https://doi.org/10.22201/dgtic.ctud.2023.1.1.16

Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 52 - 61

Implementación de la técnica de Mapeo Objeto-Relacional en el desarrollo de sistemas

Información del reporte:

Para citar este reporte técnico:

Ortega Ramírez, C. R. (2023). Implementación de la técnica de Mapeo Objeto-Relacional en el desarrollo de sistemas. *Cuadernos Técnicos Universitarios de la DGTIC, 1* (1), páginas (52 - 61). https://doi.org/10.22201/dgtic.ctud.2023.1.1.16

Cristian Ricardo Ortega Ramírez

Dirección General de Cómputo y de Tecnologías de Información y Comunicación

Universidad Nacional Autónoma de México

cristian.ortega@comunidad.unam.mx

ORCID: 0009-0003-1073-6492

Resumen:

Se abordan los principales conceptos implicados en la técnica *ORM* (*Object Relational Mapping*). Se describe el proceso realizado para su implementación en el desarrollo de la segunda versión del Sistema Integral de Personal Académico con el *framework Symfony* y la librería *Doctrine*, mediante su interfaz de línea de comandos. La incorporación de esta técnica brindó mayor flexibilidad a la hora de manipular los datos, y permitió la reducción del tiempo empleado para la codificación del sistema en aproximadamente un 37% con respecto a la primera versión.

Palabras clave:

Mapeo Objeto-Relacional, programación orientada a objetos, bases de datos, Symfony, Doctrine.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.16 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 53 - 61

1. INTRODUCCIÓN

Un sistema de información es comúnmente descrito como un conjunto de componentes que interactúan entre sí para recoger, procesar, almacenar y proveer la información necesaria para apoyar la toma de decisiones y el control de una organización. Entre los componentes que conforman un sistema de información destacan la tecnología de procesos, que viene determinada principalmente por los lenguajes y técnicas de programación utilizadas, así como la tecnología empleada para la gestión de los datos (Laudon y Laudon, 2006; Whitten, Bentley y Dittman, 2004, citados por Fernández Alarcón, 2006, p. 13).

Gómez-Díaz et al. (2022) sostienen que uno de los principales desafíos de los sistemas de información, refiere a la complejidad de alinear el código de programación con las estructuras de la base de datos, ya que un desarrollador además de dominar el lenguaje de programación empleado, deberá de comprender y aplicar mediante el lenguaje de consulta estructurado (*SQL*), las sentencias y el código necesarios para la conexión e interacción de su aplicación con una base de datos.

Es así que cobra especial relevancia la implementación de una técnica que permita mapear las estructuras de base de datos sobre una estructura lógica de objetos, con el fin de simplificar, y acelerar el desarrollo del sistema, permitir a los desarrolladores "liberarse" de la escritura o generación manual de código *SQL* y realizar las operaciones *CRUD* (creación, recuperación, actualización y eliminación, por sus siglas en inglés) sobre los datos de forma indirecta (Muro, 2023).

La primera versión del Sistema Integral de Personal Académico fue desarrollada por un equipo de trabajo conformado por un solo programador y un diseñador. Posteriormente, nuevos requerimientos que impactaron de forma importante en la estructura de datos dio como resultado una nueva versión del SIPA, incorporando al equipo de trabajo a un especialista en bases de datos con la finalidad de implementar un mecanismo que brindara mayor flexibilidad a los procesos de manipulación e interacción con la estructura de la base y los datos en sí mismos para acelerar el desarrollo del sistema. Adicionalmente, se integró al equipo a un especialista en *DevOps*[1] para agilizar el despliegue del sistema en entornos de desarrollo, pruebas y producción.

A continuación, se describe el proceso de implementación de la técnica *ORM* para el desarrollo de la segunda versión del SIPA, misma que contribuyó a proveer a los desarrolladores con una capa de abstracción que encapsula la lógica de los datos, lo que facilita su gestión a través de objetos, así como la reutilización de código, para acelerar el desarrollo y puesta en marcha del sistema.

2. OBJETIVO

Reportar los resultados y beneficios obtenidos al implementar la técnica de Mapeo Objeto-Relacional en el desarrollo de la segunda versión del SIPA.

[1] El ingeniero de *DevOps* incorpora procesos, herramientas y metodologías para equilibrar las necesidades durante todo el ciclo de vida del desarrollo de *software*, desde la programación y la implementación hasta el mantenimiento y las actualizaciones

https://doi.org/10.22201/dgtic.ctud.2023.1.1.16 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 54 - 61

3. DESARROLLO

3.1 SELECCIÓN DE TECNOLOGÍA

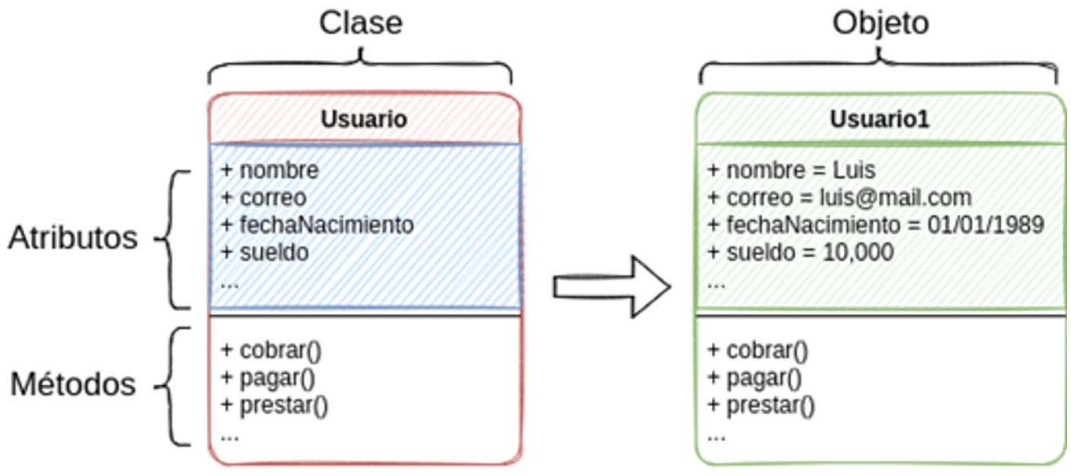
3.1.1 LENGUAJE DE PROGRAMACIÓN

Para la codificación del sistema se optó por utilizar *PHP*, lenguaje empleado en la versión previa del SIPA y con el que el programador principal tenía mayor experiencia, además de ser el lenguaje de programación del lado del servidor más utilizado en sistemas web (Web Technology Surveys [W3Techs], s. f.). El lenguaje de programación *PHP* está basado en el paradigma de programación orientada a objetos (*POO*), con un gran impacto en el desarrollo de sistemas en tanto que facilita la simplificación de la escritura, mantenimiento y reutilización de código. Asimismo, *PHP* permite representar un programa como una colección de objetos que colaboran entre sí para realizar determinadas tareas. Se entiende por objeto a una entidad que se encuentra en situaciones o problemas de nuestro mundo real, y que está conformado tanto por atributos que representan sus datos o características, como por métodos que responden a las acciones u operaciones que puede realizar. La representación abstracta de una colección de objetos que comparten una misma estructura y un mismo comportamiento se denomina clase, y es a partir de ella que el objeto se materializa (Pérez Montero y Hernández Pérez, 2019).

En la figura 1 se muestra una representación visual de una clase y un objeto derivado de dicha clase, ambos con sus correspondientes atributos y métodos.

Figura 1

Representación visual de una clase y un objeto



https://doi.org/10.22201/dgtic.ctud.2023.1.1.16 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 55 - 61

3.1.2 FRAMEWORK PHP

Se optó por el uso del *framework*[2] *Symfony*, debido a su capacidad para realizar actualizaciones de forma segura en el esquema de la base de datos a través de migraciones, es decir, de bloques de código generados de manera automática por el framework al comparar la definición de las clases con la estructura actual real de la base de datos (Altube Vera, 2021).

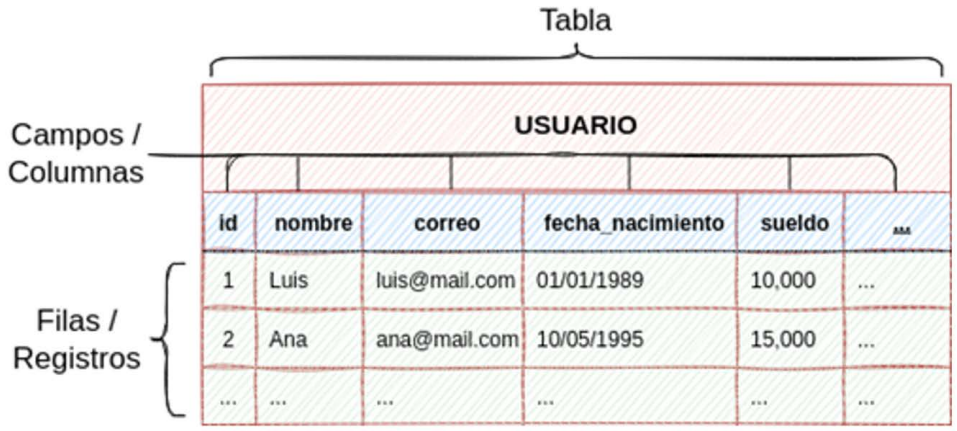
3.1.3 SISTEMA MANEJADOR DE BASES DE DATOS (SMBD)

Como los sistemas de información requieren almacenar datos relacionados entre sí de una manera estructurada es imprescindible el uso de un Sistema Manejador de Bases de Datos. De acuerdo con el DB-Engines (2023), el grupo de las bases de datos relacionales es el más consolidado del sector de las TIC y el de mayor uso hoy en día. Una base de datos relacional estructura información en tablas conformadas por campos y filas, además de relacionar dicha información mediante la unión de tablas, lo que facilita la comprensión sobre la relación entre varios datos (Google, s. f.).

En la figura 2 se muestra la representación visual de una tabla con sus correspondientes campos y filas.

Figura 2

Representación visual de una tabla en una base de datos



La herramienta a utilizar para la gestión de los datos fue el *SMBD* el cual se empleé en la primera versión del sistema. Dado que ya se contaba con información almacenada del SIPA en su primera versión, utilizar el mismo SMBD permitió acelerar el tiempo de migración de los datos existentes, y garantizar su integridad. Dicha migración de datos implicó ejecutar un proceso de *ETL* (Extracción, Transformación y Carga, por sus siglas en inglés) que se llevó a cabo paralelamente a la codificación del sistema.

[2] Un *framework* es un marco de trabajo que proporciona una estructura base para desarrollar un proyecto con objetivos específicos, en otras palabras, es un tipo de plantilla que sirve como punto de partida para organizar y crear *software*.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.16 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 56 - 61

3.1.4 LIBRERÍA *ORM*

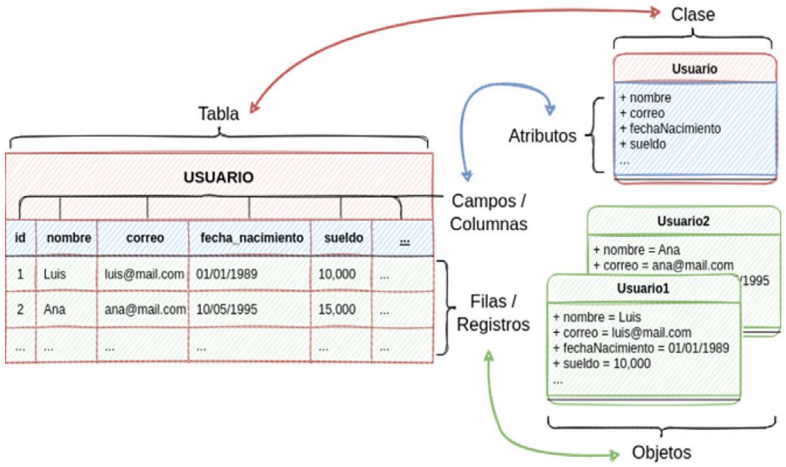
Para la gestión y acceso a los datos se implementó la técnica *ORM*, que brinda mejores niveles de abstracción respecto a otras técnicas de acceso a bases de datos. Existen dos patrones de diseño utilizados en el Mapeo Objeto-Relacional (*Active Record* y *Data Mapper*), y si bien ambos patrones tienen el mismo propósito, difieren en la forma en que abordan la asignación y la responsabilidad de las operaciones de base de datos (Kolodka, 2023).

Finalmente, para la implementación de la técnica *ORM* se decidió utilizar la librería *Doctrine*, la cual utiliza el patrón *Data Mapper* y es la utilizada por defecto para un proyecto creado con *Symfony*, lo cual permite aprovechar el amplio abanico de herramientas que ofrece para la implementación de dicha librería. Asimismo, se consideró la experiencia de los nuevos integrantes del equipo de trabajo en el uso de otras librerías que utilizan el patrón *Data Mapper*.

Doctrine permite realizar un Mapeo Objeto-Relacional al trasladar los datos de una base de datos relacional a un sistema de clases y de objetos, donde las tablas, campos y filas serían el equivalente a clases, atributos y objetos, respectivamente. En la figura 3 se muestra una representación visual del mapeo realizado por un *ORM*.

Figura 3

Representación visual de un Mapeo Objeto-Relacional



3.2 IMPLEMENTACIÓN DE LA TÉCNICA *ORM*

Los cambios en la funcionalidad requerida para la segunda versión del SIPA implicaron realizar una reestructura de la base de datos original, mediante la implementación de *Doctrine* desde un enfoque *Model First* para generar un Modelo Entidad-Relación que cubriera dichos requerimientos, lo que permitió trabajar de forma paralela la migración de los datos existentes y la codificación del sistema.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.16 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 57 - 61

Cabe señalar que el enfoque *Model First* consiste en realizar, en primera instancia, el modelado de la aplicación mediante alguna herramienta como *UML*, y posteriormente generar la estructura de la base de datos y el código a partir de dicho modelo.

Para iniciar con la implementación del *ORM*, se debe crear un nuevo proyecto *Symfony* que, a su vez, precisa de la instalación previa de la paquetería[3] que requiere el *framework* en el entorno de desarrollo. A continuación, se integran las instrucciones que generarán la estructura de un nuevo proyecto *Symfony,* a través de la *CLI* (*Command-Line Interface*):

```
$ composer create-project symfony/skeleton[:"<symfony_version>"] <app_name>
$ cd <app_name>
$ composer require webapp
```

Una vez creada la estructura del proyecto *Symfony,* se procede a la instalación de la librería *Doctrine* y del paquete "*symfony/maker-bundle*", para dar inicio a la generación asistida de código para la implementación del *ORM*:

```
$ composer require symfony/orm-pack
$ composer require --dev symfony/maker-bundle
```

Finalmente, se configura la conexión con el *SMBD* lo que implica editar el archivo "*.env*" para agregar a la siguiente línea sustituyendo cada campo con los valores correspondientes:

```
DATABASE_URL="<smbd_name>://<db_user>:<db_password>@<db_host>:<db_port>/<db_name>?serverVersion=<smbd_version>"
```

En este punto, se puede proceder con la implementación del *ORM* partiendo del modelo diseñado para el sistema, como se muestra en la figura 4.

Figura 4

Modelo Entidad-Relación

[3] Los requerimientos técnicos para comenzar a trabajar con *Symfony* son: *PHP* versión 8.1 o superior; las librerías *Ctype, iconv, PCRE, Session, SimpleXML* y *Tokenier*; *Composer* para el manejo de dependencias para *PHP*.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.16 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 58 - 61

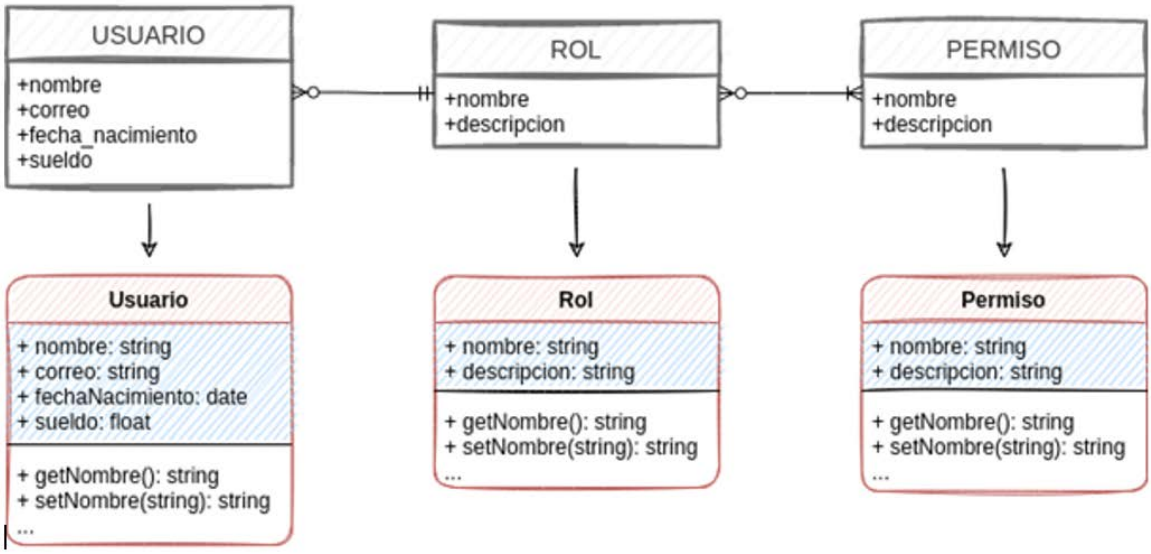
Para comenzar a generar el código *PHP* de las clases con sus correspondientes atributos, se ejecuta un comando proporcionado por el paquete "*symfony/maker-bundle*":

```
$ php bin/console make:entity
```

Una vez ejecutado el comando, un asistente será el encargado de guiar la generación del código de una *Clase PHP* que servirá para mapear dicha entidad, lo que requiere especificar: nombre de la clase, nombre de los atributos, tipo de dato a utilizar para cada atributo, longitud (para los tipos de datos que apliquen) e indicar si el valor del atributo puede ser nulo en la base de datos. En la figura 5, se muestra una representación de las clases generadas a partir de las entidades del modelo.

Figura 5

Representación visual de las clases generadas a partir de las entidades del modelo



Una vez creadas las clases para cada entidad es necesario crear sus relaciones considerando los requerimientos del sistema y preponderando si se deben mapear uno o los dos sentidos que se integran en una relación.

Por ejemplo, en el modelo mostrado, existe la entidad *USUARIO* que se relaciona con la entidad *ROL*, vista la relación desde la primera entidad hacia la segunda, significa que un *USUARIO* tiene un *ROL* asignado, pero si se observa la relación en el sentido inverso, implica que un *ROL* puede estar asignado a uno o más *USUARIOS*. Entonces, si los requerimientos del sistema indican que es preciso conocer los usuarios y sus roles asignados, convendrá mapear la relación de *USUARIO* a *ROL*, en el caso contrario, si se deben conocer los usuarios a los que se ha asignado cada rol, sería apropiado mapear la relación desde el *ROL* hacía *USUARIO*; otro escenario posible es aquel en que las dos situaciones sean requerimientos del sistema, en cuyo caso mapear ambos sentidos de la relación será lo indicado.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.16 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 59 - 61

Para generar las relaciones, se ejecuta nuevamente el comando previo y cuando el asistente pregunta el nombre de la clase que se desea crear, se ingresa el nombre de la clase existente para modificarla; posteriormente se ingresa el nombre del atributo que servirá para mapear la relación, y se especifica como tipo de dato *relation*. En este punto el asistente preguntará por el nombre de la entidad con la que se desea relacionar la entidad actual, tras lo cual se indica el tipo de relación (muchos a uno, uno a muchos, muchos a muchos o uno a uno); opcionalmente el asistente permitirá crear la relación inversa.

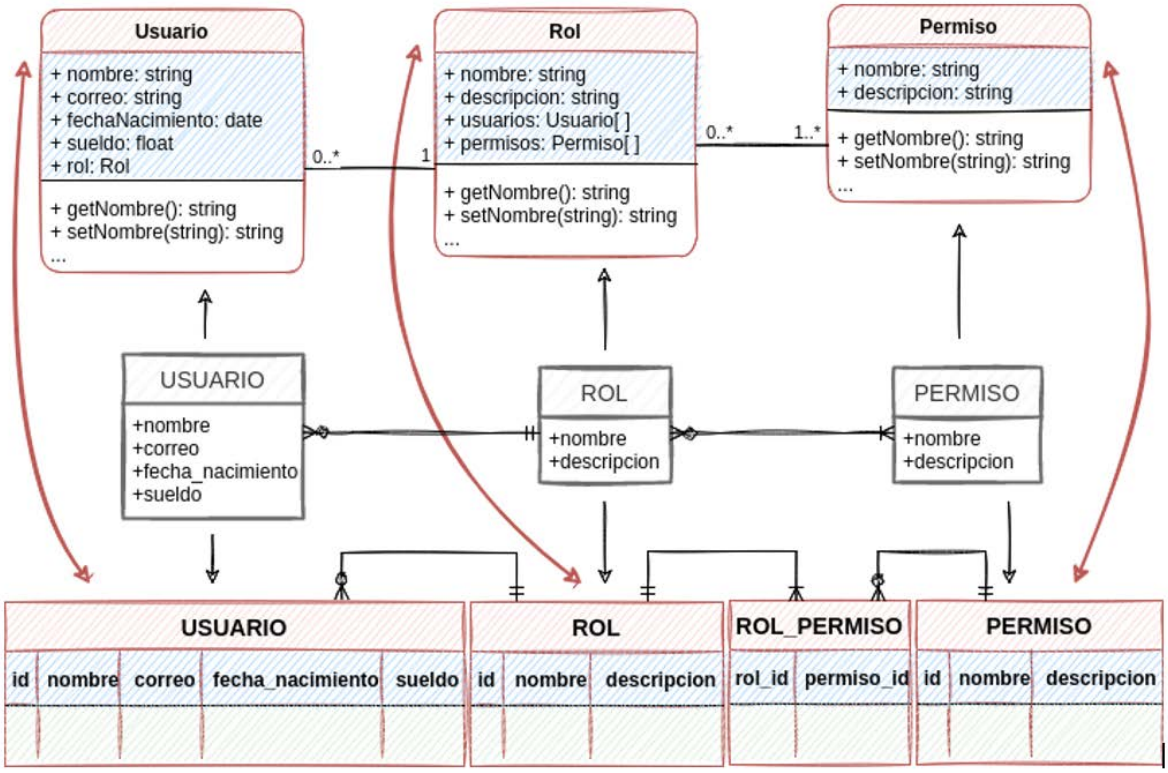
Finalmente, para la construcción de la estructura de la base de datos, se genera una migración y se aplican dichos cambios en la base de datos a través de los siguientes comandos:

```
$ php bin/console make:migration
$ php bin/console doctrine:migration:migrate
```

Lo anterior da como resultado el conjunto de elementos necesarios (clases y tablas) para comenzar a desarrollar el sistema con la implementación de la técnica *ORM* como se muestra en la figura 6.

Figura 6

Mapeo Objeto-Relacional a partir del enfoque model first



https://doi.org/10.22201/dgtic.ctud.2023.1.1.16 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 60 - 61

4. RESULTADOS

Se logró implementar de forma exitosa la técnica *ORM* en el desarrollo del SIPA con los criterios de selección tecnológica anteriormente descritos (*PHP* + *Symfony* + *MariaDB* + *Doctrine*), de manera que fue posible separar las tareas de desarrollo entre los integrantes del equipo y trabajar de forma paralela en diferentes aspectos del sistema (generación de entornos, programación y migración de datos).

Fue posible validar los beneficios ofrecidos por la técnica *ORM*, en cuanto a agilizar el desarrollo del sistema mediante una programación centrada en la lógica de objetos, lo que permitió realizar el Mapeo Objeto-Relacional de 18 tablas y 25 relaciones, en un tiempo estimado de cuatro horas y reducir en, aproximadamente, un 37% el tiempo efectivo empleado en la codificación de la segunda versión del SIPA (cinco meses) con respecto al tiempo que tomó la codificación de la primera versión (ocho meses), la cual no implementó la técnica *ORM*.

Finalmente, la mayoría de la literatura disponible sobre la implementación de la técnica *ORM* sugiere utilizar una librería bajo el enfoque *Active Record* cuando se desea priorizar el tiempo de desarrollo por considerar que el enfoque *Data Mapper* resulta más intimidante debido a su elevada curva de aprendizaje y configuración más compleja; sin embargo, los beneficios obtenidos en el desarrollo del SIPA dada la implementación exitosa de la técnica *ORM* con una librería que sigue el enfoque *Data Mapper*, enfatizan la importancia de considerar las capacidades del equipo de desarrollo por encima de las características de la tecnología a implementar.

4.1 TRABAJOS FUTUROS

Si bien al seleccionar la tecnología empleada para el almacenamiento de los datos, se optó por utilizar *MariaDB* priorizando la optimización de tiempo para la migración de los datos existentes, otra de las principales ventajas que ofrece la técnica *ORM* es la de proporcionar independencia del *SMBD*, por lo que, en una futura actualización, se podría cambiar de forma transparente en caso de ser necesario. La alternativa más conveniente de acuerdo con un análisis realizado por Choina y Skublewska-Paszkowska (2022) sería utilizar *PostgreSQL* al ser el manejador de bases de datos que presentó el mejor rendimiento en conjunto con *Doctrine*.

5. CONCLUSIONES

Implementar una técnica de Mapeo Objeto-Relacional en el desarrollo de un sistema brinda mayor flexibilidad a la hora de manipular los datos al proveer mecanismos para la generación de código y sentencias *SQL* de forma automatizada, lo que permite aumentar la productividad de los programadores y evita la escritura repetitiva de código para ejecutar operaciones *CRUD*.

Si bien, la implementación de la técnica *ORM* en el SIPA resultó exitosa y por ello se sugiere emplear en el desarrollo de otros sistemas que requieran gestionar información en una base de datos, siempre se debe considerar la magnitud del proyecto, los plazos para su entrega, las capacidades del equipo de desarrollo y las tecnologías a emplear, para determinar su viabilidad.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.16 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 61 - 61

REFERENCIAS BIBLIOGRÁFICAS

Altube Vera, R. (2021, mayo 21). *Laravel vs Symfony: Qué framework PHP elegir. openwebinars.net.* https://openwebinars.net/blog/laravel-vs-symfony-que-framework-php-elegir/

Choina, M., & Skublewska-Paszkowska, M. (2022). *Performance analysis of relational databases MySQL, PostgreSQL and Oracle using Doctrine libraries*. Journal of Computer Sciences Institute, 24, 250-257. https://doi.org/10.35784/jcsi.3000

DB-Engines. (2023). *DB-Engines ranking*. DB-Engines. Recuperado el 18 de septiembre de 2023, de https://db-engines.com/en/ranking

Fernández Alarcón, V. (2006). *Desarrollo de sistemas de información: Una metodología basada en el modelado* (1a ed.). UPC, S.L., Edicions. https://doi.org/10.5821/ebook-9788498800708

Gómez-Díaz, M. S., Casillas-Rodríguez, F. J., Juárez Guerra, L., Castellanos Nolasco, E., y Uribe López, U. (2022). *Análisis e implicaciones de la implementación del mapeo relacional de objetos en la programación orientada a objetos. Innovación y Desarrollo Tecnológico Revista Digital,* 14(4), 983-988.

Google. (s.f.). *¿Qué es una base de datos relacional?* Google Cloud. https://cloud.google.com/learn/what-is-a-relational-database?hl=es-419

Kolodka, P. (2023, julio 5). *Understanding abstraction levels in database interactions:* DAL, DAO, Raw Queries, Query Builder, ORM and Repository. Medium. https://levelup.gitconnected.com/understanding-abstraction-levels-in-database-interactions-dal-dao-raw-queries-query-builder-4819d607b0d6

Laudon, K. C., y Laudon, J. P. (2012). *Sistemas de información gerencial* (12a ed.). Pearson Educación.

Muro, J. A. (2023). ¿Qué es un ORM? Deloitte Spain. https://www2.deloitte.com/es/es/pages/technology/articles/que-es-orm.html

Pérez Montero, E. L., & Hernández Pérez, F. de M. (2019). *Object oriented programming: Easy to create*. I+ T+ C- Research, Technology and Science, 1(13), 96-100.

Web Technology Surveys. (s.f.). *Usage statistics and market share of server-side programming languages for websites*. Recuperado el 25 de octubre de 2023, de https://w3techs.com/technologies/overview/programming_language