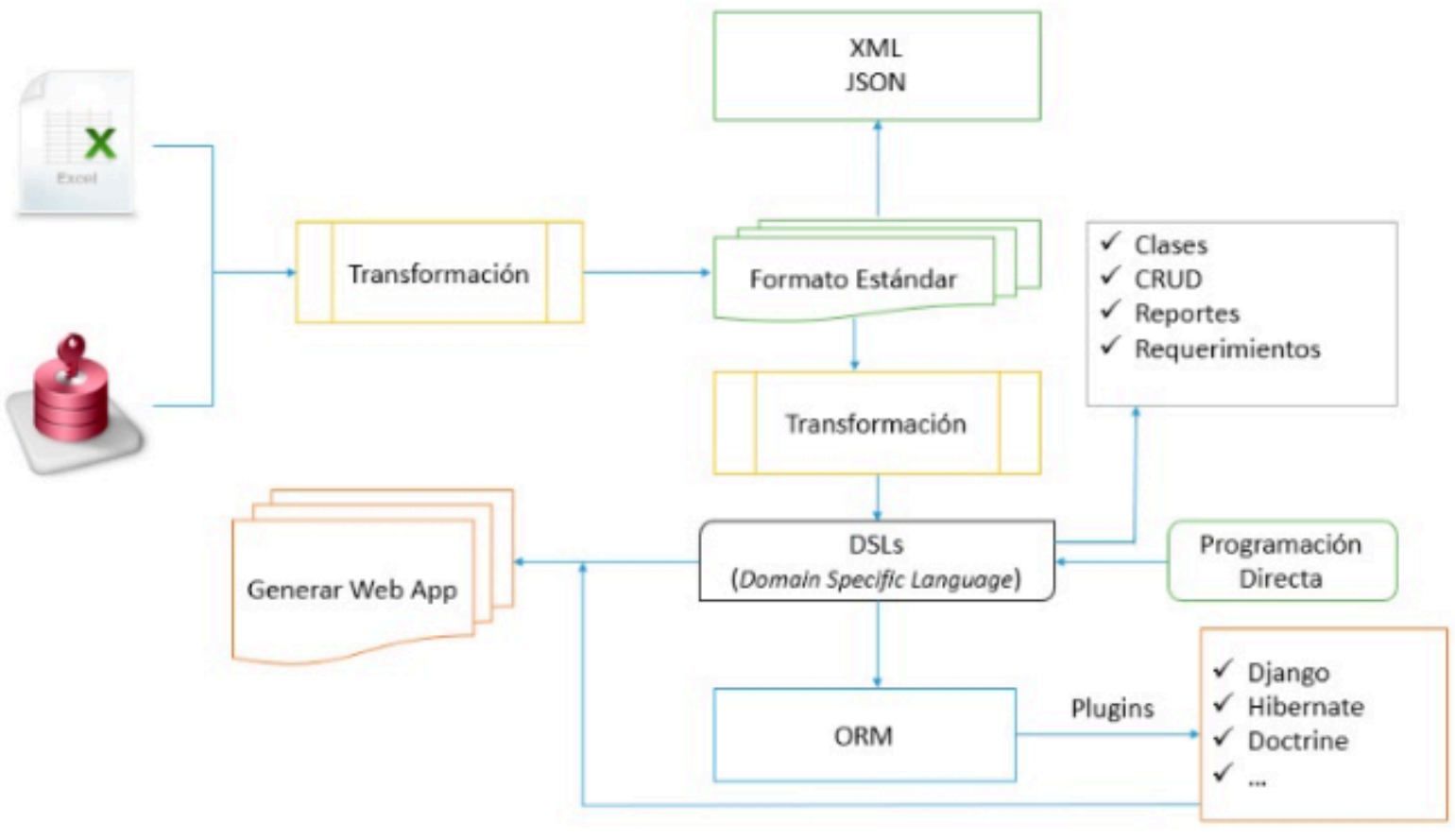
Es importante señalar que las aplicaciones de bases de datos utilizan el lenguaje SQL para la creación de consultas y reportes. En el modelo propuesto se obtienen estas consultas en formato SQL para aprovecharlas en futuros reportes de la nueva aplicación.

---

4. Propuesta de modelo general.

El modelo propuesto para aplicaciones de oficina (Figura. 3) debe proporcionar la información necesaria para transformar estas aplicaciones en un formato estándar común para ambos tipos de aplicaciones. El formato estándar, no solo proporcionará información para el diseño de la aplicación, sino también los datos almacenados en éstas para su extracción, transformación e ingreso a la futura aplicación que se desea crear.

Figura 3
Modelo de transformaciones propuesto.



A partir de los modelos propuestos anteriormente, se propone un modelo general para la trasformación de estas aplicaciones de oficina en un formato estándar, independiente de la plataforma. Se proponen los formatos XML (Figura. 5) y JSON (Figura. 6), ambos formatos pueden ser interpretados y transformados a una serie de lenguajes de dominio específico (DSL) que servirán de base para la creación de la aplicación web.

El formato estándar, almacena los datos (Figura. 4) y a su vez proporciona la información necesaria para la creación de los DLS (Figura. 6). Estos DSL crean los modelos para las clases, entidades, requerimientos y reportes.

El modelo propuesto permite generar, a partir de la interpretación de los DSL, una aplicación web con implementaciones para las funcionalidades básicas, CRUD (*Create, Update, Delete*), así como reportes obtenidos de las aplicaciones de oficina. Igualmente se propone la generación de modelos para diferentes plataformas.

La propuesta de la presente investigación intenta llegar a un enfoque de generación de código que permita a los desarrolladores crear aplicaciones corriendo de forma automática con las funcionalidades básicas, y de ser posible con reportes predefinidos en los modelos.

Por otra parte, muchos *frameworks* para desarrollo de aplicaciones web implementan técnicas de ingeniería dirigida por modelos, entre ellos se destacan Ruby on Rails, Django, Symfony, entre otros.

Otras de las propuestas van encaminadas a crear modelos para estos *frameworks*. A través de *plugins* que permitan interpretar los DSLs, se pueden obtener diferentes ORMs (*Object Relational Mapping*) como *Django*, *Hibernate*, *Doctrine*, entre otros. Los modelos obtenidos serán la base para generar aplicaciones web en diferentes plataformas.

Figura 4
Ejemplo de formato JSON con los
datos de la aplicación anterior

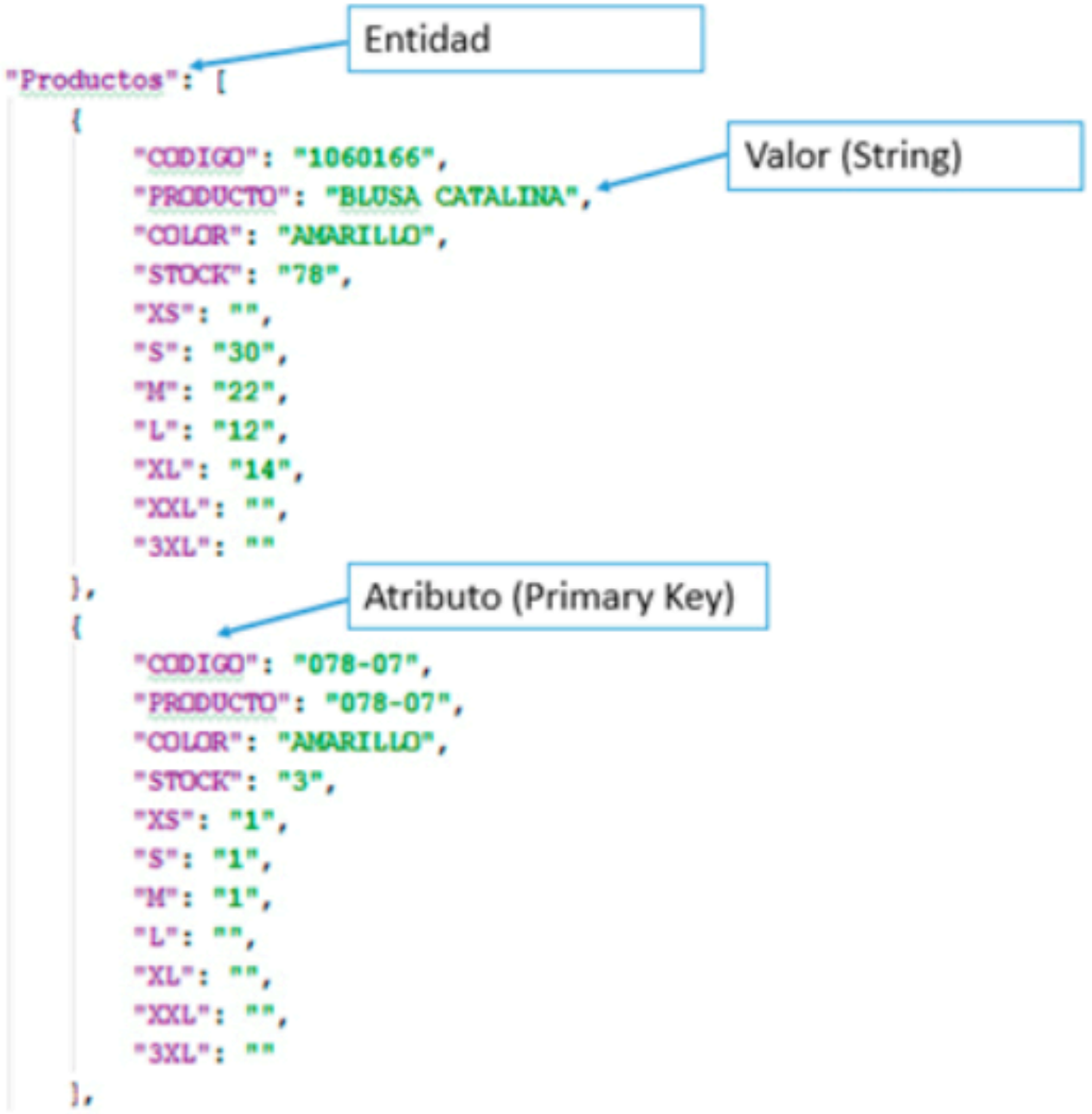


Figura 5
Ejemplo de formato XML
con la definición del modelo

```
<productos orderby="codigo">
  <codigo>
    <require>True</require>
    <primaryKey>True</primaryKey>
    <dataType>Varchar</dataType>
    <maxLength>30</maxLength>
    <minLength>6</minLength>
    <control>textInput</control>
    <charset>UTF-8</charset>
    <helpText> código...</helpText>
  </codigo>
  <producto>
    <require>True</require>
    <dataType>Varchar</dataType>
    <maxLength>50</maxLength>
    <control>textInput</control>
    <unique>True</unique>
    <charset>UTF-8</charset>
    <helpText>producto...</helpText>
  </producto>
  <color>
    <require>True</require>
    <dataType>Integer</dataType>
    <control>select</control>
   <foreignKey>tblColor</foreignKey>
    <index>true</index>
    <helpText> color...</helpText>
  </color>
</productos>
```

Figura 6
Ejemplo de formato JSON
con la definición del modelo

```
{
 "Productos": [ {
    "CODIGO":
     {
      "require": true,
      "primaryKey": true,
      "dataType": "Varchar",
      "maxLength": 30,
      "minLength": 6,
      "control": "textInput",
      "helpText": "código...",
      "order": "asc"
     },
     "PRODUCTO":
     {
      "require": true,
      "unique": true,
      "dataType": "Varchar",
      "maxLength": 50,
      "control": "textInput",
      "charset": "UTF-8",
      "helpText": "producto..."
     },
      "COLOR":
      {
       "require": true,
       "dataType": "Integer",
       "control": "select",
       "foreignKey": "tblColor",
```


Los atributos definidos para cada campo proporcionan la información para el diseño del modelo entidad relación, la creación de la base de datos y la información requerida para la validación de datos y generación de formularios HTML.

5. Conclusiones

El análisis y transformación de aplicaciones de oficina, en formato estándar, es un proceso complejo, éste requiere definir claramente las características mínimas que deben cumplir las aplicaciones objeto de actualización.

Es necesario definir con detalle los atributos de cada campo, así como los diferentes valores que se van a utilizar. Estos son la base del modelo utilizado posteriormente para la creación de los DSLs. En la presente investigación se muestra algunos ejemplos del formato para la definición de los modelos.

Los modelos propuestos cumplen con los requisitos para convertir una aplicación de oficina en un modelo estándar que sirva como base para crear modelos para diferentes frameworks como django, entre otros.

El análisis de aplicaciones de oficina, para su transformación en un modelo estándar, requiere definir parámetros que permitan establecer los requerimientos necesarios para el éxito del proceso. Los modelos estándares obtenidos deben ser transformados en varios lenguajes de dominio específico.

El próximo paso es definir la sintaxis concreta del lenguaje de dominio específico como base para el proceso de creación de modelos que puedan ser transformados en otros modelos para diferentes *Frameworks* existentes actualmente.

Referencias bibliográficas

Barbier, F. D. (2011). Model Driven Reverse Engineering. *Increasing Legacy Technology Independence., 125*.

Bruneliere, H. (2010). MoDisco: a generic and extensible framework for model driven reverse engineering. . *IEEE/ACM international conference on automated software engineering.* (págs. 173-174). New York: ACM.

Cunha , J., Fernandes, J., Mendes, J., Pacheco, H., & Saraiva, J. (2012). Bidirectional Transformation of Model-Driven Spreadsheets. *Quinta Conferencia Internacional* (págs. 28-29). Praga, República Checa: Springer.

Cunha, J., Saraiva, J., & Visser, J. (2009). From spreadsheets to relational databases and back. *Partial evaluation and program manipulation* (págs. 179-188). New York: ACM.

Fernandes, P., cunha, J., mendes, J., & Pereira, R. (2015). Design and implementation of queries for model-driven spreadsheets. Central European Functional Programming School. *Springer International*, 459-478.

Frost, B. H. (2012). *United State Patente nº 11/777,588.*

J. Visser. Comput. Sci., 2.–2. 2. (2008). Coupled transformation of schemas, documents, queries, and constraints. *Electr. Notes Theor. 200*, págs. 3-23. Elsevier.

Molina, F. a. (2009). "Integrating usability requirements that can be evaluated in design time into Model Driven Engineering of Web Information Systems. *Advances in Engineering Software*, 1306-1317.

Paige , R., Dimitrios S, K., & Matragkas, N. (2014). Spreadsheets are Models Too Position Statement. (págs. 9-10). The York research database.

Pozle, A. (2011). Actualizaciones dinámicas de componentes gráficos en .NET Framework. *IEEE Xplore* (págs. 322-330). Berna, Suiza: IEEE.

Rivero, J., Grigera, J., Rossi, G., Robles, E., Montero, F., & Gaedke, M. (2014). Mockup-Driven Development:Providing agile support for Model-Driven Web Engineering. . *Information & Software Technology*, 670-687.

---

1. Gerente general empresa Creativos SA. Master en Tecnologías para la Educación, alexiscm72@hotmail.com

2. Docente investigador Universidad Ecotec, (Ecuador)Facultad de Sistemas y Telecomunicaciones, Master en Nuevas Tecnologías para la Educación, mramirez@ecotec.edu.ec

---

Revista ESPACIOS. ISSN 0798 1015
Vol. 40 (Nº 28) Año 2019

[Índice]