ISSN: 2594 - 2905

ARQUITECTURA DE UN SISTEMA DE INFORMACIÓN CON ACCESO A BASES DE DATOS CON INTEGRIDAD REFERENCIAL DINÁMICA Y USO DE APIS GUBERNAMENTALES Y BANCARIAS

ARCHITECTURE OF AN INFORMATION SYSTEM WITH ACCESS TO DATABASES WITH DYNAMIC REFERENCE INTEGRITY AND USE OF GOVERNMENT AND BANKING APIS

Kevin Arisai Maldonado Cordova[1], Juan Ramos Ramos[2], María Janaí Sánchez Hernández[3], José Juan Hernández Mora[4], Elizabeth Cuatecontzi Cuahutle[5]

[1] Ingeniero en Tecnologías de la Información y Comunicaciones. [2] Maestro en Ciencias Computacionales y Telecomuniciones. [3] Maestra en Ciencias en Ciencias Computacionales. [4] Doctor en Excelencia Docente. [5] Maestra en Dirección de Ingeniería de Software

[1,2,3,4,5] Tecnológico Nacional de México/ Instituto Tecnológico de Apizaco. División de Estudios de Posgrado e Investigación.
[1] maldonadokef951@gmail.com, [2] juan.rr@apizaco.tecnm.mx, [3] janai.sh@apizaco.tecnm.mx, [4] juan.hm@apizaco.tecnm.mx, [5] elizabeth.cc@apizaco.tecnm.mx

Resumen – El objetivo de este trabajo es describir la propuesta de una arquitectura de software implementada en el diseño y desarrollo de un sistema de información con acceso a bases de datos con integridad referencial dinámica. Para el desarrollo del proyecto se utilizó la metodología de investigación y desarrollo tecnológico, realizando actividades propias de una investigación, en combinación con herramientas ágiles de desarrollo de software para la ejecución del proyecto de intervención, empleando SCRUM, XP y Kanban, así como los patrones arquitectónicos M-V-T y Microservicios, dando como resultado un modelo arquitectónico de software de 3 capas con acceso a microservicios para el uso de pasarelas de pago y timbrado ante el SAT, necesarios para gestionar cobros y otros servicios dentro de una entidad gubernamental.

Palabras Clave: Arquitectura de software, Patrones de diseño, Modelo de 3 capas, integridad referencial dinámica.

Abstract -- The aim of this work is to describe the proposal of a software architecture implemented in the design and development of an information system with access to databases with dynamic referential integrity. For the development of the project, the research and technological development methodology was used, carrying out research activities, in combination with agile software development tools for the execution of the intervention project, using SCRUM, XP and Kanban, as well as the architectural patterns M-V-T and Microservices, resulting in a 3-layer software architectural model with access to microservices for the use of payment gateways and stamping before the SAT, necessary to manage collections and other services within a governmental entity.

Key words **–Software architecture, design patterns, 3-layer model, dynamic referential integrity.**

INTRODUCCIÓN

La arquitectura de un producto de software no solo comprende los componentes de software, sino también las propiedades visibles, así como sus relaciones. Para esto es necesario contar con una evaluación para determinar las habilidades para el cumplimiento de los atributos de calidad que son necesarios para los sistemas de software [1]. Para poder satisfacer los requerimientos funcionales, así como no funcionales es necesario establecer procesos dentro de la organización del sistema, a esto se le conoce como diseño arquitectónico.

Los patrones arquitectónicos son un elemento determinante al momento de definir la arquitectura de software [2], los cuales se conocen como un marco de referencia para el diseño, y dan estructura a la construcción de una aplicación y definen características básicas de su comportamiento. Dentro de los patrones de arquitectura se encuentran distintas clases y depende de las necesidades del sistema, por lo que generalmente se hace uso del más conveniente al modelo de negocio.

El despliegue de la aplicación requiere patrones arquitectónicos e implementación de otras tecnologías como microservicios, facilitando la entrega e implementación continua en diferentes entornos [3]. Sin embargo, el desarrollo de aplicaciones web debe permitir entregas continuas de versiones para así ofrecer flexibilidad y reusabilidad, ya sea en un módulo o en el sistema. Esta característica se puede implementar con los servicios Web RESTful dentro de las aplicaciones para la reducción de tiempos, preparación del ambiente de desarrollo y despliegue a producción.

Fecha de Recepción: abril 2022 | Fecha de Aceptación: junio 2022

Un problema que se presenta a menudo en el diseño e implementación de una Base de Datos Relacional (BDR), es la necesaria incorporación de nuevas tablas que sean imprescindibles por cambios en los requerimientos de los usuarios, implicando a su vez que esas nuevas entidades de datos implementen integridad referencial para asegurar los valores de los atributos específicos de otras tablas [4], por ejemplo, un catálogo de información predefinido.

En el presente trabajo se propone la arquitectura de un sistema de información que accede a bases de datos con integridad referencial dinámica, haciendo uso de microservicios (APIs) de índole gubernamental y bancarios, para realizar procesos de cobro y timbrado ante el Servicio de Administración Tributaria (SAT), posteriores a los pagos relacionados dentro de una entidad gubernamental.

DESARROLLO

Marco teórico y estado del arte

Arquitectura de software y patrones arquitectónicos

Dentro de cualquier sistema de información es necesario establecer la arquitectura para su desarrollo [5]. Las bases de una arquitectura se definen como componentes (módulos) organizados dentro de un software [6], la cual es una o más estructuras del sistema, incluyendo atributos externos y visibles con relación entre ellos.

La primera etapa dentro de los procesos de software es el diseño arquitectónico, ya que es la unión fundamental entre el diseño e ingeniería de requerimientos. Las actividades detalladas dentro del diseño de un sistema se comprenden como patrones arquitectónicos, que dan al equipo de desarrollo una solución de problemas comunes. A los patrones se les considera una descripción abstracta de buenas prácticas de desarrollo [7].

El patrón M-T-V (Model-Template-View), es utilizado por el Framework Django, mismo que funciona en base a una arquitectura de 3 capas, separando la especificación de los modelos de datos, de la lógica de negocio y la interfaz del usuario final [8].

Framework

Para la construcción de aplicaciones o software, los desarrolladores utilizan una estructura o plantilla. Esta estructura se le conoce como framework (marco de trabajo), el cual es la base para la organización y desarrollo del sistema y el conjunto de clases, objetos y componentes que trabajan bajo una arquitectura de reutilización para aplicaciones a fines.
La división de los componentes de un sistema se basa usualmente en el patrón arquitectónico de 3 capas, constituido por la capa de presentación, la capa de lógica de negocio y la capa de acceso a datos.

Bases de datos

La integridad referencial es la encargada de asegurar los valores que contiene la relación con el conjunto de atributos que se visualizan en otra relación para un conjunto de atributos [9].

Las tres características del modelo relacional son: 1) Estructuras de datos simples, 2) Suministro de fundamentos sólidos para la consistencia de datos y 3) Manipulación de datos [10].

La Integridad Referencial Dinámica consiste en una solución que permite la inclusión de nuevas entidades de datos (catálogos) a partir de las cuales se podrá aplicar el concepto de integridad referencial de manera dinámica, a nivel de base de datos, para nuevos atributos. Se utiliza una tabla para almacenar las posibles entidades que representan nuevos datos requeridos, una segunda tabla que indica si cada nueva entidad requiere integridad referencial y una tercera tabla se utiliza para guardar los valores (catálogos de datos) que serán tomados para validar una integridad referencial [4].

Metodología

La investigación cuantitativa está basada en la realidad existente y la investigación cualitativa trata de demostrar cómo es la realidad y su significado, mientras que la investigación y desarrollo tecnológico demuestra que la realidad puede ser modificada y parte del cambio es ejecutado por el ser humano con afirmaciones particulares, operativas y ejecutables [11].

Las etapas para realizar dentro de la metodología de investigación y desarrollo tecnológico son las indicadas en la Figura 1.

iPSUMTEC
ISSN: 2594 - 2905

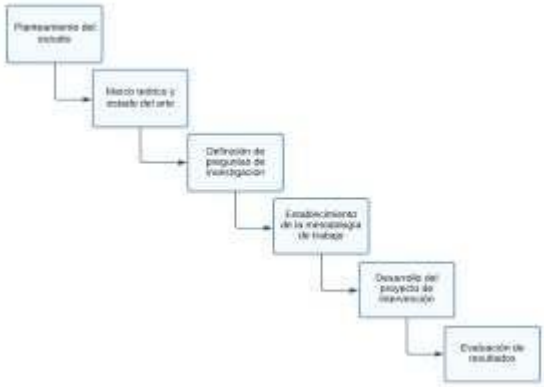

Figura 1. Metodología de investigación y desarrollo tecnológico (Fuente Propia).

Planteamiento del estudio

El problema principal consiste en determinar la arquitectura idónea de un sistema de información que acceda a una base de datos con integridad referencial dinámica y que permita la conexión con soluciones de terceros mediante APIs, tales como las pasarelas de pago o servicio de timbrado ante el SAT.

Marco teórico y estado del arte

Contemplando las definiciones previas dentro del apartado del estado del arte, se tiene como base teórica y estado del arte para el entendimiento para el desarrollo de una arquitectura de un sistema de información con acceso a bases de datos relacionales con integridad referencial dinámica.

Definición de preguntas de investigación

En un proceso de investigación y desarrollo tecnológico es necesario plantear los posibles escenarios basados en una modificación del mundo real. El desarrollo del proyecto de intervención permitirá modificar esa concepción actual. Los supuestos de interacción con el usuario y su percepción de uso de una nueva herramienta permiten definir las hipótesis que sustentan esta investigación. En relación con el desarrollo planteado se define la siguiente pregunta:

- ¿Cuál es la arquitectura que requiere un sistema de información que permita acceder a Bases de Datos con tablas que incluyen integridad referencial dinámica?

Establecimiento de la metodología de trabajo

El desarrollo de este proyecto, que se basa en la combinación de la metodología de investigación y desarrollo tecnológico con herramientas ágiles para la gestión de proyectos de software (Figura 2). Se prevén las siguientes actividades:

- Proceso de establecimiento de la investigación y desarrollo tecnológico con sus fases establecidas.
- Integración de las metodologías de desarrollo tecnológico e integración de metodologías de desarrollo de software (ingeniería de software).
- Propuesta de una arquitectura con acceso a bases de datos relacionales con integridad referencial dinámica.
- Integración de Microservicios en la arquitectura propuesta.

Figura 2. Combinación de la metodología de investigación y desarrollo tecnológico con herramientas ágiles de ingeniería de software (Fuente Propia).

Desarrollo del proyecto de intervención

Para el desarrollo de la solución propuesta se propone el uso de metodologías para gestión de proyectos y desarrollo de software. Se hace uso se SCRUM para gestionar las actividades correspondientes a cada una de las nuevas funcionalidades de la herramienta informática a desarrollar, a través de la planeación de Sprints.

Cada sprint se integra por un conjunto de historias de usuario, gestionándolas mediante un tablero de Kanban. De esta manera se hace uso de las mejores prácticas de las metodologías de SCRUM, Kanban y XP

En el proceso inicial se establece un sprint de inicio denominado “Sprint 0”, que se encarga del análisis y diseños generales del sistema [12].

Lo importante del Sprint 0 es reutilizar partes del software generado por la ingeniería y así crear procesos rápidos en desarrollo; el Sprint define la infraestructura del sistema y junto con los requisitos funcionales de calidad y los procesos de negocio serán los que definan el contexto en los requisitos funcionales y de diseño.

DISCUSIÓN Y ANÁLISIS DE RESULTADOS

Para contemplar el desarrollo, fue necesario establecer los procesos de ingeniería de software, así como la arquitectura de software que abarca elementos, propiedades visibles de las relaciones y funcionalidades para cumplir con los requisitos deseados [2]. Además, fue necesario decretar los procesos y metodologías para la implementación de la propuesta del sistema de información establecido como caso de aplicación.

Teniendo como base la investigación y desarrollo tecnológico, se obtiene como resultado de la arquitectura propuesta un diseño fundamentado en el modelo de 3 capas del Framework Django (Model-View-Template), con escalabilidad del sistema, haciendo uso de microservicios, como se muestra en la Figura 3, donde se hace uso de APIs para complementar las necesidades de la arquitectura.

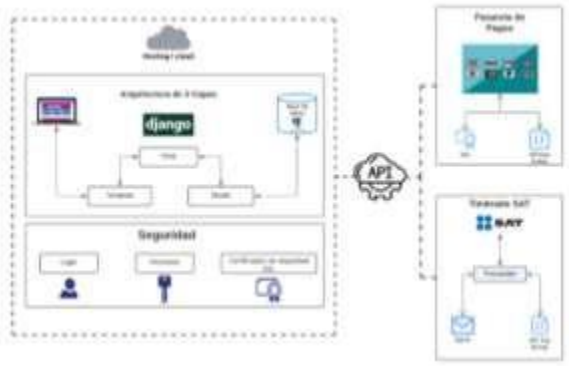

Figura 3. Diagrama arquitectónico propuesto

(Fuente Propia).

El modelo arquitectónico propuesto se fundamenta en una arquitectura de 3 capas, tomando como referencia el patrón de diseño establecido en el Framework Django, conectando con los servicios externos de pasarelas de pago y timbrado en el SAT mediante el uso de APIs. A continuación, se describe cada una de las capas implementas:

- Capa de presentación (Template): su principal responsabilidad es el manejo de la interacción con el cliente, por medio de las interfaces para hacer peticiones y poder visualizarlas.
- Capa de lógica de negocio (Vista): se encuentra la funcionalidad total del sistema, estableciendo la interacción entre la capa de presentación y la capa de datos.
- Capa de acceso a datos (Model): es la encargada de realizar la comunicación con la base de datos física, a petición de la capa de lógica de negocio.

En relación con el modelo de datos que da soporte al caso de aplicación para el sistema de información propuesto, se diseñó una base de datos con integridad referencial dinámica que facilita la incorporación de nuevas tablas, las cuales representan en la mayoría de los casos, catálogos de referencia de información para la operación del propio sistema [4]. El modelo conceptual se integra de manera apropiada con la arquitectura resultante de esta investigación.

Para asegurar el sistema, es necesario contar con las credenciales de usuario (Login), así como los distintos tipos de roles, privilegios o permisos que se le asignan relacionados con el sistema.

El uso de la arquitectura propuesta ha permitido crear un sistema escalable, flexible con separación de cada una de sus funcionalidades, adoptando servicios externos a través del uso de APIs. Esta propuesta difiere de una arquitectura monolítica, la cual se enfrenta a deficiencias inherentes a la relación intrínseca de los módulos para su funcionamiento [13].

La implementación de microservicios requiere los siguientes pasos u operaciones para trabajar con el sistema:

a) Conexión

La conexión se realiza con el proveedor del microservicio para hacer un enlace al host que devuelve la información requerida, a través de una URL, la cual funciona como puente para realizar las solicitudes.

b) Seguridad

La seguridad para realizar la conexión la determina el proveedor del microservicio, el cual emite la clave de conexión (clave API) y la clave API, estableciendo el enlace de forma segura con el cliente.

c) Implementación

Para implementar microservicios con el sistema, es necesario crear rutas de comunicación y almacenar información en la base de datos, brindando a los usuarios servicios y operaciones transparentes y fáciles.

Al cumplir con las necesidades de implementación de microservicios dentro de una arquitectura de 3 capas, es posible tener una conexión estable y segura para usar sus servicios en el sistema desarrollado.

Finalmente, el diseño del sistema se realiza conforme al modelo monolítico de 3 capas, el cual tiene una orientación dentro de su funcionamiento al modelo cliente-servidor, con una extensión al uso de microservicios y así estableciendo la perspectiva de abstracción de la arquitectura de software desde 3 vistas distintas, como lo son la del desarrollador, del usuario y de funcionalidad, lo cual se muestra en la Figura 4.

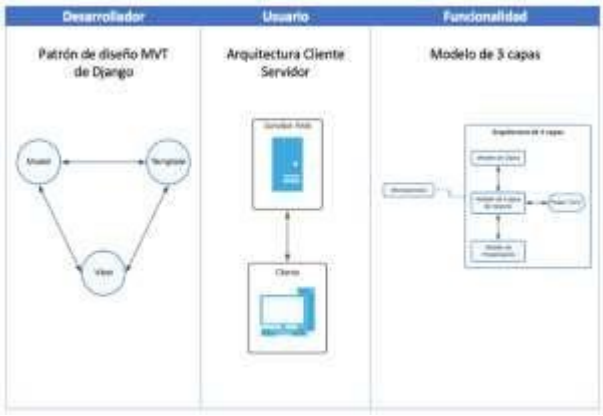

Figura 4. Perspectivas de abstracción de la arquitectura de software (Fuente Propia).

CONCLUSIONES

Con el modelo arquitectónico propuesto es posible incorporar un modelo de 3 capas y el consumo de recursos a través de APIs de terceros como lo son pasarelas de pago y timbrado. Esta arquitectura permite la conexión a bases de datos con integridad referencial dinámica, de manera flexible y transparente, sin ninguna modificación a la estructura de modelos propuestos en la capa correspondiente.

La solución de conectividad entre una aplicación desarrollada en un modelo de 3 capas y la necesidad de usuario para timbrar cobros realizados ante el SAT es posible implementarla mediante la solución que ofrezca un proveedor de facturación, preferentemente ofreciendo
soporte técnico para la implementación en distintos lenguajes de programación, facilitando la integración con una aplicación desarrollada en un lenguaje como Python, contando con servidores de prueba, incorporación de código de verificación y facilitando la generación de prueba iniciales.

Las pasarelas de pagos, a través de sus APIs, facilitan la generación del cobro por servicios, productos u otros conceptos utilizando medios electrónicos. Su incorporación en una aplicación web se realiza a través del uso de una interacción dependiente del lenguaje de programación utilizado.

Con base en el consumo de microservicios, la integración con el modelo arquitectónico propuesto permite la creación de una herramienta que está disponible al usuario final, facilitando la gestión de cobros y otros servicios en una entidad gubernamental para modernizar el sistema tradicional de cobro, manteniendo la trasparencia en el ejercicio de los recursos públicos y facilitando al usuario final herramientas actuales para servicios en línea, vía internet.

TRABAJO A FUTURO

Para el desarrollo de trabajos futuros se contemplan los siguiente:

1. Generación de microservicios (APIs), para que aplicaciones externas puedan consumir o consultar la información del sistema y sus catálogos de información, según necesidades.

2. Evaluación e integración de la arquitectura para la incorporación de otros módulos de la administración interna gubernamental.

BIBLIOGRAFÍA

[1] Blas, María Julia; Leone, Horacio Pascual; Gonnet, Silvio Miguel. Modelado y verificación de patrones de diseño de arquitectura de software para entornos de computación en la nube. RISTI.2019;35(17):1-17.

[2] Navarro, M. E., Moreno, M. P., Aranda, J., Parra, L., Rueda, J. R., & Pantano, J. C. Selección de metodologías ágiles e integración de arquitecturas de software en el desarrollo de sistemas de información. In XIX Workshop de Investigadores en Ciencias de la Computación. Buenos Aires; 2017. P 632-636.
[3] Huanca Torres, FA. Arquitectura para el desarrollo e implementación de servicios web [Tesis de Maestría]. Puno, Perú. Universidad Nacional del Altiplano. 2017. Recuperado a partir de: http://repositorio.unap.edu.pe/handle/UNAP/10789
[4] Jiménez Is-B, Ramos J, Hernández JJ, Tlapalamatl C. Dynamic Referential Integrity Model for Relational Databases. International Journal of Science and Research (IJSR) [Internet]. 2020; 9(7): 312-316. Doi: https://www.ijsr.net/get_abstract.php?paper_id=SR20627024418
[5] Bass L, Clements P, Kazman, R. Software architecture in practice. 2da ed. México Addison-Wesley,2003.
[6] Pressman R. Ingeniería del software: Un enfoque práctico. México: Eddison Wesley; 2010.
[7] Sommerville I. Ingeniería del software. México: Addison-Wesley; 2011.
[8] EspiFreelancer. Qué es el patrón MVT (Model Template View) [Internet]. EspiFreelancer. [citado 15 de enero de 2022]. Recuperado a partir de: https://espifreelancer.com/mtv-django.html
[9] Silberschatz, A, Korth, HF, Sudarshan S, Pérez FS, Santiago AI, Sánchez AV. Fundamentos de bases de datos. 5ta ed. México: McGraw-Hill; 2006.
[10] Tamayo Alzate, A, Duque Méndez, N Mecanismos de seguridad e integridad en un sistema de bases de datos. [Internet]. Universidad Nacional de Colombia - Sede Manizales; 2001 [citado: 2022, febrero] Universidad Nacional de Colombia Sede Manizales Facultad de Administración Departamento de Informática y Computación.
[11] Casaño CDLC. Metodología de la investigación tecnológica en ingeniería. Ingenium [Internet]. 2016 [citado: 2022, mazo]; 1(1): 43-46. doi: http://dx.doi.org/10.18259/ing.2016007
[12] Edwin Rafael Mago - Germán Harvey Alférez. El Papel de la Arquitectura de software en Scrum. Publicado en SG #30 https://sg.com.mx/revista/30/el-papel-la-arquitectura-software-scrum
[13] Padilla Velasco JI, Rodriguez Ruiz AI, Parra Alvira HA. Arquitectura basada en microservicios para aplicaciones web. Tecnol. Investig. Academia TIA [Internet]. 10 de enero de 2020 [citado 9 de mayo de 2022];7(2):12-20. Disponible en: https://revistas.udistrital.edu.co/index.php/tia/article/view/13364
[14] Navarro, M. E., Moreno, M. P., Aranda, J., Parra, L., Rueda, J. R., & Pantano, J. C. Integración de arquitectura de software en el ciclo de vida de las metodologías ágiles. In XIX Workshop de Investigadores en Ciencias de la Computación. Buenos Aires; 2017. P 566-569.
[15] Padilla Velasco JI, Rodriguez Ruiz AI, Parra Alvira HA. Arquitectura basada en microservicios para aplicaciones web. Tecnol. Investig. Academia TIA [Internet]. 10 de enero de 2020 [citado 7 de abril de 2022];7(2):12-20. Disponible en: https://revistas.udistrital.edu.co/index.php/tia/article/view/13364
[16] Sánchez JC. Metodología de la investigación científica y tecnológica. Ediciones Díaz de Santos; 2004.

ROLES DE DISTRIBUCIÓN	AUTOR (ES)
Conceptualización Metodología Curación de datos Software Redacción	Kevin Arisai Maldonado Cordova
Conceptualización Metodología Administración del proyecto Supervisión	Juan Ramos Ramos
Validación	María Janaí Sánchez Hernández
Recursos	José Juan Hernández Mora
Supervisión	Elizabeth Cuatecontzi Cuahutle