http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 12, n. 3, p. 196-208, set.-dez. 2019 – ISSN 1983-3652
DOI: 10.17851/1983-3652.12.3.196-208

SISTEMA PARA ANÁLISIS DE MUESTRA DE UROCULTIVO A PARTIR DE LA CURVA DE CRECIMIENTO

SYSTEM FOR ANALYSIS OF UROCULAR SAMPLE FROM THE GROWTH CURVE

Omar Mar Cornelio
University of Informatics Sciencis: La Habana, Cuba
omarmar@uci.cu

Leyanys Acosta Calderón
University of Informatics Sciencis: La Habana, Cuba
lacalderon@uci.cu

Karla García Benítez
University of Informatics Sciencis: La Habana, Cuba
kbenitez@uci.cu

RESUMEN: En los laboratorios clínicos de los centros médicos, se realizan un conjunto de análisis a partir de la recogida de muestras biológicas, en organizaciones pequeñas como policlínicos y hospitales no especializados. Se realiza mediante la aplicación de diversos sistemas para el procesamiento semiautomático de las muestras. Actualmente en Cuba uno de los sistemas que realiza este proceso es el Driamic. Sin embargo, al no poseer soporte técnico, no ha sido posible corregir la generación de falsos positivos en los resultados de la muestra, además no existen algoritmos que muestren las curvas de crecimiento. La presente investigación describe una solución a la problemática planteada a partir de la elaboración de un sistema basado en tecnologías libres que implementa un conjunto de algoritmos computacionales, que permiten graficar el crecimiento bacteriano, a partir de la unidad formadora de colonias (UFC)/ml que presenta una muestra en tiempo real, mediante su curva de crecimiento. Se generan además un conjunto de reportes que facilita el diagnóstico clínico y permite la visualización de los datos generados por el procesamiento de la muestra de urocultivo de un paciente.
PALABRAS CLAVES: sistema de Gestión; curva de crecimiento; urocultivo.

ABSTRACT: In the clinical laboratories of the medical centers, a set of analyzes is made from the collection of biological samples, in small organizations such as polyclinics and non-specialized hospitals. It is done by applying various systems for the semi-automatic processing of the samples. Currently in Cuba one of the systems that makes this process is the Driamic. However, since there is no technical support, it has not been possible to correct the generation of false positives in the results of the sample, in addition there are no algorithms that show the growth curves. The present investigation describes a solution to the problematic raised from the elaboration of a system based on free technologies that implement a set of computational algorithms, that allow to graph the bacterial growth, from one of the unit forming of colonies (UFC)/ml that shows a sample in real time, through its growth curve. A set of reports that facilitates the clinical diagnosis and allows the visualization of the data generated by the processing of a patient's urine sample is also

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 12, n. 3, p. 196-208, set.-dez. 2019 – ISSN 1983-3652
DOI: 10.17851/1983-3652.12.3.196-208

generated.
KEYWORDS: management system; growth curve; uroculture.

1 Introducción

Las Tecnologías de la Información y las Comunicaciones (TIC) se han extendido como apoyo a los procesos fundamentales que son realizados por el hombre, La ciencia de la salud es unos de los campos más beneficiados por el uso de las tecnologías como parte del diagnóstico médico. Como soporte al diagnóstico médico con el uso de las tecnologías son introducidos los laboratorios clínicos automatizado (HERNÁNDEZ, 2016, p. 8). Un laboratorio clínico automatizado permite realizar un mayor número de determinaciones en un menor tiempo, aumentándose la eficiencia en el proceso de diagnóstico médico.

Diversos han sido los laboratorios clínicos automatizados desarrollado como soporte al diagnóstico médico (FIGUEROA, 2017, p. 18), (RONQUILLO ALVAREZ, 2015, p. 54), (TERCEROS, LAZARTE, 2017, p. 18). En Cuba el Centro Nacional de Investigaciones Científica (CNIC) entre los resultados obtenidos se relacionan las investigaciones en el campo de la Urología. Un resultado a destacar en este campo los representa la herramienta DIRAMIC como parte del laboratorio clínico (ESPINO, 2010, p. 79).

La herramienta DIRAMIC en su proceso de mejora continua de la calidad, se le detectaron no conformidades identificándose las siguientes deficiencias:

- La herramienta posee informaciones irrelevantes en los campos de entradas de los datos de los urocultivos.
- Se muestran resultados de falsos negativos relacionados con microorganismos de crecimiento lento.
- No implementa mecanismos de validación que restrinja la introducción de información duplicada.
- No implementa visualización del crecimiento de los microorganismos de una muestra en tiempo real.

A partir del análisis sobre las deficiencias identificadas se provoca gestión de datos irrelevante, almacenamiento de informaciones innecesarias lo que trae como consecuencias que se ralenticen las búsquedas de informaciones en la generación de reportes. Además no se identifican las distintas fases del crecimiento bacteriano, para un resultado preciso en el procesamiento de los datos de los urocultivos de los pacientes.

Problemas como los antes mencionados han sido abordados en la literatura científica a partir del estudio de las curvas de crecimiento de la unidad formadora de colonias (UFC)/ml, lectura en tiempo real y el comportamiento de los microorganismos existentes dentro de una muestra urinaria (DÍAZ, 2017, p. 7; LEONARD *et al.*, 2017, p. 25). A partir de la problemática antes mencionada se define como objetivo de la presente investigación desarrollar un sistema para el análisis de muestra de urocultivo a partir curvas de crecimiento de la unidad formadora de colonias (UFC)/ml utilizándose

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 12, n. 3, p. 196-208, set.-dez. 2019 – ISSN 1983-3652
DOI: 10.17851/1983-3652.12.3.196-208

tecnologías libres para su desarrollo.

2 Materiales y método

Para el desarrollo del sistema de gestión de muestras de los urocultivos, es necesario adoptar un grupo metodologías, herramientas y tecnologías. A continuación, se explica, las herramientas y tecnologías adecuadas en el desarrollo de la propuesta.

En el presente trabajo se decidió emplear la metodología AUP-UCI, que propone 4 fases (Inicio, Elaboración, Construcción, Transición) para el ciclo de vida de los proyectos, empleándose el escenario No 2 que modela el negocio con Modelo Conceptual y modelar el sistema mediante Casos de Uso del Sistema.

Para formalizar el desarrollo del sistema se requiere que cada una de las partes que comprende el desarrollo del software sea visualizado, especificado y documentado con lenguaje común (CORNELO, 2008, p. 9). Para lo cual se empleó el Lenguaje Unificado de Modelado (UML) por sus siglas en ingles Unified Modeling Language. Para el modelado de la solución se utilizó el Visual Paradigm para UML 8.0, herramienta profesional que soporta el ciclo de vida completo de desarrollo de un software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad (PRESSMAN, 2008, p. 280).

Los Sistemas de Gestión de Bases de Datos (SGBD), convierten el acceso a los datos y su gestión en una aplicación cerrada, interponiéndose entre los usuarios y los ficheros, y haciéndose cargo de todos los problemas de explotación, mantenimiento y comprobación de los datos. De esta manera el usuario pierde de vista todos los detalles relativos al almacenamiento físico de los datos al utilizarlos sólo a través de un lenguaje conceptual sencillo (RIPOLL, 2008, p. 10). Par el desarrollo de la propuesta se utilizó PostgreSQL 9.4 ya que es un sistema de gestión de bases de datos objeto-relacional. Utiliza un modelo cliente/servidor y usa multiprocesos para garantizar la estabilidad del sistema (SARRÍA, 2006).

Se utiliza como entorno de trabajo Hibernate que tiene como objetivo facilitar la persistencia de objetos Java en bases de datos relacionales y al mismo tiempo la consulta de estas bases de datos para obtener objetos, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones. Es una herramienta ORM (Mapeador Objeto - Relación) de código abierto (PUETAMAN, TOLEDO, CASTRO, ROSERO, 2016, p. 9; RODRÍGUEZ, ROELY, 2017, p. 68).

Se seleccionó IntelliJ idea como entorno de desarrollo integrado, es compatible con la finalización de código basado en texto. Ofrece una lista de los símbolos más relevantes aplicables en el contexto actual. Posee una función avanzada de finalización de código que enumera los símbolos aplicables, accesible a través de métodos o captadores en el contexto actual. Le permite utilizar fácilmente constantes o métodos estáticos y agrega automáticamente las declaraciones de importación necesarias para evitar un error de compilación (BASELGA, CARRETERO, MARTÍN, 2016. p. 38).

Se decidió utilizar Java como lenguaje de programación. De acuerdo con Sun Microsystems, Java es "simple, orientado a objetos, tipificado estáticamente, compilado,

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 12, n. 3, p. 196-208, set.-dez. 2019 – ISSN 1983-3652
DOI: 10.17851/1983-3652.12.3.196-208

independiente de arquitectura, multi-procesos, con recolector de basura, robusto, seguro y ampliable.

Para el desarrollo de interfaces visuals se utilizó JavaFX v11.0 es un nuevo lenguaje diseñado para animaciones y gráficos. Permite la vinculación de componentes escritos en otros lenguajes (principalmente en Java) con rapidez, código simple y completamente orientado a objetos. Es una familia de productos y tecnologías pensados para el desarrollo de “Rich Internet Applications” (RIA’s). Este producto es la contrapartida de SUN, para las herramientas de desarrollo de este tipo de aplicaciones, como Flash de ADOBE y Silverlight de Microsoft.

3 Resultado y discusiones

El Modelo Conceptual tiene como objetivo identificar los conceptos significativos en el negocio objeto de estudio, así como exponer los atributos y las asociaciones existentes entre ellos. Dichas entidades no son componentes del software, representan el escenario actual antes de implementar la propuesta de solución. La Figura 1 muestra el modelo conceptual que representa la descripción del proceso.

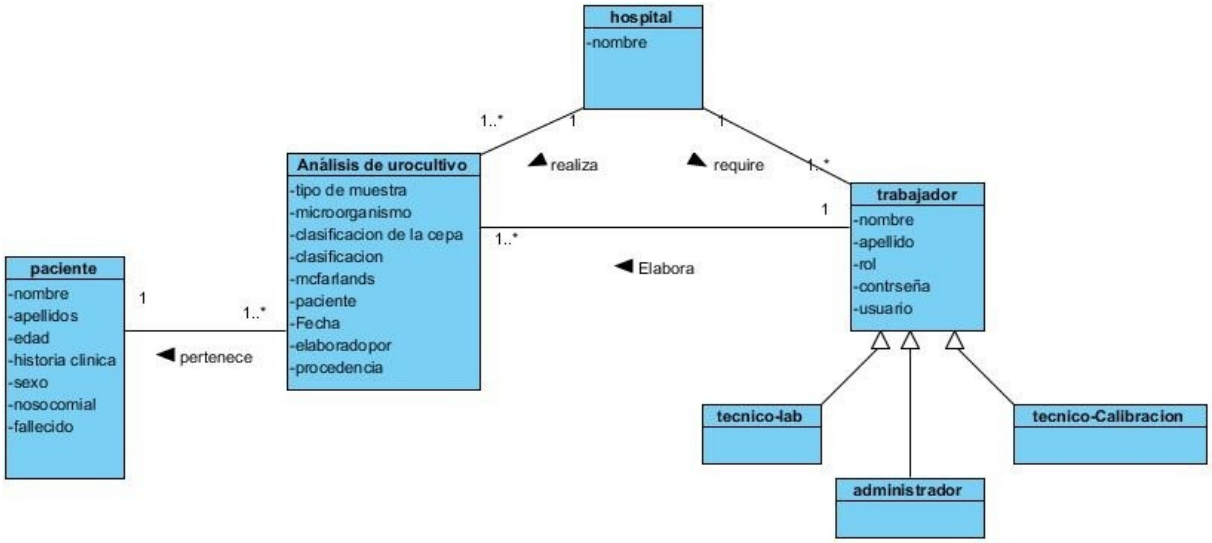

Figura 1: Diagrama de modelo conceptual.
Fuente: elaboración propia.

Los principales conceptos representados son:

- Hospital: Centro hospitalario donde se lleva a cabo el análisis de los urocultivos.
- Paciente: Persona que contiene la muestra que más tarde será analizada.
- Urocultivo: Muestra urinaria dada por el paciente para ser examinada.
- Trabajador: Personal capacitado para trabajar en los laboratorios de microbiología.
- Administrador: Persona encargada de gestionar los usuarios.
- Tecnico_lab: Persona encargada de dar los resultados del análisis de las muestras.

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 12, n. 3, p. 196-208, set.-dez. 2019 – ISSN 1983-3652
DOI: 10.17851/1983-3652.12.3.196-208

- Tecnico_calibrador: Persona encargada de la calibración del sistema.

A partir de la identificación de los principales conceptos asociados al dominio del problema, como parte del proceso de desarrollo de Software, se realiza la Ingeniería de Requisitos, donde se lleva a cabo el proceso de descubrir, analizar, escribir y verificar los servicios y restricciones, del sistema que se desea producir. El proceso se realiza mediante la obtención, el análisis, la especificación, la validación y la administración de los requerimientos del software. Los requisitos de software son las necesidades de los clientes, los servicios que los usuarios desean que proporcione el sistema de desarrollo y las restricciones en las que debe operar (SOMMERVILLE, 2005).

Para la solución propuesta se identificaron 28 Requisitos Funcionales que se relacionan a continuación: RF1. Autenticar usuario (administrador), RF2. Insertar los datos del usuario (administrador), RF3. Modificar los datos del usuario (administrador), RF4. Eliminar usuario (administrador), RF5. Buscar usuario a partir de los datos insertado (administrador), RF6. Listar usuarios (administrador), RF7. Cambiar contraseña de usuarios (tec lab, tec calibración, administrador), RF8. Leer la concentración de células de cada muestra de urocultivo (tec lab), RF9. Insertar datos de una muestra de urocutivo (tec lab), RF10. Mostrar los datos de una muestra del urocultivos (tec lab), RF11. Modificar los datos de una muestra de urocultivo (tec lab), RF12. Detener la lectura de la muestra seleccionada (tec lab), RF13. Eliminar los datos de una muestra de urocultivo que fue detenida (tec lab), RF14. Buscar los datos de un paciente por historia clínica (tec lab), RF15. Insertar los datos del paciente que tiene una muestra de urocultivo (tec lab), RF16. Modificar los datos del paciente que tiene una muestra de urocultivo (tec lab), RF17. Insertar los datos del paciente (tec lab), RF18. Modificar los datos del paciente (tec lab), RF19. Eliminar los datos del paciente (tec lab), RF20. Buscar los datos de un paciente (tec lab), RF21. Listar los datos del paciente (tec lab), RF22. Buscar los datos de una muestra de urocultivo (tec lab), RF23. Listar los datos de una muestra de urocultivo (tec lab), RF24. Modificar datos de una muestra de urocultivo registrado (tec lab), RF25. Eliminar datos de una muestra de urocultivo registrado (tec lab), RF26. Imprimir reporte del urocultivo (tec lab), RF27. Calibración del sistema automático (tec calibración), RF28. Calibración del calibrador independiente. (tec calibración).

Para el contexto de la presente investigación se identificaron los actores del sistema. Los representan las distintas personas que intervienen en la ejecución del sistema, los actores representan funciones que una persona real realiza (WEITZENFELD, 2005, p. 5). Se identificaron los siguientes actores:

- Administrador: Personal encargado de asignar los permisos a los usuario.
- Tec lab: Personal encargado de llevar a cabo la realización de los análisis de urocultivos, así como la gestión de todos los pacientes del sistema.
- Tec Calibración: Personal encargado de realizar la calibración del de sistema en cuestión.

A partir de la definición de los actores del sistema, se establece los Casos de Usos del Sistema, se documentan con el empleo del Diagrama de Caso de Uso. El conjunto de Casos de Uso representa todas las interacciones posibles que se describirán en los

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 12, n. 3, p. 196-208, set.-dez. 2019 – ISSN 1983-3652
DOI: 10.17851/1983-3652.12.3.196-208

requerimientos del sistema. Los actores en el proceso, que pueden ser individuos u otros sistemas. (SOMMERVILLE, 2005, p. 54; CORIA, ZAWADSKI, 2019, p. 11; MAR, CABRERA, 2016, p. 7). La Figura 2 muestra el Diagrama de Casos de Uso de la propuesta.

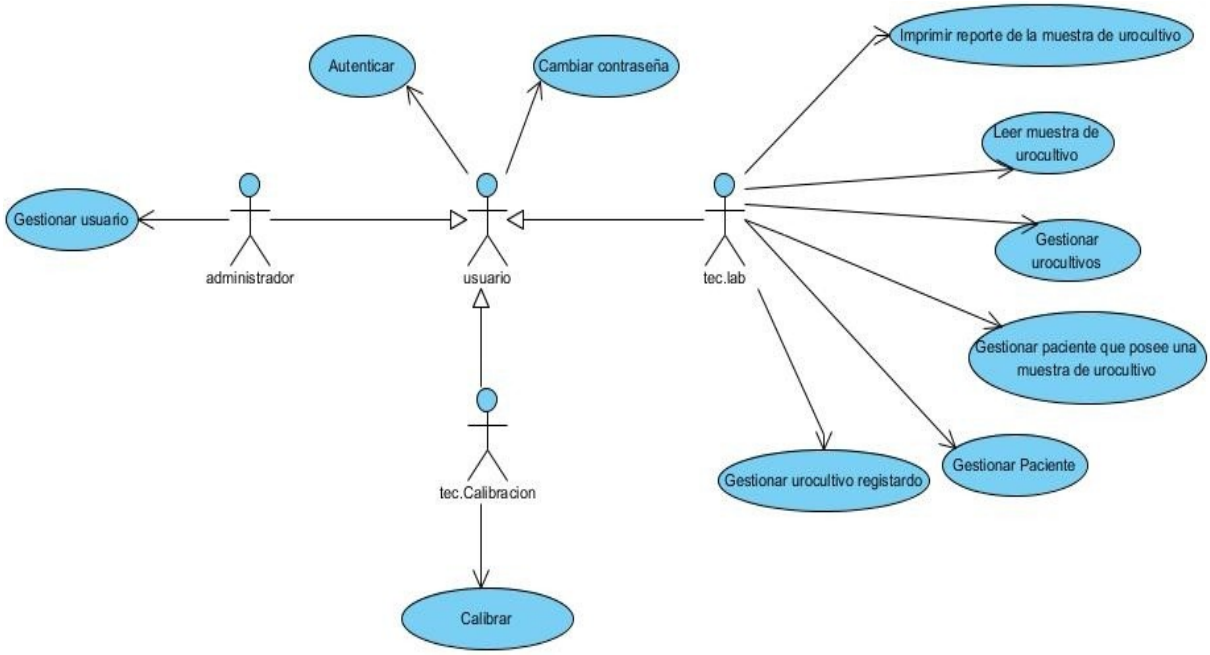

Figura 2: Diagrama de Caso de Uso del Sistema.
Fuente: elaboración propia.

La definición de los Casos de Usos de Sistema permite realizar la agrupación de los requisitos del sistema en Casos de Usos del Sistema (CUS), para la presente investigación se realizó la agrupación de los 28 requisitos funcionales en 10 CUS tal como se describe a continuación:

- CUS1 (Autenticar usuario): RF1
- CUS2 (Gestionar usuario): RF2, RF3, RF4, RF5, RF6
- CUS3 (Cambiar contraseña de usuario): RF7
- CUS4 (Leer datos de las muestras de urocultivo): RF8, RF9, RF10
- CUS5 (Gestionar urocultivo): RF11, RF12, RF13, RF14, RF15
- CUS6 (Gestionar paciente que posee una muestra de urocultivo): RF16, RF17, RF18
- CUS7 (Gestionar paciente): RF19, RF20, RF21, RF22, RF23
- CUS8 (Gestionar urocultivo registrado): RF24, RF25, RF26
- CUS9 (Imprimir reporte de la muestra de urocultivo): RF27
- CUS10 (Calibrar): RF28

A partir del diagrama de Caso de Uso del Sistema se identificaron los patrones de Caso de Uso empleados:

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 12, n. 3, p. 196-208, set.-dez. 2019 – ISSN 1983-3652
DOI: 10.17851/1983-3652.12.3.196-208

CRUD (del inglés, Creating, Reading, Updating, Deleting) completo vigente en los CUS: “Gestionar urocultivo”, “Gestionar paciente”, “Gestionar urocultivo registrado”, “Gestionar paciente que posee una muestra de urocultivo”, “Gestionar usuario”

Múltiples actores (herencia o generalización entre actores) con roles comunes: el Administrador, tec.Calibración y tec.lab se comportan como un actor Usuario que se relaciona con el CUS “Autentica usuario” o “Cambiar contraseña”.

Para la realización del sistema a implementar se define utilizar el patrón arquitectónico Modelo-Vista-Controlador (MVC), ya que permite una separación de conceptos, de forma tal que el desarrollo del sistema esté estructurado de una mejor forma y facilite la programación de manera paralela e independiente en las diferentes capas, además de brindarle una mayor escalabilidad (ALMEIRA, 2007, p. 12). Como característica principal se destaca la manera que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos: el modelo, la vista y el controlador.

Mediante el diagrama de despliegue se muestra la configuración de los nodos que participan en la ejecución y los componentes que residen en ellos. Muestra la configuración de los elementos de procesamiento en tiempo de ejecución, los componentes de software, hardware, procesos y objetos que los ejecutan. Este diagrama es útil para ilustrar la arquitectura física de un sistema (JACOBSON, 2000, p. 125). La Figura 3 muestra el diagrama de despliegue de la solución propuesta.

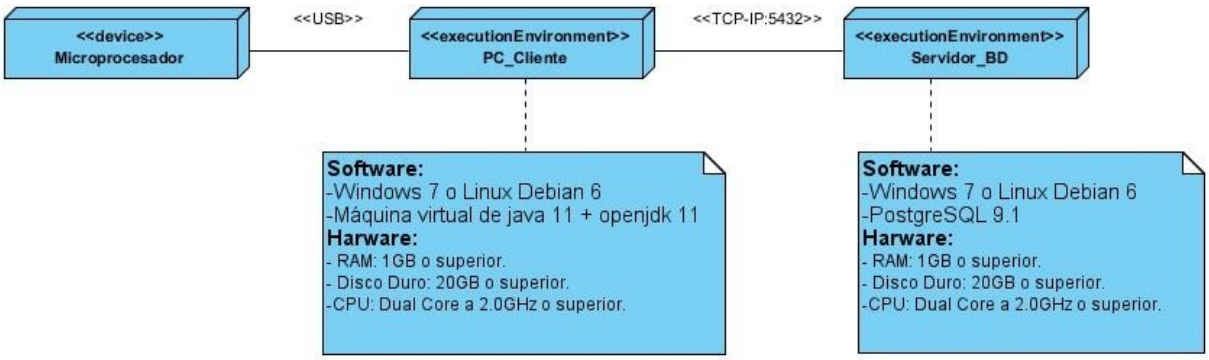

Figura 3: Diagrama de despliegue.
Fuente: elaboración propia.

Para interactuar con el sistema es necesario contar con una PC_Cliente que tenga instalado la máquina virtual de java 11 y el openjdk 11, además necesita conectarse con el micropocesador por puerto usb que permite la comunicación con el equipo automático. Los datos persistentes resultantes del sistema se almacenan en el servidor de bases de datos (Servidor_BD), accesible mediante el protocolo TCP-IP (protocolo de control de transmisión de datos entre computadoras, del inglés, Transmission Control Protocol – Internet Protocol).

Como resultado del despliegue general se obtiene el sistema propuesto, la figura 4 muestra una pantalla en ejecución de la propuesta en la que se introducen las muestras de urocultivo para su procesamiento por el sistema.

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 12, n. 3, p. 196-208, set.-dez. 2019 – ISSN 1983-3652
DOI: 10.17851/1983-3652.12.3.196-208

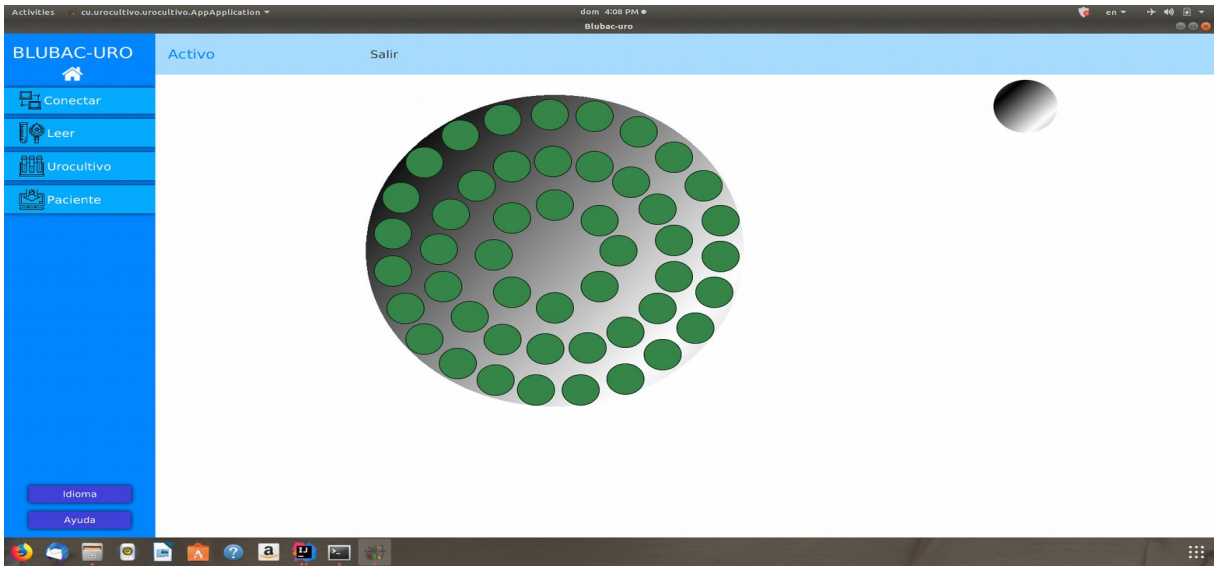

Figura 4: Interfaz leer muestra de Urocultivo.
Fuente: elaboración propia.

La Figura 5 muestra una pantalla en ejecución después de procesas una muestra de urocultivo donde se muestra la curva de crecimiento. La vista muestra además los datos atribuidos al paciente objeto de estudio y es almacenada para su posterior análisis por los especialistas.

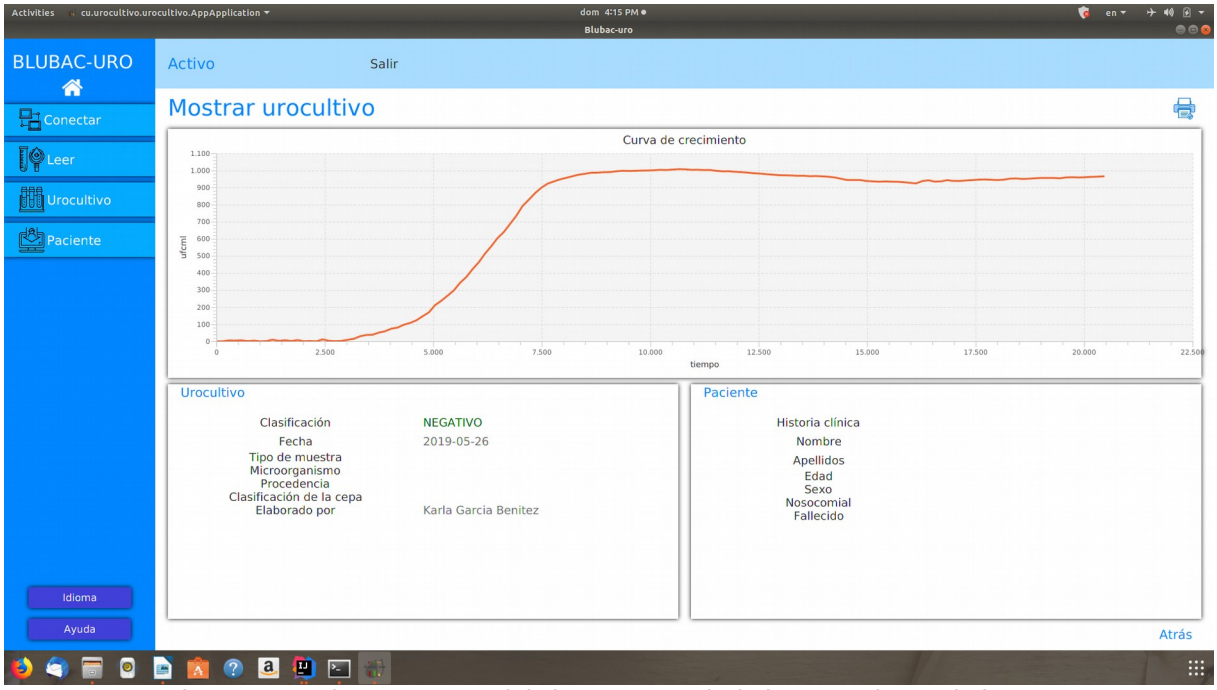

Figura 5: Interfaz respuesta del sistema a partir de la curva de crecimiento.
Fuente: elaboración propia.

Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 12, n. 3, p. 196-208, set.-dez. 2019 – ISSN 1983-3652
DOI: 10.17851/1983-3652.12.3.196-208

codificación. El objetivo de las mismas es encontrar el máximo número posible de errores con una cantidad manejable de esfuerzo aplicado en un período realista de tiempo (PRESSMAN, 2008, p. 345). Con las mismas se garantiza que el producto final funcione como fue diseñado e implemente de manera correcta los requisitos identificados.

Para dar inicio a las pruebas de un software lo primero es describir una estrategia de prueba donde quede plasmado los niveles de prueba a tratar, así como los tipos de prueba a emplear en cada nivel, los métodos de prueba a aplicar, así como las técnicas a utilizar para cada método. Las estrategias de pruebas describen y verifican el enfoque de la misma. Luego de realizar una estrategia de prueba se decidió aplicar pruebas de unidad y funcionalidad para medir el cumplimiento del objetivo de la presente investigación.

Las pruebas de unidad se realizarán con el objetivo de comprobar que el sistema implementado, entendido como una unidad funcional de un programa independiente, está correctamente codificado. Para la generación de casos de prueba de unidad, se decidió utilizar la técnica de camino básico del método de caja blanca y la técnica de partición equivalente del método de caja negra; para probar de la manera más completa posible el sistema implementado.

Para el método de caja blanca: las pruebas unitarias son una de las piedras angulares de AUP. Todos los sistemas deben de pasar las pruebas unitarias antes de ser liberados o publicados. Por otra parte, las pruebas deben ser definidas antes de realizar el código (“Test-driven programming”). Que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del código. En este sentido, el sistema y el conjunto de pruebas debe ser guardado junto con el código, para que pueda ser utilizado por otros desarrolladores, en caso de tener que corregir, cambiar o recodificar parte del mismo.

La detección y corrección de errores cuando se encuentra un error (“bug”), éste debe ser corregido inmediatamente, y se deben tener precauciones para que errores similares no vuelvan a ocurrir. Así mismo, se generan nuevas pruebas para verificar que el error haya sido resuelto (MORENO, 2013, p. 10). Las pruebas unitarias realizadas a través del sistema que ofrece Spring llamado test, se le fueron realizadas a las clases Grafica, Paciente, Urocultivo, GraficaRepository y AppApplication. La Figura 6 muestra el resultado de las pruebas unitarias que se realizan al sistema, utilizando el módulo test.

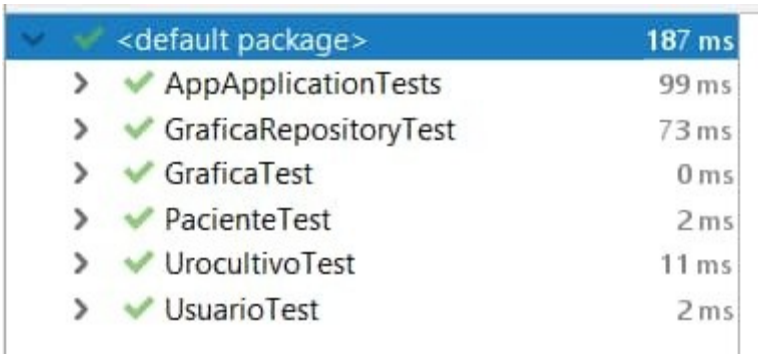

Figura 6: Resultado de las pruebas.
Fuente: elaboración propia.

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 12, n. 3, p. 196-208, set.-dez. 2019 – ISSN 1983-3652
DOI: 10.17851/1983-3652.12.3.196-208

El método de prueba de caja negra, también denominada prueba de comportamiento, se centra en los requisitos funcionales del software. O sea, permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales del programa.

Para el caso de las pruebas de caja negra, se aplicará la técnica de partición equivalente, la cual divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar (PRESSMAN, 2008, p. 332; CRUZ, TANDAZO, 2017, p. 20).

Para el desarrollo de la prueba, el Diseño de Casos de Prueba (DCP) para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. A continuación, se muestra un ejemplo de los DCP realizados, específicamente el del CUS Gestionar urocultivo.

DCP del CUS: Gestionar urocultivo

Descripción general: El CUS se inicia cuando el usuario selecciona de la interfaz “leer”, el círculo que representa una muestra de urocultivo y culmina con la realización de una de las acciones: Insertar, Mostrar, Modificar, Detener la lectura de la muestra seleccionada o Eliminar los datos de una muestra de urocultivo que fue detenida

Condiciones de ejecución: El usuario debe estar autenticado en el sistema, así como contar con los permisos para realizar alguna de las operaciones antes mencionadas. En el caso de que se pretenda Modificar, Mostrar o Detener alguna muestra de urocultivo, este debe haber sido insertado anteriormente al sistema, para Eliminar la muestra debe haber sido detenida con anterioridad. En el caso de que se pretenda Insertar los datos de alguna muestra de urocultivo, el usuario solo podrá realizar la acción si la muestra de urocultivo fue leída e insertada en la base de datos, en ese caso podrá insertar los datos o no ejecutar la acción hasta se haya leído una muestra de urocultivo.

Resultado de la aplicación de las pruebas: Como parte de la ejecución de las pruebas de caja negra se realizaron 12 casos de pruebas en 3 iteraciones de pruebas para comprobar el correcto funcionamiento del sistema. En la primera iteración se identificaron 8 no conformidades (NC). Una vez corregidas, se procedió a realizar una segunda en la que se identificaron 3 nuevas NC y finalmente se realizó una iteración en la que no se encontró deficiencias que, razón por la que se definió no realizar más iteraciones.

Para mayor entendimiento de este proceso se muestra en la Tabla 1 la descripción de las iteraciones realizadas durante las pruebas por cada iteración.

Tabla 1: Resultado de la aplicación de las pruebas.

			4 Usab - Estética y diseño	Medio
1		**8**		

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 12, n. 3, p. 196-208, set.-dez. 2019 – ISSN 1983-3652
DOI: 10.17851/1983-3652.12.3.196-208

2	**3**	2 Validación	Alto
3	**0**	No se identifican	No procede

Fuente: elaboración propia.

Las iteraciones de pruebas realizadas permitieron garantizar que el sistema cumplió satisfactoriamente los requerimientos identificados garantizándose que cumpla con las especificaciones que se trazaron obteniéndose un sistema que satisface las necesidades de la problemática planteada.

4 Conclusiones

Durante el proceso de desarrollo se obtuvieron y especificaron los requisitos a tener en cuenta para el cumplimiento de las necesidades planteadas como problemática de la investigación que permitieron la correcta implementación del sistema propuesto.

Las tecnologías y herramientas utilizadas para el desarrollo de la solución propuesta, se basaron en plataformas soberanas de tecnologías libre a partir de las cuales se alcanzaron los resultados deseados.

La realización de las distintas pruebas de software permitió comprobar el cumplimiento de los requisitos especificados para la propuesta de sistema garantizándose un producto libre de errores.

Referências

ALMEIRA, A. S. A. V. P. C. *Arquitectura de Software*: Estilos y Patrones Universidad Nacional De La Patagonia San Juan Bosco. Argentina, 2007.

BASELGA, S. V.; CARRETERO, A. B.; MARTÍN, J. R. Los medios comunitarios, libres y ciudadanos en el Estado español. Territorios, tecnologías y valores. *Cultura, lenguaje y representación*: revista de estudios culturales de la Universitat Jaume I, 15, p. 99-118, 2016.

CORIA, J. C. G.; ZAWADSKI, K. J. Aplicación de la biotecnología en cítricos para el desarrollo de plantas libres de patógenos en Paraguay. *Revista de Ciencia y Tecnología*, p. 16-21, 2019.

CORNELO, J. E. G. *¿Qué es UML?* El lenguaje de Modelado Unificado, 2008.

CRUZ, I. T. M.; TANDAZO, K. I. C. Incidencia del software libre en el sector comercial del Ecuador: una revisión literaria. *Espirales revista multidisciplinaria de investigación*, 1(10), 2017.

Belo Horizonte, v. 12, n. 3, p. 196-208, set.-dez. 2019 – ISSN 1983-3652
DOI: 10.17851/1983-3652.12.3.196-208

DÍAZ, V. Caracterización molecular de genes de resistencia a ß-lactámicos en aislados clínicos de Escherichia coli provenientes de urocultivos y pruebas de inhibición con péptidos de Boana rosenbergi y Rana sp. *PUCE*, 2017.

ESPINO, M.; ÁLVAREZ, E.; ZAYAS, Á.; CONTRERAS, R. Detección de cepas productoras de b-lactamasas de espectro extendido por el sistema DIRAMIC: Comparación con los métodos doble difusión con discos y E-test. *Revista chilena de infectología*, 27, 2010.

FIGUEROA, L. Normatividad relacionada al control de calidad analítica en los laboratorios clínicos del Perú. *Acta Médica Peruana*, 34(3), p. 237-243, 2017.

HERNÁNDEZ, B. *Plataforma tecnológica para automatización del proceso de recolección de muestras de laboratorio de referencia*, 2016. Disponible: http://ri.ibero.mx/bitstream/handle/ibero/477/016165s.pdf?sequence=1. Consultado: 25 dez. 2019.

JACOBSON. *El proceso unificado de desarrollo de software.* P. E. S.A Ed., 2000.

LEONARD, I.; VARGAS, B.; UVIDIA, H., TORRES, V.; VERDECIA, D.; RAMÍREZ, J.; ANDINO, M. Las curvas de crecimiento de dos pastos introducidos en Ecosistemas Amazónicos. REDVET. *Revista Electrónica de Veterinaria*, 18(7), 2017.

MAR, O.; CABRERA, M. Práctica de Microbiología y Parasitología Médica integrado al Sistema de Laboratorios a Distancia en la carrera de Medicina. *Rev. Ciencias Médicas de Pinar del Río*, v. 20, n. 2, p. 174-181, 2016.

MORENO, S. *Manejo de sistemas de información para la organización de procesos de la gerencia de protección y aseguramiento de ingresos de la compañía*, 2013.

PRESSMAN, R. S. *Ingeniería de Software*. Sexta Edición ed., 2008.

PUETAMAN, I. M. A.; TOLEDO, R. A. J.; CASTRO, D. M. E.; ROSERO, D. C. Desarrollo de Apps: un estudio comparativo entre frameworks libres y privativos. *Boletín Informativo CEI*, 2(3), 2016.

RIPOLL, L. Q. Sistemas de gestión de bases de datos. Master en Ingenería Medioambiental y Gestión del Agua. *Escuela de Negocios*, 2008.

RODRÍGUEZ, C.; ROELY, D. *Arquitectura de hardware y software libres para un dispositivo de rastreo de vehículos en tiempo real*. Universidad Central “Marta Abreu" de Las Villas. Facultad de Ingeniería, 2017.

RONQUILLO ALVAREZ, R. M. *Diseño de un prototipo de software para la administración de laboratorios clínicos, enfocado en el cálculo de costo y estimación de ganancia o perdida por determinación/pruebas*. Universidad de Guayaquil Facultad de Ciencias Matemáticas y Física Carrera, 2015.

Belo Horizonte, v. 12, n. 3, p. 196-208, set.-dez. 2019 – ISSN 1983-3652
DOI: 10.17851/1983-3652.12.3.196-208

SARRÍA, F. A. *Programación en SQL con PostgreSQL.* 2006.

SOMMERVILLE, I. *Ingenieria del Software*. Pearson Educacion, S.A ed. v. 7, 2005.

TERCEROS, M.; LAZARTE, A.; BALDELOMAR, F.; CHUCUSEA, M. Enfisema lobar congénito. Presentación de caso clínico. *Revista Científica Ciencia Médica*, 20, p. 44-46, 2017.

WEITZENFELD, A. *Ingeniería de software orientada a objetos con UML, Java e Internet*. Cengage Learning Editores, 2005.

Recebido em dia 03 de junho de 2019.
Aprovado em dia 30 de setembro de 2019.