Recibido 12 Ene. 2023 *ReCIBE, Año 12 No.1, May. 2023*

Aceptado 20 Feb. 2023

Sistema de Alerta Temprana epidemiológica animal

Animal Epidemiological Early Warning System

Edgar Montiel Cruz[1]
Felipe Hernández González[1]
Alejandro Luis Collantes Chávez-Costa[1]

[1]Universidad Autónoma de Quintana Roo

Resumen En la isla de Cozumel, México, se encuentran especies endémicas y ferales, mismas que se encuentran amenazadas por sus reducidos tamaños poblacionales y por enfermedades. Para el manejo de una problemática que se pudiera presentar y toma de decisiones, se propone un prototipo de sistema de alerta temprana epidemiológica animal, construida sobre una plataforma Web bajo el modelo cliente-servidor. Se siguió la metodología del ciclo de vida del software junto con el modelo incremental. Para la definición de los requerimientos se utilizó el Lenguaje Unificado de Modelado (Unified Model Languaje) integrando el desarrollo guiado por pruebas (Test Driven Development). Para la codificación se utilizó el lenguaje de programación JavaScript y MySQL como motor de base de datos. La implementación de la interfaz de usuario se realizó con el apoyo de las librerías Vue.js y Vuetify.js. Los módulos del sistema comprenden el catálogo de especies, captura de casos que incluyen datos de georeferencia y consultas de través de un mapeo de casos y un módulo de alertas automáticas based en el uso de semáforos similar al del SARS-COV 19.

Palabras clave: Sistema de Alerta temprana, Desarrollo Web.

Abstract: Endemic and feral species are found on the island of Cozumel, Mexico, which are threatened by their small population sizes and diseases. For the management of a problem that could arise and decision-making, a prototype of an animal epidemiological early warning system is proposed, built on a Web platform under the client-server model. The software life cycle methodology was followed following the incremental model. For the definition of the requirements, the Unified Model Language was used, integrating Test Driven Development. For coding, the programming language JavaScript and MySQL were used as the database engine. The implementation of the user interface was carried out with the support of the Vue.js and Vuetify.js libraries. The modules of the system include the catalog of species, capture of cases that include georeference data and queries through a case mapping and eaerly warning module which uses a traffic light system very similar SARS-COV 19.
Keywords: Early Warning System, Web Development

1 Introducción

La isla de Cozumel ubicada en el estado de Quintana Roo, México, es el primer destino de arribo de turistas de crucero del país. Tan sólo en el primer semestre de 2019 recibió 2,406,908 (Sectur, 2019) por esta vía, sin contar otros medios como el aéreo o cruce de ferry de pasajeros o de vehículos terrestres. En las áreas de selva virgen se encuentran especies endémicas, como Nasua nelson (coatí), Procyon pygmaeus (mapache enano), y Pecari tajacu nanus (Pecarí de Collar) (Universidad Nacional Autónoma de México, 2012), mismas que se encuentran amenazadas por sus reducidos tamaños poblacionales, por lo que pueden ser consideradas vulnerables a eventos destructivos como enfermedades. Estas especies comparten con animales domésticos, enfermedades infectocontagiosas, como lo son la rabia y el distemper canino. Al avanzar la mancha urbana, es posible que se produzcan contactos entre las mismas y por consiguiente un contagio puede ocurrir (Frausto, Ihl, Rojas, 2016). Para contribuir a la conservación de las especies endémicas de la isla de Cozumel, el presente proyecto tiene la finalidad de elaborar un prototipo de sistema de alerta temprana epidemiológica animal, que ayude a la toma de decisiones en el control de enfermedades que amenazan la fauna endémica de la Isla de Cozumel.

Un sistema de alerta temprana se define como "*el conjunto de herramientas, dispositivos de control, capacidades de gestión e instrumentos tecnológicos que las instituciones claves identifican para difundir la información de manera oportuna a las comunidades expuestas a un riesgo, y cuyo resultado son medidas de mitigación orientadas a reducir los efectos de los*

desastres naturales y las pérdidas económicas y de vidas, así como las lesiones" (Domínguez, Lozano, 2019).

2 Métodos o Metodología

2.1 Ciclo de vida del software

Para el desarrollo del sistema de alerta temprana se siguieron las siguientes etapas del ciclo de vida del software:

- Investigación Preliminar. Se tomó en cuenta el contexto de la situación o problemática a resolver, mediante entrevistas con los usuarios.
- Definición de los Requerimientos. Se identificaron las características que debe tener el nuevo sistema, la información que deben producir y las características operacionales. El modelado de los requerimientos se hizo mediante la definición de casos de uso utilizando la notación ***UML*** (Unified Modeling Language) el cual se basa en el Paradigma Orientado a Objetos(***POO***) (Bruegge, Dutoit, 2009). Esto se ilustra en la figura 1.

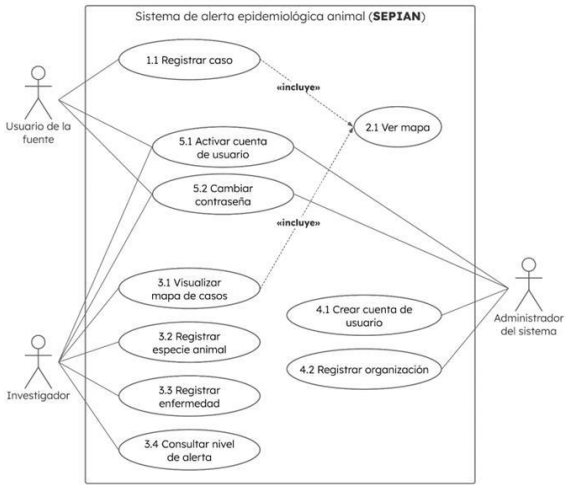

Figura 1. Casos de uso del sistema de alerta temprana animal.

- Definición de la Frontera del Problema. Se determinó con base a la definición de los requerimientos expresados en las entrevistas.
- Definición del Modelo Lógico de la Base de Datos. Se determinaron los detalles que establecen la forma en la que el sistema cumplirá con los requerimientos identificados durante la fase de análisis.
- Diseño Modular. En esta etapa se dividieron o fragmentaron los procesos del sistema en funciones separadas que deberán cumplir con ciertas condiciones.
- Diseño de la interfaz hombre-máquina. Se diseñó el conjunto de elementos con los que un usuario va a interactuar con la aplicación de software para llevar a cabo una actividad o proceso. Comúnmente se denomina ***GUI*** (Graphic User Interface).
- Desarrollo del prototipo. Se transformó el diseño de los requerimientos en programas que cubran con la funcionalidad esperada. Una vez que se desarrolló el prototipo, se siguió el modelo de desarrollo de software incremental el cual consiste en lo siguiente: Después de contar con el desarrollo inicial de la arquitectura completa del sistema, se añaden incrementos y versiones parciales del mismo. Cada incremento tiene su propio ciclo de vida y se agregó funcionalidad adicional o mejorada sobre el sistema. Conforme se completó cada etapa, se verificó e integró la versión con las demás versiones ya completadas del sistema. Esto se puede ver en la figura 2.

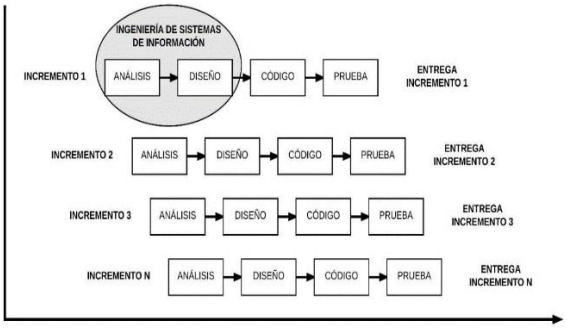

Figura 2. Modelo Incremental. Fuente: (Pressman,2001)

2.1.1 Desarrollo guiado por pruebas

Siguiendo esta práctica de programación, primero se definieron las pruebas o lo que el software tiene que hacer que es muy similar a la definición de casos de uso definidos en ***RUP*** (Rational Unified Process).

Posteriormente se elaboró el código fuente que pasó las pruebas satisfactoriamente. Una vez conseguido esto, se refactorizaró el código escrito para ser parte del producto de software.

De esta forma el código es menos susceptible a fallas a la vez que se acelera el proceso de desarrollo. Para la codificación de las pruebas se utilizó la librería ***Jest*** que corre bajo ***Node.js***.

Para la codificación del código fuente de todos los módulos incluyendo los pruebas, se empleó el ***IDE*** (Integrated Development Environment) Visual Studio Code en su última versión. Para el control de versiones lo más adecuado fue un repositorio en Gitlab cuyo manejo está integrado en Visual Studio Code.

2.1.2 Arquitectura del sistema

Se utilizó un patrón cliente-servidor y específicamente una arquitectura en capas. Se opta por el enfoque en capas a razón de su afinidad con el ciclo de vida del software y desarrollo incremental (Sommerville, 2011), haciéndolo adecuado para la utilización de una metodología de desarrollo ágil. Marston (2012) define las siguientes capas:

- Capa de presentación.
- Capa de lógica de negocio.
- Capa de acceso a datos.

La arquitectura del sistema por capas del sistema de alerta temprana animal se aprecia en la figura 3.

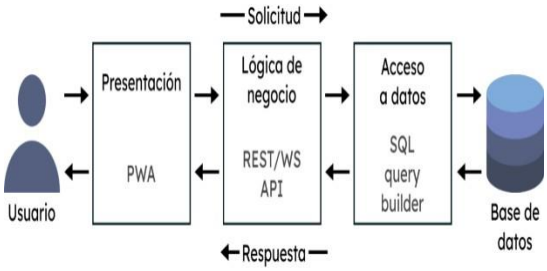

Figura 3. Arquitectura del sistema. Fuente: Elaboración propia

Implementación de las capas

En la capa de presentación será implementada la interfaz gráfica de usuario para el sistema. Se plantea el desarrollo de una Aplicación Web Progresiva (***PWA***, por sus siglas en inglés) la cual funciona independientemente del sistema operativo o dispositivo en el que se use. Una alternativa es el desarrollo de aplicaciones cliente nativas para cada sistema operativo requerido (Biørn-Hansen, Majchrzak, Grønli, 2017).

Se utilizó el lenguaje de programación JavaScript para el código fuente. En el cliente el motor de ejecución lo proveen los navegadores Web y en el servidor, el motor lo provee ***Node.js***. Del lado del cliente se utilizaron las bibliotecas ***Vue.js*** y ***Vuetify.js*** que permiten la creación de una Interfaz Gráfica de Usuario de aspecto moderno. ***Vue.js*** permite la implementación de Progresive Web Applications, las cuales se comportan como una aplicación nativa de la plataforma dónde se ejecutan, ya sean computadoras de escritorio o dispositivos móviles (Lei, Ma, Tan,2014). En el lado del servidor se optó por ***MySQL*** como tecnología de base de datos, pudiendo ser reemplazada por cualquier otra que sea soportada en el entorno ***de Node.js***.

También se empleó la librería ***FeathersJS*** para crear una ***API REST*** (Aplication Programming Interface basada en el protocolo ***REST***) que proporcionará el acceso a los recursos de la base datos a través del protocolo ***HTTPS*** (Hypertext Transfer Protocol Secure) e implementa la funcionalidad de autenticación de usuarios usando la tecnología ***JSON Web Tokens***. En el servidor, se utilizó ***Nginx*** como proxy reverso para el acceso a la API, ya que proporciona balanceo de carga de ser requerido en el futuro. Adicionalmente el código del cliente requiere ser servido de forma estática mediante ***HTTPS***, por lo que ***Nginx*** puede realizar esta función y servir a la ***API*** con un nombre de dominio y el código del cliente con otro.

Las tecnologías utilizadas se resumen en la figura 4.

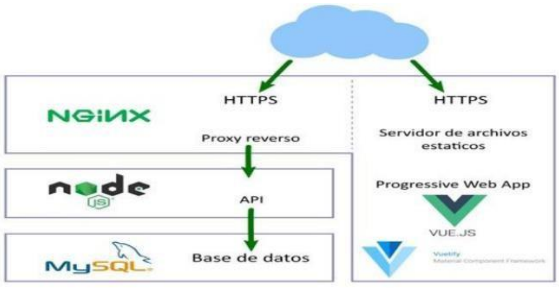

Figura 4. Tecnologías aplicadas a la arquitectura del sistema. Fuente: Elaboración propia

3 Resultados

3.1 Desarrollo del sistema

La interfaz del módulo principal del sistema se aprecia en la figura 5.

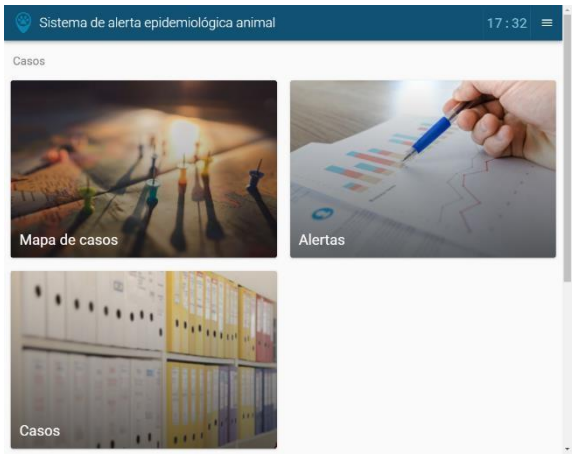

Figura 5. Módulos del sistema de alerta temprana animal.

Módulos principales: Casos, Mapa de Casos y Alertas.

La función del Módulo Casos es registrar la información de los casos de manera individual que incluye información georeferenciada como se puede apreciar en la figura 6.

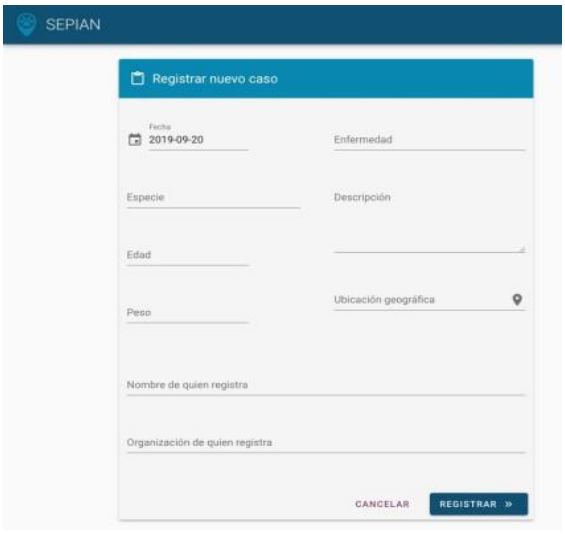

Figura 6. Módulo de captura de casos.

La opción Mapa de Casos permite consultas de manera dinámica al especificar parámetros como la enfermedad o un rango de fechas. El resultado de una consulta se muestra en la figura 6.

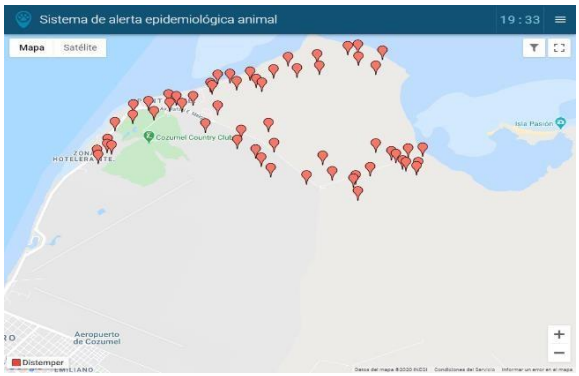

Figura 7. Módulo de Consulta de Casos.

Las alertas del sistema, se definen por cada enfermedad registrada y se parametrizan con base a un nivel de incremento. Esto se aprecia en la figura 8.

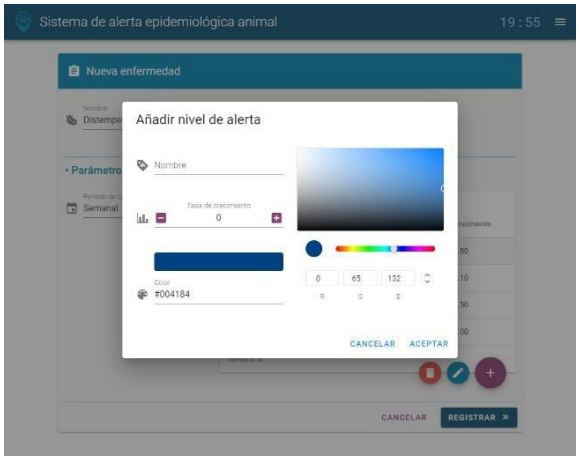

Figura 8. Ejemplo de alertas automáticas.

De esta manera, se asocia este parámetro a un sistema de semáforo con códigos de color exactamente al que se utiliza por el gobierno de México para la contingencia sanitaria del SARS-COV-19: Verde (sin peligro), Amarillo (peligro bajo), Naranja (peligro moderado) y Rojo (peligro máximo).

Con esta información el sistema es capaz de generar alertas de manera automática. Esto se aprecia en la figura 9.

Figura 9. Ejemplo de alertas automáticas por rango de fechas. Fuente: Elaboración propia.

3.2 Desarrollo guiado por pruebas

El desarrollo guiado por pruebas (TDD, acrónimo de Test-Driven Development) es un enfoque donde convergen el diseño pruebas y el código.

Los pasos son los siguientes:
1.- Se identifica una funcionalidad con base en las reglas de negocio.

2.- Se diseña y construye la prueba, la cual es totalmente automatizada. Una vez termina se procede a ejecutarla.

3.- Si la prueba se supera con éxito, se procede a integrar una nueva funcionalidad, es decir se inicia otra vez el paso 1.

4.- En caso de que la prueba falle, se tienen que hacer las correcciones a la misma para una correcta implementación refactorizando el código. Una vez completado este proceso se vuelve a correr la prueba.

La figura 10 resume el proceso TDD.

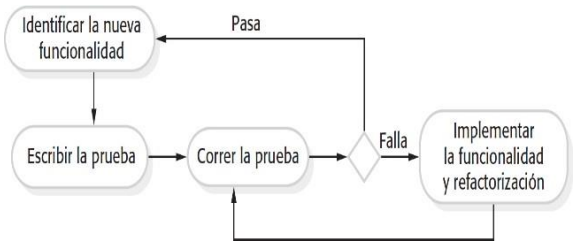

Figura 10. El desarrollo guiado por pruebas. Fuente: Basado en Dookhun, Nagowah, 2019.

El desarrollo dirigido por pruebas también funciona como una forma de documentar el desarrollo del código.

.

Un ejemplo del resultado del desarrollo guiado por pruebas se aprecia en la figura 11. Se utilizó el framework JUnit 5.

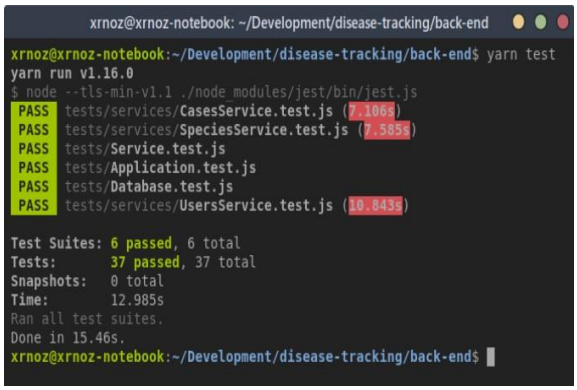

Figura 11. Resultado del Desarrollo Guiado por Pruebas para el Módulo de Casos.

4 Discusión

El haber seguido el ciclo de vida del software junto el desarrollo incremental permitió cumplir con el desarrollo del sistema. La implementación del desarrollo guiado por pruebas se complicó un poco en virtud que el equipo de trabajo no tenía experiencia ni con la metodología ni con las herramientas para llevarlas a cabo. Solo tenía experiencia en hacer pruebas de caja blanca para determinar los errores del software y corregirlos.

La definición de los parámetros de las alertas sufrió modificaciones en virtud que el usuario cambió los requerimientos iniciales que especificaban valores estáticos en lugar de valores que pudieran modificarse con el paso del tiempo según se necesitara con base al comportamiento de una enfermedad en específico. Se implementó un módulo para el administrador del sistema mismo que permite la creación de cuentas de usuario, que serán creadas para las personas que se encargarán de la captura de información.

Las consultas de casos por rango de fechas y el despliegue de estos en un mapa, fue una de las funcionalidades que el usuario consideró de mayor utilidad ya que le permitirá visualizar el comportamiento de una enfermedad y su área geográfica asociada. Permitirá el

establecimiento de cercos de seguridad sanitaria y acciones para control y erradicación de un brote.

.

5 Conclusiones

Se diseñó y construyó un Sistema de Alerta Temprana Animal que cumple con los requerimientos del usuario y tiene tres módulos completamente funcionales: Casos, Mapa de Casos y Alertas. El primero permite la captura de enfermedades nuevas en caso de que no se haya capturado con anterioridad. El segundo permite la generación de consultas por un rango de fechas y despliegue visual a través del mapeo de los casos con base a la información georefereneciada. El módulo de alertas permite la definición de las mismas con base al establecimiento de parámetros que pueden actualizarse a través del tiempo. Utiliza el sistema de semáforos como los empleados para el monitoreo del SARS-COV 19 para el despliegue visual de la información.

.

6 Trabajo a futuro

Es necesario que el sistema cuente con un módulo especial que permita explotar la información que se está registrando mediante la generación de estadísticas para complementar el módulo de consultas cuyos resultados de las mismas, se basan en un rango de fechas para generarlos. De esta manera se podrá dar seguimiento a la evolución de un brote que se presente, hacer comparaciones entre enfermedades, así como las bases para realizar un mapa de riesgos tomando la información georeferenciada de cada caso. También es necesario que el sistema genere gráficas para que para un mejor seguimiento del comportamiento y evolución de las enfermedades que se presenten.

.

Referencias bibliográficas

Dookhun, A. S., & Nagowah, L. (2019). Assessing the effectiveness of test-driven development and behavior-driven development in an industry setting. En 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE) (pp. 365-370). https://doi.org/10.1109/ICCIKE47802.2019.900432

Biørn-Hansen, A., Majchrzak, T. A., & Grønli, T.-M. (2017). Progressive Web Apps: The Possible Web-native Unifier for Mobile Development. Proceedings of the 13th International Conference on Web Information Systems and Technologies - Volume 1: WEBIST, 344–351. https://doi.org/10.5220/0006353703440351

Bruegge, B., & Dutoit, A. H. (2009). Object-oriented software engineering using UML, patterns, and Java (3a ed.). Pearson.

Dominguez, Lozano (2014). Recuperado el 12 de diciembre de 2019 de http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0370-39082014000300007

Frausto, O., Ihl, T., & Rojas, J. (2016). Atlas de Riesgos de la Isla de Cozumel, México. Teoría Y Praxis, 12(Especial, Octubre 2016), 74–93.

Sommerville, I. (2011). Ingeniería de Software (9a ed.). Pearson Educación.

Lei, K., Ma, Y., & Tan, Z. (2014). Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js. 2014 IEEE 17th International Conference on Computational Science and Engineering, 661-668. https://doi.org/10.1109/CSE.2014.142

Marston, T. (2012). What is the 3-Tier Architecture? Recuperado el 30 de agosto de 2019 de http://www.tonymarston.net/php-mysql/3-tier-architecture.html