



PROYECTO FINAL - MEDIVOUCHER

Validación y Verificación de Software

- Juan J. Miño
- Dennis Ocaña
- Oscar Albuja
- Martin Lomas

09 de julio, 2024



Contenido

1	Identificación del problema dentro de las restricciones del proyecto	4
1.1	Introducción.....	4
1.2	Descripción general	4
1.2.1	Objetivo	4
1.2.2	Alcance.....	4
1.2.3	Justificación.....	4
1.2.4	Supuestos.....	5
1.2.5	Restricciones.....	5
1.2.6	Riesgos	5
1.3	Requisitos específicos	5
2	Selección de metodologías integrales para solucionar el problema	6
2.1	Establecimiento de propuestas de solución	6
2.1.1	Especificación de requisitos.....	7
2.1.2	Historias de usuario	1
2.2	Descripción de la funcionalidad y uso de Time to market implementados	1
2.3	Descripción y justificación de los tipos de prueba seleccionados.....	3
2.3.1	Casos de uso con Selenium.....	3
2.3.2	Análisis Estático de Código con SonarCloud	4
2.3.3	Pruebas de carga con JMeter	4
3	Diseño de una propuesta técnica para el problema	5
3.1	Diseño de clases de la implementación (Diagramas de clases y ER).....	5
3.2	Descripción de la implementación del sistema informático	6
3.3	Casos de prueba.....	8
3.4	Resultados de la ejecución de los casos de prueba.....	9
3.5	Informe de resultados de la prueba	11
3.5.1	Reporte de Resultados Caso de Prueba con ID: SCA-003.....	11
3.5.2	Reporte de Resultados Caso de Prueba con ID: P-018	12
3.5.3	Reporte de Resultados Caso de Prueba con ID: C-003	13
3.6	Video final	13
3.7	Código fuente	14
4	Anexos	14

4.1	Anexo 1: IEEE 830	14
4.2	Anexo 2: Casos de prueba	14
4.3	Anexo 3: Sistema desplegado	14

1 Identificación del problema dentro de las restricciones del proyecto

1.1 Introducción

El proyecto MediVoucher tiene como objetivo desarrollar un sistema que genere y gestione cupones para productos farmacéuticos, además de la administración básica de marcas, presentaciones y cadenas farmacéuticas. Este sistema busca mejorar la accesibilidad a medicamentos esenciales para grupos vulnerables y fomentar la fidelización de clientes a través de promociones estratégicas.

MediVoucher sirve como la base de este proyecto de validación y verificación de software, cuyo objetivo fundamental es aplicar conocimientos especializados para analizar, diseñar e implementar una solución que permita realizar un diseño de pruebas para validar el funcionamiento del sistema.

1.2 Descripción general

1.2.1 Objetivo

El objetivo principal de MediVoucher es generar y gestionar cupones para productos farmacéuticos. Además de la administración básica de marcas, presentaciones, cadenas farmacéuticas, y objetos promocionales.

1.2.2 Alcance

El sistema generará cupones mediante códigos alfanuméricos según las presentaciones de los medicamentos existentes, clasificados por marcas, y divididos por cadenas farmacéuticas. Permitirá la gestión (CRUD) de todas las entidades involucradas dentro del sistema.

1.2.3 Justificación

El desarrollo de este sistema se justifica por su capacidad para impulsar el flujo de clientes hacia las cadenas farmacéuticas mediante la implementación estratégica de promociones en medicamentos. Al ofrecer descuentos y beneficios concretos, el sistema no solo fomenta un incremento en las ventas, sino que también promueve la fidelización de los clientes al proporcionar valor agregado a su experiencia de compra.

MediVoucher está diseñado para prestar apoyo significativo a individuos bajo medicación continua, particularmente aquellos que se encuentran en grupos de alto riesgo, como son las personas mayores, diabéticas, y otros. Facilitando el acceso a medicamentos esenciales a través de cupones y descuentos. El sistema contribuye a mejorar la adherencia al tratamiento y la gestión de la salud de estos pacientes, al mismo tiempo que reduce la carga económica que representan los costos farmacéuticos continuos.

1.2.4 Supuestos

Dentro del diseño de la propuesta del sistema se establecieron los siguientes supuestos que garantizan el correcto funcionamiento del sistema:

- Se supone que se posee la información de los productos que las farmacéuticas están dispuestas a reducir su precio final para ofrecer el respectivo descuento.
- Se supone que se cuenta con diversas cadenas farmacéuticas dispuestas a colaborar con el aplicativo MediVoucher.
- Se supone que cada una de las cadenas farmacéuticas maneja una variedad de marcas y presentaciones.

1.2.5 Restricciones

Además de los supuestos, se establecieron las siguientes restricciones dentro de la propuesta, las cuales deberán ser tomadas en cuenta en su diseño y desarrollo:

- La aplicación debe desarrollarse únicamente con software libre.
- La aplicación no cuenta con ningún presupuesto para su realización.
- La aplicación debe realizarse a través de OpenXava.
- La aplicación debe estar lista antes del mes de julio del 2024.

1.2.6 Riesgos

Es importante tomar en cuenta los posibles riesgos involucrados en el desarrollo del sistema para poder tener una prever posibles soluciones y cambios al cronograma del proyecto, es por eso por lo que se identificaron los siguientes riesgos:

- Cortes de luz que disminuyen la productividad del equipo y el tiempo de trabajo disponible.
- Falta de presupuesto que limita las capacidades de implementación y tecnologías a disposición.
- Riesgo de que las herramientas de software libre seleccionadas no se integren eficientemente, lo que podría complicar el desarrollo y mantenimiento del sistema.
- Riesgo de no cumplir con los plazos de entrega establecidos, afectando las evaluaciones y la entrega final del proyecto.

1.3 Requisitos específicos

MediVoucher está diseñado para ofrecer una gestión eficiente de cupones para productos farmacéuticos, garantizando funcionalidades esenciales para el manejo de información y promociones, es por esto por lo que se establecen los siguientes requisitos funcionales y no

funcionales a un alto nivel. La especificación de requisitos con mayor detalle se encuentra en la siguiente sección del documento:

Requisitos funcionales

- **Generación de cupones:** Capacidad para crear cupones alfanuméricos únicos aplicables a promociones específicas.
- **Gestión de datos:** Facilidades para administrar información detallada sobre marcas, presentaciones de medicamentos y cadenas farmacéuticas, incluyendo la localización y ofertas especiales.
- **Interfaz de usuario:** Desarrollo de una interfaz clara y accesible, optimizada para facilitar la visualización de ofertas y la gestión de información relevante.

Requisitos no funcionales:

- **Usabilidad:** Diseño intuitivo y fácil manejo, destinado a usuarios de entre 18 y 40 años, con adaptabilidad a diversos dispositivos y plataformas.
- **Rendimiento y Disponibilidad:** Alta capacidad de respuesta, con tiempos de carga menores a 2 segundos para operaciones críticas, y alta disponibilidad para garantizar un acceso continuo y sin interrupciones.
- **Mantenibilidad y Portabilidad:** Facilidad de mantenimiento y actualización, con documentación detallada para soportar las operaciones y futuras mejoras. El sistema debe ser compatible con diferentes entornos de hosting y navegadores web.

2 Selección de metodologías integrales para solucionar el problema

2.1 Establecimiento de propuestas de solución

- **Solución 1:** Desarrollo de un aplicativo Web basado en OpenXava
La solución propuesta utilizando OpenXava se enfoca en el desarrollo de un sistema web para la gestión de cupones farmacéuticos. OpenXava, un framework de código abierto para Java, permite una implementación rápida y eficiente gracias a su enfoque basado en modelos, que automatiza la creación de interfaces de usuario y la lógica de persistencia. Esta solución facilitaría la generación y administración de cupones alfanuméricos, así como la gestión de marcas, presentaciones y cadenas farmacéuticas. Con OpenXava, se lograría una interfaz intuitiva y accesible, optimizada para la visualización y manejo de ofertas, lo cual es crucial para mejorar la accesibilidad a medicamentos y fomentar la fidelización de clientes. Además, su compatibilidad con diversas bases de datos y dispositivos móviles asegura una amplia adaptabilidad y mantenibilidad del sistema.

- **Solución 2:** Aplicación instalable

La aplicación instalable sería una solución robusta y versátil que los usuarios podrían instalar en sus computadoras personales. Al ejecutarla, se abriría una interfaz intuitiva y amigable que permitiría la gestión de cupones farmacéuticos sin necesidad de conexión a Internet. Los usuarios podrían registrar sus cuentas, crear nuevos cupones y editar los existentes. Además, podrían realizar búsquedas avanzadas, filtrar cupones por fecha de vencimiento o descuento, y exportar sus datos a archivos locales. En resumen, esta solución proporcionaría una experiencia de usuario fluida y confiable, ideal para aquellos que prefieren trabajar sin depender de la web.

- **Solución 3:** Desarrollar una aplicación web utilizando el framework Laravel

La aplicación web basada en Laravel y PHP sería una plataforma centralizada para la gestión de cupones farmacéuticos. Los usuarios accederían a través de un navegador web y se autenticarían en la plataforma. Los administradores tendrían la capacidad de crear y editar cupones, mientras que los usuarios podrían ver y aplicar los cupones disponibles. La aplicación incluiría funciones avanzadas, como la asignación de roles y permisos para los usuarios, notificaciones automáticas sobre nuevos cupones o cambios en los existentes, y la generación de informes detallados sobre el uso de cupones. En resumen, esta solución ofrecería una experiencia web completa, escalable y fácil de mantener, ideal para empresas farmacéuticas o cadenas de farmacias.

2.1.1 Especificación de requisitos

Los requisitos especificados se encuentran detallados en el formato IEEE 830 adjunto en el Anexo 1 y subido en el enlace del repositorio de Github. Sin embargo, aquí se encuentran los más generales y fundamentales:

- **Generación de Cupones:** Capacidad para crear cupones alfanuméricos únicos para promociones específicas.
- **Verificación de Cupones:** Capacidad para verificar la validez cupones alfanuméricos únicos para promociones específicas.
- **Gestión de Datos:** Administrar información detallada sobre marcas, presentaciones de medicamentos y cadenas farmacéuticas a través de la creación, edición y eliminación con su respectiva visualización de estas.
- **Interfaz de Usuario:** Desarrollo de una interfaz clara y accesible, optimizada para la visualización de ofertas y gestión de información relevante.
- **Inicio de Sesión:** Un inicio de sesión de administrador para realizar cambios en la aplicación, generar cupones y verificar cupones.

2.1.2 Historias de usuario

EPICA				HISTORIA DE USUARIO				OTROS DATOS DE LA EPICA O HISTORIA DE USUARIO						
ID Epica	Como (Rol)	Deseo...	Para...	ID Historia de Usuario	Como (Rol)...	Deseo....	Para....	Criterios de Aceptación	Prioridad	Estimación	Dependencias	Sprint	Estado	Comentarios
EPIC01	Usuario	Tener un sistema de gestión de promociones para medicamentos	fidelizar mis clientes y ofrecer descuentos											
				HU01	Usuario	Poder iniciar sesión al sistema haciendo uso de credenciales	Hacer uso del aplicativo	Se inicia sesión por medio de la pantalla de inicio de sesión	1	1		1	Terminado	Ninguno
				HU02	Usuario	Poder crear registros de farmacias	Añadir nueva información	Es posible añadir registros en la base de datos con todas sus validaciones necesarias y los campos requeridos por el cliente	2	4		1	Terminado	Ninguno
				HU03	Usuario	Poder leer los registros de farmacias	Conocer la información existente	Es posible leer los registros de la base de datos en una tabla	2	1	HU02	1	Terminado	Ninguno
				HU04	Usuario	Poder actualizar los registros de farmacias	Modificar la información existente	Es posible modificar registros en la base de datos con todas sus validaciones necesarias y los campos requeridos por el cliente	2	3	HU03	1	Terminado	Ninguno
				HU05	Usuario	Poder eliminar los registros de farmacias	Eliminar la información existente	Es posible eliminar los registros de la base de datos en una tabla	2	1	HU03,HU4	1	Terminado	Ninguno
				HU06	Usuario	Poder crear registros medicamentos	Añadir nueva información	Es posible añadir registros en la base de datos con todas sus validaciones necesarias y los campos requeridos por el cliente	2	4		1	Terminado	Ninguno
				HU07	Usuario	Poder leer los registros de medicamentos	Conocer la información existente	Es posible leer los registros de la base de datos en una tabla	2	1	HU06	1	Terminado	Ninguno

				HU08	Usuario	Poder actualizar los registros de medicamentos	Modificar la información existente	Es posible modificar registros en la base de datos con todas sus validaciones necesarias y los campos requeridos por el cliente	2	3	HU07	1	Terminado	Ninguno
				HU09	Usuario	Poder eliminar los registros de medicamentos	Eliminar la información existente	Es posible eliminar los registros de la base de datos en una tabla	2	1	HU07,HU08	1	Terminado	Ninguno
				HU10	Usuario	Poder crear registros de presentaciones	Añadir nueva información	Es posible añadir registros en la base de datos con todas sus validaciones necesarias y los campos requeridos por el cliente	2	4		1	Terminado	Ninguno
				HU11	Usuario	Poder leer los registros de presentaciones	Conocer la información existente	Es posible leer los registros de la base de datos en una tabla	2	1	HU10	1	Terminado	Ninguno
				HU12	Usuario	Poder actualizar los registros de presentaciones	Modificar la información existente	Es posible modificar registros en la base de datos con todas sus validaciones necesarias y los campos requeridos por el cliente	2	3	HU11	1	Terminado	Ninguno
				HU13	Usuario	Poder eliminar los registros de presentaciones	Modificar la información existente	Es posible modificar registros en la base de datos con todas sus validaciones necesarias y los campos requeridos por el cliente	2	1	HU011,H12	1	Terminado	Ninguno
				HU14	Usuario	Poder crear registros de promociones	Añadir nueva información	Es posible añadir registros en la base de datos con todas sus validaciones necesarias y los campos requeridos por el cliente	2	4		1	Terminado	Ninguno

				HU15	Usuario	Poder leer los registros de promociones	Conocer la información existente	Es posible leer los registros de la base de datos en una tabla	2	1	HU14	1	Terminado	Ninguno
				HU16	Usuario	Poder actualizar los registros de promociones	Modificar la información existente	Es posible modificar registros en la base de datos con todas sus validaciones necesarias y los campos requeridos por el cliente	2	3	HU15	1	Terminado	Ninguno
				HU17	Usuario	Poder eliminar los registros de promociones	Modificar la información existente	Es posible modificar registros en la base de datos con todas sus validaciones necesarias y los campos requeridos por el cliente	2	1	HU15,HU16	1	Terminado	Ninguno
				HU18	Usuario	Poder crear registros de cupones	Añadir nueva información	Es posible añadir registros en la base de datos con todas sus validaciones necesarias y los campos requeridos por el cliente	2	7		1	Terminado	Ninguno
				HU19	Usuario	Poder leer los registros de cupones	Conocer la información existente	Es posible leer los registros de la base de datos en una tabla	2	1	HU18	1	Terminado	Ninguno
				HU20	Usuario	Poder comprobar la existencia de un cupón	verificar la códigos de cupones proporcionados por los clientes	Es posible leer los comprobar la existencia de un cupón en la base de datos.	2	5	HU19	1	Terminado	Ninguno
										Total de puntos del sprint:	50			

2.2 Descripción de la funcionalidad y uso de Time to market implementados

Para poder aplicar los conocimientos de la clase de validación y verificación del software, se requiere un sistema real y funcional al cual se le pueda aplicar las pruebas. Este sistema debe ser desarrollado con una relativa facilidad y en poco tiempo. Es por eso por lo que se seleccionó una herramienta como OpenXava que permite desarrollar sistemas con estas características. A continuación, se presenta una descripción detallada de OpenXava.

¿Qué es OpenXava?

OpenXava es un framework de desarrollo orientado a la productividad para aplicaciones web y móviles en Java. Este framework es de código abierto y utiliza un enfoque basado en modelos para automatizar la interfaz de usuario y la lógica de persistencia a partir de anotaciones en simples clases Java. La promesa central de OpenXava es permitir a los desarrolladores escribir solo el código necesario para definir la lógica de negocio y las estructuras de datos, mientras que el framework se encarga de generar automáticamente las vistas y los controladores necesarios para una aplicación web funcional.

Una de las características distintivas de OpenXava es que no requiere que los desarrolladores escriban HTML, CSS, JavaScript, SQL, ni incluso JSPs para las interfaces de usuario; en lugar de ello, estos elementos se generan automáticamente a partir del modelo de datos definido en Java. Esto simplifica significativamente el proceso de desarrollo, ya que permite a los desarrolladores centrarse en la lógica de negocio subyacente sin preocuparse por los detalles de implementación de la interfaz de usuario o la base de datos.

OpenXava también soporta una amplia variedad de bases de datos, lo que permite su integración en diversos entornos empresariales sin estar atado a un proveedor específico de bases de datos. Además, incluye soporte para las aplicaciones móviles adaptativas, asegurando que las aplicaciones creadas con OpenXava funcionen bien tanto en dispositivos móviles como en escritorios sin necesidad de desarrollo adicional específico para móviles.

El desarrollo en OpenXava es altamente modular, lo que significa que las aplicaciones pueden ser desarrolladas, probadas y mantenidas en componentes más pequeños y manejables. Esta modularidad facilita la gestión del código a gran escala y mejora la capacidad de mantenimiento a largo plazo de las aplicaciones desarrolladas con el framework.

¿Cómo funciona OpenXava?

OpenXava simplifica el desarrollo de aplicaciones mediante la utilización de un enfoque basado en modelos. Este enfoque permite a los desarrolladores definir la estructura y el comportamiento de su aplicación a través de clases Java anotadas, sin necesidad de preocuparse por el código de bajo nivel para la interfaz de usuario o la gestión de la base de datos. Aquí te explico en detalle cómo funciona este proceso:

- **Modelo de dominio:** En OpenXava, todo comienza con el modelo de dominio, que son simplemente clases Java anotadas que representan los objetos de negocio (entidades) y sus relaciones. Estas clases están anotadas con metadatos que describen cómo se deben presentar los datos en la interfaz de usuario y cómo interactuar con la base de datos.
- **Generación automática:** Utilizando las anotaciones en estas clases, OpenXava genera automáticamente las interfaces de usuario y las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) necesarias para la aplicación. Esto incluye formularios, listas, y diálogos sin que el desarrollador tenga que escribir HTML, CSS o JavaScript.
- **Desarrollo Front-End:** Aunque la interfaz de usuario se genera automáticamente, OpenXava permite personalizaciones a través de XML o directamente desde el código Java. Esto ofrece flexibilidad para ajustar la interfaz de usuario según las necesidades específicas del negocio sin salirse del modelo declarativo.
- **Persistencia de datos:** Para la persistencia de datos, OpenXava se apoya en JPA (Java Persistence API), lo que le permite trabajar con cualquier base de datos compatible con JPA. Esto abstrae la mayoría de las operaciones de la base de datos, permitiendo que el desarrollador se concentre en la lógica de negocio sin preocuparse por los detalles de la implementación de la base de datos.
- **Despliegue y pruebas:** Las aplicaciones desarrolladas con OpenXava pueden ser desplegadas en cualquier servidor de aplicaciones Java EE, como Tomcat o WildFly. Además, OpenXava facilita la integración de pruebas automáticas y la configuración de pruebas unitarias, lo que es crucial para la validación y verificación del software.
- **Modularidad y componentes:** OpenXava es modular, permitiendo que los componentes de la aplicación sean desarrollados, probados y mantenidos de manera independiente. Esto mejora la escalabilidad de las aplicaciones y facilita la gestión del ciclo de vida del software.

¿Para qué funciona OpenXava?

OpenXava es un framework optimizado para el desarrollo de aplicaciones empresariales que necesitan manejar grandes volúmenes de datos y procesos complejos. Está especialmente diseñado para entornos donde se requieren sistemas como ERPs y CRMs, permitiendo a los desarrolladores enfocarse en la lógica de negocio gracias a su generación automática de interfaces y operaciones CRUD. Esta capacidad hace que OpenXava sea ideal para proyectos con tiempos de desarrollo ajustados y recursos limitados.

Además, OpenXava facilita la adaptabilidad y escalabilidad de las aplicaciones, características esenciales en mercados dinámicos donde los requisitos del negocio pueden cambiar rápidamente. La integración con diversas bases de datos y su compatibilidad con Java EE permiten que las aplicaciones sean robustas y escalables, mientras que su soporte para el desarrollo multiplataforma asegura una buena adaptabilidad en dispositivos móviles y de escritorio.

El mantenimiento y las actualizaciones de las aplicaciones desarrolladas con OpenXava son simplificados por su estructura modular y su enfoque en la reusabilidad del código. Los cambios

en los modelos de negocio se reflejan rápidamente con mínimas modificaciones en el código, reduciendo los costos y el tiempo de mantenimiento. Además, su soporte integrado para internacionalización facilita la localización de aplicaciones en varios idiomas, un aspecto crucial para empresas con presencia global.

¿Por qué fue elegida como la herramienta óptima?

OpenXava fue seleccionada como la herramienta óptima para el proyecto MediVoucher principalmente por su capacidad de acelerar el desarrollo de aplicaciones robustas sin necesitar un conocimiento profundo de frameworks de UI o SQL. Esto es particularmente valioso en un contexto académico como el de una clase de validación y verificación de software, donde el enfoque está en la calidad y eficacia del software más que en la programación detallada de cada componente. Además, OpenXava apoya la agilidad en el desarrollo y pruebas, permitiendo iteraciones rápidas y modificaciones sin costes prohibitivos. Esto alinea perfectamente con los objetivos del curso de proporcionar una plataforma sólida para aplicar prácticas de validación y verificación de software en un escenario realista.

2.3 Descripción y justificación de los tipos de prueba seleccionados

2.3.1 Casos de uso con Selenium

Los casos de uso son escenarios específicos que describen las interacciones entre el usuario y el sistema para lograr un objetivo funcional dentro de la aplicación. Estos casos de uso son esenciales para asegurar que todas las funcionalidades de MediVoucher se comporten como se espera en situaciones del mundo real. Para automatizar y ejecutar estos casos de uso se selecciona Selenium, una herramienta robusta para la automatización de pruebas en aplicaciones web. Selenium permite simular las acciones de los usuarios, como clics, ingreso de texto, y navegación por la aplicación, para verificar que cada paso del caso de uso se ejecuta correctamente y produce los resultados esperados.

MediVoucher es un sistema destinado a generar y gestionar cupones para productos farmacéuticos, lo cual implica una interacción constante con interfaces de usuario por parte de farmacias y clientes. Es crucial que todas las funcionalidades, desde la creación y redención de cupones hasta la administración de usuarios y farmacias, sean intuitivas y libres de errores. La prueba de casos de uso con Selenium asegura que el flujo de trabajo del usuario final sea suave y sin interrupciones, reduciendo así las posibilidades de errores en la operación real del sistema que podrían afectar la experiencia del usuario y la eficiencia del sistema.

Selenium fue elegido por varias razones estratégicas:

- **Compatibilidad Multiplataforma:** Selenium soporta todos los navegadores web principales y puede ejecutarse en diferentes sistemas operativos, lo cual es vital para MediVoucher dado su alcance y necesidad de funcionar a través de diversas plataformas y dispositivos.

- **Flexibilidad:** Selenium puede manejar pruebas simples y complejas, adaptándose a las necesidades del proyecto.
- **Comunidad y Recursos:** Al ser una de las herramientas de automatización más populares y ampliamente utilizadas, Selenium cuenta con una vasta comunidad de desarrolladores. Esto ofrece un soporte extenso y recursos de aprendizaje, facilitando la resolución de problemas y la expansión de la cobertura de prueba.

2.3.2 Análisis Estático de Código con SonarCloud

El análisis estático de código es una técnica de verificación que se utiliza para evaluar el código fuente sin ejecutar el programa. La herramienta SonarCloud analiza el código en busca de errores, vulnerabilidades de seguridad, y malos olores de código ("code smells"), ayudando a mantener un estándar de calidad y seguridad en el desarrollo. Este tipo de prueba es esencial para detectar problemas tempranamente en el ciclo de desarrollo y garantizar que el código cumple con los estándares de calidad antes de su implementación.

Para MediVoucher, que implica la gestión de datos sensibles y operaciones críticas como la generación y validación de cupones para medicamentos, asegurar la integridad y seguridad del código es primordial. Un análisis estático con SonarCloud permite identificar y mitigar riesgos de seguridad y fallos lógicos que podrían comprometer la funcionalidad del sistema y la protección de datos de los usuarios. Esta práctica contribuye significativamente a la fiabilidad y robustez del sistema, factores clave para su éxito y adopción por parte de cadenas farmacéuticas y usuarios finales.

SonarCloud fue seleccionado por su integración continua y capacidad de análisis en la nube, lo que facilita la revisión de código en proyectos desarrollados con metodologías ágiles y en entornos de desarrollo colaborativos. Su compatibilidad con múltiples lenguajes de programación y frameworks, junto con su facilidad de uso y configuración, lo hacen ideal para un entorno educativo y proyectos de desarrollo rápido como MediVoucher. Además, SonarCloud ofrece una visualización detallada de los resultados del análisis, lo que permite a los desarrolladores identificar rápidamente áreas de mejora y entender mejor las prácticas de codificación segura y eficiente.

2.3.3 Pruebas de carga con JMeter

Las pruebas de carga son esenciales para evaluar cómo el sistema MediVoucher maneja un volumen significativo de usuarios y transacciones simultáneas. Este tipo de prueba simula un entorno donde múltiples usuarios interactúan con el sistema al mismo tiempo, realizando diversas operaciones como la generación de cupones, la validación de estos y la administración de farmacias y medicamentos. El objetivo es identificar cómo el rendimiento del sistema se ve afectado bajo carga pesada y determinar si puede sostener su operatividad bajo condiciones de estrés sin degradación significativa del rendimiento.

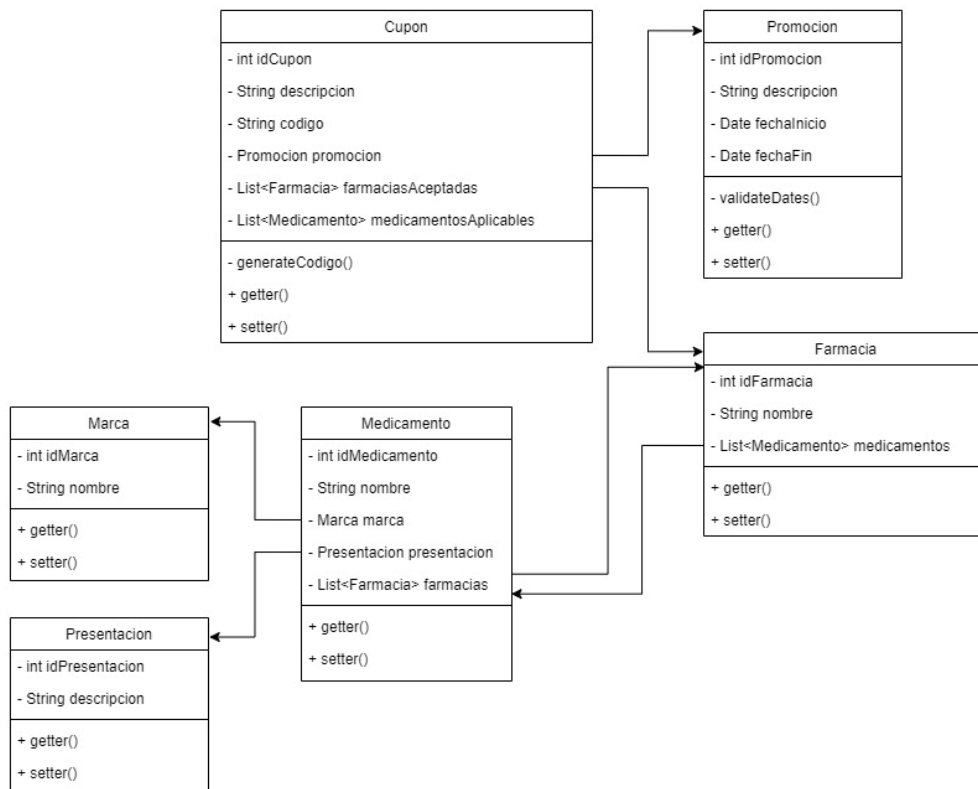
Dado que MediVoucher está diseñado para facilitar la distribución y gestión de cupones en un entorno farmacéutico, es crucial que el sistema sea capaz de manejar picos de demanda. Las pruebas de carga aseguran que MediVoucher pueda operar de manera eficiente y sin fallos, incluso bajo presión, garantizando que todos los usuarios tengan acceso ininterrumpido y una experiencia fluida. Estas pruebas ayudan a identificar y resolver cuellos de botella y problemas de rendimiento antes de que el sistema sea puesto en producción, minimizando el riesgo de fallos que puedan afectar a los usuarios finales y a la reputación del sistema. La selección de pruebas de carga para MediVoucher es crítica por varias razones:

- **Previsión de Crecimiento:** A medida que más farmacias y usuarios se integren al sistema, es vital anticipar el incremento en la carga y asegurarse de que el sistema pueda escalar adecuadamente.
- **Estabilidad y Confiabilidad:** MediVoucher debe ser estable y fiable en todo momento. Las pruebas de carga permiten probar la resiliencia del sistema y su capacidad para recuperarse de condiciones de alta demanda.
- **Optimización del Rendimiento:** Estas pruebas proporcionan datos valiosos que pueden ser utilizados para optimizar el rendimiento del sistema, ayudando a mejorar la gestión de recursos y la eficiencia general del sistema.

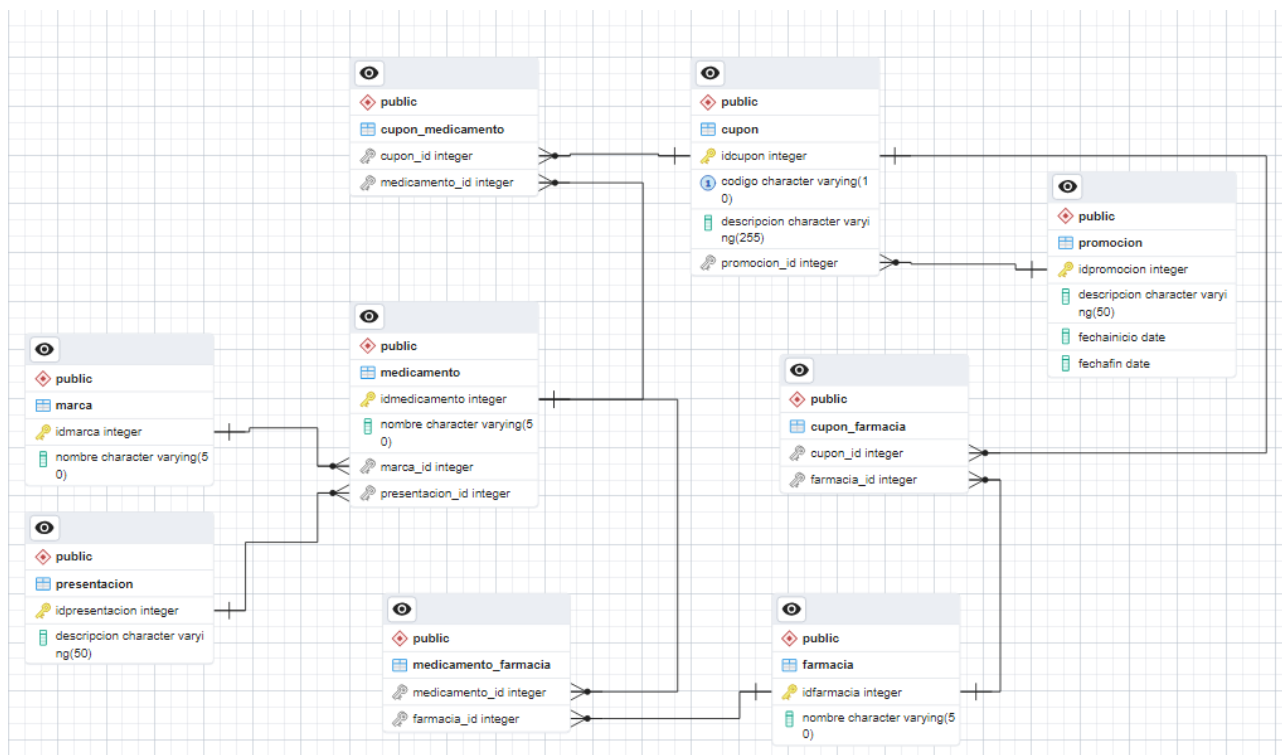
3 Diseño de una propuesta técnica para el problema

3.1 Diseño de clases de la implementación (Diagramas de clases y ER)

A continuación, se presenta el diagrama de clases propuesto para el sistema MediVoucher. Este diagrama fue la base para la creación del modelo del sistema en OpenXava:



Una vez creadas las clases en OpenXava, este se encarga de crear las tablas correspondientes en la base de datos, que en este caso fue una de PostgreSQL alojada en la nube (Render). Este es el diagrama ER de la base de datos generada:



3.2 Descripción de la implementación del sistema informático

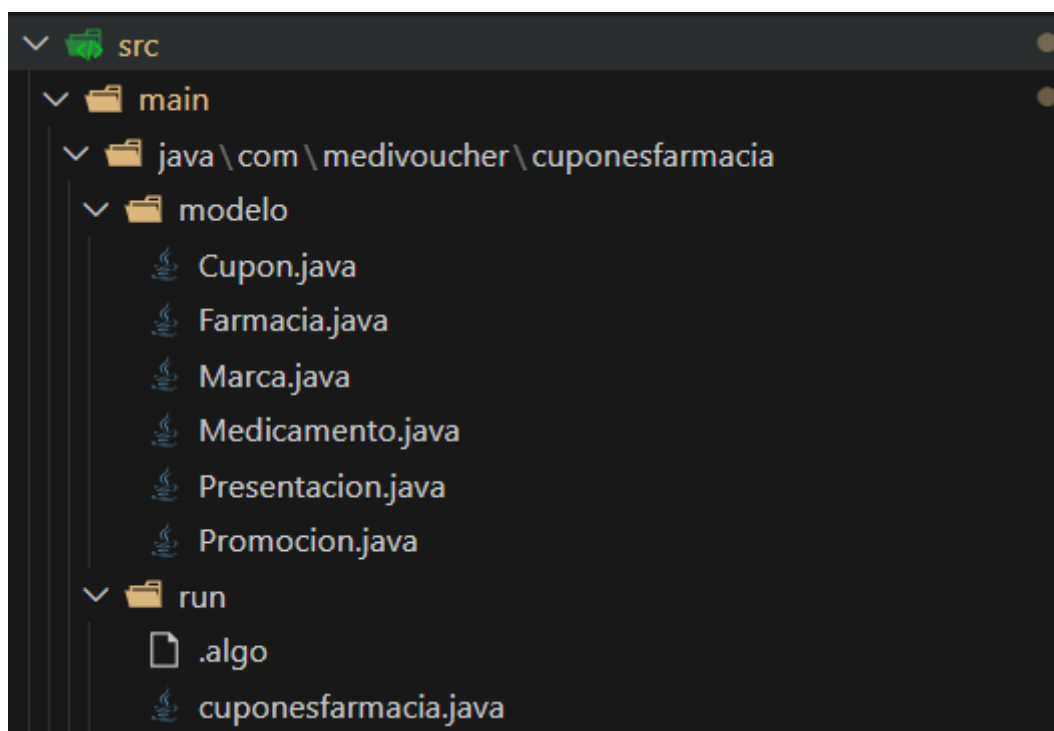
El proyecto MediVoucher ha sido desarrollado utilizando OpenXava, un framework de alta productividad para el desarrollo de aplicaciones empresariales en Java. Este framework permite la rápida creación de aplicaciones web y móviles basadas en la arquitectura Java Enterprise, aprovechando una arquitectura centrada en el dominio donde las clases de negocio se enriquecen con anotaciones que automatizan la interfaz de usuario, la persistencia, y el API.

Estructura y Tecnologías Utilizadas:

- **Framework:** OpenXava
- **Lenguaje de Programación:** Java
- **Sistema de Gestión de Bases de Datos (DBMS):** PostgreSQL
- **Plataforma de Despliegue de la Base de Datos:** Render
- **Sistema Operativo de Desarrollo:** Para el desarrollo se utilizó Windows 11.
- **IDE:** Visual Studio Code y OpenXava.
- **Herramientas de producción:**
 - **Instancia:** EC2 micro
 - **JRE:** Java 13
 - **Servidor Web:** Tomcat 8
 - **MVN:** 3.9.8
 - **Sistema Operativo:** Debian 12

La estructura del proyecto se organiza en diversos paquetes y clases, representados en la siguiente estructura:

- **Paquete modelo:** Contiene las clases de entidad que representan la lógica de negocio y la estructura de datos del sistema MediVoucher. Estas clases son:
 - **Cupon.java:** Define los atributos y métodos para la gestión de cupones en el sistema.
 - **Farmacia.java:** Gestiona la información relacionada con las farmacias asociadas a los cupones.
 - **Marca.java:** Representa las marcas de medicamentos disponibles en el sistema.
 - **Medicamento.java:** Contiene la lógica para la gestión de los medicamentos ofertados.
 - **Presentacion.java:** Administra las diferentes presentaciones de los medicamentos.
 - **Promocion.java:** Encapsula las promociones aplicables a diferentes productos o medicamentos.



Utilizando OpenXava, la definición de estas clases con anotaciones permite generar automáticamente tanto la interfaz de usuario como las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para cada entidad. Además, se genera la estructura de base de datos correspondiente en PostgreSQL. La plataforma de despliegue Render facilita la operación continua del sistema en un entorno de producción, garantizando alta disponibilidad y escalabilidad.

La generación automática de UI y lógica de persistencia a través de OpenXava reduce significativamente el tiempo de desarrollo y elimina la necesidad de escribir código repetitivo para las operaciones comunes de la base de datos y la interfaz de usuario. La integración con PostgreSQL, mediante JPA (Java Persistence API), asegura una gestión eficiente y profesional de los datos, permitiendo transacciones seguras y un rendimiento optimizado.

3.3 Casos de prueba

Para garantizar la calidad y funcionalidad del sistema MediVoucher, se ha definido una tabla estructurada de casos de prueba que comprende varios campos esenciales para documentar y ejecutar cada prueba de manera efectiva. A continuación, se describe la función y el propósito de cada campo en la tabla de casos de prueba:

- **ID:** Un identificador único para cada caso de prueba. Este ID facilita la referencia rápida a un caso de prueba específico y ayuda en la organización y seguimiento de las pruebas a lo largo del ciclo de vida del desarrollo del software.
- **Nombre:** Un nombre descriptivo y conciso que identifica el caso de prueba. Este debe ser claro y reflejar la esencia de la prueba para facilitar la comprensión del objetivo del caso a cualquier miembro del equipo.

- **Descripción:** Una descripción detallada del caso de prueba que explica qué funcionalidad o característica del sistema se está probando. Este campo debe proporcionar suficiente información para entender el contexto y el alcance del caso de prueba.
- **Precondiciones:** Las condiciones que deben cumplirse antes de ejecutar el caso de prueba. Esto puede incluir la configuración del sistema, estados específicos de la aplicación, o datos necesarios que deben estar presentes.
- **Pasos:** Una enumeración detallada de los pasos a seguir para llevar a cabo el caso de prueba. Esta sección debe ser precisa y detallada para garantizar que la prueba pueda ser reproducida consistentemente por diferentes miembros del equipo o en futuras iteraciones.
- **Resultado esperado:** La descripción de lo que se espera que ocurra una vez que se hayan ejecutado los pasos del caso de prueba. Este resultado debe alinearse con los requisitos y especificaciones del sistema para verificar si el caso de prueba pasa o falla.
- **Resultado actual:** El resultado que se obtiene al ejecutar el caso de prueba. Este campo se llena durante la ejecución de la prueba y se compara con el "Resultado esperado" para determinar si el caso de prueba ha pasado o fallado.
- **Estado:** El estado actual del caso de prueba, que puede incluir términos como "Pendiente", "En progreso", "Correcto", "Fallo", o "Bloqueado". Este campo ayuda a rastrear el progreso en la ejecución y resolución de las pruebas.
- **Referencias:** Enlaces o referencias a documentos, tickets de bugs, o requisitos específicos que están relacionados con el caso de prueba. Este campo es útil para proporcionar contexto adicional y justificación para la prueba, facilitando la trazabilidad y la auditoría de las pruebas.

El documento de Excel con los casos de prueba detallados se encuentra adjunto en el anexo número 2. También se encuentra cargado en el repositorio de Github del código fuente.

3.4 Resultados de la ejecución de los casos de prueba

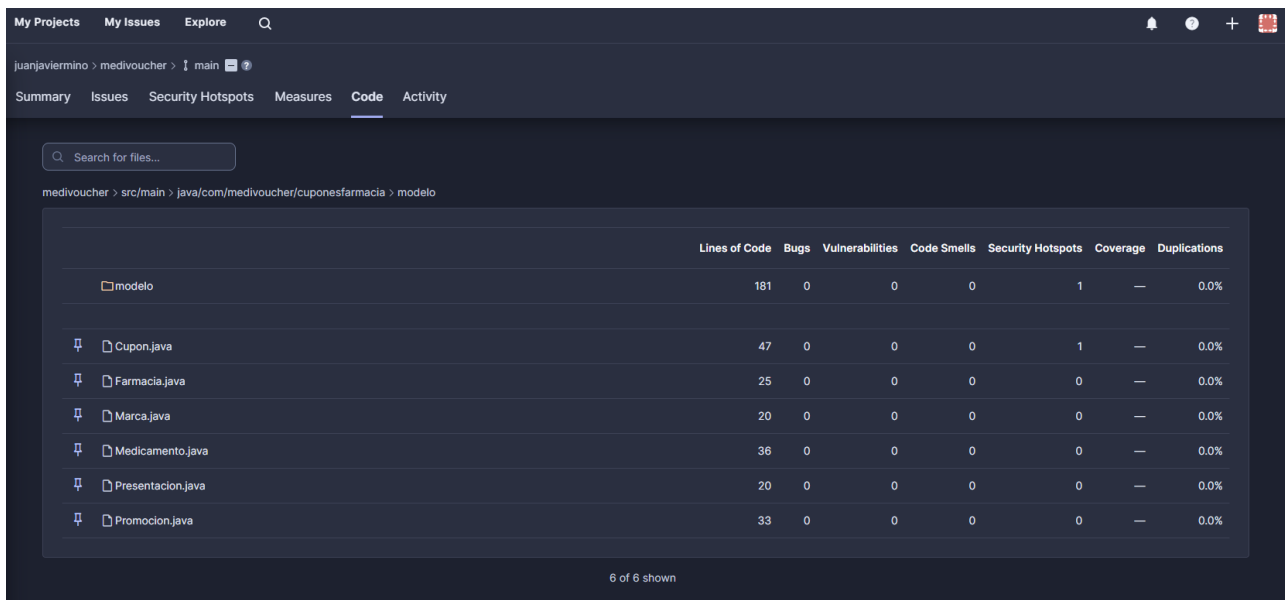
Casos de uso:

Para las pruebas de casos de uso se realizaron 21 casos de prueba. Dentro de estos casos de prueba solamente uno de ellos presentó un fallo (4.7%) entre los resultados esperados y los actuales. Este caso de prueba y su defecto se redactan de una manera más detallada en la siguiente sección, sin embargo, se puede adelantar que fue en la generación de cupones (ID del caso de prueba: P-018). Lo que ocurrió en este caso de prueba fue que el código del cupón no se

creó cuando se ingresó un nuevo cupón. Este solamente apareció cuando se crea la entidad y solucionándose al editarla. Este defecto es crítico en severidad, pues afecta directamente a la funcionalidad más importante del negocio: la generación de cupones, y tiene una prioridad alta pues sin esta funcionalidad trabajando de manera correcta, el software no puede continuar en producción.

Análisis de código estático:

Para el análisis de código estático se presentaron seis casos de prueba, uno por cada clase del modelo del sistema, siendo estos los archivos más importantes del mismo. Entre estos casos de prueba se tuvo uno que no cumplió del todo con los estándares propuestos de cero bugs, vulnerabilidades, code smells, y security hotspots (16.7%). Este es el caso de prueba con ID: SCA-003 en el cual se analizó el código de la clase Cupon. La razón por la cual este caso de prueba falla es porque se encontró un item en security hotspot, el cual se detalla de mejor manera en la siguiente sección del documento.



	Lines of Code	Bugs	Vulnerabilities	Code Smells	Security Hotspots	Coverage	Duplications
modelo	181	0	0	0	1	—	0.0%
Cupon.java	47	0	0	0	1	—	0.0%
Farmacia.java	25	0	0	0	0	—	0.0%
Marca.java	20	0	0	0	0	—	0.0%
Medicamento.java	36	0	0	0	0	—	0.0%
Presentacion.java	20	0	0	0	0	—	0.0%
Promocion.java	33	0	0	0	0	—	0.0%

Pruebas de Carga:

Se realizaron un total de 4 pruebas de carga, cada uno con un escenario distinto para así simular los diversos tipos de actividad que pueden existir durante el ciclo de vida de la aplicación de Medivoucher.

La aplicación supero 3 de las 4 pruebas de carga y falló en una de manera no significativa, por lo cual se considera de prioridad baja y no debería afectar a la aplicación en el entorno productivo en el cual está desplegada. Gracias a estas pruebas podemos asumir que la aplicación podrá manejar la carga de hasta 1000 usuarios simultáneos los cuales podrán utilizarla de manera fluida y sin inconvenientes.

3.5 Informe de resultados de la prueba

A continuación, se presenta el reporte de todos aquellos defectos encontrados al momento de realizar las pruebas.

3.5.1 Reporte de Resultados Caso de Prueba con ID: SCA-003

Título: Falla en Análisis Estático de Seguridad para la Clase Cupon.

Descripción detallada: El caso de prueba SCA-003 se diseñó para evaluar la calidad del código de la clase Cupon.java mediante la herramienta SonarCloud, buscando identificar bugs, vulnerabilidades, code smells y security hotspots. Aunque el análisis no reveló bugs, vulnerabilidades ni code smells, se detectó un security hotspot, lo cual indica un potencial riesgo de seguridad que necesita ser investigado y resuelto para garantizar la seguridad de la aplicación.

Pasos para reproducir:

- Abrir la interfaz de SonarCloud.
- Configurar un nuevo proyecto vinculado al repositorio de GitHub que contiene el código de Cupon.java.
- Ejecutar el análisis estático de código en SonarCloud para que lea y analice el código completo de la clase Cupon.java.
- Revisar el informe generado por SonarCloud que detalla las métricas y hallazgos de seguridad.

Defecto específico: El análisis reveló la presencia de 1 security hotspot en la clase Cupon.java. Los security hotspots son áreas en el código que podrían ser vulnerables a ataques si no se manejan adecuadamente.

medivoucher > src/main > java/com/medivoucher/cuponesfarmacia > modelo > Cupon.java

medivoucher	Lines	Duplications	Bug	Vulnerability	Code Smell	Security Hotspot	
src/.../medivoucher/cuponesfarmacia/modelo/Cupon.java	59	0.0%	0	0	0	1	

1

package com.medivoucher.cuponesfarmacia.modelo;

Este security hotspot se encuentra en el método generateCodigo (línea 53), y detalla: Criptografía débil, pues la generación de cupones con randomAlphanumeric no es segura ni robusta para generación de códigos.

Entorno:

- Herramienta de Análisis: SonarCloud
- Repositorio de Código: GitHub
- Lenguaje de Programación: Java
- Clase Analizada: Cupon.java

Datos propios:

- Líneas de Código: 47
- Bugs: 0
- Vulnerabilidades: 0
- Code Smells: 0
- Security Hotspots: 1

Severidad: Alta - Los security hotspots requieren una revisión manual para evaluar la gravedad real del potencial riesgo de seguridad y tomar las medidas correctivas necesarias para mitigar cualquier amenaza.

3.5.2 Reporte de Resultados Caso de Prueba con ID: P-018

















Título: Falla en la Generación Correcta de la Estructura de Cupones.

Descripción detallada: El caso de prueba P-018 se diseñó para validar la correcta generación de cupones dentro del sistema MediVoucher, asegurando que todos los componentes necesarios del cupón, como el código, estén presentes según los requisitos establecidos. Durante la ejecución del caso de prueba, se encontró que, aunque el sistema permitió la creación del cupón, la estructura de este no incluía el código alfanumérico esencial para su validez y uso, lo cual es un componente crítico según los casos de uso definidos.

Pasos para reproducir:

- Iniciar sesión en la aplicación MediVoucher con credenciales de administrador.
- Navegar a la pestaña "Cupones".
- Seleccionar de las listas desplegables la farmacia, el medicamento, la presentación y la promoción correspondientes.
- Generar el cupón utilizando el botón o función designada.
- Revisar la estructura del cupón generado para asegurar que incluya todos los componentes necesarios, en especial el código alfanumérico.

Defecto específico: El cupón generado carece del código alfanumérico, lo cual impide su correcta utilización y rastreo dentro del sistema. Esto representa un fallo en la lógica de generación de cupones o en la implementación de las funciones de generación dentro del sistema.

							
		<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	
		<input type="checkbox"/>	5	Insulina de larga duración 2X1			1 Regreso a clases
		<input type="checkbox"/>	2	30% en insulina	YLEIOTPUNT		1 Regreso a clases
		<input type="checkbox"/>	6	20% menos en tabletas masticables	IWJSW9E897		4 Día del médico
			Σ				Σ

Entorno:

- Sistema Operativo: Windows
- Navegador Web: Chrome
- Versión del Sistema: MediVoucher v1.0

Datos propios: Se verificó que todos los pre-requisitos como la creación previa de farmacias, medicamentos, presentaciones y promociones estaban correctamente configurados y disponibles para la selección.

Severidad: Alta - La ausencia del código alfanumérico en los cupones no solo impide su uso efectivo, sino que también compromete la integridad del proceso de validación y rastreo de

cupones dentro del sistema MediVoucher. Este defecto requiere una corrección inmediata para asegurar la funcionalidad y fiabilidad del sistema en la generación y manejo de cupones.

3.5.3 Reporte de Resultados Caso de Prueba con ID: C-003

Título: Fallo en Cumplimiento de Latencia Durante Análisis de Carga para 1000 Usuarios.

Descripción detallada:

El caso de prueba C-003 fue diseñado para evaluar la capacidad del servidor web de manejar 1000 usuarios simultáneos, con el objetivo de mantener una latencia promedio inferior a 325 ms.

Durante la prueba, se observó que la latencia promedio excedió el umbral establecido, alcanzando 352 ms, indicando una posible sobrecarga o ineficiencia en el manejo de altas cargas por parte del servidor.

Pasos para reproducir:

- Asegurar que la aplicación está desplegada en el entorno productivo.
- Configurar el software de prueba de carga para simular 1000 usuarios distribuidos en 1 segundo.
- Establecer el número de hilos de ejecución conforme al número de usuarios simulados.
- Configurar el número de ciclos de prueba a 1 para concentrar la carga en un breve periodo.
- Iniciar la prueba de carga y monitorear la latencia y el rendimiento del servidor.

Defecto específico: El servidor no cumplió con el criterio de aceptación de mantener una latencia promedio inferior a 325 ms, registrando en cambio una latencia de 352 ms bajo las condiciones de prueba especificadas.

Entorno:

- Sistema Operativo del Servidor: Debian 12.
- Software de Prueba de Carga: JMeter.

Datos propios:

- Número de Usuarios Simulados: 1000
- Duración de la Prueba: 1 segundo
- Latencia Observada: 352 ms

Severidad: Baja- Si bien la latencia observada es superior a la esperada, no es lo suficientemente significativa para ser una inconveniencia o notada en el uso diario de la aplicación.

3.6 Video final

El video final con la descripción del proyecto y la presentación de los resultados de las pruebas propuestas se encuentra en este enlace de Canva:

https://www.canva.com/design/DAGKeQFs48s/JZstWg_PtfmC6-uOmBFNNA/edit?utm_content=DAGKeQFs48s&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Además, el video también está cargado en el repositorio de Github detallado en la siguiente sección del documento.

3.7 Código fuente

En el repositorio de Github cuyo enlace se adjunta a continuación, se encuentran los siguientes archivos:

- Código fuente del sistema.
- PDF con el formato IEEE 830.
- Excel con los casos de prueba y sus resultados.
- Video final.

Enlace del repositorio: <https://github.com/juanjaviermino/valid-verif-sw-proyecto-final>

En el Anexo 3 igualmente se puede encontrar el enlace al proyecto desplegado. Para ingresar al sistema el usuario y la contraseña son **admin**.

4 Anexos

4.1 Anexo 1: IEEE 830

Enlace al documento: https://udlaec-my.sharepoint.com/:w/g/personal/dennis_ocana_udla_edu_ec/EUX8lgYzPrBCoP4FmXFQMqMB7utB3lqlIT2ie7Kp_RwVqA?e=OgwnXI

4.2 Anexo 2: Casos de prueba

Enlace al Excel: https://udlaec-my.sharepoint.com/:x/g/personal/juan_mino_arboleda_udla_edu_ec/EflUU8TrZoJJoxUFSolxRqYBZBVaJnmAae7mB7idExCAxQ?e=ttT1O5

4.3 Anexo 3: Sistema desplegado

Enlace al sistema: <http://medivoucher.mindsoftdev.com:8080/cuponesfarmacia/>