UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
LABORATORIO DE REDES MÓVILES, INALÁMBRICAS Y DISTRIBUIDAS (ICARO)

Sistema de Gestión de Inventario para los equipos e insumos del Centro Educativo de la Asociación de Profesores de la Universidad Central de Venezuela (CEA-PUCV) utilizando código QR

Trabajo Especial de Grado presentado ante la ilustre Universidad Central de Venezuela, por los bachilleres:

Juárez Andrés C.I. 19.204.865

Villegas Roosevelt C.I. 20.006.753

Tutores:

Profa. Morales Ana

Profa. Villapol María

Caracas, agosto 2014

Agradecimientos

En primer lugar a Dios Todopoderoso por llevarme de la mano en el camino de la vida, por siempre enseñarme el camino del bien y por sobre todo enseñarme a no desfallecer ante las dificultades.

A mis padres, Gladys y José Andrés, por el apoyo incondicional durante cada etapa de mi vida y en las decisiones que he tomado a lo largo de esta, por ser mis guías y ejemplos de vida, por enseñarme que con amor, paciencia y perseverancia todo se alcanza.

A mis hermanos, por siempre estar tan dispuesto a aportar su ayuda sin importar las dificultades que esto suponga.

A Argriet Fermín, mi novia y mejor amiga, por acompañarme a lo largo de toda mi carrera universitaria, por ayudarme a tomar las mejores decisiones y hacerme sonreír en los momentos más difíciles.

A la familia Fermín Lampert por ser mi segundo hogar.

A mi padrino José Buceta, por estar siempre pendiente de mi carrera universitaria, así como de prestar su ayuda en cualquier momento.

A mi familia que aunque muy lejos que se encuentren nunca faltó por ningún medio un mensaje de aliento, gratitud y felicitaciones durante toda mi vida.

Andrés A. Juárez M.

Antes de todo, tengo que agradecer a mis maravillosos padres, Emilia y Roosenvelt, que a parte de darme la vida, han dado y sacrificado todo por mí, por que sea una persona de bien, de buenos valores y principios. Que siempre me han apoyado en todo y han sido el sustento de mi vida. No puedo pedir mejores padres, los amo mamá y papá.

Agradezco a mi tía Olga, por estar siempre a mi lado, por apoyarme y ser una excelente amiga; por guiarme y enseñarme a ser una mejor persona.

A la familia Fermín Lampert, por servirnos de apoyo durante todo este trabajo especial de grado, en especial a Argenis, a la sra. Margriet y a Sisi.

Quiero también agradecerle a mi compañero de tesis Andrés, con quien formé amistad a principio de la carrera, y la cual se fortaleció durante este proyecto; por una amistad que dure para siempre. Por ser responsable y atento con el proyecto. Lo logramos.

A todas las amistades que hice durante todos estos años en la carrera, quienes nos ayudamos mutuamente para finalmente terminar esta etapa de nuestras vidas.

Y a los futuros tesistas, solo les digo: constancia y dedicación, valdrá la pena.

El tiempo es lo más preciado y valioso que tenemos, debemos aprovecharlo cada día de nuestras vidas, dando el mayor esfuerzo y dedicación a todo lo que hacemos, para no voltear atrás y poder arrepentirnos.

Roosevelt J. Villegas F.

A la profa. Ana Morales y la profa. María Villapol, por su tutoría, por habernos confiado este proyecto y estar a total disposición ante cualquier eventualidad.

A los profesores Sergio y Jossie que nos guiaron durante el proceso de seminario. Al profesor Antonio Silva por prestarnos su ayuda, brindándonos sus conocimientos para resolver eventualidades del proyecto.

A la familia Fermín Lampert, por prestarnos en más de una oportunidad un ambiente cálido y hogareño en el cual trabajar en este proyecto.

Andrés y Roosevelt.

Resumen

Sistema de Gestión de Inventario para los equipos e insumos del Centro Educativo de la Asociación de Profesores de la Universidad Central de Venezuela (CEAPUCV) utilizando código QR

Autores:

Andrés Juárez.

Roosevelt Villegas.

Tutores:

Profa. Ana Morales.

Profa. María Villapol.

Un inventario es la manera organizada de documentar la cantidad de bienes en existencia que posee una entidad. Realizar un control de inventario puede ser de múltiples beneficios dependiendo del ente que lo ejecute, así como los intereses que tenga este último por mantener un control de sus bienes en existencia, además de conseguir información de las ganancias o pérdidas que se estén generando por alguna de las clasificaciones de los bienes. Actualmente en el Centro Educativo de Asociación de Profesores de la Universidad Central de Venezuela (CEAPUCV) no existe un sistema de control de inventario que permita controlar de forma adecuada los bienes que posee el centro educativo, como tampoco existe un procedimiento para determinar el origen y el destino de alguno de estos bienes. Es por esto que el objetivo de este Trabajo Especial de Grado, es crear un sistema web de control de inventario haciendo uso de la tecnología de código abierto QR y del método de desarrollo de software ágil XP, con el fin de proveer una adecuada solución a la situación anteriormente mencionada que padece el CEAPUCV, a través de un sistema robusto y seguro el cual ofrezca una experiencia confortable a los usuarios.

Palabras Claves: Código QR, Sistema de gestión de inventarios, Ruby on Rails, CEAPUCV y XP.

Índice general

Índice de figuras

Índice de tablas

Introducción

En un principio las tareas cotidianas del hombre se llevaban a cabo de forma manual. A medida que estas tareas fueron tomando más complejidad, se empezaron a originar problemas como: falta de precisión, disminución de la calidad en los resultados finales, así como también la prolongación de los tiempos de producción. Todos estos problemas mencionados, se redujeron significativamente haciendo uso de tecnologías que colaboran con la automatización de los procesos. Dentro de las tareas que se llevaban a cabo de forma manual, se encuentra la gestión de inventarios.

El inventario no es más que realizar una documentación formal donde se tiene en cuenta las cantidades existentes de ciertos bienes pertenecientes a un ente, persona o comunidad; en términos de contabilidad se tienen en cuenta otros factores como precios unitarios, sumas parciales por grupos, importes, procedencia, destino y clasificaciones de los bienes del inventario.

El control de un inventario puede satisfacer a distintas entidades, como pequeñas o grandes empresas, entidades del estado sin dejar por fuera a las unidades educativas; en estas últimas entidades el control de inventario puede estar o no orientado a la forma contable, casos en los cuales solo se necesite llevar un control de las cantidades en existencia de los distintos bienes que puede poseer una institución educativa, sin tomar en cuenta los precios unitarios ni las ganancias o pérdidas que generen dichos bienes reflejados en los resultados de los reportes que se puedan obtener.

Actualmente el Centro Educativo de Asociación de Profesores de la Universidad Central de Venezuela (CEAPUCV) no lleva a cabo ningún control de inventario sobre los bienes que posee, como: libros, mobiliario, equipos de computación, materiales de laboratorios, equipos deportivos, insumos de oficina, electrodoméstico, medicamentos, etc. Además no posee un control en donde pueda identificar el origen de los bienes que poseen, debido a que el instituto recibe equipos para su uso a través de distintos medios, como la sociedad de padres y representantes, y asignaciones que realiza el Instituto de Previsiones del Profesor (IPP). Por lo tanto al no tener identificado el origen de cada equipo, el IPP puede disponer de estos aún así no hayan sido ellos quiénes lo suministraron. Razón por la cual este trabajo de investigación está orientado al desarrollo de un sistema Web de control de inventario automatizado haciendo uso de la tecnología de códigos QR. Se hace uso de los códigos QR, ya que estos proporcionan características como el corto

tiempo de lectura e interpretación, la posibilidad de leerlos con el uso de teléfonos inteligentes aprovechando los dispositivos disponibles, poder leer el código desde cualquier angulo con un espectro de 360 grados y una recuperación de datos de hasta el 30 %. Dentro de los códigos QR se pueden almacenar datos de cada uno de los bienes como: el nombre, la marca, el autor, etc., así como también una URL, por la cual se puede acceder al sistema.

Durante el desarrollo de la investigación se obtuvo información detallada de como se llevan a cabo los procesos de entrada y salida de los bienes, quienes son los responsables de mantener un sistema de inventario, identificar los distintos tipos de usuarios que va a soportar el sistema; para ofrecer una solución segura y confiable, categorizar los bienes que se desean inventariar y asignar esas categorías a los diferentes usuarios una vez conocidos.

El sistema web ha sido desarrollado haciendo uso de herramientas de código abierto para reducir costos al momento de futuros mantenimientos y actualizaciones al mismo, dentro de las herramientas tenemos el lenguaje Ruby bajo su *framework* Rails. Como manejador de base de datos se tiene MySQL; por ultimo para hacer que la aplicación web sea usable y también que posea un diseño que se pueda adaptar a las resoluciones de los dispositivos móviles, se utilizó Twitter Bootstrap.

Finalizado el desarrollo del trabajo de investigación, el CEAPUCV cuenta con un sistema confiable, seguro y usable para llevar a cabo las tareas de control de inventario. No depende de un trabajo pesado como lo es un trabajo manual al momento de generar reportes de todos los bienes. Se puede disponer con el mismo sistema para poder identificar el origen de cada uno de los bienes que ingresen al sistema, en caso de conocerlo. El que el sistema esté desarrollado en una plataforma web, facilita el acceso al mismo a través de las computadoras de las oficinas, pero a su vez desde los dispositivos móviles del personal que tenga acceso al mismo.

Este trabajo especial de grado se divide en seis capítulos; en el capítulo I “Planteamiento del problema” se describe cual es problema y su solución. En el capítulo II “Marco Conceptual” se definen todos los conceptos necesarios relevantes de los sistemas de gestión de inventario, el código QR, así como el método de desarrollo para realizar este proyecto. En el capítulo III “Herramientas Existentes de Sistemas de Gestión de Inventario” se describen tres herramientas de sistemas de gestión inventarios, los cuales comparten similitudes al sistemas que se quiere implementar. En el capítulo IV “Marco Aplicativo” se describe cuales fueron los ajustes hechos al método XP, las historias de usuarios, iteraciones y pruebas hechas. El capítulo V contiene las pruebas de usabilidad y los resultados de los mismos. Y por último la Conclusión y la Bibliografía.

Capítulo 1

Planteamiento del Problema

Actualmente el Centro Educativo de la Asociación de Profesores de la Universidad Central de Venezuela (CEAPUCV) vive una realidad en la que carece de un adecuado sistema de gestión de materiales. En el presente capitulo se detallará esta situación y se hará una propuesta de solución a esta.

1.1. Planteamiento

El CEAPUCV es la unidad educativa que presta sus servicios a los hijos de los docentes de la Universidad Central de Venezuela (UCV). Al igual que muchas instituciones educativas, esta integra actividades recreativas, deportivas, clases de computación, así como servicios de enfermería. Dichas actividades o servicios exigen de la presencia de ciertos equipos o insumos para la correcta realización de las mismas, por ejemplo: las actividades deportivas requieren equipamiento como balones, bates, mallas, colchonetas, raquetas, etc. El servicio de enfermería requiere insumos como inyectadoras, medicamentos, termómetros, etc.

El IPP es el ente encargado de proveer al colegio los equipos o insumos que necesite para cumplir con su labor. Debido a la realidad económica que vive el país, en la actualidad el IPP ve afectada su labor de proveer al CEAPUCV, y la comunidad que hace vida dentro del centro educativo, más específico la Sociedad de Padres y Representantes, ha tomado medidas en el asunto para conseguir de alguna manera los mencionados equipos e insumos. Dentro de las medidas tomadas se encuentran actividades como rifas, bazares y donaciones, entre otras.

Esta situación conlleva al primer problema, el CEAPUCV no posee un mecanismo adecuado que se encargue de establecer el origen (ya sea del IPP o de las actividades realizadas por la Sociedad de Padres y Representantes) de los materiales que adquiere el colegio. Por otra parte la institución tampoco posee un sistema de gestión de dichos materiales, en el que se mantenga un control de asignación, entrada y salida. A excepción de la enfermería la cual posee un sistema,

pero que presenta algunas deficiencias, como la falta de un control de roles de quién o quiénes manejan el sistema, es decir, que cualquier persona puede acceder al mismo y realizar cambios sin tener en cuenta la importancia de los cambios que se estén realizando; así alguien ajeno a la enfermería puede eliminar todos los insumos que se tienen registrados en el sistema de forma explícita o por error, cuando en su lugar el encargado de poseer acceso a borrar datos de forma masiva debería ser el personal de la enfermería. Tampoco posee una interfaz de usuario usable, lo que dificulta la compresión de las funciones del sistema.

Se conoce el proyecto del sistema encargado de la biblioteca, desarrollado por un grupo de estudiantes de la Facultad de Ciencias de la Universidad Central de Venezuela, cuyo propósito es solo encargase de la gestión de los libros pertenecientes a la biblioteca, sin embargo este proyecto aunque desarrollado, todavía no está puesto en producción dentro del colegio.

Resumiendo, se tienen los siguientes problemas, no se posee el sistema adecuado para establecer el origen de los bienes que adquiere el colegio; el único sistema que posee el colegio presenta deficiencias. Y además no existe un sistema que facilite la gestión del control del inventario que englobe todos los bienes que posee la institución. Como solución, dichos bienes podrían identificarse con el uso de una etiqueta, la cual pudiese ser implementada con el código QR, por todos los beneficios que esta tecnología ofrece como: el corto tiempo de lectura e interpretación, la posibilidad de leer los códigos con el uso de teléfonos inteligentes, poder leer el código desde cualquier angulo de 360 grados y una recuperación de datos de hasta el 30 %.

Entonces, cómo dar solución a estos problemas que vive el CEAPUCV a continuación se pretende plantear una propuesta formal de solución.

1.2. Propuesta

Debido a los tiempos actuales en que se viven en donde la tecnología ha evolucionado de una forma exponencial y donde la mayoría de herramientas tecnológicas están al alcance de la mano, se propone como una posible solución realizar un sistema web de inventario automatizado que se encargue de la gestión de los equipos e insumos que dispone el CEAPUCV, incluyendo el uso de los códigos QR para desempeñarse dentro del proceso de registro e identificación de los equipos e insumos.

1.3. Objetivo General

Desarrollar un sistema de gestión de inventario web utilizando la tecnología de código QR, que permita llevar el control y gestión de los equipos e insumos en el CEAPUCV.

1.4. Objetivos Específicos

- Identificar los requerimientos funcionales y no funcionales del sistema de gestión de inventario web del CEAPUCV.
- Investigar sobre el uso e implementación de la tecnología de código QR.
- Diseñar un sistema de gestión de inventario web utilizando la tecnología de código QR para el CEAPUCV.
- Desarrollar el sistema de gestión de inventario web utilizando la tecnología de código QR para el CEAPUCV.
- Realizar las pruebas funcionales y de aceptación necesarias para comprobar el correcto funcionamiento del sistema de inventario.

1.5. Método de desarrollo

El método de desarrollo ágil de software que se decidió implementar fue XP, ya que éste es el que mejor se adapta a las necesidades de desarrollo para realizar este proyecto. Sin embargo no todas las características se ajustan a la perfección; por lo que se decidió hacer una configuración al método. A continuación se detalla esta configuración.

Una de las características de XP, es que se realizan "pequeñas entregas". En este proyecto se proporcionan "medianas entregas", el cual consiste en entregas de versiones lo más completas posibles, cercanas al producto final.

En este proyecto se decidió complementar las historias de usuarios, con documentos formales, para así tener un mejor enfoque al momento de diseñar y desarrollar el sistema de inventario a realizar. Entre estos documentos formales que se decidió agregar están: el mapa relacional de la base de datos, los casos de uso y la arquitectura del sistema.

Estos ajustes se realizaron para tener una mejor eficiencia y calidad de la entrega del producto final.

1.6. Justificación

Al carecer el centro educativo de un sistema adecuado de gestión de equipos e insumos, hacer uso de las tecnologías más recientes no solo garantiza dar solución a este problema, sino que también se asegura que las posibles actualizaciones o modificaciones que existan en el futuro puedan fácilmente acoplarse a la solución propuesta. La explicación de este ultimo punto es

sencilla, el utilizar las últimas versiones de las herramientas de desarrollo, asegura que posibles nuevas versiones sean totalmente compatibles con la que se desarrollará.

Una posible solución es el uso de la tecnología *Radio Frequency Identification* (RFID, identificación por radiofrecuencia), para la identificación de los equipos registrados en el inventario. Pero pese a las muchas ventajas que ofrece la tecnología RFID [1], al querer sustituir los códigos de barra, se puede decir que no es del todo viable para la realización de este proyecto. Esto se debe al hecho de que el mercado de los dispositivos y etiquetas para la implementación de RFID en Venezuela es muy reducido y como consecuencia los precios de los mismos son muy elevados a diferencia de sus precios reales. Debido a ello se puede adquirir los dispositivos y etiquetas en el exterior, pero el sistema cambiario de monedas hace casi imposible esta solución.

Por estas razones se propone el uso de códigos de barra, como medida de identificación de productos y/o materiales, específicamente a los códigos QR, ya que estos aportan características distintas y mejores al resto de los códigos de barra. Dentro de estas características que se toman en cuenta, y que proporcionan el peso suficiente para inclinar la decisión a este tipo de código, tenemos: el corto tiempo de lectura e interpretación del código debido a las especificaciones explicadas en el capítulo 2 sección 2.1.2, la posibilidad de leer los códigos con el uso de teléfonos inteligentes y así aprovechar los dispositivos disponibles y no realizar gastos adicionales en la compra de lectores específicos. La amplia gama de aplicaciones de software libre que existen para la lectura e interpretación de estos códigos, para utilizar los teléfonos inteligentes. La ventaja de poder leer el código desde cualquier angulo con un espectro de 360 grados y una recuperación de datos de hasta el 30 %, en caso de que la etiqueta sufra de algún daño, con el nivel de corrección H, del cual depende la versión de la etiqueta; y en consecuencia, su tamaño. Para imprimir los códigos QR basta con tener a disposición una impresora que realice su trabajo a 300 puntos por pulgada (ppp), cabe acotar que cualquier impresora láser actual cumple con esta característica. Son estas las características que motivan a proponer los códigos QR como sistema de identificación dentro del sistema de inventario a desarrollar.

1.7. Alcance

Se definen como alcance del presente proyecto los siguientes puntos:

- Desarrollar un sistema de inventario web (control y gestión de bienes, suministros e insumos) utilizando la tecnología de código QR.
- El sistema se limita a automatizar funcionalidades básicas de un inventario para el control y gestión de bienes, suministros y servicios. Estas funcionalidades resumen en la captación del origen„ administración del ingreso y egreso de los equipos, suministros e insumos, control de ubicación de los mismos, así como también la posibilidad de realizar consultas

del estado de los bienes y generación de reportes en formato PDF ó impresos. Sin tomar en cuenta el conjunto completo de características y especificaciones que involucran los sistemas de inventarios complejos, ya que estas no fueron contempladas en el conjunto de requerimientos funcionales provistos por los usuarios finales.

Así mismo, no se contemplará en el alcance de éste proyecto:

- La integración con el sistema de biblioteca desarrollada por alumnos de la Universidad Central de Venezuela.
- No se llevará un control sobre aspectos como costo-precio, ni la depreciación del valor de los equipos e insumos el cual está presente en los sistemas de inventarios comunes, así cómo el manejo de la fecha de vencimiento de los medicamentos y productos de enfermería, ya que no son requisitos funcionales del sistema.

Capítulo 2

Marco Conceptual

En este capítulo se expone los conceptos necesarios para poder entender de manera satisfactoria el proyecto realizado. Se muestran conceptos como el código QR, y todas sus características, así como también que es un sistema de inventario, y sus funciones. Se muestran las características de las herramientas utilizadas para el desarrollo de este proyecto, como el *framework* Ruby on Rails y el manejador de base de datos MySQL.

2.1. Código QR

La tecnología que se abarca en la presente sección es códigos de barra, debido al importante papel que han jugado en el ámbito de la identificación y automatización de la gestión de productos y/o materiales. Luego se profundiza más en los códigos de QR, tecnología en que se basa este proyecto.

2.1.1. Código de Barras

Desde el momento en que las industrias automatizan sus producciones se hace necesario la identificación unívoca de sus productos con respecto a otros, como solución a este dilema surgen los códigos de barras, los cuales permiten la identificación de forma automática y precisa.

Definido según GS1 (*Global Standard 1*) [2] como "una identificación única y no ambigua de un producto, localización y/o servicio que se representa mediante un símbolo compuesto por barras oscuras y espacios claros paralelos entre sí y de anchos variables" y como "la representación gráfica, mediante barras y espacios, de un conjunto de caracteres numéricos, o alfanuméricos que permiten la identificación inequívoca de artículos". A continuación se hará una descripción detallada de como esta conformado el código de barras.

Historia

Desarrollado y patentado por primera vez por Joseph Woodland y Bernard Silver, [3] la idea surge por la petición de un presidente de una cadena de alimentos quién solicita a uno de los decanos del Instituto de Tecnología de Drexel, Filadelfia, donde Silver estudiaba, desarrollar un método para la lectura automática de la información de sus productos. Planteado el problema y con la motivación necesaria Woodland y Silver comienzan a trabajar juntos en el proyecto, la primera idea de Woodland fue desarrollar una tinta sensible a los rayos ultravioleta, idea de la cual se logró desarrollar un prototipo, pero la idea tuvo que ser descartada por la inestabilidad del sistema y sus altos costos; pronto se tuvo que volver a empezar desde cero. No fue sino hasta el 20 de octubre de 1949 cuando se presenta la patente titulada como "método de clasificación de productos mediante reconocimiento de formas". En la Figura 2.1 se muestra lo que fue el primer código de barras similar a un ojo, formado por círculos concéntricos.

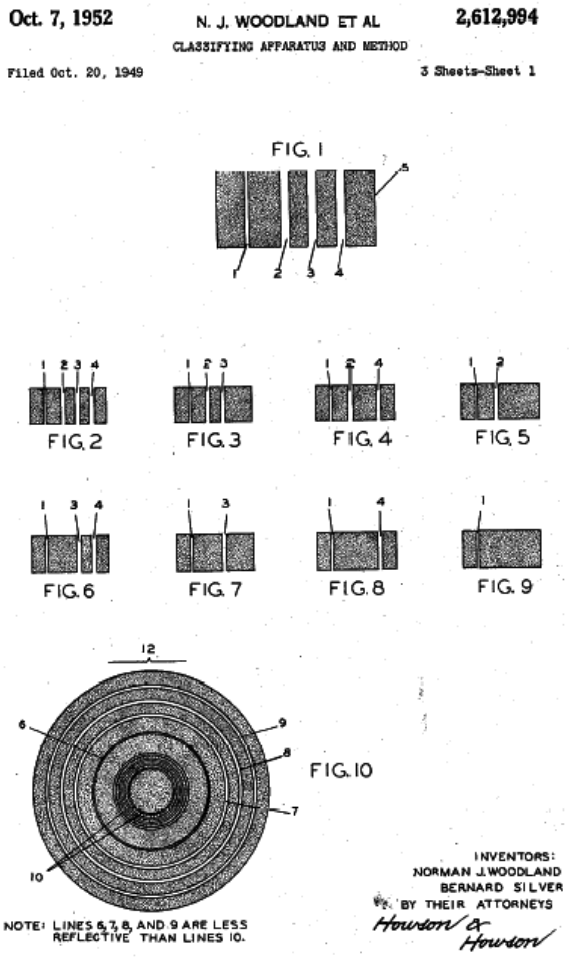


Figura 2.1: Primer código de barras
tomada de: http://www.codigodebarras.pe/wp-content/uploads/2012/05/bullseye-barcode.jpg

Descrito por sus creadores como un símbolo, esta constaba de cuatro líneas blancas sobre un fondo oscuro, donde la primera era tomada como referencia para la distancia de las tres restantes. La patente americana se registra el 07 de octubre de 1952 bajo el número 2.612.994.

Ninguno de los creadores ganaron dinero con su idea multimillonaria debido a que vendieron la patente a RCA (Radio Corporation of America) en 1952 por solo una pequeña suma de dinero, mucho antes de que se comercializara la tecnología. Esta patente venció en 1969, 5 años antes de su participación en el campo industrial. Bernard Silver falleció en 1962 a los treinta y ocho años de edad en un accidente automovilístico y no logra ver el uso comercial de su invención. Por su parte Woodland era empleado de IBM (International Business Machines) y perteneció al equipo encabezado por George Lauren para el desarrollo de un símbolo UPC (Universal Product Code) que en 1973 la empresa presentó, el cual es el conocido en la actualidad como código de barras. Woodland siguió trabajando en IBM hasta su jubilación en 1987. En 1992 fue galardonado con la Medalla Nacional de Tecnología e Innovación la cual le fue entregada por el entonces presidente George H. W. Bush. En el año 2011 los nombres de los padres del código de barras entran en el Salón de la Fama de los Inventores de los Estados Unidos. Woodland muere el 9 de diciembre de 2012 con 91 años de edad [4, 5].

Simbología

Definiendo como símbolo al código de barras impreso, se entiende por simbología a la correspondencia entre la información y el código que la representa [6], es decir, la simbología es el correspondiente a lo que sería un idioma que debe ser comprendido por el lector del símbolo. Existen distintos tipos y pueden ser clasificados en dos grandes grupos teniendo en cuenta dos criterios diferentes:

El primer grupo distinguido por ser continuo o discreto, donde los símbolos continuos comienzan con un espacio o con una barra; a diferencia de los discretos que comienzan y terminan siempre con una barra y el espacio entre caracteres es simplemente ignorado debido a que no es lo suficientemente ancho.

El segundo grupo caracterizado por su dimensionalidad pueden ser bidimensionales o multidimensionales, donde en los símbolos bidimensionales las barras pueden ser anchas o angostas, por otro lado el ancho de las barras en símbolos multidimensionales son el múltiplo de una constante determinada (X).

Tipos de Simbologías

Existen distintas simbologías dentro del mundo de los códigos de barras, aunque esto es así, la mayoría de las personas están acostumbradas a ver uno o dos de estos tipos, los cuales son de uso común en la industria. Sin embargo a continuación se presentan algunas de las simbologías más comunes:

UPC/EAN

Son los símbolos de los productos que pueden ser escaneados en caja, orientados para su uso en la industria alimenticia e industria minorista [6]. Los símbolos UPC poseen una longitud fija de 12 dígitos, poseen un tamaño compacto y razonable. En la Figura 2.2 se observan las diferencias que existen entre la simbología UPC y EAN, entre las cuales se pueden señalar la cantidad de dígitos que en UPC es 12 y en EAN es 13, el dígito extra de EAN es el primer dígito de UPC repetido al principio de EAN.

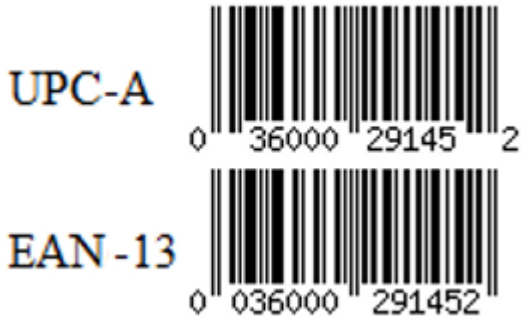


Figura 2.2: Simbología UPC/EAN
tomada de: http://faculty.berea.edu/pearcej/CSC226/tasks/UPC-EAN.png

El último dígito que se conoce como "carácter de control de módulo" en UPC se encuentra fuera del código de barras mientras que en EAN se encuentra dentro del mismo.

El carácter de control de módulo posee un algoritmo para la verificación del mismo. El cual se enuncia a continuación:

Teniendo en cuenta los dígitos del código UPC de la Figura 2.2: 036000291452 y recordando que el último dígito es el carácter de control no lo tomamos en cuenta para seguir el algoritmo.

1. Se suman los dígitos impares: 0+6+0+1+5 = 14. El resultado de la suma se multiplica por 3: 14 x 3 = 42

2. Se suman los dígitos pares: 3+0+0+9+4 = 16

3. Se suman los resultados obtenidos en el paso 1 y 2: 42+16= 58. Se calcula mod 10 al resultado: 58 mod 10 = 8

4. Si el resultado es distinto de 0 se le resta a 10 el resultado obtenido: 10 - 8 = 2. Este resultado es el que debe coincidir con el carácter de control de módulo.

Código 39

Dada la necesidad de algunos de insertar letras dentro del código de barras surge Código 39, simbología que hace dicha inclusión y que es usado como un estándar para usos no relacionados con la alimentación [6, 7]. Pero como punto negativo para esta simbología podría llegar a ser muy largo, lo que en ocasiones no conviene para el tamaño de la etiqueta en donde será impreso. En la Figura 2.3 se muestra un ejemplo de cómo seria un código de barras con esta simbología.

Figura 2.3: Código 39
tomada de http://html.rincondelvago.com/000697776.png

Código 128

Debido a las debilidades del código 39 como su corta gama de caracteres y su gran tamaño, surge el Código 128, simbología que amplía la selección de caracteres y reduce su tamaño haciendo de la Figura un símbolo denso y compacto [6, 7]. Utilizado en los ambientes de transporte donde el tamaño de la etiqueta es un factor importante. Como imagen de referencia tenemos la Figura 2.4.

Figura 2.4: Código 128
tomado de: http://www.conpresencia.com/images/c128.gif

Entrelazado de 2 a 5

Lo que se observa en la Figura 2.5 también se conoce como ITF, es una simbología muy utilizada en la industria del transporte y almacenaje [6, 7].

Figura 2.5: Entrelazado de 2 a 5
tomada de: http://www.tecno-symbol.com/imgs/simbolog%EDas/itf.gif

Empleado comúnmente en las cajas de cartón corrugado. A diferencia del anterior este solo utiliza caracteres numéricos. Aunque es un símbolo pequeño es ligeramente más largo que el UPC cuando su codificación es de 10 dígitos. Una característica de esta simbología es que representa mayormente números pares, en caso de que el número sea impar se antepone un cero al principio del mismo.

Posnet

Esta simbología es tal cual se observa en la Figura 2.6 y la misma es para uso exclusivo del Servicio Postal de Estados Unidos (USPS por sus siglas en inglés) [7].

Figura 2.6: Posnet
tomada de: http://www.tecno-symbol.com/imgs/simbolog%EDas/postnet.gif

Estructura de los códigos de barra

Como se observa en la Figura 2.7, los tres primeros dígitos corresponden al país de donde pertenece el código de barras, los siguientes 4 dígitos son de la empresa a la cual pertenece dicho código. A continuación los siguientes 5 son los dígitos pertenecientes al producto y por último el dígito ya presentado como "caracter de control de módulo"; esta fue la estructura con respecto a los dígitos.

Figura 2.7: Estructura del código de barras
tomada y adaptada de:
http://www.elarsenal.net/2012/02/28/cumple-codigo-de-barras-25-anos-en-mexico/

La estructura con respecto al símbolo y según la imagen es la siguiente: los recuadros identificados con la letra A se conocen como "quiet zone" son las zonas blancas al principio y al final de cada código de barras, su objetivo es que el lector óptico distinga entre un código y otro. Las zonas identificadas con la letra B son las zonas de Inicio y Fin, constan de una combinación de barras que indican exactamente eso, el inicio y el fin del código. El espacio nombrado con la letra C es la zona donde se ubican los caracteres de datos donde entran los de empresa y producto. Por último volvemos a tener el espacio reservado para el "carácter de control de módulo" identificado con el literal D.

Pasos para implementar el código de barras

La GS1 Venezuela® asociación civil sin fines de lucro quien se encarga de desarrollar, administrar e implementar el sistema GS1, sugiere seguir los siguientes pasos para la implementación del mismo[8, 9]:

Paso 1: obtener prefijo de compañía. El fabricante de productos debe dirigirse a la GS1 Venezuela® para afiliarse y obtener el permiso de la utilización de dicho prefijo de compañía el cual está compuesto por el código del país que en el caso de Venezuela es el 759 y el código de empresa el cual es asignado por la mencionada asociación. Es válido acotar que aunque esta asignación se hace a nivel local son unívocos a nivel mundial.

Paso 2: asignación de números. Aprobado el paso 1 la empresa procede a asignar los códigos a sus productos teniendo en cuenta que cualquier variación entre tipo de producto llámese: marca, sabor, tamaño, promoción, etc., requieren un código diferente. GS1 Venezuela® proporciona a las empresas cursos de capacitación en los cuales se les da a conocer a las mismas las reglas de asignación de código de productos según las necesidades de cada empresa.

Paso 3: seleccionar el sistema de impresión. Se pueden aplicar distintas técnicas de impresión: serigrafía, flexografía, litografía, rotograbado y offset; para la impresión de etiquetas se debe hacer uso de impresoras ya sean térmicas, láser, Ink Jet o Transferencia Térmica.

Recomendaciones por parte de GS1 Venezuela®: si la cantidad de etiquetas es muy grande pedir a un proveedor de impresión de códigos, la impresión de las mismas, en caso contrario obtener el material indicado para realizar esta actividad por cuenta propia de la empresa, así como también solicitar a GS1 Venezuela® contacto de empresas que proveen dichos servicios o material para estas actividades y otras relacionadas.

Paso 4: seleccionar el escenario de escaneo para el código de barras. Las características del código de barras dependerá del lugar donde va a ser escaneado el producto. Ejemplos de escenarios para la actividad pueden ser puntos de ventas, centros de distribución, etc. Una vez se tenga definido el escenario se puede pasar a escoger una simbología.

Paso 5: seleccionar la simbología adecuada del código de barras. Ya teniendo establecido el escenario donde se escanearán los códigos de barras, se procede a escoger la simbología las cuales se contemplaron ya en este capítulo y según dicho escenario se puede escoger una u otra simbología, sin embargo GS1 Venezuela® recomienda participar en sus cursos de capacitación los cuales también guían a las empresas a escoger el que mejor se ajuste a las necesidades de las mismas y cumpliendo los estándares GS1.

Paso 6: seleccionar las dimensiones del código de barras. Es en este paso donde inicia el proceso de diseño del código de barras, seleccionando el tamaño que debe poseer y este se ve ajustado dependiendo de la simbología a utilizar y cómo será impreso.

Las unidades de consumo deben poseer un tamaño máximo y un tamaño mínimo. Al tamaño original (100 %) que se usará como patrón se le conoce como "Factor de Magnificación 1", como tamaño mínimo no se puede descender de un 80 % al Factor 1 y como tamaño máximo se dispone de hasta un 200 % sobre el patrón.

Aún así GS1 Venezuela® recomienda el uso de una película maestra en el proceso de impresión

del código de barras. En la Tabla 2.1 se hace una relación entre el tipo de impresión y el factor de magnificación que se puede utilizar.

Tabla 2.1: Relación tipo impresión y factor de magnificación
tomada de: http://www.gs1ve.org/pasoscodigo.php

Tipo Impresión	Factor Mínimo de Magnificación
offset	80 %
flexografía	100 %
serigrafía	110 %
litografía	80 %
tipografía	80 %

Paso 7: generar texto del código de barras. El texto debajo de los códigos de barras es de norma obligatoria en la gran mayoría de los casos. Y la razón para esto es básica, ya que si llegase a ocurrir un error en el código o este es dañado existe el código escrito para su inserción en el sistema de forma manual. En resumen el texto escrito es un respaldo de la información que almacena el código de barras.

Paso 8: seleccionar combinaciones legibles para el código de barras. La base para una buena lectura es establecer un buen contraste entre las barras y el fondo para que el lector pueda cumplir su labor de forma satisfactoria, por lo que se recomienda que las barras sean de un tono más oscuro que el fondo.

Aunque la combinación de barras negras y fondo blanco es la más usual y la más idónea, existen otras posibles y viables combinaciones que quizás respeten y se ajusten al diseño del producto. Al momento de escoger la combinación es importante realizar una prueba de color para asegurar que la lectura puede realizarse sin inconvenientes. En algunos casos aunque la combinación al ojo humano sea la adecuada, para el lector puede no serla, esto puede ser ocasionado por el sustrato sobre el cual se ha impreso el código. Para aclarar este punto se muestra en la Figura 2.8 un conjunto de combinaciones válidas y en la Figura 2.9 otro conjunto de combinaciones, pero no válidas.

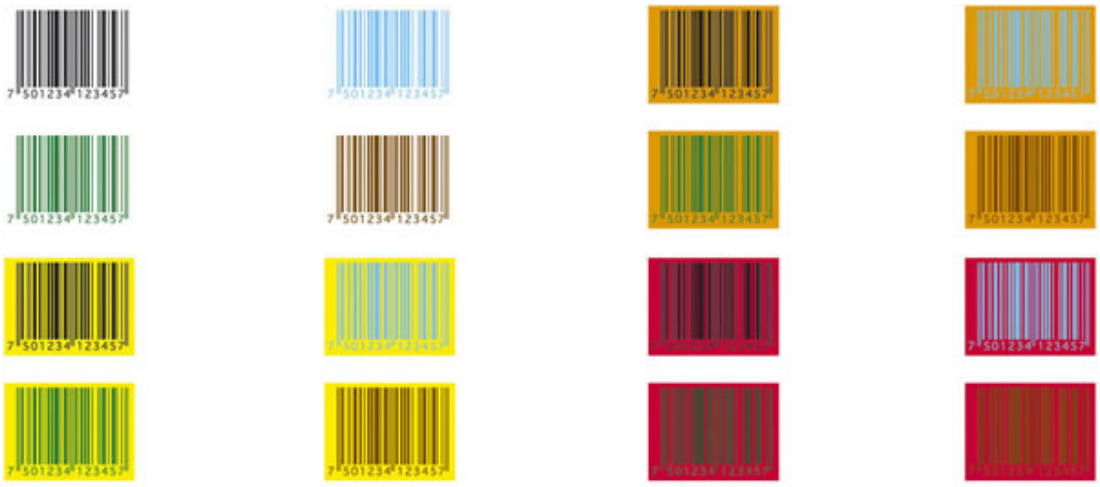


Figura 2.8: Combinaciones válidas
tomada de: http://www.gs1ve.org/images/1-aso-boletines_clip_image015_0000.jpg

Figura 2.9: Combinaciones inválidas
tomada de: http://www.gs1ve.org/images/1-aso-boletines_clip_image016_0000.jpg

Paso 9: considerar la ubicación y posición del código de barras. La ubicación del código de barras es indispensable para hacer del proceso de lectura un proceso ágil. Se debe tener en cuenta diferentes recomendaciones en cuanto a la ubicación del símbolo, entre las cuales se tienen: ubicarlo en la base natural del producto, tomar en cuenta el proceso de empaque del producto para evitar que el símbolo sea dañado, aplastado, distorsionado, doblado o cortado durante el mismo.

Con respecto a la posición de símbolo es recomendable seguir el sentido de impresión del mismo y de esta forma evitar errores mínimos que son imperceptibles al ojo humano pero que podrían presentar problemas para el lector.

Paso 10: verificar la calidad del código de barras. Con el pasar del tiempo las empresas tanto nacionales como extranjeras se han vuelto exigentes en cuanto a la calidad de la impresión del código de barras. Existe la Norma ISO/IEC 15416 "Especificaciones Técnicas de Calidad de Impresiones de Códigos de Barras" describe el método para verificar la calidad de la simbología del código de barras después de haber sido impresa.

GS1 Venezuela® proporciona la verificación de los códigos de barra a aquellos que estén afiliados a la asociación aplicando la Norma ISO/IEC.

2.1.2. Códigos de dos dimensiones 2D

La característica primordial de estos códigos, la cual diferencia principalmente de los códigos de una dimensión (1D), es que pueden almacenar más información [10]. Dependiendo del tipo exacto, se puede lograr almacenar hasta casi 7000 caracteres. Hasta hace algunos años se necesitaban de unos lectores especiales para poder trabajar con este tipo de códigos, gracias al avance tecnológico actualmente alguno de estos pueden ser leídos por cualquier smartphone con una simple aplicación y el uso de la cámara. A continuación se nombran los códigos de barra en 2D más famosos y algunas de sus características:

Código PDF417

Es un código de alta densidad, el cual en los últimos años ha ganado campo y se ha ido estableciendo como un estándar en la industria de identificación, presentándose en documentos como DNI (Documento Nacional de Identidad), permisos de conducir, servicios sociales, etc. Lo práctico de esta simbología radica en que puede almacenar 1,1 kilobytes de información que puede contener datos biométricos como huellas dactilares, firmas, fotografías, etc. En la Figura 2.10 se puede observar este código.

Figura 2.10: PDF417
tomada de: http://www.tecno-symbol.com/imgs/simbolog%EDas/pdf417.gif

Estas simbologías son entonces las más utilizadas, se ha nombrado el campo de desenvolvimiento de cada una nombrando sus fortalezas y debilidades en algunas de las mismas, así como también los factores que forzaron sus sustituciones. Así como estas, existen muchas más simbologías las cuales han ido evolucionando y llegando a las que se conocen hoy en día.

Código DataMatrix

Según GS1 [11] "puede ser impreso como un símbolo cuadrado o rectangular, compuesto por puntos individuales o cuadrados, la representación es un cuadriculado ordenado de puntos oscuros y claros, bordeado por un buscador de patrón". Se puede observar un ejemplo de como sería en la Figura 2.11.

Figura 2.11: DataMatrix
tomada de: http://www.codigodebarras.com/tema.php?ID=dos_dimensiones

Características:

- Capaz de codificar todos los caracteres ASCII.
- Con una capacidad de hasta 2000 caracteres.
- El tamaño se ajusta a la cantidad de datos que contiene.
- Utilizada en la industria detallista para codificar por lote y fecha de fabricación.

Código MaxiCode

Tal como se puede ver en la Figura 2.12, el código MaxiCode está compuesto por pequeños módulos hexagonales que rodean a tres círculos concéntricos ubicados en el medio de la Figura [12]. Los hexágonos negros representan el 1 en binario y el los blancos representan el 0. Este código fue creado por UPS (United Parcel Systems) que es una de las grandes empresas de paquetería del mundo junto con empresas como FedEx, DHL y USPS.

Figura 2.12: MaxiCode
tomada de: http://www.codifikar.com.ar/images/maxicode.gif

Características:

- Codifica todos los caracteres ASCII.
- Posee una capacidad de 93 caracteres .
- Siempre posee el mismo tamaño el cual es de 28.14mm x 26.91mm.

Código Quick Response

El código *Quick Response Code* (QR, código de respuesta rápida) fue diseñado en Japón en el año 1994 por una compañía automotriz llamada Denso Wave, la cual era subsidiaria de Toyota [2]. El éxito probado del QR se debe precisamente a su estándar abierto y a que su decodificación puede realizarse con cualquier teléfono móvil con cámara sin ser importante la calidad de ésta. En Japón está muy extendido su uso y es raro que los dispositivos móviles no vengan con una aplicación incorporada para poder decodificarlos o con la facilidad de ser adquiridos en el *store* del sistema operativo. En la Figura 2.13 se puede detallar un ejemplo del código QR.

Figura 2.13: Código QR
tomada de: http://www.codigodebarras.com/tema.php?ID=dos_dimensiones

Características

- El estándar japonés fue publicado en enero de 1998 y el estándar ISO (ISO/IEC18004) fue aprobado en junio de 2000.
- Se diseñó con la intención de ser decodificado a alta velocidad.
- Adoptado como el estándar 2D en toda Europa.
- Codifica todos los caracteres ASCII además de información binaria como imágenes.
- Puede codificar hasta 7089 caracteres.
- El tamaño depende de la cantidad de información codificada.
- Posee 40 versiones y 4 grados de corrección (L, M, Q, H).
- Cada versión posee un tamaño, empezando por la versión 1 con un tamaño de 21x21 módulos, aumentando en 4 módulos por cada versión siendo la segunda 25x25 módulos hasta la llegar a la cuadragésima de 177x177.

A demás ofrece características que proporcionan ventajas que otros códigos no proporcionan:

- QR tiene la capacidad de corregir errores. En el caso de que parte del código resultara dañado existe la posibilidad de restaurar datos. Existen diferentes niveles de corrección de datos pudiendo llegar hasta corregir el 30 % de la información. El sistema de corrección de errores se basa en Reed Solomon.
- Puede ser leído a mayor velocidad que el resto de los códigos y puede ser leído desde cualquier orientación en los 360°.

- Las especificaciones del código están abiertas al público lo que ha permitido crear lectores de QR de muy bajo costo o incluso gratuito.
- A consecuencia de lo anterior se han desarrollado muchas aplicaciones, y debido a que los teléfonos de la actualidad poseen cámara y soportan aplicaciones; haciendo de estos dispositivos la mejor elección para sus uso como lectores de códigos QR; donde la mayoría de estas aplicaciones son gratuitas.

Estructura del código QR

A continuación se mencionan y explican las diferentes partes que conforman la estructura de un código QR [13].

- **Símbolo**: es toda la imagen QR Code, formada por *módulos*, que conforman los datos en la *región de codificación*, y los *patrones de función*.
- **Codeword:** conjunto de 8 módulos que puede tener diferentes formas dependiendo de su localización en el símbolo y que sirve para almacenar información codificada de los datos o de la corrección de errores.
- **Codeword restante**: codeword de relleno para llenar posiciones sin codeword asignado para completar la capacidad total del *símbolo*. Va detrás de los codewords de corrección de error.
- **Información de formato:** patrón codificado que contiene información sobre el grado de corrección de errores con el que se han codificado los datos de la región de codificación y el tipo de máscara que se les ha aplicado.
- **Información de versión** : patrón codificado en los símbolos de versión 7 o superior que contiene información que indica la versión del símbolo.
- **Módulo**: cuadro blanco o negro que en conjunto componen el símbolo QR Code.
- **Patrón de función:** partes del código QR que no contiene los datos codificados, sino información necesaria para la decodificación de éstos. Los patrones de función son: patrón de localización, separador, patrón de alineamiento y patrón temporizador.
- **Patrón localizador:** patrón de función que existe por triplicado en el símbolo, situado en las esquinas superiores y la inferior izquierda. Sirven para calcular la orientación rotacional del símbolo.
- **Patrón temporizador**: secuencia alternada de módulos blancos y negros que ayuda a calcular las coordenadas de los módulos del símbolo.

- **Patrón de alineamiento:** patrón de función que permite resincronizar las coordenadas de correlación de la imagen QR Code ante posibles distorsiones moderadas de ésta.
- **Región de codificación**: región del *símbolo* no ocupada por *patrones de función* y sí por codewords de datos y de corrección de errores, y también por la *información de formato* y *versión*.
- **Separador**: patrón de función formado por *módulos* blancos, cuyo ancho es de un módulo y que separa los patrones localizadores del resto del símbolo.
- **Versión:** tamaño del símbolo que puede ir desde la versión 1 con 21x21 *módulos* hasta la 40 con 177x177 módulos. Dependiendo de la versión, el *símbolo* puede tener o no algunos de los elementos descritos y en diferente número; especialmente *patrones de alineamiento* y la *información de versión*.
- **Zona silenciosa:** zona que rodea al símbolo que debe estar en blanco (negro en caso de reflectancia inversa) para delimitar correctamente sus bordes, debe tener una anchura mínima de 4 módulos.

Los códigos QR están formados por módulos colocados en una estructura cuadrada. Ésta estructura contiene la región de codificación y los patrones de función, que son: localizador, separador, temporizador y de alineamiento, tal como se muestran en la Figura 2.14. El símbolo debe estar rodeado en sus cuatro lados por una zona de silencio [13].

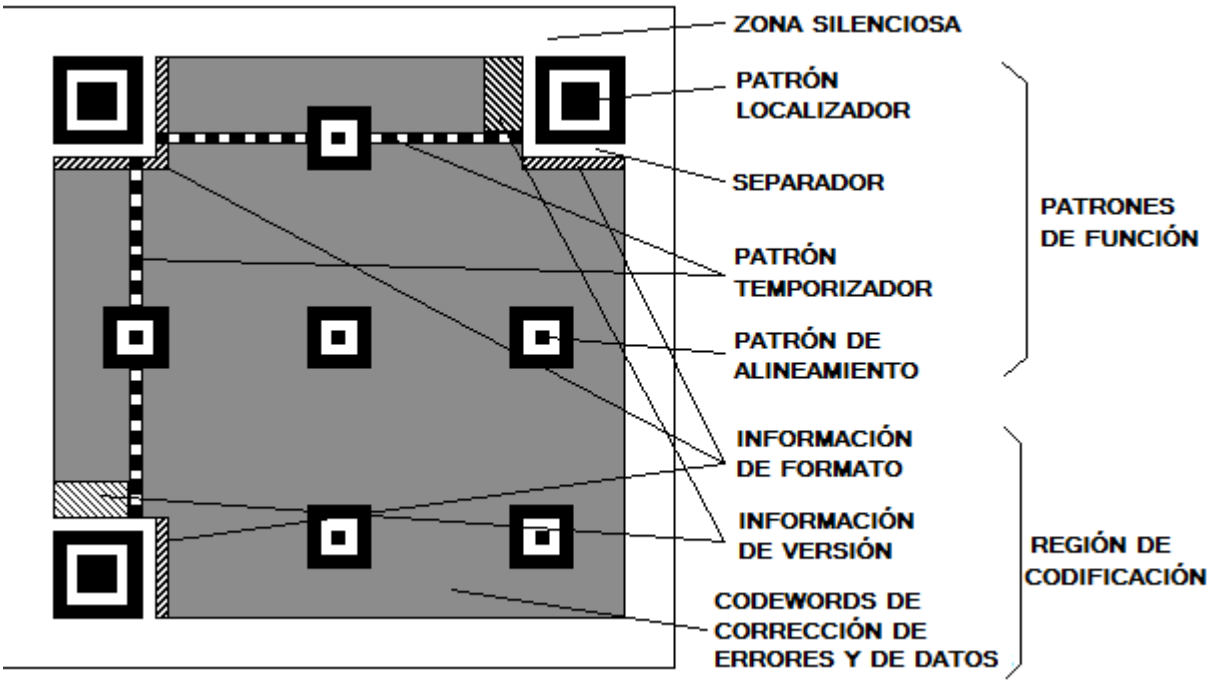


Figura 2.14: Estructura QR
tomada de: [13]

Hay 40 versiones, cada una con un número de módulos, la versión 1 tiene 21x21 módulos y la 40 177x177; el número de módulos se incrementa de 4 en 4 de una versión a otra, como se muestra en las figuras 2.15 y 2.16. Las versiones también se diferencian en el número de codewords que contienen y el de patrones de alineamiento, algunas tienen bits de relleno pero otras no. Las versiones anteriores a la 7 no tienen información de versión, la versión 1 no tiene ningún patrón de alineamiento. Todas tienen tres patrones localizadores, dos patrones temporizadores, tres separadores y la información de formato por duplicado.

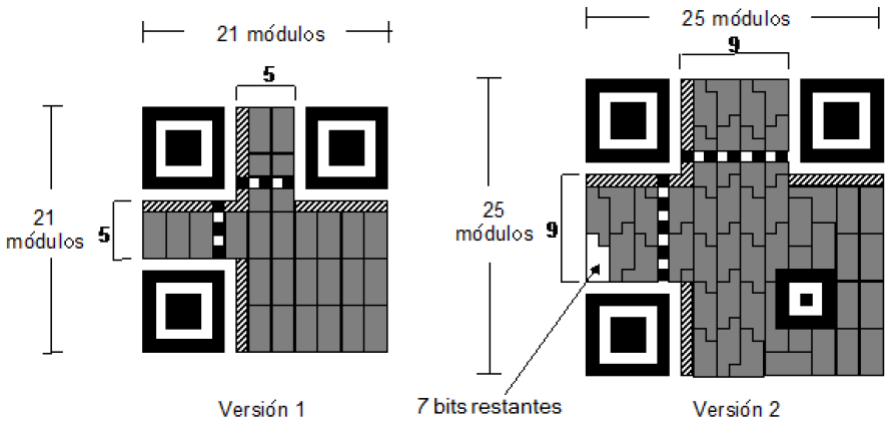


Figura 2.15: Estructura de código QR versiones 1 y 2
tomada de: [13]

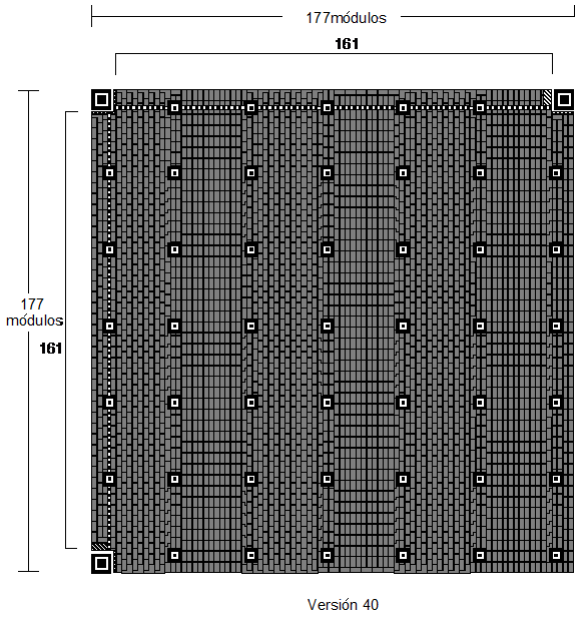


Figura 2.16: Estructura de código QR versión 40
tomada de: [13]

El patrón localizador se sitúa en las esquinas superior izquierda, superior derecha e inferior izquierda del símbolo QR Code [13]. Está formado por un cuadrado relleno de 3x3 módulos negros, rodeado de un cuadrado de 5x5 módulos blancos que a su vez está rodeado por otro cuadrado de 7x7 módulos negros como se muestra en la Figura 2.17. Será muy difícil encontrar un patrón de módulos similar a este en otras partes del símbolo. Tener éxito en encontrar los 3 patrones localizadores de un símbolo supone poder calcular la orientación en el campo de visión de éste.

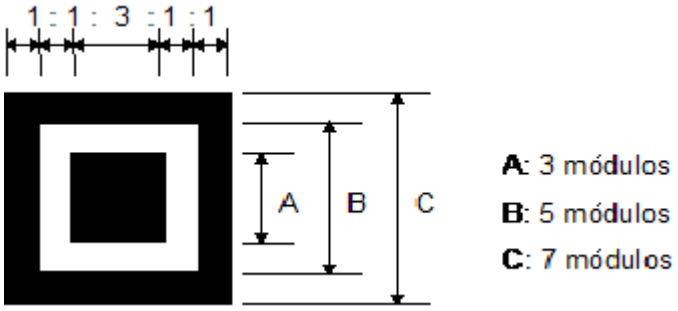


Figura 2.17: Patrón localizador
Tomada de: [13]

Los separadores están formados por módulos blancos y rodean los bordes de los patrones localizadores que dan a la parte interior del símbolo.

Los patrones temporizador son dos, uno vertical y otro horizontal. Están formados por una línea o columna de módulos blancos y negros alternados, comenzando y terminando en un módulo negro. Posibilitan que la versión del símbolo y las coordenadas de los módulos puedan ser determinadas. El temporizador horizontal cruza la fila número 6 entre los separadores superiores y el vertical igual pero cruzando la columna 6.

Los patrones de alineamiento están formados por un módulo negro, rodeado de un cuadrado de 3x3 módulos blancos que a su vez está rodeado por otro cuadrado de 5x5 módulos negros. Su número en el símbolo varía según la versión.

La región de codificación contiene los codewords que representan los datos, también contiene codewords de corrección de errores, la información de formato y la información de versión en la mayoría de casos.

La zona de silencio debe tener un grosor de 4 módulos rodeando los cuatro bordes del símbolo.

Proceso de codificación

El proceso de codificación de un código QR se compone de 7 pasos [13], los cuales se enumeran y describen a continuación:

1. Análisis de datos:

Se hace un análisis de los caracteres a codificar, para de esta manera encontrar el modo óptimo de codificarlos y así ahorrar la mayor cantidad de espacio. Además se puede tomar en cuenta que como las capacidades de los símbolos QR, aumentan de manera discreta en cada versión, no siempre es necesaria alcanzar la máxima eficiencia. Ahora se describen los modos de codificación.

Modo numérico: codifica caracteres numéricos del conjunto [0,9]. Normalmente, 3 caracteres numéricos se representan con 10 bits en este modo.

Modo alfanumérico: codifica un conjunto de 45 caracteres. Los 10 dígitos del modo numérico, los 26 caracteres del alfabeto internacional y 9 símbolos más (ESPACIO, $, %, *, +, -, . , /, :). Normalmente 2 caracteres alfanuméricos se representan con 11 bits en este modo.

Modo Byte: los datos se codifican con 8 bits por carácter en código ASCII, aunque podría definirse un conjunto alternativo.

Modo Kanji: codifica los caracteres Kanji del alfabeto japonés de acuerdo con el sistema Shift Jis basado en JIS X 0208 (estándar japonés que describe el código QR). Cada carácter de 2 bytes es compactado en un codeword de 13 bits.

Modo interpretación de canal extendido (ICE): permite definir interpretaciones diferentes del conjunto de caracteres por defecto. Provee un método consistente para especificar interpretaciones particulares.

2. Codificación de los datos:

Los datos se convierten en un flujo de bits consistente en uno o más subconjuntos de modos diferentes. Si se utiliza el ICE por defecto, el flujo de bits comienza por el primer indicador de modo. Si se utilizaran uno o más ICEs diferentes del por defecto, el flujo de bits comenzaría con la cabecera ICE seguida del primer subconjunto.

Los subconjuntos se componen de indicador de modo (4 bits), de contador de caracteres y de los datos. Cada subconjunto de modo empieza con el bit más significativo del indicador de modo y termina con el bit menos significativo del flujo de datos. No existen separadores entre los subconjuntos ya que su tamaño y lugar de inicio están delimitados inequívocamente por el indicador de modo y el contador de caracteres.

El indicador de modo tendrá los siguientes valores:

- Numérico = 0001
- Alfanumérico = 0010
- Byte = 0100

- Kanji = 1000
- ICE = 0111
- Fin de mensaje = 0000

Se presentan los bits del contador de caracteres en la Tabla 2.2.

Tabla 2.2: Bits del contador de caracteres
tomada de: [13]

Versión	Modo numérico	Modo alfanumérico	Modo byte	Modo kanji
1-9	10	9	8	8
10-26	12	11	16	10
27- 40	14	13	16	12

En el **modo numérico**, los datos de entrada se dividen en grupos de tres dígitos, cada grupo se convierte en su equivalente binario de 10 bits. Si el número de dígitos no es múltiplo de tres, los uno o dos últimos son convertidos a 4 o 7 bits. Los datos binarios resultantes son entonces concatenados y se les pone como prefijo el indicador de modo y el indicador contador de caracteres.

En el modo alfanumérico, a cada carácter se le asigna un valor entre 0 y 44 acorde a la Tabla 2.3:

Tabla 2.3: tabla de codificación alfanumérico
tomada y adaptada de: [13]

Char	Valor	Char	Valor	Char	Valor	Char	Valor	Char	Valor	Char	Valor	Char	Valor	Char	Valor
0	0	6	6	C	12	I	18	O	24	U	30	SP	36	.	42
1	1	7	7	D	13	J	19	P	25	V	31	$	37	/	43
2	2	8	8	E	14	K	20	Q	26	W	32	%	38	:	44
3	3	9	9	F	15	L	21	R	27	X	33	*	39		
4	4	A	10	G	16	M	22	S	28	Y	34	+	40		
5	5	B	11	H	17	N	23	T	29	Z	35	-	41		

Los caracteres se dividen en grupos de dos caracteres codificados en 11 bits. El valor del primer carácter se multiplica por 45 y el valor del segundo se le suma al de este producto. La suma es entonces convertida en un número de 11 bits. Si el número de caracteres de entrada no es múltiplo de dos, el último carácter se codifica en un número de 6 bits. Los datos binarios resultantes son concatenados y se les pone como prefijo el indicador de modo y el indicador contador de caracteres.

En el **modo Byte**, el valor de cada carácter es el mismo que el de su correspondiente codeword de 8 bits. Los valores resultantes se concatenan y se les pone como prefijo el indicador de modo y el indicador contador de caracteres.

3. Corrección de errores:

El código QR permite hacer corrección de errores; esto lo hace mediante el método Reed-Solomon [14]. Para ello se generan una serie de codewords de corrección de errores, los cuales se añaden a los datos. En el código QR existen cuatro niveles de corrección de errores, estos pueden ser elegidos por el usuario que requiere crear el símbolo. Los niveles se clasifican en:

- L (Low): capacidad de recuperación de 7 %
- M (Medium): capacidad de recuperación de 15 %
- Q (Quality): capacidad de recuperación de 25 %
- H (High): capacidad de recuperación de 30 %

Este sistema puede corregir dos tipos de codewords erróneos, los que provocan que un carácter no se pueda decodificar (borrón) y los que provocan que no se decodifique en otro carácter erróneo (error).

La formula matemática para la corrección de errores es: X + 2Y <= a - b

Donde:

x = número de borrones

y = número de errores

a = número de codewords de corrección de errores para borrones

b = número de codewords de corrección de errores para errores

4. Colocación de codewords en la matriz

La mayoría de codewords se representaran en el símbolo como un bloque de 2x4 módulos o 4x2 módulos. Otros tendrán formas irregulares por tenerse colocar adyacentes a algún patrón de función, como se muestra en la siguiente Figura 2.18:

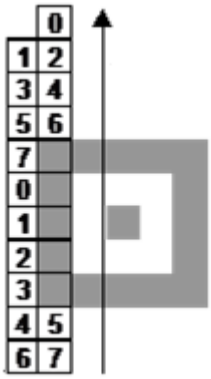


Figura 2.18: Módulo adyacente a patrón de función
tomada de: [13]

Se crea una matriz en blanco con el tamaño correspondiente a la versión del símbolo a crear. A este tamaño hay que añadir la región de silencio rodeando el símbolo que debe ser de 4 módulos de tamaño. Se añaden los tres patrones localizadores y los separadores, también los patrones temporizadores, que tendrán un tamaño diferente según la versión. Los módulos correspondientes, a la información de versión y de formato, se dejan en blanco de momento. Los patrones de alineamiento también se pueden colocar.

La región de codificación se rellena con la secuencia de codewords (los codewords normalmente tendrán la forma de 2x4 módulos), empezando por la parte inferior derecha del símbolo, subiendo primero hasta llegar al patrón localizador, luego se pasa a la columna adyacente izquierda y se baja hasta llegar al limite inferior del símbolo, para luego volver a subir. Así los codewords se colocan en zigzag. En la Figura 2.19 se muestra como sería la colocación de codewords en un símbolo versión 2.

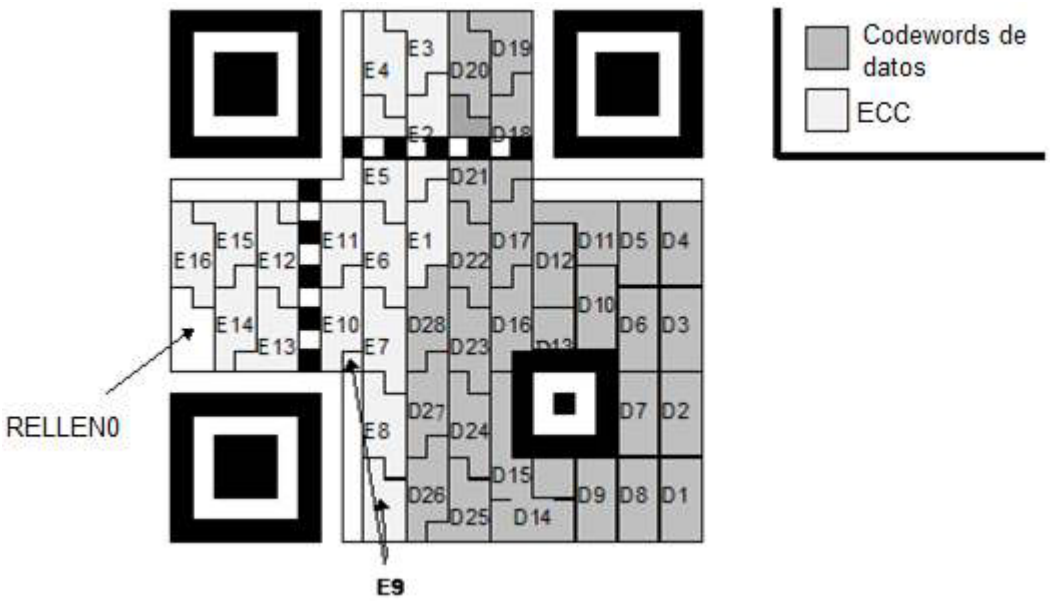


Figura 2.19: Colocación de codewords de la versión 2 de código QR
tomada de: [13]

El orden de los bits de cada codeword varía cuando el codeword se coloca hacia arriba o hacia abajo en el símbolo. Esto se muestra en la Figura 2.20.

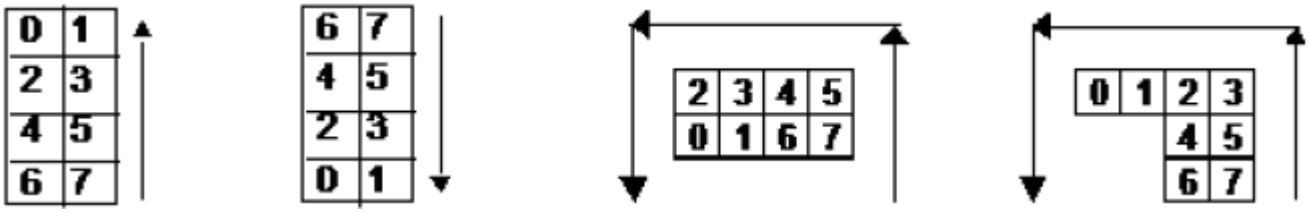


Figura 2.20: Modo de colocación de los codewords
tomada de: [13]

5. Enmascarar lo datos

Para mejorar la decodificación de los símbolos es preferible que el número de módulos blancos y negros del símbolo estén equilibrado [13]. El patrón 1011101 debe ser evitado ya que se puede confundir con el patrón localizador. Para conseguir que el número de módulos blancos y negros estén equilibrados después de crear la matriz del símbolo, debemos aplicar una máscara de datos a esta. La máscara no se aplica a los patrones de función ni a la información de formato y versión.

La máscara se aplica a los módulos mediante una operación XOR. Los patrones posibles y su código para la información de formato se pueden observar en la Tabla 2.4:

Tabla 2.4: Patrones y códigos de formato
tomada de: [13]

Código de patrón de máscara de Datos	Fórmula
000	$(i + j) \bmod 2 = 0$
001	$i \bmod 2 = 0$
010	$j \bmod 3 = 0$
011	$(i + j) \bmod 3 = 0$
100	$((i \text{ div } 2) + (j \text{ div } 3)) \bmod 2 = 0$
101	$(i\,j) \bmod 2 + (i\,j) \bmod 3 = 0$
110	$((i\,j) \bmod 2 + (i\,j) \bmod 3) \bmod 2 = 0$
111	$((i+j) \bmod 2 + (i\,j) \bmod 3) \bmod 2 = 0$

Para la versión 1 de código QR, los patrones serían como se muestran en la Figura 2.21:

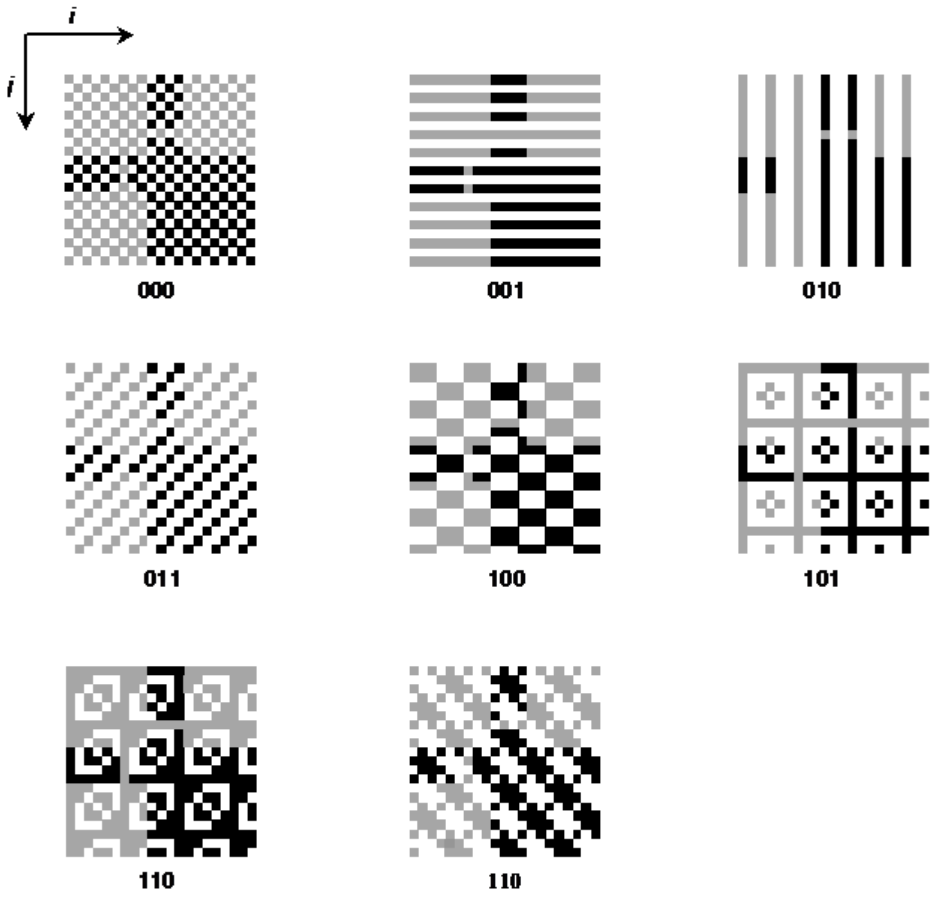


Figura 2.21: Patrones de enmascarado
tomada de: [13]

Siendo para las figuras 2.21 y 2.4: i el número de módulos en sentido horizontal y j el número de módulos en sentido vertical.

Los módulos grises pertenecen a patrones de función, o son de información de versión o formato, y no se les debe aplicar la máscara.

6. Información de formato

La información de formato consiste en una secuencia de 15 bits, de los cuales 5 contienen datos y los otros 10 son para reconstruir los 5 anteriores en caso de existir un error al decodificar. Los primeros dos bits de los cinco indican el nivel de corrección de error usado: L = 01, M = 00, Q = 11, H = 10. Los otros tres bits indican el patrón de la máscara de datos usada.

Una vez calculados los 15 bits se les debe aplicar mediante XOR la máscara 101010000010010, esto es para evitar que pudiera generarse una información de formato compuesta solo por ceros. La información de formato se coloca en la matriz del símbolo por duplicado en las zonas establecidas. En la Figura 2.22 se muestra como sería en un símbolo de versión 1.

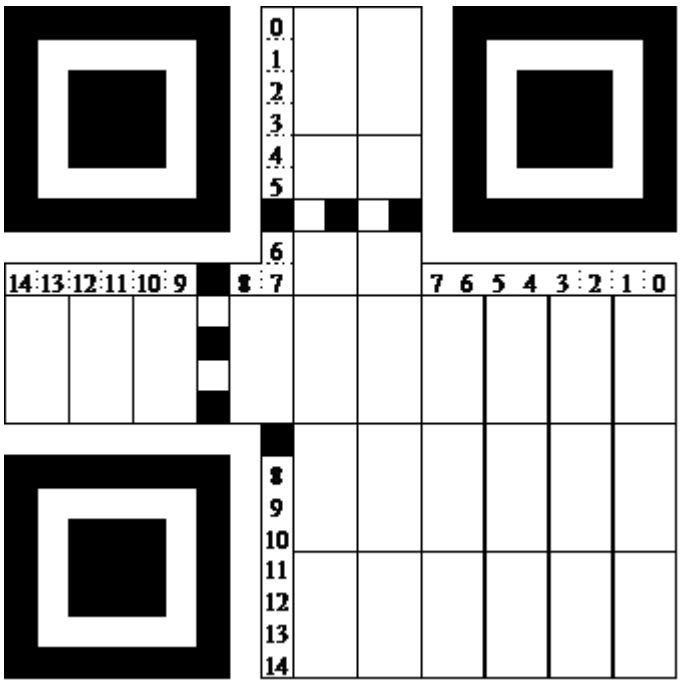


Figura 2.22: Colocación de formato en código QR versión 1
tomada de: [13]

7. Información de versión

Esta información solo se aplica a los símbolos de versión 7 o superior. Está formada por 18 bits, 6 de ellos son de datos y los otros 12 para corregir errores. Los 6 bits de datos contienen codificada la versión del símbolo. 000111 representaría la versión 7, 101000 representaría la versión 40. No es necesario enmascarar esta información como pasaba en el apartado anterior puesto que las versiones que provocarían una cadena de ceros no tienen información de versión.

La información de versión se coloca en la matriz del símbolo por duplicado en las zonas establecidas, vistas en la Figura 2.14. La forma de los bits se muestran en la Figura 2.23.

Figura 2.23: posicionamiento de la versión
tomada de: [13]

Proceso de decodificación

El proceso de decodificación es el inverso al de codificación [13]. A partir de un símbolo obtenemos unos datos en forma de caracteres. Este proceso se puede dividir en 8 pasos:

1. Obtener una imagen del símbolo y a partir de ella se crea una matriz de unos y ceros, reconociendo los módulos negros y blancos. Para eso se debe binarizar correctamente la imagen, en el proceso de binarización de la imagen es de vital importancia el cálculo del color del umbral que decidirá que píxeles serán negros o blancos.

2. Leer la información de formato. Obtener el nivel de corrección de errores y con ello detectar y corregir posibles errores. Obtener el tipo de patrón de mascara de datos usado.

3. Leer la información de versión si está, para determinar la versión del símbolo; en el caso de no estar presente la versión del símbolo, es decir en casos menores a la versión siete, se calcula midiendo el ancho del símbolo contando la cantidad de módulos.

4. Aplicar la máscara de datos a la matriz de unos y ceros en la región de codificación mediante una operación XOR. Para de esta manera deshacer la máscara aplicada en el proceso de codificación.

5. Obtener los codewords de datos y de error de la región de codificación teniendo en cuenta su orden de colocación.

6. Detectar los posibles errores en los codewords de datos, utilizando los codewords de error y corregirlos.

7. Dividir los codewords corregidos en segmentos según los indicadores de modo y contadores de caracteres encontrados.

8. Decodificar los caracteres de acuerdo a su modo de codificación y concatenar los resultados para crear la cadena decodificada resultante.

Para ubicar los patrones de localización, se debe encontrar el patrón 1011101. Como ahora se tiene píxeles y no bits se debe buscar una proporción de píxeles alternados en negro y blanco de $1:1:3:1:1$ como se puede detallar en la Figura 2.17, en el momento en que se tenga un prospecto a patrón localizador se procede a analizarlo.

Una vez encontrados los tres patrones localizadores, hay que determinar la orientación del símbolo. Para ello se analizan las posiciones de los centros de cada patrón y se detecta cuál de ellos es el patrón superior izquierdo. El hecho de haber detectado los patrones, da una idea de cuál es el ancho de módulo en píxeles de la imagen.

Para definir bien la región de codificación es necesario encontrar los patrones temporizadores. Estos patrones comienzan y terminan en un borde de los patrones localizadores, con una separación de un módulo blanco; este módulo blanco pertenece a un patrón separador.

En la Tabla 2.5 se puede detallar las capacidades para cada versión de las 40 del código QR.

Tabla 2.5: Capacidades de las versiones de QR
tomada de: [13]

Versión	Número de módulos por lado	Módulos de patrones de función	Módulos de información de formato y versión	Módulos de datos	Codewords de datos	Bits restantes
1	21	202	31	208	26	0
2	25	235	31	359	44	7
3	29	243	31	567	70	7
4	33	251	31	807	100	7
5	37	259	31	1079	134	7
6	41	267	31	1383	172	7
7	45	390	67	1568	196	0
8	49	398	67	1936	242	0
9	53	406	67	2336	292	0
10	57	414	67	2768	346	0
11	61	422	67	3232	404	0
12	65	430	67	3728	466	0
13	69	438	67	4256	532	0
14	73	611	67	4651	581	3
15	77	619	67	5243	655	3
16	81	627	67	5867	733	3
17	85	635	67	6523	815	3
18	89	643	67	7211	901	3
19	93	651	67	7931	991	3
20	97	659	67	8683	1085	3
21	101	882	67	9252	1156	4
22	105	890	67	10068	1258	4
23	109	898	67	10916	1364	4
24	113	906	67	11796	1474	4
25	117	914	67	12708	1588	4
26	121	922	67	13652	1706	4
27	125	930	67	14628	1828	4
28	129	1203	67	15371	1921	3
29	133	1211	67	16411	2051	3
30	1374	1219	67	17483	2185	3
31	141	1227	67	18587	2323	3
32	145	1235	67	19723	2465	3
33	149	1243	67	20891	2611	3
34	153	1251	67	22091	2761	3
35	157	1574	67	23008	2876	0
36	161	1582	67	24272	3034	0
37	165	1590	67	25568	3196	0
38	169	1598	67	26896	3362	0
39	173	1606	67	28256	3532	0
40	177	1614	67	29648	3706	0

Tabla comparativa de códigos de barra 2D

En la Tabla 2.6 se exponen las características de los diferentes códigos de barra 2D estudiados anteriormente, para luego así compararlos cada uno y luego escoger cual es el que mejor se adapta a las características requeridas.

Tabla 2.6: Comparación códigos 2D
Juárez y Villegas (2014)

Características	PDF417	DataMatrix	MaxiCode	QR
Codificación	ASCII	ASCII	ASCII	ASCII e información binaria
Tamaño Máximo Información	1850 texto 2170 Dígitos	1 a 2000 caracteres	93 caracteres	Hasta 7089 caracteres
Tamaño de Impresión	Depende de los niveles de seguridad	Depende de la cantidad de datos	1.1" x 1"	Depende de la versión y la cantidad de datos
Aplicaciones	Licencias, identificaciones, Aduana	Servicios postales	Mensajería y paquetería	Amplia diversidad de aplicaciones: publicidad, identificaciones, complementos, turismo, etc
Leído por teléfonos inteligentes	No	Si	No	Si

Uno de los criterios de comparación fue con la cantidad de caracteres soportados, dando como resultado a el código QR con la mayor cantidad de caracteres. En los últimos años QR ha ganado popularidad, lo que ha generado un gran uso del mismo, por tal motivo se han implementado una mayor cantidad de aplicaciones que lo soportan, a la vez que ha crecido su soporte técnico; todo esto inclina a una tendencia al uso de código QR. De los cuatros códigos de barras 2D estudiados, solo dos pueden ser leídos por teléfonos inteligentes, siendo uno de estos el código QR.

Habiendo analizado todas estas características, se obtiene como resultado el uso del código QR para su implementación sobre los demás códigos de barras 2D.

2.2. Sistema de Inventario

En esta sección se explica que es un sistema de inventario, su importancia dentro de una organización, así como su implementación utilizando la tecnología de códigos QR.

2.2.1. ¿Qué es un Sistema de Inventario?

Un inventario se llama a la existencia de cualquier artículo o recurso utilizado en una organización [15]. Un Sistema de Inventario es un conjunto de normas, métodos, procedimientos, políticas y controles utilizados para el monitoreo de la cantidad de artículos disponibles, la determinación de los niveles que se deben mantener, el momento de reponer la existencia de algún artículo y el tamaño que deben tener los pedidos. Este sistema puede ser manual o automatizado.

La historia de los inventarios, comienza desde la antigüedad, cuando los pueblos, debido a las épocas de escasez, deciden almacenar grandes cantidades de alimentos, para hacer frente a ellas; así se idea un mecanismo de control para su reparto. El inventario es una manera de mantener cierto control en la mercancía, insumos y materias primas que sean parte de la actividad económica de la organización, para así tener un desarrollo óptimo y generar un crecimiento de la misma [16].

2.2.2. Objetivos de los Sistemas de Inventario

Uno de los principales objetivos de los Sistemas de Inventario, es proveer o distribuir adecuadamente los materiales necesarios a la organización [16]. Colocándolos a disposición en el momento indicado, para así evitar aumentos de costos y perdidas de los mismos. Permitiendo satisfacer correctamente las necesidades reales de la organización, a las cuales debe permanecer constantemente adaptado. Por lo tanto la gestión de inventarios debe ser atentamente controlada y vigilada. Se reconocen los siguientes objetivos:

1. Administración de los materiales y/o productos para su control.
2. Determinar la inversión óptima de los inventarios de acuerdo a las posibilidades financieras de la organización.
3. Contar con suficientes activos fijos en existencia para hacer frente a la demanda de las actividades de la organización.
4. Hacer predicciones sobre las necesidades del inventario.

2.2.3. Administración de inventarios

Es la eficiencia en el manejo adecuado del registro, de la rotación y evaluación del inventario de acuerdo a como se clasifique y qué tipo de inventario tenga la organización, ya que a través de todo esto se determina los resultados de una manera razonable, pudiendo establecer la situación de la organización y las medidas necesarias para mejorar o mantener dicha situación [16]. La administración de inventario determina la cantidad de inventario que deberá mantenerse, la fecha en que deberán colocarse los pedidos y las cantidades de unidades que se deben ordenar. La administración de inventario, en general, se centra en tres aspectos básicos:

1. Cuantas unidades deberían ordenarse en un momento dado.
2. En qué momento deberían ordenarse el inventario.
3. Que artículos del inventario merecen una atención especial.

2.2.4. Sistema de procesamiento de transacciones de inventario

Un sistema de procesamiento de transacciones de inventario consiste de un conjunto de procedimientos para el registro de las transacciones de inventario [15]. En estos sistemas se proveen puntos de control, que son ubicaciones físicas de transferencia de artículos de inventario. En estos puntos de control se documenta y registra la transferencia del artículo de forma manual o automática. Parte de la información que debe documentarse es el número del artículo, la cantidad, la ubicación y el estado de los artículos. El registro de transacciones de forma automática se apoya en el uso de tecnologías informáticas. Generalmente los registros de transacciones se documentan en una base de datos conectada a la red interna de la organización. El registro físico es realizado utilizando sistemas de identificación, por ejemplo: códigos de barra, escáners y lectores de caracteres ópticos o microchips programados para responder a una determinada señal del espectro radioeléctrico.

2.2.5. Sistemas de inventarios y códigos QR

En los sistemas de inventarios se necesita un mecanismo para la identificación de todos los materiales y/o productos que en este se encuentra. Esto en consecuencia de poder saber en todo momento posible donde y en que estado se encuentran estos materiales, para así poder llevar con mayor eficacia y eficiencia el control de los mismos. Debido a las características de los códigos QR para poder almacenar información, a la versatilidad de estos para su portabilidad y al fácil acceso que se tiene para el uso de su tecnología, debido al auge de dispositivos capaces de procesar este tipo de código, son una de las mejores opciones para implementarlos como sistema de procesamiento de transacciones de los materiales dentro de un inventario.

Para poder implementar el código QR en un sistema de inventario automatizado se debe:

- Clasificar todos los materiales del inventario con un código único que los identifique.
- Registrar el código único junto con los datos que se quieren almacenar del material o producto en el inventario automatizado.
- Utilizar dicho código para generar el código QR con algún generador de los mismos.
- Imprimir dicho código y adherirlo al material o producto.

Luego de este proceso, si se quiere verificar la información del material, simplemente se escanea el código QR y el procesador de estos códigos desplegará toda la información almacenada por éste.

En términos generales este es el método mas usado para implementar los códigos QR como sistema de identificación de los productos o materiales dentro de un inventario.

2.2.6. Herramientas Existentes de Sistemas de Gestión de Inventario

En este capítulo se presentarán herramientas de sistemas automatizadas de inventarios y su funcionamiento; esto con el motivo de explorar las similitudes de estas herramientas existentes con el sistema de inventario que se pretende implementar.

Se presentará un resumen de cada herramienta estudiada, exponiendo su funcionamiento y operatividad, realizando una conclusión con la herramienta con más similitudes al sistema de gestión de inventario propuesto a desarrollar. A continuación se presentan las herramientas estudiadas.

StockBase POS

Es una herramienta gratuita y completa de gestión comercial, orientada a la administración de comercios o pequeñas y medianas empresas. Permite generar todo tipo de informes y estadísticas a partir de la información introducida, lo que permite visualizar la evolución financiera de la empresa, de manera mensual, trimestral o anual [17].

Esta herramienta no tiene un soporte de conexión online. Dicha característica hace que sea segura contra ataques informáticos, especialmente en sistema que no tienen conexión a Internet. Posee una base de datos donde se puede llevar el control de artículos, clientes, proveedores y vendedores. StockBase puede ser instalado en modo monousuario o en modo multiusuario.

En una sola ventana incluye todo lo necesario para administrar eficazmente una empresa o negocio:

- Permite ejercer un control exhaustivo sobre las existencias disponibles del inventario.
- Permite administrar listas de precios y costos.
- Permite registrar y administrar muchos tipos de datos sobre clientes, proveedores, cuentas bancarias, etc.
- Permitir realizar compras, ventas, emisión e impresión de tickets, facturas y recibos.

Se muestra en la Figura 2.24 la pantalla de inicio de StockBase POS.

Figura 2.24: Pantalla de StockBase POS
tomada de captura de pantalla

Inventoria Stock Manager

Es un programa de inventario gratuito, control de reservas, adquisiciones y presentación de informes [18]. Inventoria Stock Manager es un software de gestión de reservas profesional y fácil de usar, dando a las empresas de todos los tamaños la capacidad de controlar eficientemente todos sus inventarios, desde una pequeña tienda a una cadena de tiendas y almacenes en una

o más ubicaciones. Este software de gestión del inventario permite administrar fácilmente las cantidades de existencias, transferir entre almacenes, establecer alertas de bajo nivel, generar y enviar pedidos de compra. Inventoria está diseñado para ser tan intuitivo de usar como sea posible, por lo que tras una instalación rápida, se podrá optimizar los procesos de inventario dentro de minutos.

Las principales características que posee este programa son:

- Supervisa e informa sobre los costos, niveles de existencias y promedios.
- Escanea en códigos de barras para agregar nuevos elementos.
- Importa el inventario actual con un archivo CSV.
- Añade notas, URLs y fotos a los elementos.
- Establece unidades de venta de artículos que se venden por peso o paquetes.
- Agrupa elementos comunes en categorías para el fácil control de reservas.
- Configura tasas de impuestos, GST e IVA para aplicar a los productos.
- Establece avisos de existencias para saber cuando se debe reordenar.
- Ve rápidamente los niveles de inventario generales, o por ubicación o categoría.
- Ve el historial del producto cuando fueron recibidos o vendidos elementos.
- Acciones de transferencia entre almacenes.
- Mantiene una base de datos de los clientes y proveedores.
- Crea pedidos de compra y los envía por correo directamente a los proveedores.
- Interfaz simple y fácil de usar.

En la Figura 2.25 se muestra un ejemplo de la interfaz gráfica de Inventoria Stock Manager.

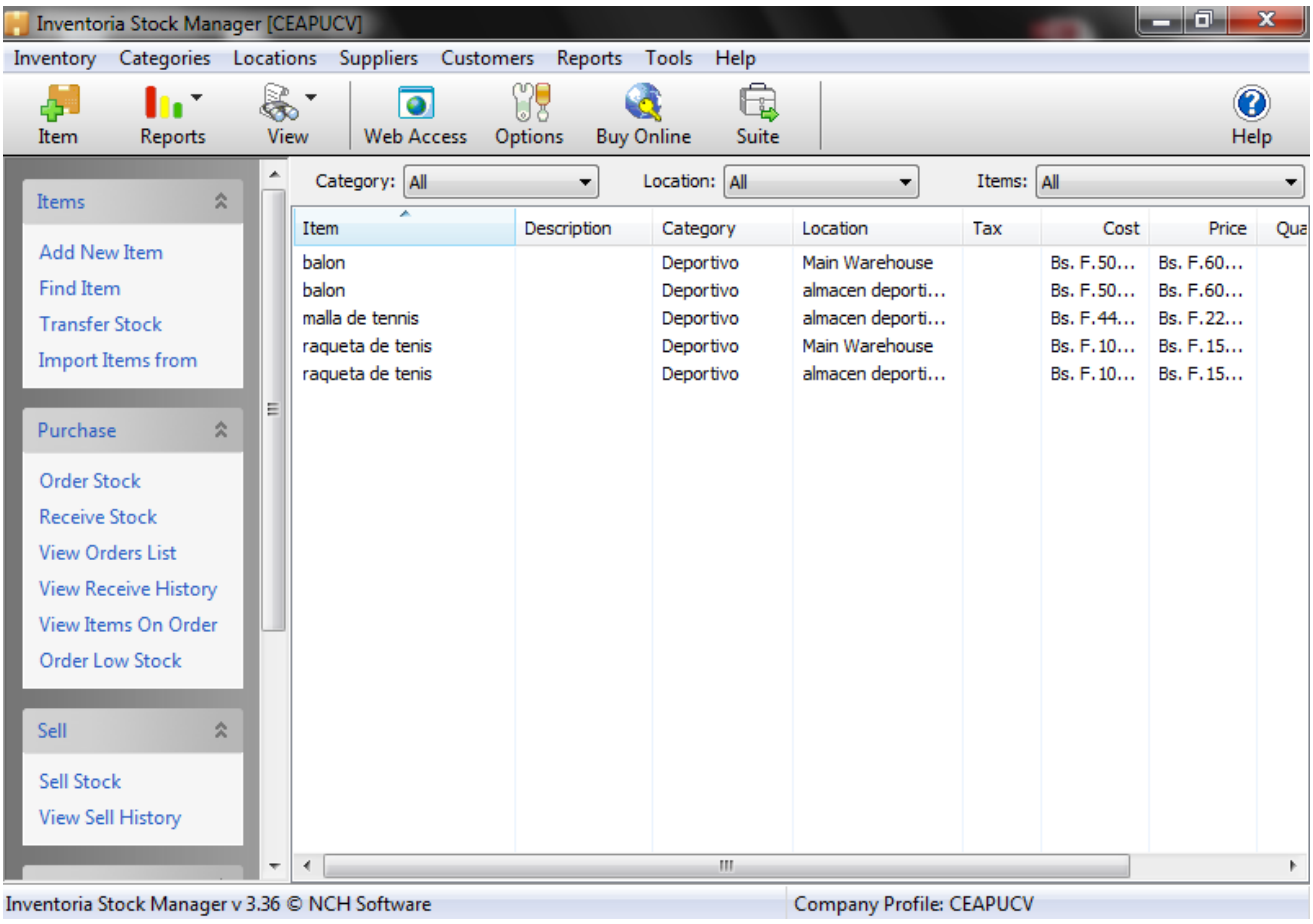


Figura 2.25: Interfaz gráfica de Inventoria Stock Manager
tomada de captura de pantalla

Birtud

Birtud es un sistema de inventario en línea gratuito para pequeños negocios o proyectos que permite manejar el inventario de una manera segura y fácil [19]. En este sistema de inventario se puede controlar la existencia de cualquier tipo de producto relevante para cualquier negocio o proyecto (repuestos, insumos, productos en venta, artículos de escritorio, etc.). Para agregar algún producto basta con definir un nombre distintivo que permita identificarlo fácilmente, la unidad de medida en la que se manejará en el inventario, y se habilita un campo opcional donde se puede agregar todos los atributos, códigos y/o comentarios adicionales que se desee.

Una de las característica más importante es que permite compartir el inventario con socios. La plataforma está diseñada para soportar múltiples lenguajes siendo estos inglés y español; así como multiusuario: varios usuarios de una organización pueden estar conectados con su propia cuenta y permisos de acceso. Posee integración con Excel para importar y exportar compilaciones de inventario para su uso fuera de línea. Además al ser una aplicación Web, está disponible en todo momento y ha sido diseñada para ser utilizada en todo dispositivos con un navegador

moderno. Birtud fue creado en su esencia para pequeños negocios y organizaciones, pero técnicamente, este sistema puede ser utilizado por cualquiera, incluso para llevar el inventario de artículos de la casa.

Modo de uso de Birtud Primeramente se accede a la pagina web de Birtud: http://www.birtud.com/, a continuación se registra como usuario, y luego de este proceso se estará en la pagina principal (tablero). En este tablero mostrado en la Figura 2.26, se observa varias opciones: Productos, Almacenes, Socios, Ayuda y Opciones.

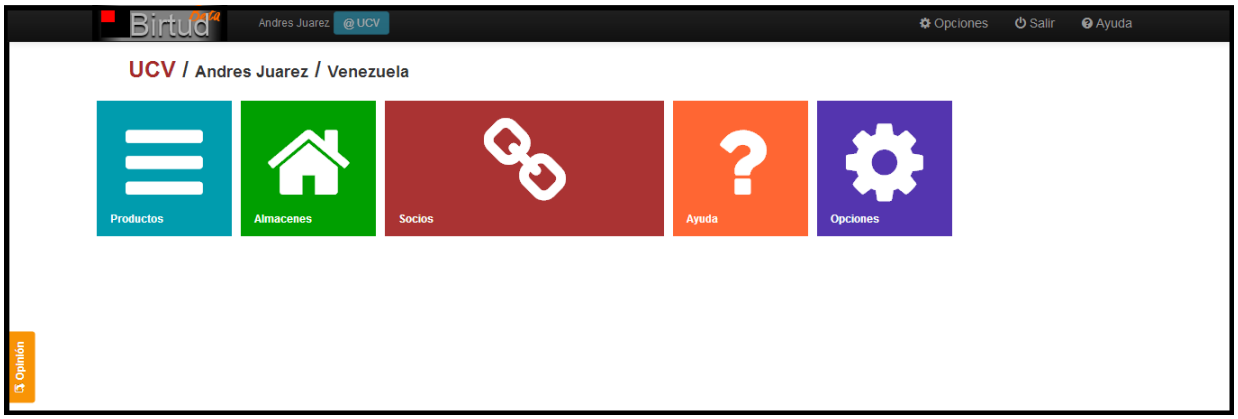


Figura 2.26: Menú Birtud
tomada de: https://app.birtud.com/App#ve/ucv/computación

- En el enlace Productos se pueden crear todos los productos que se manejarán en el inventario y listar todos los productos existentes. Un ejemplo de creación de producto se muestra en la Figura 2.27 y una lista de producto ya creados se muestra en la Figura 2.28.

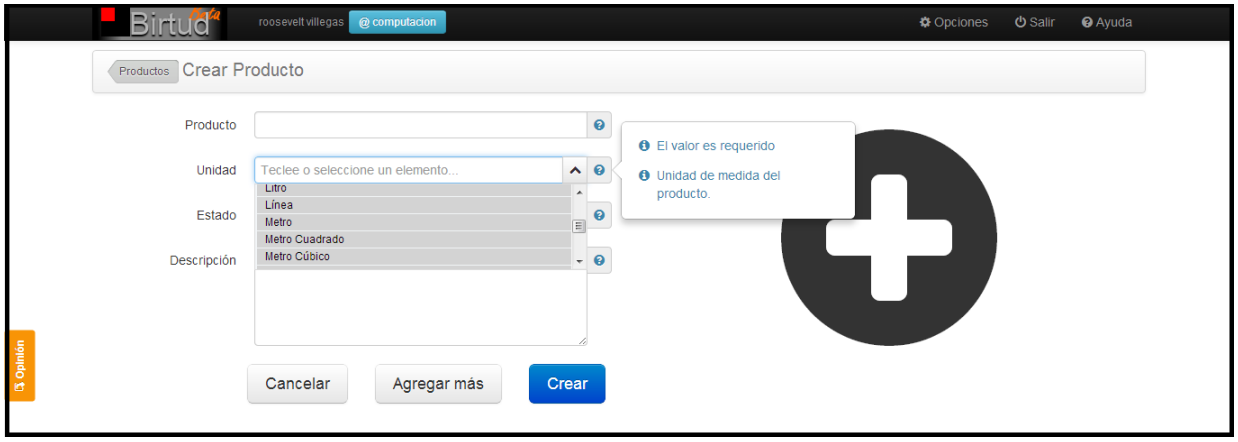


Figura 2.27: Crear producto.
tomada de: https://app.birtud.com/App#ve/ucv/computacion/products/add

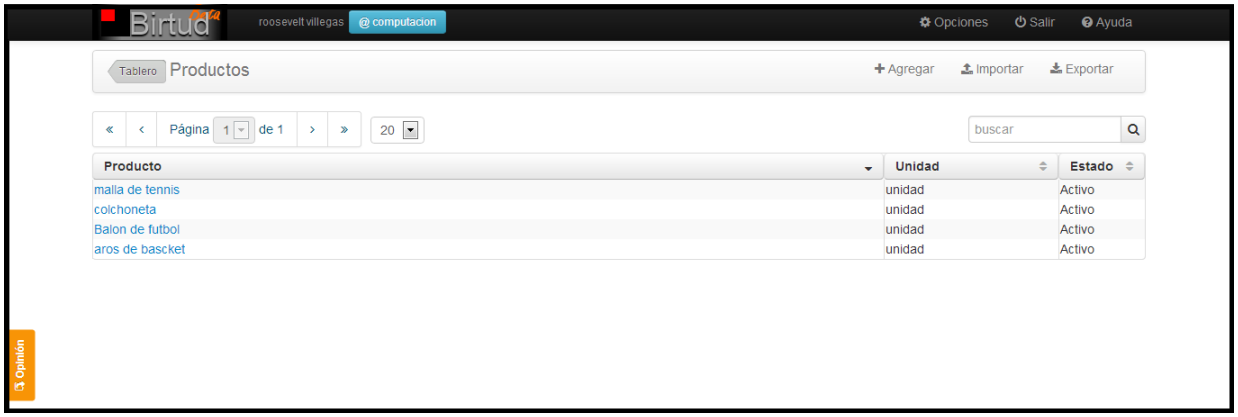


Figura 2.28: Lista de productos.
tomada de: https://app.birtud.com/App#ve/ucv/computación/products

- En el enlace Almacenes, se pueden crear y listar los almacenes del negocio o proyecto. Esto se puede ver en la Figura 2.29.

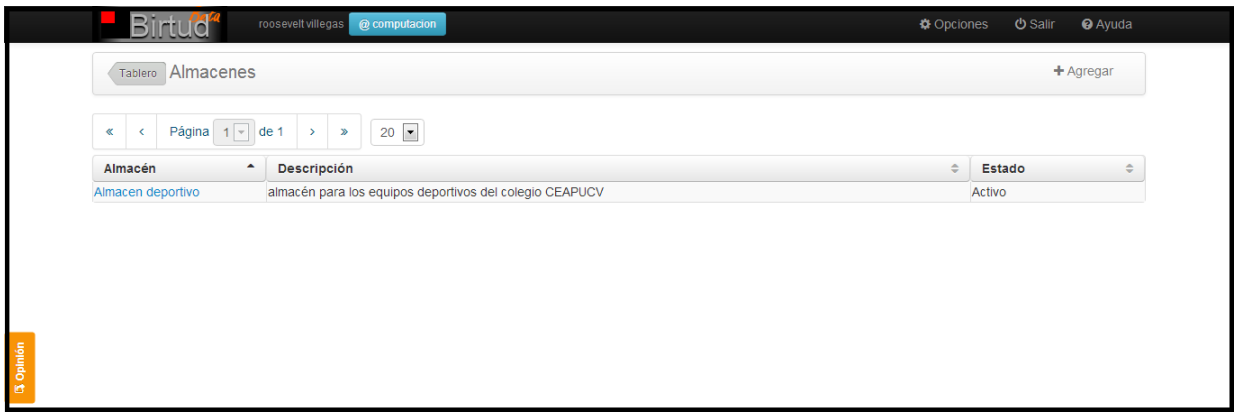


Figura 2.29: Almacenes.
tomada de: https://app.birtud.com/App#ve/ucv/computación/warehouses

- El enlace Socios te permite invitar y/o hacer formar parte del proyecto a los usuarios seleccionados.
- El enlace Ayuda despliega un documento, el cual es el manual del sistema.
- El enlace Opciones permite realizar ciertos cambios, como cambiar la contraseña de la cuenta, administrar los usuarios asociados al proyecto o registrar una nueva área de trabajo.

Las dos primeras herramientas estudiadas para la gestión de inventario, aunque cumplían las necesidades básicas de un sistema de inventario, estas estaban más orientadas hacia un ambiente

comercial o de negocios, por lo que sus principales características incluyen métodos de facturación, métodos de pago, notas de crédito, etc. Dichas características se desvían del objetivo principal del sistema de gestión de inventario que se quiere implementar. Sin embargo, la herramienta de gestión de inventario Birtud es la que mejor se ajusta a las necesidades de la solución a implementar, ya que además de ser una aplicación online, esta no se centra en funciones de negocios, por lo que se mantiene lo más simple y general posible, teniendo funcionalidades necesarias como el control de roles. Todas estas características dan como resultado a Birtud como la herramientas que más se acerca o tiene similitudes con el sistema de gestión de inventario que se propone implementar en este proyecto. Por tal motivo, se toma como referencia esta herramienta, su interfaz gráfica, su control de roles y su usabilidad.

2.3. Ruby on Rails

Ruby on Rails, conocido también como RoR o simplemente Rails, es un *framework* para aplicaciones web de código abierto usando el lenguaje de programación Ruby, el cual está orientado a objetos [20]. Este *framework* sigue el paradigma de programación Modelo, Vista y Controlador (MVC). Fue creado con el propósito de hacerlo lo más simple posible, con la posibilidad de desarrollar aplicaciones escribiendo menos código que con el uso de otros *frameworks*, además de hacerlo con un mínimo de configuraciones. El lenguaje en el cual está basado Rails, Ruby, permite la metaprogramación, es decir el uso de programas que escriben o manipulan otros programas (o a sí mismos) como datos, de lo cual Rails hace uso, en consecuencia la sintaxis de este es muy legible y fácil de usar. El formato oficial de paquetes o librerías oficiales para Rails son las gemas.

2.3.1. Filosofía

Entre las filosofías de RoR se tiene DRY (*Don't Repeat Yourself* , no te repitas) y las convenciones sobre configuraciones. DRY significa que no se deba repetir alguna definición que ya se haya hecho en otro lugar. En Rails los componentes están integrados, de manera que no hay necesidad de establecer puentes entre estos. Un ejemplo de esto se puede ver con *ActiveRecord*, que es el ORM (*Object-Relational Mapping*, mapeo objeto-relacional) de RoR, las definiciones de las clases no necesitan especificar los nombres de las columnas; Ruby puede averiguarlos a partir de la propia base de datos, de forma que definirlos tanto en el código como en el programa sería redundante.

Las convenciones sobre configuraciones se trata de que Rails en vez de configuraciones, define convenciones y anotaciones. Esto quiere decir que el programador sólo necesita definir aquella configuración que no es convencional. Un ejemplo puede ser: si hay una clase Historia en el

modelo, la tabla correspondiente de la base de datos es historias, pero si la tabla no sigue la convención (por ejemplo blogposts) debe ser especificada manualmente.

2.3.2. Arquitectura MVC de Ruby on Rails

La rapidez en el desarrollo de proyectos con Rails está fundamentada en la idea de construir la aplicación separando de forma clara las capas de Modelo (Datos), Vista (Presentación) y Controlador (Funciones y métodos) para reducir el acoplamiento entre la lógica del negocio y la de presentación. En la Figura 2.30 se muestra una arquitectura Modelo-Vista-Controlador (MCV).

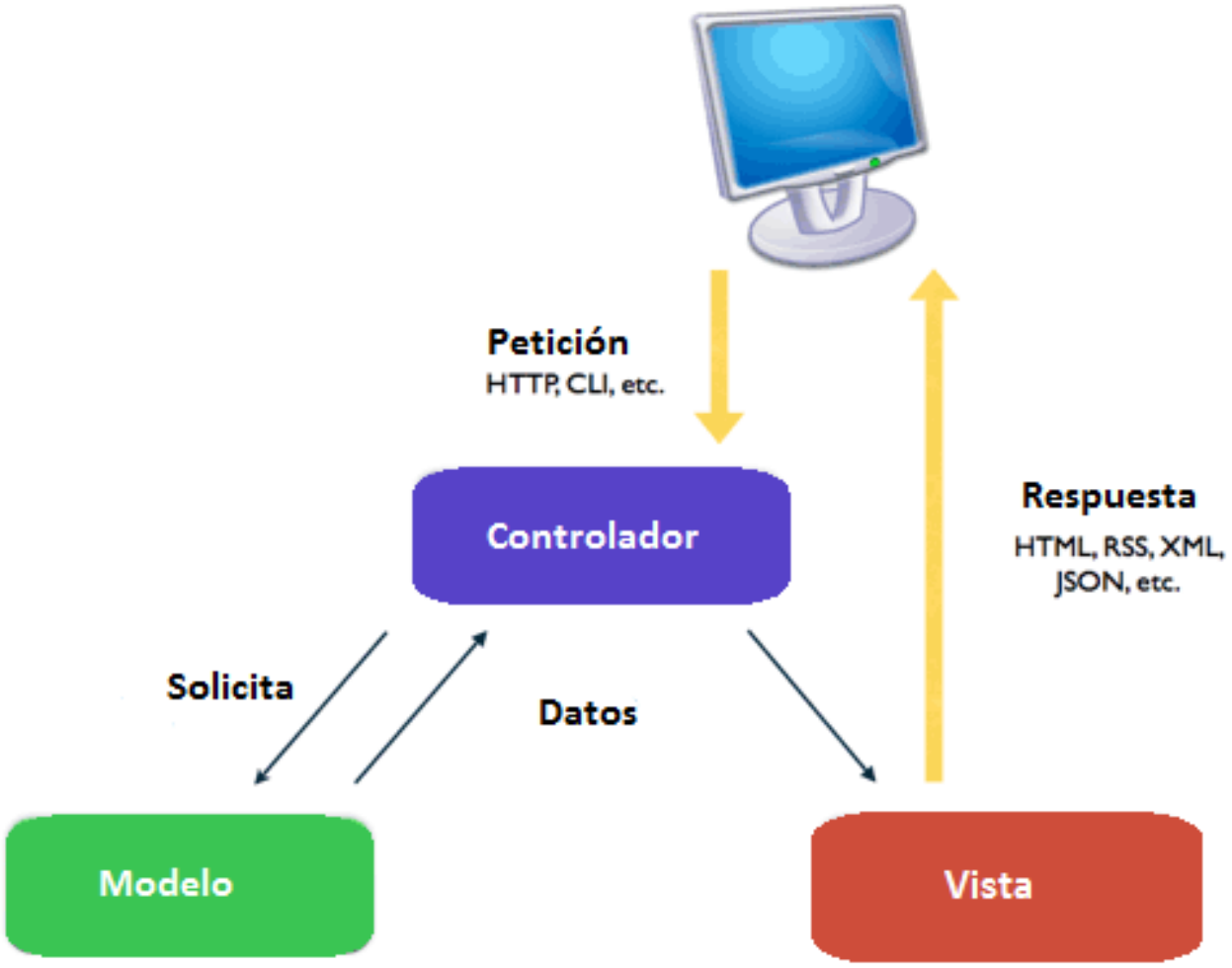


Figura 2.30: Arquitectura MVC
Juárez y Villegas (2014)

Modelo: en las aplicaciones web orientadas a objetos sobre bases de datos, el Modelo consiste en las clases que representan a las tablas de la base de datos. En RoR, las clases del Modelo son gestionadas por *ActiveRecord*. Por lo general, lo único que tiene que hacer el programador, es heredar de la clase *ActiveRecord::Base*, y el programa averiguará automáticamente qué tabla usar y qué columnas tiene.

El modelo representa:

- Las tablas de la base de datos.
- Migraciones (expresan cambios en la base de datos).
- Reglas para manipular los datos.

Vista: en el paradigma MVC, es la lógica de visualización, o cómo se despliegan los datos de las clases del Controlador. Con frecuencia en las aplicaciones web la vista consiste en una cantidad mínima de código incluido en HTML. El método que emplea Rails usar Ruby Empotrado (archivos.rhtml, desde la versión 2.x en adelante de RoR archivos.html.erb), que son básicamente fragmentos de código HTML con algo de código en Ruby. También pueden construirse vistas en HTML y XML con Builder o usando el sistema de plantillas. Es necesario escribir un pequeño fragmento de código en HTML para cada método del controlador que necesita mostrar información al usuario. El "maquetado" o distribución de los elementos de la página se describe separadamente de la acción del controlador y los fragmentos pueden invocarse unos a otros.

Controlador: en MVC, las clases del Controlador responden a la interacción del usuario e invocan a la lógica de la aplicación, que a su vez manipula los datos de las clases del Modelo y muestra los resultados usando las Vistas. En las aplicaciones web basadas en MVC, los métodos del controlador son invocados por el usuario usando el navegador web.

La implementación del Controlador es manejada por el *ActionPack* de Rails, que contiene la clase *ApplicationController*. Una aplicación Rails simplemente hereda de esta clase y define las acciones necesarias como métodos.

2.3.3. Gemas

Las gemas son plugins o códigos añadidos a los proyectos RoR, que permiten nuevas funcionalidades como nuevos *create* o nuevas funciones predefinidas (como inicio de sesión de usuarios).

Para la realización de este proyecto se hizo uso de las gemas: 'rails', 'sqlite3', 'sass-rails', 'uglifier', 'coffee-rails', 'jquery-rails', 'turbolinks', 'jbuilder', 'sdoc', 'mysql2', 'sorcery', 'rqrcode', 'therubyracer', 'cancan', 'validates_timeliness', 'execjs', 'wicked_pdf' y 'wkhtmltopdf-binary'.

2.4. Estado de Transferencia Representacional (REST)

REST significa *Representational State Transfer* y es una arquitectura de software hecha para diseñar y para construir aplicaciones Web, que se basa en el concepto de la representación de recursos[21]. Un recurso puede ser cualquier concepto coherente al que se haga referencia como una URL. La representación de un recurso es un documento que capta el estado actual del mismo. Un ejemplo de esto es una página HTML, o HAML. ReST también utiliza todas las operaciones básicas o métodos del protocolo HTTP: GET, POST, DELETE, UPDATE. Esto evita tener que utilizar protocolos complejos como RPC (*Remote Procedure Call* o llamada a procedimiento remoto) o SOAP (*Simple Object Access Protocol* o protocolo simple de acceso a objetos).

ReST está basado además en protocolos de comunicación que pueden utilizar cache, que poseen una arquitectura cliente-servidor y que no tengan estado. Este tipo de arquitectura utiliza peticiones HTTP para crear, actualizar, leer y para eliminar datos. Por lo tanto las aplicaciones ReST utilizan el protocolo HTTP para las cuatro operaciones CRUD (*Create/Read/Update/Delete*). Las aplicaciones ReST son auto-contenidas, por lo que cada petición lleva consigo toda la información que requiera el servidor para completar sus operaciones.

En cuanto a programación, las aplicaciones ReST son alternativas ligeras a protocolos pesados y complejos como RPC o a servicios Web como SOAP o WSDL.

Entre las propiedades de los servicios Web se encuentran:

- Son independientes a la plataforma.
- Independencia del lenguaje de programación utilizado.
- Está basado en estándares, en este caso HTTP o incluso HTTPS para encriptación.
- Pueden utilizarse en la presencia de Firewalls.
- Para cuestiones de seguridad como nombre de usuario y claves utiliza tokens.
- Los servicios ReST utilizan XML en sus respuestas como una manera de organizar datos, mientras que en las peticiones no se utiliza casi, esto debido a la simplicidad de dichas peticiones. En las peticiones simplemente se utiliza el método HTTP GET o el método HTTP POST y sus parámetros en el URL sin comprometer la seguridad. Pero las aplicaciones ReST no tienen que utilizar obligatoriamente XML, al contrario de SOAP. También puede utilizar JSON o CSV.

2.5. Twitter Bootstrap

Bootstrap es un *framework* que simplifica el proceso de creación de diseños web combinando CSS y JavaScript [22]. Ha sido desarrollado por Twitter.

Bootstrap tiene un soporte completo para HTML5 y CSS3, y es compatible con la mayoría de los navegadores web. La información básica de compatibilidad de sitios web o aplicaciones esta disponible para todos los dispositivos y navegadores. Existe un concepto de compatibilidad parcial que hace disponible la información básica de un sitio web para todos los dispositivos y navegadores. Por ejemplo, las propiedades introducidas en CSS3 para las esquinas redondeadas, gradientes y sombras son usadas por *Bootstrap*. Esto extiende la funcionalidad de la herramienta. Desde la versión 2.0 también soporta diseños sensibles. Esto significa que el diseño gráfico de la página se ajusta dinámicamente, tomando en cuenta las características del dispositivo usado (Computadoras, tabletas, teléfonos móviles). Bootstrap es de código abierto y está disponible en GitHub.

Estructura y función de Twitter Bootstrap

Bootstrap es modular y consiste esencialmente en una serie de hojas de estilo LESS (*stylesheet language* o lenguaje de hojas de estilo) que implementan la variedad de componentes de la herramienta. Una hoja de estilo llamada bootstrap.less incluye los componentes de las hojas de estilo. Los desarrolladores pueden adaptar el mismo archivo de Bootstrap, seleccionando los componentes que deseen usar en su proyecto.

2.6. MySQL

MySQL, que significa *My Structured Query Language* o Lenguaje de Consulta Estructurado, es un sistema de gestión de base de datos relacional, que soporta multihilo y multiusuario. Está licenciado bajo GNU *General Public License* o Licencia publica general (GPL GNU), licencia usada en el software libre. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente [23].

Las principales características de MySQL son[24]:

- Interioridades y portabilidad.
  - Funciona en diferentes plataformas.
  - APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl.
  - Proporciona sistemas de almacenamiento transaccionales y no transaccionales.

- Usa tablas en disco B-tree (*MyISAM*) muy rápidas con compresión de índice.
- El servidor está disponible como un programa separado para usar en un entorno de red cliente/servidor. Dicho programa puede usarse en un entornos donde no hay red disponible.

- Escalabilidad y límites.
  - Soporte a grandes bases de datos. Se usa MySQL Server con bases de datos que contienen 50 millones de registros. También se puede usar MySQL Server con 60.000 tablas y cerca de 5.000.000.000.000 de registros.
  - Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2). Un índice puede usar prefijos de una columna para los tipos de columna CHAR, VARCHAR, BLOB, o TEXT.

- Conectividad
  - Los clientes pueden conectar con el servidor MySQL usando sockets TCP/IP en cualquier plataforma.

- Clientes y herramientas
  - MySQL server tiene soporte para comandos SQL para chequear, optimizar, y reparar tablas.

2.7. Metodología

La Metodología de la Investigación se considera y se define como la disciplina que elabora, sistematiza y evalúa el conjunto del aparato técnico procedimental del que dispone la Ciencia, para la búsqueda de datos y la construcción del conocimiento científico [25]. La Metodología consiste entonces en un conjunto más o menos coherente y racional de técnicas y procedimientos cuyo propósito fundamental apunta a implementar procesos de recolección, clasificación y validación de datos y experiencias provenientes de la realidad, y a partir de los cuales pueda construirse el conocimiento científico.

La metodología sirve a las ciencias como el soporte conceptual y procedimental suficiente para asegurar la aplicabilidad, pertinencia y validez de los procesos de investigación científicos de manera de cumplir con las exigencias y protocolos que cada disciplina exige para considerar a cada producción como un hallazgo de carácter científico.

2.7.1. Metodología de desarrollo de software

A principios de la década de los 90's, surgió un enfoque que fue bastante revolucionario para su momento ya que iba en contra de toda creencia de que mediante procesos altamente definidos se iba a lograr obtener software en tiempo, costo y con la requerida calidad. Se dio a conocer en la comunidad de Ingeniería de Software con el nombre de RAD (*Rapid Application Development* o Desarrollo Rápido de Aplicaciones). RAD consistía en un entorno de desarrollo altamente productivo, en el que participaban grupos pequeños de programadores utilizando herramientas que generaban código en forma automática tomando como entradas sintaxis de alto nivel. En general, se considera que este fue uno de los primeros hitos en pro de la agilidad en los procesos de desarrollo [26].

La historia de las Metodologías Ágiles y su apreciación como tales en la comunidad de la Ingeniería de Software, tiene sus inicios en la creación de uno de los métodos más utilizados como arquetipo: XP (*eXtreme Programming* o Programación extrema), que nace de la mente de Kent Beck en el año 1990.

2.7.2. XP (eXtreme Programming o Programación extrema)

XP es una método ágil centrado en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software [26], promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. En XP se considera que los cambios en los requisitos o requerimientos durante el ciclo de desarrollo son naturales. El modelo de trabajo de XP hace énfasis en el análisis, el cual solo se hace una vez, y luego el diseño, implementación y pruebas se van iterando, como se muestra en la Figura 2.31.

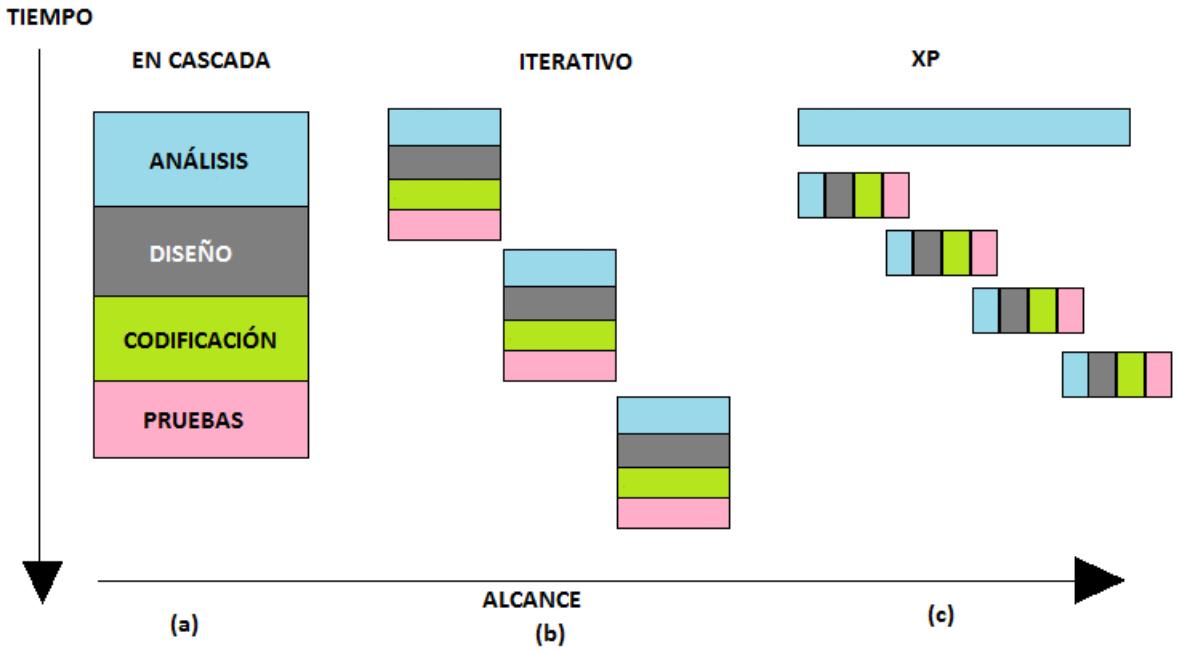


Figura 2.31: Modelo trabajo XP
Juárez y Villegas (2014)

Prácticas XP

La principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione [26]. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación ideal de las siguientes prácticas:

- **El rol de la planificación:** hay una comunicación frecuente del cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.
- **Diseño simple:** se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto.
- **Pruebas:** la producción de código está dirigida por las pruebas unitarias.
- **Refactorización:** es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. Se mejora la estructura interna del código sin alterar su comportamiento externo.

- **Programación en parejas:** toda la producción de código debe realizarse con trabajo en parejas de programadores. Esto conlleva ventajas implícitas; menor tasa de errores, mejor diseño, mayor satisfacción de los programadores, etc.
- **Propiedad colectiva del código:** cualquier programador puede cambiar cualquier parte del código en cualquier momento.
- **Estándares de programación:** XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible.

Ciclo de vida

El ciclo de vida ideal para XP consiste en cinco fases [26]:

1. **Exploración:** los clientes plantean los requerimientos funcionales del sistema a desarrollar. Los desarrolladores basado en los requerimientos funcionales dados, se plantea los requerimientos no funcionales del sistema.
2. **Planificación:** el equipo de desarrollo otorga, estima prioridades y cantidad de esfuerzo que se debe aplicar para cada requerimiento de usuario.
3. **Iteraciones:** el desarrollo del proyecto debe estar compuesto por iteraciones. Cada iteración puede estar compuesta por varias de las siguientes fases:

   a) Planificación: captación de requerimientos (historias de usuarios).

   b) Diseño: se hace un glosario de términos para que el equipo de trabajo pueda evitar confusiones y se deben tomar en cuenta las funcionalidades extra por cada ciclo.

   c) Codificación: código simple, legible y modular.

   d) Pruebas: pruebas funcionales y/o de aceptación.

4. **Producción:** pruebas adicionales y revisiones de rendimiento.
5. **Muerte del proyecto:** cuando ya no existan más requerimientos de usuario. Se genera la documentación final y no se realizan más cambios en la arquitectura del sistema.

Las historias de usuario

Son las técnicas utilizadas para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer,

sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible.

Las historias de usuario son descompuestas en tareas de programación y asignadas a los programadores para ser implementadas durante una iteración.

Roles XP

Los roles de acuerdo con la propuesta original de Beck son[26]:

Programador: el programador escribe las pruebas unitarias y produce el código del sistema.

Cliente: escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.

Encargado de pruebas: ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

Encargado de seguimiento: proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.

Entrenador: es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

Consultor: es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.

Gestor: es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

Proceso XP

El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos [26]:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El programador construye ese valor de negocio.
4. Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos.

Capítulo 3

Marco Aplicativo

Por las características del método de desarrollo de software XP, es el que mejor se acopla al desarrollo que se desea seguir para realizar el proyecto. Sin embargo no todas las características se ajustan a la perfección, por tal motivo deben hacerse ciertos ajustes de este método a utilizar.

Se ajusta el concepto de la característica de "pequeñas entregas" a uno de "medianas entregas", el cual consiste en entregas de versiones lo más completas posibles, cercanas al producto final.

Este método describe entre sus características las historias de usuario, que sencillamente representan de alguna manera un poco informal, los requisitos funcionales o no funcionales del software a implementar, los cuales de cierto modo reemplazan los documentos funcionales formales que se deben realizar para el diseño y desarrollo de un software. Por esta razón, en este proyecto se decidió complementar las historias de usuarios con documentos formales, para así tener un mejor enfoque al momento de diseñar y desarrollar el sistema de inventario a realizar. Entre estos documentos formales que se decidió agregar están: el mapa relacional de la base de datos, los casos de uso y la arquitectura del sistema.

Todos los ajustes anteriormente mencionados, son los necesarios para que el método de desarrollo de software XP se adapte a la implementación de este proyecto.

En la implementación del código QR de los equipos o insumos, el número de la versión depende del nivel de corrección del mismo. En este caso, se decidió utilizar el nivel de corrección M, el cual mantiene un equilibrio entre el número de caracteres que puede almacenar y la tasa de corrección de errores.

A continuación se presenta el desarrollo de este TEG: exploración, planificación, iteraciones y las pruebas.

3.1. Exploración

En esta fase del ciclo de vida de XP de este proyecto, se plantean los requerimientos del cliente (los usuarios finales), es decir, los requerimientos funcionales del sistema. También se plantean los requerimientos que el equipo de desarrollo quiere que cumpla el sistema, para tener un software de calidad y funcional. Estos requerimientos son los llamados requerimientos no funcionales. En esta fase también se expone la arquitectura que soportará al sistema, prototipos de la diagramación hechos para la interfaz gráfica de la aplicación y el mapa relacional de la base de datos.

3.1.1. Requerimientos funcionales

- El sistema debe permitir crear y editar a los usuarios.
- El sistema debe permitir iniciar sesión, usando los datos cédula y contraseña del usuario.
- El sistema debe restringir el acceso de los usuarios a los módulos a través de permisos asignados.
- El sistema debe permitir el ingreso de equipos e insumos nuevos o existentes en el inventario.
- El sistema debe permitir el egreso de equipos e insumos existentes del inventario.
- El sistema debe permitir generar los códigos QR de los equipos e insumos.
- El sistema debe permitir hacer consulta al inventario.
- El sistema debe permitir generar reportes de las consultas hechas al inventario.
- El sistema debe permitir la creación y la edición de las unidades de los medicamentos.
- El sistema debe permitir la creación y la edición de las presentaciones de los medicamentos.

3.1.2. Requerimientos no funcionales

- Usabilidad: el sistema debe ofrecerle al usuario la capacidad de llevar a cabo las funciones de manera intuitiva y fácil. El mismo debe poseer la capacidad de orientar al usuario si ocurre cualquier error que este haya cometido, de manera que pueda reaccionar correctamente ante la acción errónea.

- Confiabilidad: los errores generados por la interacción entre el usuario y el sistema, serán validados tanto del lado del cliente como del lado del servidor. Se mantendrá la correctitud, consistencia e integridad de los datos debido a las funciones que ofrece el sistema manejador de base de datos; de manera que estos procesos de validación de errores permanezcan transparente para el usuario.

- Seguridad: el sistema contará con validación de cédula y contraseña para poder acceder al mismo. Cada usuario tendrá permisos asignados para poder acceder a los distintos módulos que conforman al sistema.

- Portabilidad: el sistema al ser una aplicación web, deber ser soportado por los diferente navegadores, independientemente del ambiente en el cual se ejecuten.

3.1.3. Arquitectura de la aplicación

La arquitectura que se utilizará para formar parte de la solución, se puede ver en la Figura 3.1, dicha arquitectura constará con un servidor, donde estarán todos los archivos, funcionalidades, configuraciones y al cual todas las demás maquinas conectadas a la red local podrán acceder. Se podrá acceder al servidor a través de un dominio en la URL de algún explorador web, dicho dominio estará almacenado en un servidor DDNS (*Dynamic Domain Name System*, sistema de nombres de dominios dinámico) en Internet, como también la dirección IP pública del router. Se tiene que cambiar la configuración del router, para que cada petición HTTP que intente acceder al mismo, sea redireccionado a la dirección IP local del servidor.

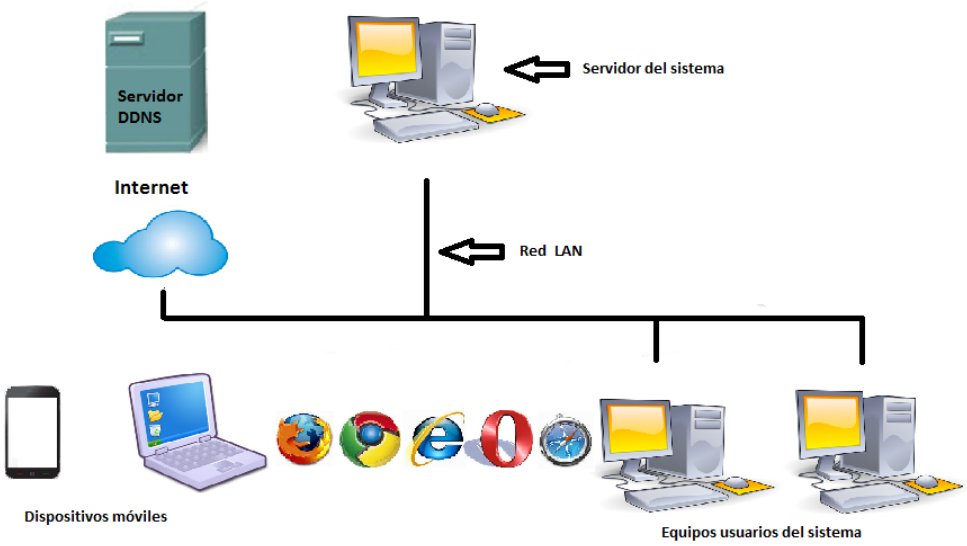


Figura 3.1: Arquitectura del sistema
Juárez y Villegas (2014)

El sistema de inventario fue desarrollado con el *framework* Ruby on Rails, el cual sigue el patrón MVC. El mismo se muestra en la Figura 3.2.

Figura 3.2: Arquitectua MVC de RoR
Juárez y Villegas (2014)

3.1.4. Prototipos de la diagramación de la interfaz

El equipo de desarrollo luego de la primera recolecta de los requerimientos funcionales que debía tener el sistema, realizó unos bosquejos que representaban una posibilidad de como luciría la interfaz gráfica del usuario. Se decidió realizar los bosquejos de las interfaces que mas se usarían en el sistema. Estos bosquejos se muestran en las Figuras 3.3, 3.4, 3.5 y 3.6.

Figura 3.3: Bosquejo de inicio de sesión
Juárez y Villegas (2014)

En la Figura 3.3, se muestra el bosquejo de inicio de sesión del sistema, el cual permite ingresar a los usuarios registrados. Este está compuesto por un cabecera con el logo y el nombre de la aplicación, y el cuerpo, el cual contiene un formulario, y este a su vez tiene dos campos, uno es el usuario, y el otro es su contraseña para poder ingresar al sistema.

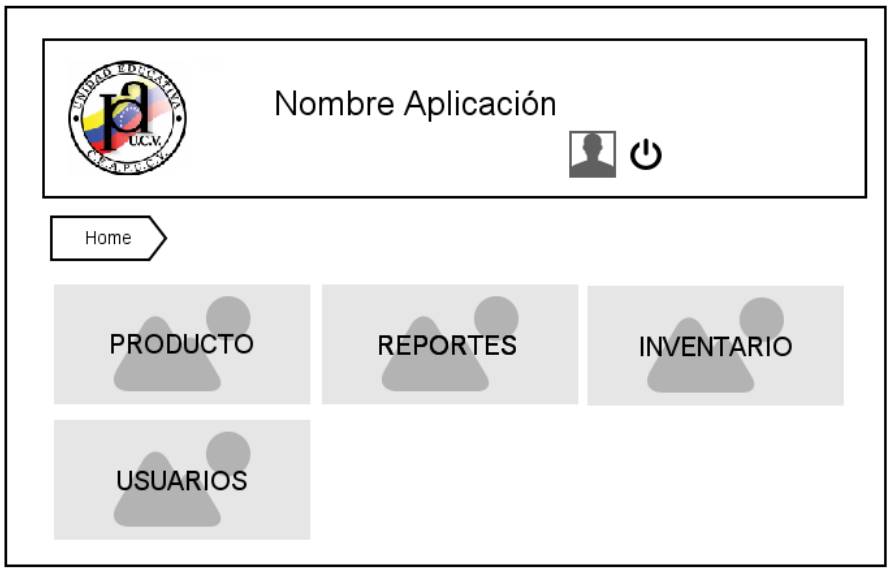


Figura 3.4: Bosquejo de inicio
Juárez y Villegas (2014)

En la Figura 3.4 se puede observar el bosquejo del inicio del sistema. Este está compuesto por varios módulos como pueden ser: USUARIOS (módulo para poder listar, editar y crear los usuarios del sistema), INVENTARIO (módulo para ingresar y egresar equipos e insumos del inventario) y REPORTES (para poder hacer consultas al inventario y generar reportes). También se puede observar en la parte superior derecha dos iconos, uno que muestra el perfil del usuario actual y el otro para cerrar sesión en el sistema.

Figura 3.5: Bosquejo Detalles del equipo
Juárez y Villegas (2014)

En la Figura 3.5 se detallan todas los datos o características de un equipo o insumo. Esta bosquejo contiene información como: la imagen del equipo, el nombre, la descripción y la cantidad en existencia en el inventario. Se puede observar el código QR correspondiente al equipo, junto con el botón para imprimir dicho código.

Figura 3.6: Bosquejo Detalle del usuario
Juárez y Villegas (2014)

En la Figura 3.6 en este bosquejo se exponen los detalles de un usuario. Este esta compuesto por información del usuario como: el nombre, el apellido, el correo, la fecha de nacimiento, rango del usuario (el permiso para acceder a través del sistema).

3.1.5. Diagrama relacional de base de datos

En la Figura 3.7 se muestra el mapa relacional de la base de datos completos, describiendo sus relaciones. Debido al tamaño y la complejidad de este, se dividió en dos figuras, para así tener una mejor compresión y legibilidad. Estas dos partes del mapa relacional se pueden observar en las figuras 3.8 y 3.9. La primera representa las tablas en las cuales se almacenan los equipos e insumos, así como sus relaciones con las tablas que almacenan datos referentes a los equipos, como son: ubicación, factura, donador, presentación, unidad y rubro. La segunda representa la tabla inventario, junto con sus relaciones con las tablas de los equipos y la de usuario.

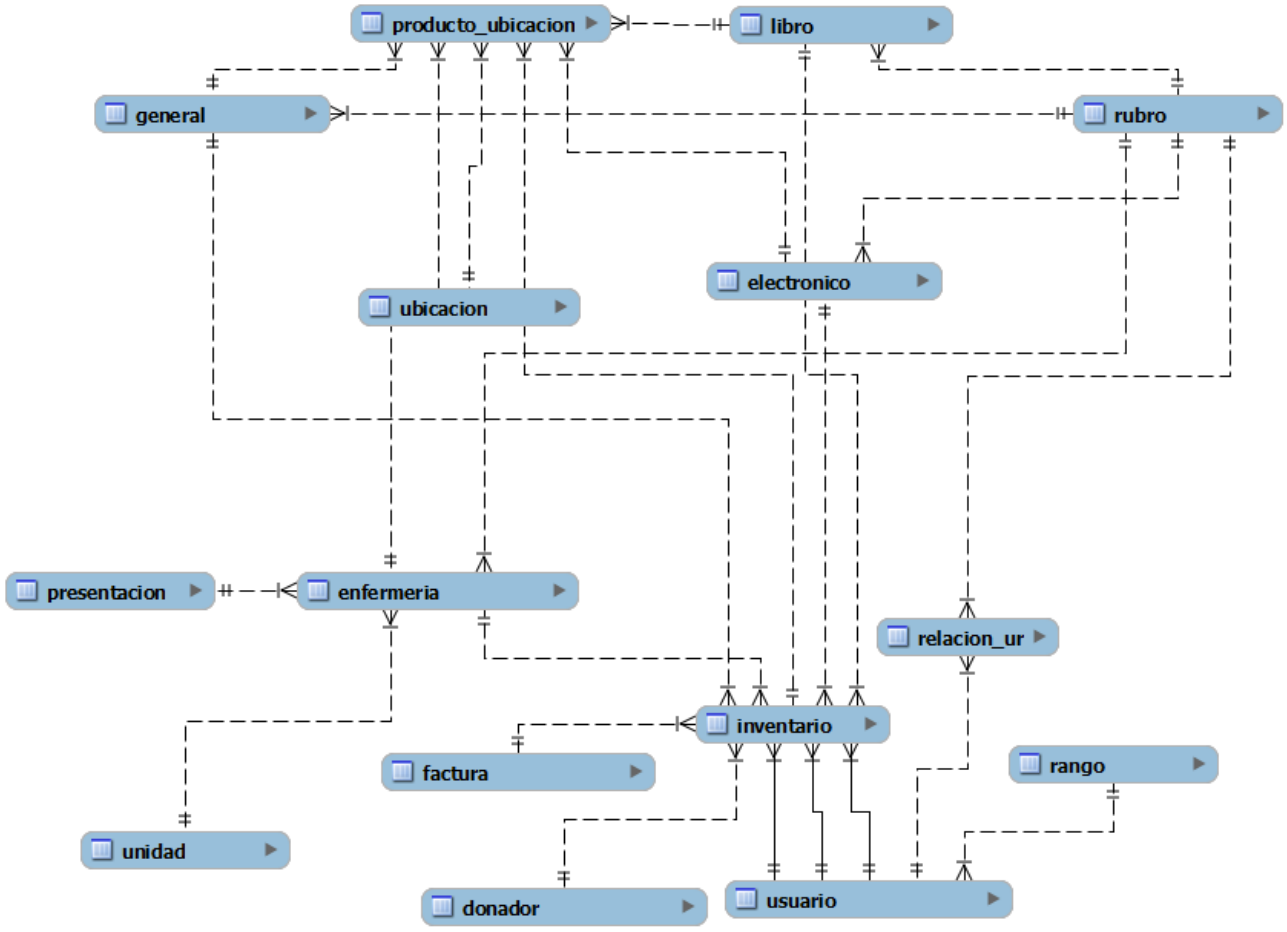


Figura 3.7: Diagrama relacional de la base de datos
Juárez y Villegas (2014)

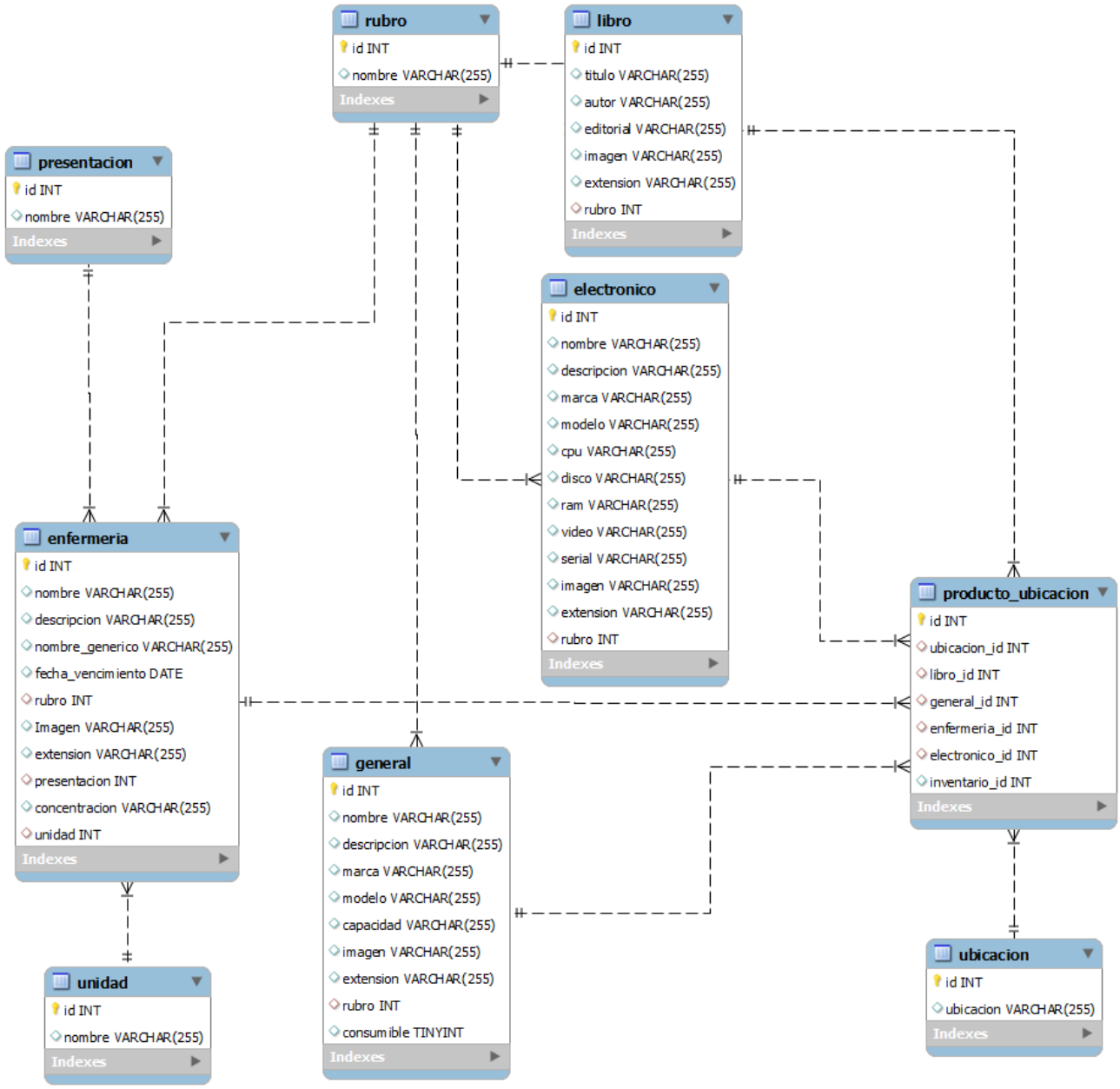


Figura 3.8: Diagrama relacional de los equipos e insumos
Juárez y Villegas (2014)

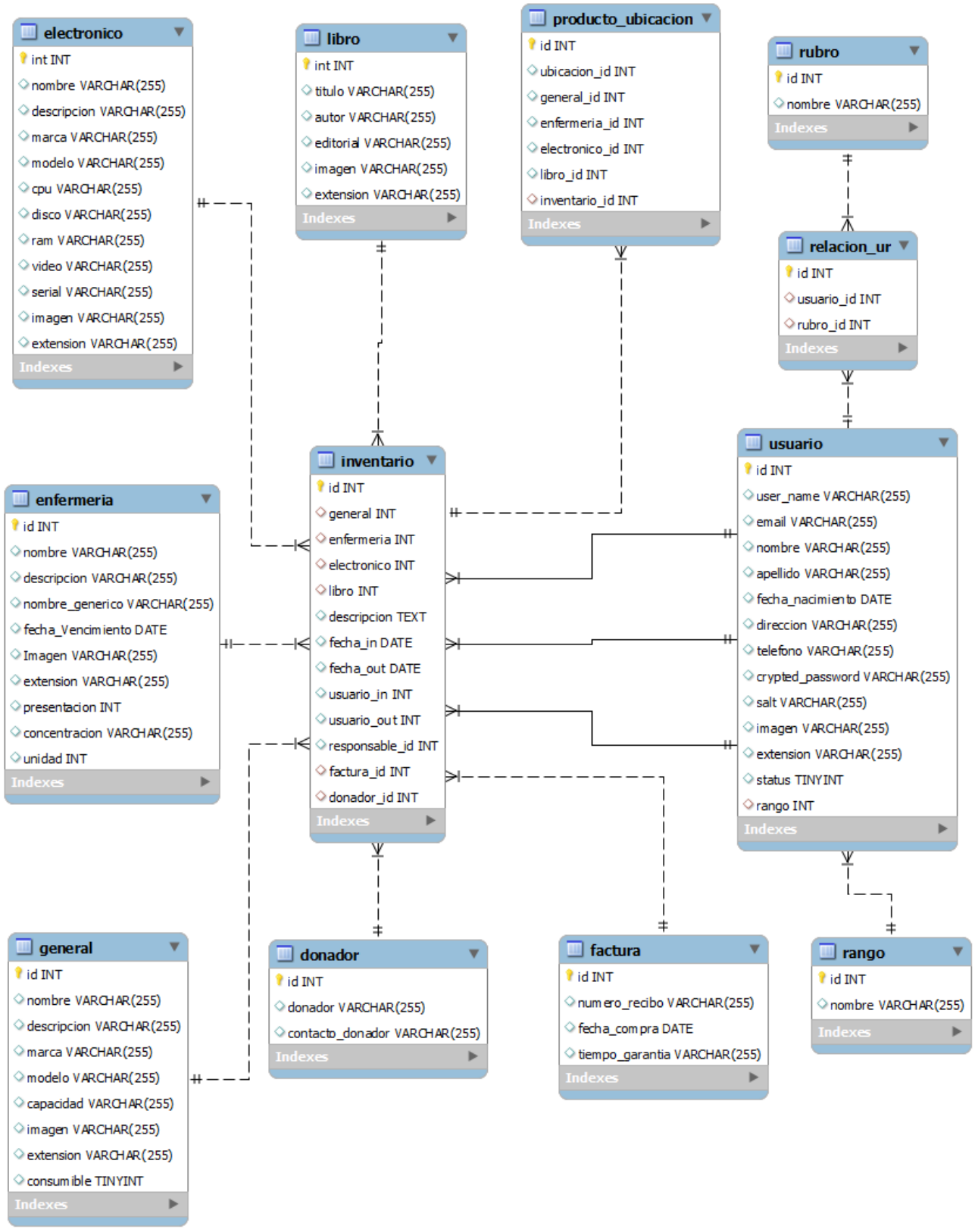


Figura 3.9: Diagrama relacional de inventario
Juárez y Villegas (2014)

3.2. Planificación

Esta fase del ciclo de vida de XP, basado en las historias de usuarios, se brindan las prioridades de los requerimientos funcionales dados por el cliente, así como también la cantidad de esfuerzo que se dispondrá para cada uno de ellos.

3.2.1. Historias de usuarios

A continuación se presentan las historias de usuarios obtenidas de los clientes.

Tabla 3.1: Historia de Usuario número 1
Juárez y Villegas (2014)

Número: 1	**Nombre**: Diseño de la aplicación
Programador: Andrés Juárez - Roosevelt Villegas	**Iteración asignada**: 0, 2, 3, 4, 5, 6, 7
Tipo: Nueva	**Tiempo estimado**: 6 meses
Descripción: se recaban los requerimientos funcionales y no funcionales de la aplicación web. Luego con estos datos se elabora el diseño, el cual contiene la arquitectura de la aplicación, casos de uso y el modelo de la base de datos	

Tabla 3.2: Historia de Usuario número 2
Juárez y Villegas (2014)

Número: 2	**Nombre**: Configuración del ambiente de trabajo
Programador: Andrés Juárez - Roosevelt Villegas	**Iteración asignada**: 0
Tipo: Nueva	**Tiempo estimado**: 10 días
Descripción: Se instalan las herramientas y configuraciones locales del sistema necesarias para el desarrollo de la aplicación	

Tabla 3.3: Historia de Usuario número 3
Juárez y Villegas (2014)

Número: 3	**Nombre**: Desarrollo de la interfaz de usuario
Programador: Andrés Juárez - Roosevelt Villegas	**Iteración asignada**: 1, 2, 3, 4, 5, 6, 7
Tipo: Nueva	**Tiempo estimado**: 6 mes
Descripción: Se comparan las interfaces gráficas de las diferentes herramientas estudiadas y se desarrolla la interfaz de la aplicación	

Tabla 3.4: Historia de Usuario número 4
Juárez y Villegas (2014)

Número: 4	**Nombre**: Registro de Usuario
Programador: Andrés Juárez - Roosevelt Villegas	**Iteración asignada**: 2
Tipo: Nueva	**Tiempo estimado**: 3 semanas
Descripción: el sistema debe permitir poder registrar usuarios con una serie de datos como: el nombre, apellido, cédula, correo etc.	

Tabla 3.5: Historia de Usuario número 5
Juárez y Villegas (2014)

Número: 5	**Nombre**: Autenticación de usuario
Programador: Andrés Juárez - Roosevelt Villegas	**Iteración asignada**: 2
Tipo: Nueva	**Tiempo estimado**: 3 semanas
Descripción: el sistema debe poder ser accedido solo por usuarios registrados. Se debe validar la cédula y la contraseña para poder ingresar al sistema	

Tabla 3.6: Historia de Usuario número 6
Juárez y Villegas (2014)

Número: 6	**Nombre**: Restricción al sistema con permisos
Programador: Andrés Juárez - Roosevelt Villegas	**Iteración asignada**: 4
Tipo: Nueva	**Tiempo estimado**: 3 semanas
Descripción: el sistema de debe poder asignar permisos a los usuarios y restringir el acceso a los módulos que la conforman con la utilización de estos permisos	

Tabla 3.7: Historia de Usuario número 7
Juárez y Villegas (2014)

Número: 7	**Nombre**: Ingreso de equipos o insumos al inventario
Programador: Andrés Juárez - Roosevelt Villegas	**Iteración asignada**: 3
Tipo: Nueva	**Tiempo estimado**: 1 mes
Descripción: el sistema debe permitir a los usuarios poder ingresar equipos o insumos al inventario	

Tabla 3.8: Historia de Usuario número 8
Juárez y Villegas (2014)

Número: 8	**Nombre**: Egreso de equipos o insumos del inventario
Programador: Andrés Juárez - Roosevelt Villegas	**Iteración asignada**: 5
Tipo: Nueva	**Tiempo estimado**: 3 semanas
Descripción: el sistema debe permitir a los usuarios poder egresar equipos o insumos al inventario	

Tabla 3.9: Historia de Usuario número 9
Juárez y Villegas (2014)

Número: 9	**Nombre**: Generar código QR de los equipos
Programador: Andrés Juárez - Roosevelt Villegas	**Iteración asignada**: 6
Tipo: Nueva	**Tiempo estimado**: 3 semanas
Descripción: el sistema debe ser capaz de generar el código QR de cualquier equipo e insumo del inventario	

Tabla 3.10: Historia de Usuario número 10
Juárez y Villegas (2014)

Número: 10	**Nombre**: Consultar el inventario
Programador: Andrés Juárez - Roosevelt Villegas	**Iteración asignada**: 7
Tipo: Nueva	**Tiempo estimado**: 1 mes
Descripción: el sistema de poseer las opciones para poder consultar la base de datos del inventario a través de el módulo consultar	

Tabla 3.11: Historia de Usuario número 11
Juárez y Villegas (2014)

Número: 11	**Nombre**: Generar reportes del inventario
Programador: Andrés Juárez - Roosevelt Villegas	**Iteración asignada**: 7
Tipo: Nueva	**Tiempo estimado**: 2 semanas
Descripción: el sistema debe poder generar un archivo PDF con los resultados previos hechos de una consulta al inventario	

Tabla 3.12: Historia de Usuario número 12
Juárez y Villegas (2014)

Número: 12	**Nombre**: Crear presentación
Programador: Andrés Juárez - Roosevelt Villegas	**Iteración asignada**: 8
Tipo: Nueva	**Tiempo estimado**: 2 semanas
Descripción: el sistema debe poder crear nuevas presentaciones para los medicamentos	

Tabla 3.13: Historia de Usuario número 13
Juárez y Villegas (2014)

Número: 13	**Nombre**: Editar presentación
Programador: Andrés Juárez - Roosevelt Villegas	**Iteración asignada**: 8
Tipo: Nueva	**Tiempo estimado**: 2 semanas
Descripción: el sistema debe poder editar las presentaciones existentes de los medicamentos en el sistema	

Tabla 3.14: Historia de Usuario número 14
Juárez y Villegas (2014)

Número: 14	**Nombre**: Crear unidad
Programador: Andrés Juárez - Roosevelt Villegas	**Iteración asignada**: 8
Tipo: Nueva	**Tiempo estimado**: 2 semanas
Descripción: el sistema debe poder crear unidades nuevas para los medicamentos en el sistema	

Tabla 3.15: Historia de Usuario número 15
Juárez y Villegas (2014)

Número: 14	**Nombre**: Editar unidad
Programador: Andrés Juárez - Roosevelt Villegas	**Iteración asignada**: 8
Tipo: Nueva	**Tiempo estimado**: 2 semanas
Descripción: el sistema debe poder editar las unidades para los medicamentos existentes en el sistema	

3.2.2. Casos de uso

A continuación se presentan los tres niveles de casos de uso que se obtuvieron a partir de los requerimientos funcionales del cliente.

Caso de uso nivel 0

En este caso de uso se modela las interacciones a nivel general de los usuarios con el sistema, tal como se muestra en la Figura 3.10. En las tablas 3.16 y 3.17 se describen los actores que interactúan con el sistema.

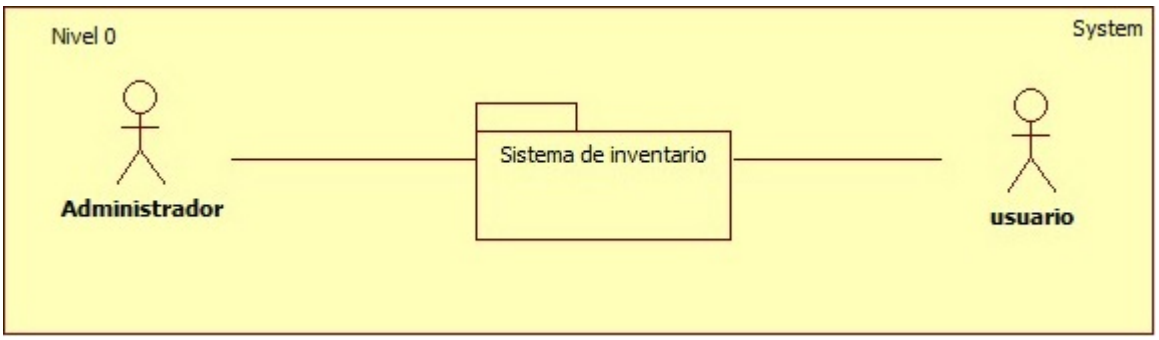


Figura 3.10: Caso de uso nivel 0
Juárez y Villegas (2014)

Tabla 3.16: Descripción del actor Administrador
Juárez y Villegas (2014)

Actor	ACT. 1 Administrador
Casos de uso	Registrar Usuarios, Editar Usuarios, Inhabilitar Usuarios, Ingresar equipos o insumos al inventario, Egresar equipos o insumos del inventario, Realizar consulta del inventario, Generar reporte, Consultar código QR, Imprimir código QR, Crear presentación, Editar presentación, Crear unidad, Editar unidad.
Tipo	Primario.
Descripción	Persona encargada de realizar los cambios y ajustes del sistema.

Tabla 3.17: Descripción del actor Usuario
Juárez y Villegas (2014)

Actor	ACT. 2 Usuario.
Casos de uso	Ingresar equipos o insumos al inventario, Egresar equipos o insumos del inventario, Realizar consulta del inventario, Generar reporte, Consultar código QR, Imprimir código QR, Crear presentación, Editar presentación, Crear unidad, Editar unidad.
Tipo	Secundario.
Descripción	Persona que podrá realizar trámites sobre el inventario.

Caso de uso nivel 1

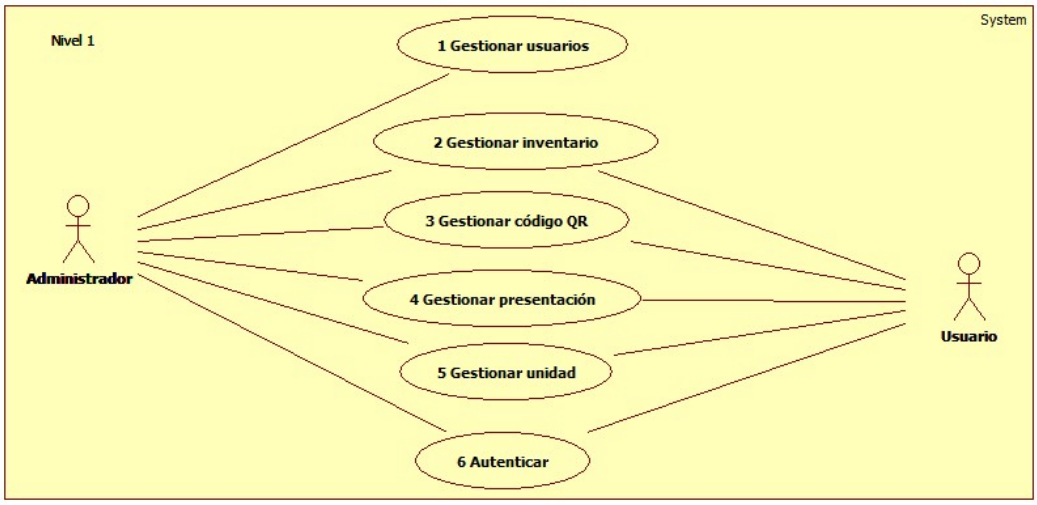


Figura 3.11: Caso de uso nivel 1
Juárez y Villegas (2014)

Caso de uso nivel 2

En la Figura 3.12 se modela los casos de uso de nivel dos de abstracción del sistema. Así mismo en las tablas 3.18, 3.19, 3.20, 3.21, 3.22, 3.23, 3.24, 3.25, 3.26 y 3.27 se describen detalladamente cada uno de los estos casos de uso.

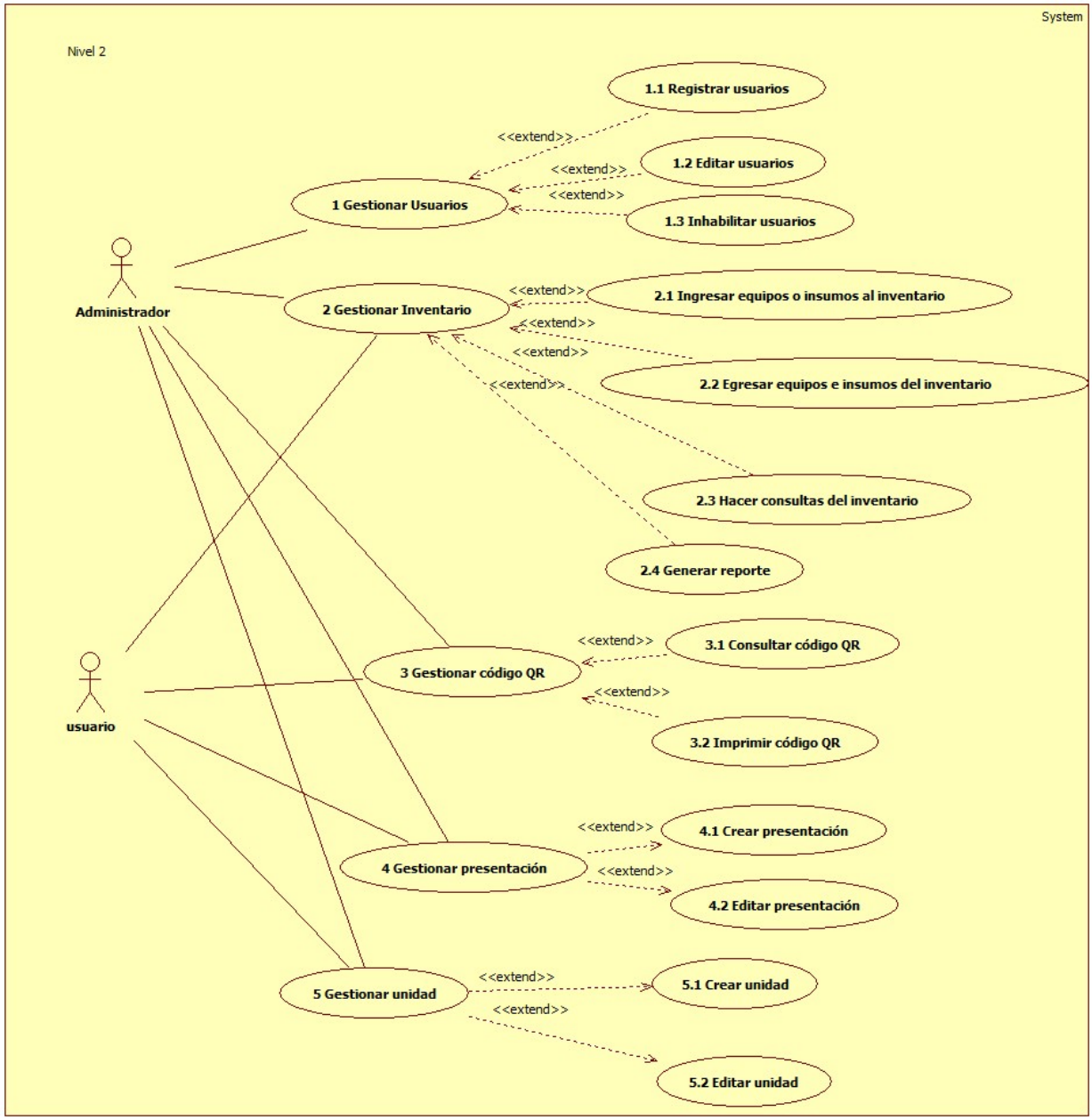


Figura 3.12: Caso de uso nivel 2
Juárez y Villegas (2014)

Tabla 3.18: Descripción del caso de uso Registrar usuarios
Juárez y Villegas (2014)

Caso de uso	CU 1 Registrar usuarios
Actor	ACT 1. Administrador
Descripción	Este caso de uso permite la creación de un nuevo usuario en la base de datos.
Precondición	El usuario ha iniciado sesión en el sistema
Flujo básico	-Ingresar al módulo de usuarios -Presionar el botón "Nuevo" -Llenar el formulario con los datos solicitados y presionar "Guardar" -El sistema envía un correo electrónico al nuevo usuario indicando su nombre de usuario y clave de acceso
Flujos alternos	-Ingresar al módulo de usuarios -Presionar el botón "Nuevo" -Llenar el formulario con los datos solicitados y presionar "Guardar" -En caso de que algún dato suministrado no cumpla con el formato adecuado o falte algún dato obligatorio se mostrará un mensaje indicando cual es el error para continuar

Tabla 3.19: Descripción del caso de uso Editar usuarios
Juárez y Villegas (2014)

Caso de uso	CU 2. Editar usuarios
Actor	ACT 1. Administrador ACT 2. Usuario
Descripción	Este caso de uso permite la edición de un usuario existente en la base de datos.
Precondición	El usuario ha iniciado sesión en el sistema
Flujo básico	El administrador puede -Acceder al módulo de usuario -Seleccionar el usuario a editar -Modificar los datos que se desea -Presionar "Guardar"
Flujos alternos	El usuario común puede: -Acceder a "Ver Perfil" -Hacer click en "Editar" -Modificar los campos que así requiera -Presionar "Guardar"

Tabla 3.20: Descripción del caso de uso Inhabilitar usuarios
Juárez y Villegas (2014)

Caso de uso	CU 3. Inhabilitar usuarios
Actor	ACT 1. Administrador
Descripción	Este caso de uso permite escoger un usuario de la base de datos e inhabilitarlo para que pueda acceder al sistema.
Precondición	El usuario ha iniciado sesión en el sistema
Flujo básico	-Ingresar al modulo de usuarios -Seleccionar el usuario a editar -Desmarcar la casilla “Estatus” -Pulsar “Guardar”
Flujos alternos	

Tabla 3.21: Descripción del caso de uso Autenticar usuario
Juárez y Villegas (2014)

Caso de uso	CU 4. Autenticar usuario
Actor	ACT 1. Administrador ACT 2. Usuario
Descripción	Este caso de uso se encarga de verificar los datos suministrados por los usuarios para acceder al sistema
Precondición	El usuario no ha iniciado sesión en el sistema
Flujo básico	-Ingresar a la URL de la aplicación -Ingresar la cédula de usuario y clave -Se muestra un mensaje de éxito en la pantalla principal de la aplicación
Flujos alternos	-Ingresar a la URL de la aplicación -Insertar nombre de usuario y clave -Se muestra un mensaje de un error

Tabla 3.22: Descripción del caso de uso Consultar código QR
Juárez y Villegas (2014)

Caso de uso	CU 5. Consultar código QR
Actor	ACT 1. Administrador ACT 2. Usuario
Descripción	Este caso de uso permite realizar una consulta de un equipo o insumo en particular.
Precondición	El usuario ha iniciado sesión en el sistema
Flujo básico	-Ingresar al módulo “Lista de equipos” -Seleccionar de la lista el equipo o insumo y presionar “ver”
Flujos alternos	-Ingresar al módulo “Gestión de inventario” -Ingresar en el módulo “Egresar” -Seleccionar de la lista el equipo o insumo y presionar “ver”

Tabla 3.23: Descripción del caso de uso Imprimir código QR
Juárez y Villegas (2014)

Caso de uso	CU 6. Imprimir código QR
Actor	ACT 1. Administrador
Descripción	Este caso de uso permite editar un rubro existente en la base de datos.
Precondición	El usuario ha iniciado sesión en el sistema
Flujo básico	-Ingresar al módulo "Lista de equipos" -Seleccionar de la lista el equipo o insumo y presionar "ver" -Presionar "Imprimir código QR"
Flujos alternos	-Ingresar al módulo "Gestión de inventario" -Ingresar en el módulo "Egresar" -Seleccionar de la lista el equipo o insumo y presionar "ver" -Presionar "Imprimir código QR"

Tabla 3.24: Descripción del caso de uso Ingresar equipos o insumos al inventario
Juárez y Villegas (2014)

Caso de uso	CU 7. Ingresar equipos o insumos al inventario
Actor	ACT 1. Administrador ACT 2. Usuario
Descripción	Este caso de uso permite al usuario ingresar un nuevo equipo o insumo al inventario
Precondición	El usuario ha iniciado sesión en el sistema
Flujo básico	-Ingresar al módulo "Gestión de inventario" -Ingresar en el módulo "Ingresar" -Rellenar los datos requeridos y presionar "Finalizar" -Se muestra un mensaje de éxito en el módulo "Gestionar inventario"
Flujos alternos	-Ingresar al módulo "Gestión de inventario" -Ingresar en el módulo "Ingresar" -Rellenar los datos requeridos y presionar "Finalizar" -si los datos son erróneos, se mostrará un(os) mensaje(s) señalando donde es el error

Tabla 3.25: Descripción del caso de uso Egresar equipos o insumos del inventario
Juárez y Villegas (2014)

Caso de uso	CU 8. Egresar equipos o insumos del inventario
Actor	ACT 1. Administrador ACT 2. Usuario
Descripción	Este caso de uso permite al usuario egresar un equipo o insumo del inventario dando a conocer el motivo por el cual se da la baja.
Precondición	El usuario ha iniciado sesión en el sistema
Flujo básico	-Ingresar al módulo "Gestión de inventario" -Ingresar en el módulo "Egresar" -Rellenar los datos requeridos y presionar "Finalizar" -Se muestra un mensaje de éxito en el módulo "Gestionar inventario"
Flujos alternos	-Ingresar al módulo "Gestión de inventario" -Ingresar en el módulo "Egresar" -Rellenar los datos requeridos y presionar "Finalizar" -si los datos son erróneos, se mostrará un(os) mensaje(s) señalando donde es el error

Tabla 3.26: Descripción del caso de uso Hacer consultas del inventario
Juárez y Villegas (2014)

Caso de uso	CU 9. Hacer consultas del inventario
Actor	ACT 1. Administrador ACT 2. Usuario
Descripción	Este caso de uso permite al usuario realizar consultas sobre la base de datos
Precondición	El usuario ha iniciado sesión en el sistema
Flujo básico	-Ingresar al módulo de "Reportes" -Seleccionar los rubros que se quiere consultar, ingresando los filtros convenientes -Presionar "Consultar"
Flujos alternos	

Tabla 3.27: Descripción del caso de uso Generar reporte
Juárez y Villegas (2014)

Caso de uso	CU 10. Generar reporte
Actor	ACT 1. Administrador ACT 2. Usuario
Descripción	Este caso de uso permite al usuario generar reportes sobre las consultas realizadas.
Precondición	El usuario ha iniciado sesión en el sistema
Flujo básico	-Ingresar al módulo de "Reportes" -Seleccionar los rubros que se quiere consultar, ingresando los filtros convenientes -Presionar "Consultar" -Presionar "Generar PDF"
Flujos alternos	

Tabla 3.28: Descripción del caso de uso Crear presentación
Juárez y Villegas (2014)

Caso de Uso	CU 11. Crear presentación
Actor	ACT 1. Administrador ACT 2. Usuario
Descripción	Este caso de uso permite al usuario crear presentaciones de los medicamentos nuevas
Precondición	El usuario ha iniciado sesión en el sistema
Flujo básico	-Ingresar al módulo de "Enfermería" -Ingresar al menú "Presentaciones" -Presionar "Nuevo" -Llenar el campo y presionar "Guardar" -se muestra un mensaje de éxito
Flujos Alternos	-Ingresar al módulo de "Enfermería" -Ingresar al menú "Presentaciones" -Presionar "Nuevo" -Llenar el campo y presionar "Guardar" -si el campo es incorrecto se muestra un mensaje de error

Tabla 3.29: Descripción del caso de uso Editar presentación
Juárez y Villegas (2014)

Caso de Uso	CU 12. Editar presentación
Actor	ACT 1. Administrador ACT 2. Usuario
Descripción	Este caso de uso permite al usuario editar presentaciones de los medicamentos existentes
Precondición	El usuario ha iniciado sesión en el sistema
Flujo básico	-Ingresar al módulo de "Enfermería" -Ingresar al menú "Presentaciones" -Seleccionar la presentación que se quiere editar y presionar "Editar" -Llenar el campo y presionar "Guardar" -se muestra un mensaje de éxito
Flujos Alternos	-Ingresar al módulo de "Enfermería" -Ingresar al menú "Presentaciones" -Seleccionar la presentación que se quiere editar y presionar "Editar" -Llenar el campo y presionar "Guardar" -si el campo es incorrecto se muestra un mensaje de error

Tabla 3.30: Descripción del caso de uso Crear Unidad
Juárez y Villegas (2014)

Caso de Uso	CU 13. Crear Unidad
Actor	ACT 1. Administrador ACT 2. Usuario
Descripción	Este caso de uso permite al usuario crear unidades de medicamentos nuevos
Precondición	El usuario ha iniciado sesión en el sistema
Flujo básico	-Ingresar al módulo de "Enfermería" -Ingresar al menú "Unidades" -Presionar "Nuevo" -Llenar el campo y presionar "Guardar" -se muestra un mensaje de éxito
Flujos Alternos	-Ingresar al módulo de "Enfermería" -Ingresar al menú "Unidades" -Presionar "Nuevo" -Llenar el campo y presionar "Guardar" -si el campo es incorrecto se muestra un mensaje de error

Tabla 3.31: Descripción del caso de uso Editar Unidad
Juárez y Villegas (2014)

Caso de Uso	CU 14. Editar unidad
Actor	ACT 1. Administrador ACT 2. Usuario
Descripción	Este caso de uso permite al usuario editar unidades de los medicamentos existentes
Precondición	El usuario ha iniciado sesión en el sistema
Flujo básico	-Ingresar al módulo de "Enfermería" -Ingresar al menú "Unidades" -Seleccionar la unidad que se quiere editar y presionar "Editar" -Llenar el campo y presionar "Guardar" -se muestra un mensaje de éxito
Flujos Alternos	-Ingresar al módulo de "Enfermería" -Ingresar al menú "Unidades" -Seleccionar la unidad que se quiere editar y presionar "Editar" -Llenar el campo y presionar "Guardar" -si el campo es incorrecto se muestra un mensaje de error

3.3. Iteraciones

Todo proyecto que siga el método de desarrollo XP, debe distribuir su desarrollo en iteraciones, los cuales harán que el proyecto progrese de forma incremental con cada una.

El desarrollo del sistema se divide en estas iteraciones, las cuales en cada una se llevará a cabo una o varias historias de usuarios. A continuación se procede a detallar las iteraciones llevadas a cabo para este sistema.

3.3.1. Iteración 0

Planificación: en esta primera iteración, resumida en la Tabla 3.32, se realiza la instalación y configuración del ambiente de desarrollo. También se hace el análisis respectivo sobre los requerimientos, obteniendo como resultado los diagramas de prototipos de interfaces de usuario 3.1.4, arquitectura 3.1.3, modelo de datos 3.1.5 en la Figura 3.7 y casos de uso 3.2.2.

Tabla 3.32: Iteración 0

Número de iteración: 0	**Número de historia**: 1 y 2
Caso de uso:	
Fecha de inicio: 10/8/2013	**Fecha fin**: 30/8/2013
Descripción: configuración del ambiente de desarrollo y diseño de la aplicación	**Tipo**: configuración y diseño

Diseño: Se realiza la instalación y configuración del servidor para Ruby on Rails sobre ambiente Linux, distribución Ubuntu. Para tener un mejor control sobre el ambiente de desarrollo de la aplicación en Rails, se instala RVM (*Ruby Versión Manager*, Manejador de Versiones de Ruby) el cual encapsula cada proyecto de Ruby con su ambiente de desarrollo, para no afectar otros proyectos de Ruby.

El primer paso para instalar RVM, es instalar CUrl, que es una herramienta de la consola de comando para realizar transferencias de archivos con sintaxis URL. Para ello se coloca en la consola:

$ \CUrl -sSL https://get.rvm.io | bash -s stable

Luego procedemos a instalar RVM, el cual instala Ruby, junto con el *framework* Rails:

$ \CUrl -sSL https://get.rvm.io | bash -s stable –ruby –rails

Ahora con el RVM, junto con Rails instalados, se procede a crear el directorio de gemas que se utilizará en la aplicación:

$ rvm gemset create tesisPiloto

El siguiente procedimiento es instalar la gema del manejador de base de datos MySQL2:

$ gem install mysql2

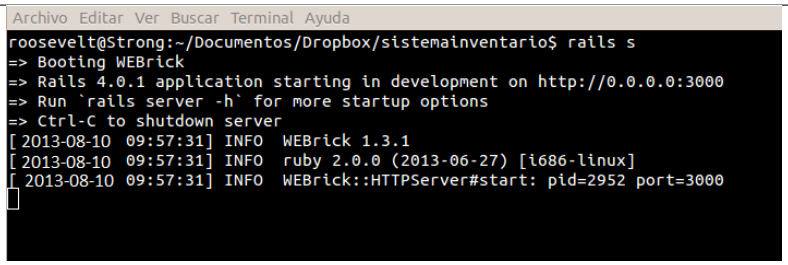
Además se agregan el resto de las gemas que vayan siendo necesarias a lo largo del desarrollo de la aplicación al archivo Gemfile, las cuales se instalan cuando se ejecuta el comando bundle install una vez modificado dicho archivo. Las gemas agregadas a lo largo de la aplicación se pueden observar en el capitulo del marco conceptual en la sección 2.3.3.

Pruebas: se realizan las pruebas funcionales de la primera iteración.

Tabla 3.33: Prueba 1

Número de caso de prueba: 1	**Número de historia de usuario:** 2
Descripción	verificar la correcta instalación de Ruby on Rails
Resultados	

Tabla 3.34: Prueba 2

Número de caso de prueba: 2	**Número de historia de usuario:** 2
Descripción	verificar que el servidor de Ruby on Rails funciona correctamente
Resultados	

3.3.2. Iteración 1

Planificación: el objetivo principal de esta iteración es crear la interfaz gráfica del sistema. Esta interfaz se realizó basándose en los bosquejos realizados en la iteración anterior. Dicha iteración se encuentra resumida en la Tabla 3.35.

Tabla 3.35: Iteración 1

Número de iteración: 1	**Número de historia**: 3
Caso de uso:	
Fecha de inicio: 02/09/2013	**Fecha fin**: 15/09/2013
Descripción: diseño y desarrollo de la interfaz gráfica del sistema	**Tipo**: desarrollo y diseño

Diseño: previo a la diagramación de los bosquejos 3.1.4. Se realizó una comparación de herramientas existentes 2.2.6, las cuales eran aplicaciones las mas parecidas en funcionalidades a lo que se quería realizar en el presente TEG. En dicha comparación, uno de los parámetros era la interfaz gráfica; y la conclusión dio como resultado la herramienta o aplicación web Birtud 2.2.6 como el modelo a seguir para realizar la interfaz gráfica. Se puede observar la interfaz gráfica en la Figura 2.26.

Para el desarrollo de la interfaz gráfica se hace uso del *framework Twitter Bootstrap* 2.5, el que simplifica el proceso de creación de diseños web combinando CSS y JavaScript.

Codificación: Debido a que esta aplicación web se realiza con el *framework* Ruby on Rails 2.3, se basa en el patrón MVC. La Vista está conformada por dos *layout*, los cuales simplemente son plantillas para las vistas comunes. Un *layout* (interno), el cual es para los usuarios que hayan iniciado sesión, y el segundo *layout* (externo) para usuarios que no hayan iniciado sesión, es decir la vista del inicio de sesión. En la Figura 3.13 se muestra el *layout* (interno).

```
<body>
  <nav class="navbar navbar-inverse navbar-fixed-top" role="navigation">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-ex1-collapse">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <% link_to "Inicio", homes_path, class: "navbar-brand" %>
      <%= link_to controller: "homes", action:"index" do %>
        <%= image_tag("ceapucv.jpg", class: "img-responsive img-home", alt: "Responsive image", size: "50x50")%>
      <% end -%>
    </div>
    <div class="collapse navbar-collapse navbar-ex1-collapse">
      <ul class="nav navbar-nav"></ul>
      <ul class="nav navbar-nav navbar-right">
        <li class="dropdown">
        <% if current_user.extension == nil %>
              <a href="#" class="dropdown-toggle" data-toggle="dropdown"><%= current_user.nombre+" "+current_user.apellido
                %> <i class="glyphicon glyphicon-user"> </i> <b class="caret"></b></a>
          <%else%>
              <a href="#" class="dropdown-toggle" data-toggle="dropdown"><%= current_user.nombre+" "+current_user.apellido
                %> <%= image_tag current_user.imagen_path, width: "25", height: "25" %> <b class="caret"></b></a>
        <%
          end%>
          <ul class="dropdown-menu">
            <li><a href="<%= usuarios_aux3_path(lu:"pe", id:current_user.id) %>">Ver Perfil</a></li>
            <li class="divider"></li>
            <li> <%= link_to "Cerrar Sesión", controller: "usuario_sessions", action:"destroy" %> </li>
          </ul>
        </li>
      </ul>
    </div><!-- /.navbar-collapse -->
  </nav>
  <div class="container">
    <!-- <div class="tope"> </div> -->
    <%unless notice == nil %>
      <div class="alert alert-success"><p id="notice" ><%= notice %></p></div>
    <%end%>
    <%= yield %>
```

Figura 3.13: Layout interno
Juárez y Villegas (2014)

Luego se tiene el código del layout (externo), el cual se usa para la vista de autenticación. Se puede observar en la Figura 3.14.

```
<!DOCTYPE html>
<html>
<head>
  <title>TesisPiloto</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <%= stylesheet_link_tag    "application", media: "all", "data-turbolinks-track" => true %>
  <%= javascript_include_tag "application", "data-turbolinks-track" => true %>
  <%= csrf_meta_tags %>
  <%= favicon_link_tag 'favicon.ico' %>
</head>
<body>
  <nav class="navbar navbar-inverse navbar-fixed-top" role="navigation">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-ex1-collapse">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <%= link_to "Sistema de inventario del CEAPUCV", usuario_sessions_path, class: "navbar-brand" %>
    </div>
    <div class="collapse navbar-collapse navbar-ex1-collapse">
      <ul class="nav navbar-nav"></ul>
      <ul class="nav navbar-nav navbar-right">
      </ul>
    </div>
  </nav>
  <div class="container">
    <div class="tope"> </div>
    <%unless notice == nil %>
      <div class="alert alert-success"><p id="notice" ><%= notice %></p></div>
    <%end%>
    <%= yield %>
  </div>
</body>
</html>
```

Figura 3.14: Layout externo
Juárez y Villegas (2014)

Pruebas: se realizan las pruebas funcionales de la iteración número uno.

Tabla 3.36: Prueba 3

Número de caso de prueba: 3	**Número de historia de usuario:** 2
Descripción	verificar el parecido de la interfaz desarrollada con los prototipos de interfaz de diagramación y la interfaz de la herramienta de inventario Web Birtud
Resultados	Figura 3.15 Interfaz desarrollada Figura 3.16 Prototipos de diagramación Figura 3.17 Interfaz de Birtud

Figura 3.15: Interfaz de usuario desarrollada
Juárez y Villegas (2014)

Figura 3.16: Interfaz de usuario de los prototipos de diagramación
Juárez y Villegas (2014)

Figura 3.17: Interfaz de usuario de Birtud
Juárez y Villegas (2014)

3.3.3. Iteración 2

Planificación: en esta iteración, cuyo resumen se refleja en la Tabla 3.37, se comienza a desarrollar los requerimientos funcionales de gestionar los usuarios (el módulo de usuarios), es decir crear y editar usuarios, además de la autenticación para poder acceder al sistema. Junto a ello sus correspondientes vistas, por lo cual también se desarrollan las interfaces gráficas para el usuario.

Tabla 3.37: Iteración 2

Número de iteración: 2	**Número de historia**: 1, 3, 4, 5
Caso de uso: 1, 2, 3 y 4	
Fecha de inicio: 16/09/2013	**Fecha fin**: 30/09/2013
Descripción: desarrollo de requerimientos funcionales de gestión de usuarios, autenticación e interfaz gráfica	**Tipo**: desarrollo

Diseño:

Codificación: siguiendo el patrón MVC que obedece Ruby on Rails, el desarrollo del registro de usuarios de esta iteración se concentra en el controlador de usuarios: **usuarios_controller.rb**, su modelo: **usuario.rb** y la vista de usuarios: **new.html.erb** mostrados en la Figura 2.30. Para la autenticación, se hizo uso de la gema "sorcery", la cual crea un controlador, una vista y modifica la tabla usuario de la base de datos.

Para el módulo de usuarios, es decir crear y editar usuarios, se creó primero el controlador, como muestra la Figura 3.18.

```
def create
 @rubros_array = Rubro.all
  # render :text => params.inspect
  @arr = params["usuario"]["rubro"].to_a
  @usuario = Usuario.new(params[:usuario].permit( :username, :email, :password, :password_confirmation, :salt, :nombre, :apellido,
     :telefono, :rango_id, :imagen, :extension, :status))
  @hoy = DateTime.current().to_formatted_s(:db)

  respond_to do |format|
    if @usuario.save
      nuevo = Usuario.last
      for i in 1..@arr.length-1
        @rub = @arr[i].to_i
        ActiveRecord::Base.connection.execute("INSERT INTO relacion_urs(usuario_id, rubro_id, created_at) VALUES(#{nuevo.id}, #{@
          rub}, '#{@hoy}')")
      end
      UserMailer.welcome_email(@usuario).deliver
      format.html { redirect_to @usuario, notice: 'Usuario creado satisfactoriamente.' }
      format.json { render action: 'show', status: :created, location: @usuario }
    else
      format.html { render action: 'new' }
      format.json { render json: @usuario.errors, status: :unprocessable_entity }
    end
  end
end
```

Figura 3.18: Función para crear usuarios del Controlador usuario
Juárez y Villegas (2014)

Esta función del controlador permite leer los datos enviados a través del formulario y guardarlos en completo orden en la base de datos; e inmediatamente después de la operación, envía un mensaje de éxito.

Luego se desarrollo la vista del formulario, que contiene la información necesaria para crear un usuario del sistema. Esto se muestra la Figura 3.19. Esta contiene todos los campos necesarios para registrar un usuario en el sistema, como: la cédula, el nombre, el apellido y otros datos.

```
def create
 @rubros_array = Rubro.all
  @arr = params["usuario"]["rubro"].to_a
  @usuario = Usuario.new(params[:usuario].permit( :username, :email, :password, :password_confirmation, :salt, :nombre,
    :apellido, :telefono, :rango_id, :imagen, :extension, :status))
  @hoy = DateTime.current().to_formatted_s(:db)

  respond_to do |format|
    if @usuario.save
      nuevo = Usuario.last
      for i in 1..@arr.length-1
        @rub = @arr[i].to_i
        RelacionUr.create(usuario_id: nuevo.id, rubro_id: @rub, created_at: "NOW()")
      end
      UserMailer.welcome_email(@usuario).deliver
      format.html { redirect_to @usuario, notice: 'Usuario creado satisfactoriamente.' }
      format.json { render action: 'show', status: :created, location: @usuario }
    else
      format.html { render action: 'new' }
      format.json { render json: @usuario.errors, status: :unprocessable_entity }
    end
  end
end
```

Figura 3.19: Formulario para crear un usuario
Juárez y Villegas (2014)

Para la validación de dicho formulario, Ruby on Rails puede crear validaciones en el modelo, las cuales son implementadas en la vista, como se muestra en la Figura 3.20. Dichas validaciones pueden ser: los campo cédula, nombre, apellido y correo electrónico no estén vacíos. Que no se permitan números o signos en el campo nombre, que el correo electrónico sea válido.

```
class EmailValidator < ActiveModel::EachValidator
  def validate_each(record, attribute, value)
    unless value =~ /\A([^@\s]+)@((?:[-a-z0-9]+\.)+[a-z]{2,})\z/i
      record.errors[attribute] << (options[:message] || "formato inválido. Ejemplo: usuario@empresa.com")
    end
  end
end
class PhoneValidator < ActiveModel::EachValidator
  def validate_each(record, attribute, value)
    unless value =~ /0(\d{3})-(\d{3})/
      record.errors[attribute] << (options[:message] || "formato inválido. Ejemplo: 0212-1234567")
    end
  end
end
class Usuario < ActiveRecord::Base
  authenticates_with_sorcery!
  validates :password, presence: {message: "No puede estar en blanco"}, on: :create
  validates_length_of :password, minimum: 5, message: "la clave debe contener mínimo 5 caracteres", if: :password, on: :create
  validates :password, confirmation: {message: "las contraseñas no coinciden"}
  validates :username, presence: {message: "No puede estar en blanco"}, length: { minimum: 6, message: "debe contener como mínimo 6
    dígitos" }, numericality: { only_integer: true, message:"campo cedula solo permite numeros" }, uniqueness: {message: "ya
    existente"}
  validates :email, presence: {message: "No puede estar en blanco"}, email: true,  uniqueness: {message: "email ya existente"}
  validates :nombre, presence: {message: "No puede estar en blanco"}, format: { with: /\A[a-zA-Záéíóú ]+\z/, message: "solo permite
    letras y espacios" }, on: :create
  validates :apellido, presence: {message: "No puede estar en blanco"}, format: { with: /\A[a-zA-Záéíóú ]+\z/, message: "solo
    permite letras y espacios" }, on: :create
  validates :telefono, presence: {message: "No puede estar en blanco"}, phone: true, on: :create

  belongs_to :rango
  has_many :relacion_ur
  has_many :usuarioIn, :class_name => "Inventario", :foreign_key => "usuarioIn_id"
  has_many :usuarioOut, :class_name => "Inventario", :foreign_key => "usuarioOut_id"
  has_many :usuarioResp, :class_name => "Inventario", :foreign_key => "responsable_id"

  attr_accessor :rubro
```

Figura 3.20: Modelo Usuario
Juárez y Villegas (2014)

Autenticación

El proceso de autenticación de este sistema se realizó con la gema "sorcery", la cual como ya se explico anteriormente, crea un controlador llamado **usuario_sessions.rb**, su vista correspondiente y agrega columnas a la tabla de usuarios de la base de datos para el proceso de autenticación mismo.

El controlador que crea sorcery se muestra en la Figura 3.21. Este recibe la información enviada desde la vista de inicio de sesión, la compara con la que está en la base de datos. Si es correcta la información, se redirige al inicio de la aplicación, en caso contrario envía un mensaje de error a la vista de inicio de sesión.

```ruby
class UsuarioSessionsController < ApplicationController
    layout "login"
def index
        render action: 'new'
    end
    def new
        if logged_in?
            redirect_to(:homes)
        else
            @usuario = Usuario.new
        end
    end
    def show
        logout
        render action: 'new'
    end

    def create
        respond_to do |format|
          if @usuario = login(params[:username],params[:password])
            if current_user.status == true
                format.html { redirect_to homes_path, notice: "Bienvenido #{current_user.nombre} #{current_user.apellido}" }
                format.xml { render :xml => @usuario, :status => :created, :location => @usuario }
            else
                format.html { flash.now[:alert] = "Fallo de LogIn: usted esta inhabilitado para ingresar al sistema."; render :actio
                    => "new" }
                format.xml { render :xml => @usuario.errors, :status => :unprocessable_entity }
            end
          else
            format.html { flash.now[:alert] = "Fallo de LogIn: revise sus datos e intente nuevamente."; render :action => "new" }
            format.xml { render :xml => @usuario.errors, :status => :unprocessable_entity }
          end
        end
    end

    def destroy
        logout
        #redirect_to(:usuario_sessions, message: "Hasta Luego")
        redirect_to usuario_sessions_path, notice: "Hasta Luego"

    end
```

Figura 3.21: Controlador de la gema sorcery
Juárez y Villegas (2014)

Luego la vista correspondiente, en otras palabras, el formulario de autenticación para acceder al sistema. Dicha vista se muestra en la Figura 3.22. Esta vista es el formulario que se despliega para poder acceder a la aplicación.

```
<div class="row">
  <div class="col-md-4"></div>
  <div class="col-md-4">

<h1>Inicio de Sesion: </h1>

<%= form_tag usuario_sessions_path, class: "form-horizontal", id: "form-login", method: :post do %>

  <div class="form-group">
    <%= label_tag :username, "Cédula:", class: "control-label" %>
    <div class="controls">
      <%= text_field_tag :username, nil, {class: "form-control", name: "username"} %>
    </div>
  </div>

  <div class="form-group">
    <%= label_tag :password, "Clave:", class: "control-label" %>
    <div class="controls">
      <%= password_field_tag :password,nil, class: "form-control", name: "password" %>
    </div>

  </div>

  <div class="form-group">
    <div class="controls actions">
      <button class="btn btn-success" type="submit"> Entrar </button>
      <a class="right"></a>
      <%= link_to '¿Has olvidado tu contraseña?', new_password_reset_path %>
    </div>
  </div>


<% end %>
  </div>
  <div class="col-md-4"></div>
</div>
```

Figura 3.22: Vista de autenticación
Juárez y Villegas (2014)

Pruebas: se realizan las pruebas funcionales de la iteración número dos.

Tabla 3.38: Prueba 4

Número de caso de prueba: 4	**Número de historia de usuario:** 4
Descripción	verificar el correcto funcionamiento de la creación de usuarios con sus validaciones
Resultados	Con los datos incorrectos: Figura 3.23 Con los datos correctos: Figura 3.24

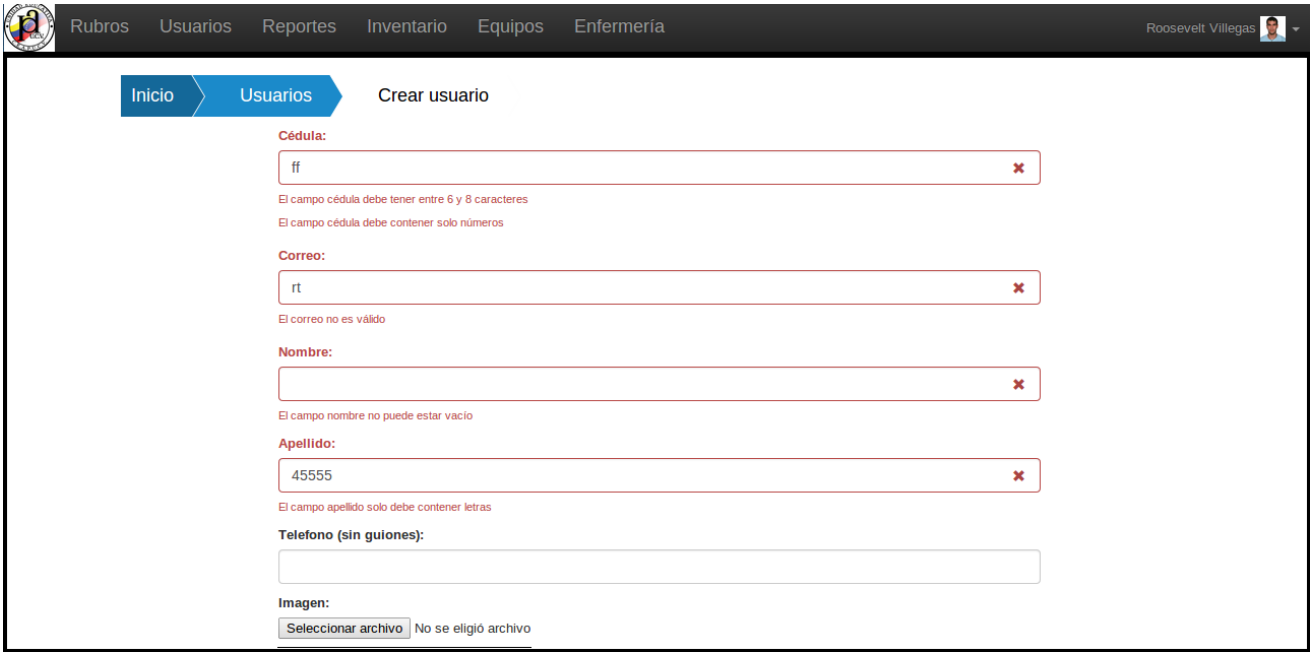


Figura 3.23: Formulario con datos incorrectos
Juárez y Villegas (2014)

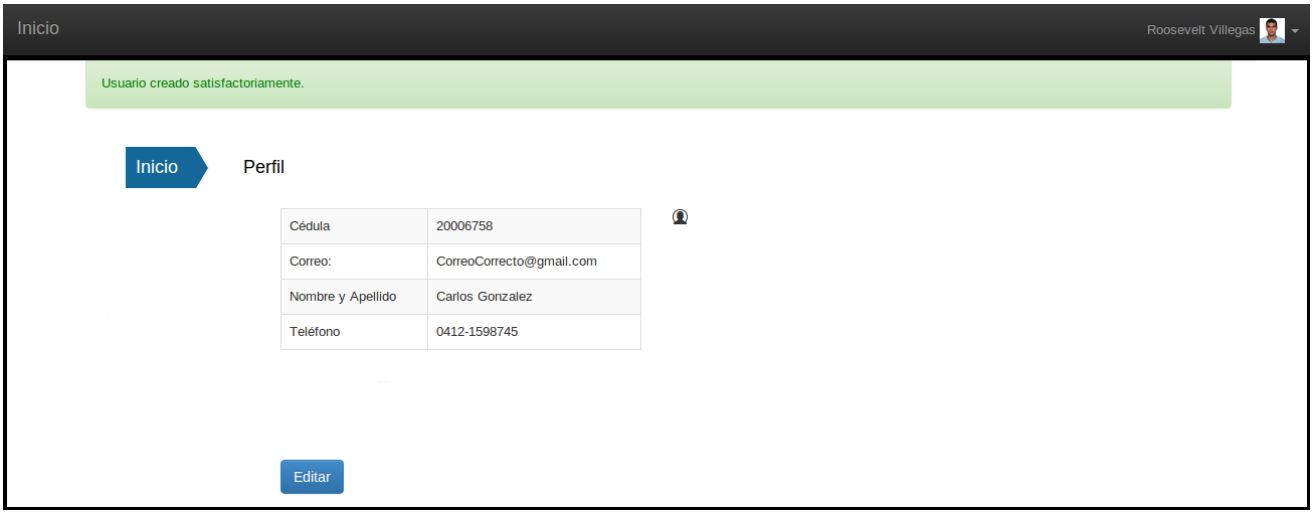


Figura 3.24: Formulario con datos correctos
Juárez y Villegas (2014)

Tabla 3.39: Prueba 5

Número de caso de prueba: 5	**Número de historia de usuario:** 5
Descripción	verificar el correcto funcionamiento de autenticación
Resultados	Con los datos incorrectos: Figura 3.25 Con los datos correctos: Figura 3.26

Figura 3.25: Inicio sesión datos incorrectos
Juárez y Villegas (2014)

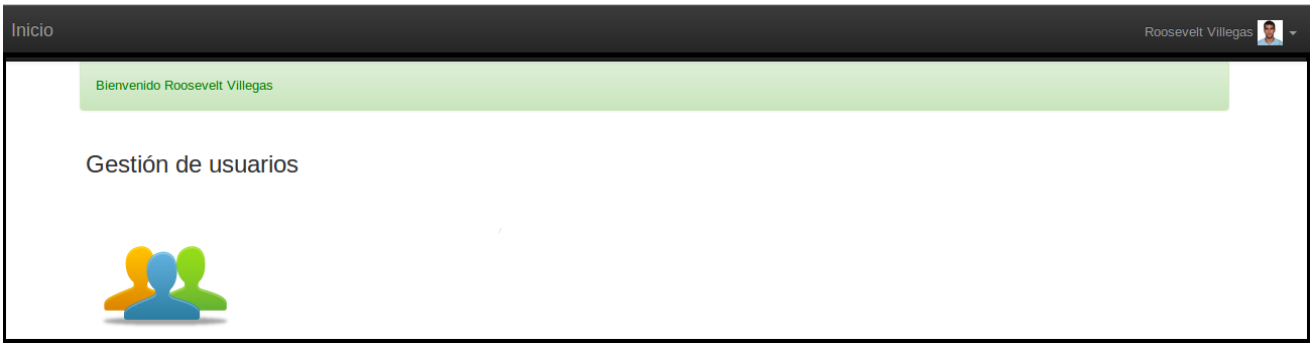


Figura 3.26: Inicio sesión datos correctos
Juárez y Villegas (2014)

3.3.4. Iteración 3

Planificación: el objetivo de esta iteración es desarrollar el módulo para ingresar equipos o insumos al inventario. Los usuarios solo estarán permitidos ingresar equipos o insumos al inventario dependiendo de los rubros a los cuales hayan sido asignados. Para dicha asignación, se crea una tabla en la base de datos para relacionar los usuarios y los rubros asignados a estos. En Tabla 3.40 se muestra la descripción de esta iteración.

Tabla 3.40: Iteración 3

Número de iteración: 3	**Número de historia**: 1, 3, 7
Caso de uso: 7	
Fecha de inicio: 01/10/2013	**Fecha fin**: 15/11/2013
Descripción: desarrollo de requerimientos funcionales de ingresar equipos o insumos al inventario e interfaz gráfica	**Tipo**: desarrollo

Diseño: uno de los requisitos, es que los usuarios no pueden ingresar todos lo rubros existentes en el sistema, es decir, que estos se vean restringidos a ingresar solo los rubros que les han sido asignados. Una vez que el usuario tenga los rubros asignados, este puede ingresar al inventario. El módulo de ingresar equipo, se desarrolló pensando en maximizar el proceso de ingreso. Ya que se puede crear una lista de equipos o insumos para ingresarlos todos de una vez, como una especie de carrito de compra virtual, donde se colocan todos los artículos y al final, cuando el usuario decida, termina el proceso. Esto con el objetivo de hacer el proceso de ingreso menos tedioso, para así no tener que ingresar un equipo o insumo, y volver a acceder al módulo de ingreso si se quiere ingresar otro.

Codificación: primero para la asignación de rubros a los usuarios, se creó una tabla en la base de datos llamada *relación_ur,* la cual contiene los id's de los usuarios de la tabla *usuario* y los id's de los rubros de la tabla *rubro,* para poder relacionarlos entre sí. Luego se modifica el formulario de creación de usuario, para poder agregarle la lista de rubros a signar al momento de crear al usuario. Esto se muestra en la Figura 3.27.

```
<div class= "form-group" >
    <%= f.label :username, "Cédula:", class: "control-label" %>
    <%= f.text_field :username, class: "form-control cedula" %>
  </div>
  <div class="form-group">
    <%= f.label :email, "Correo:", class: "control-label" %>
    <div class="controls">
      <%= f.text_field :email, class: "form-control" %>
    </div>
  </div>
<% if current_user.id == @usuario.id%>

  <div class="form-group">
    <%= f.label :password, "Contraseña:", class: "control-label" %>
    <div class="controls">
      <%= f.password_field :password, class: "form-control" %>
    </div>
  </div>
  <div class="form-group">
    <%= f.label :password_confirmation, "Confirme contraseña:", class: "control-label" %>
    <div class="controls">
      <%= f.password_field :password_confirmation, class: "form-control" %>
    </div>
  </div>
  <% f.hidden_field :salt %>

  <%else%>
  <%= f.hidden_field :password, class: "form-control clv" %>
  <%= f.hidden_field :password_confirmation, class: "form-control clvc" %>
  <%= f.hidden_field :salt %>
<%end%>
<div class="form-group field">
    <%= f.label :nombre , "Nombre:", class: "control-label"%><br>
    <%= f.text_field :nombre, class: "form-control"  %>
  </div>
  <div class="form-group field">
    <%= f.label :apellido, "Apellido:", class: "control-label" %><br>
    <%= f.text_field :apellido, class: "form-control"  %>
  </div>
  <div class="form-group field">
    <%= f.label :telefono, "Telefono:", class: "control-label" %><br>
    <%= f.text_field :telefono, class: "form-control"  %>
  </div>

    <%= f.hidden_field :salt %>

  <div class="form-group">
    <%= f.label :imagen, "Imagen: ", class: "control-label" %>
    <div class="controls">
      <%= f.file_field :imagen %>
    </div>
  </div>
  <div class="form-group">
    <%= f.label :rubro, "Rubros:", class: "control-label" %>
    <div class="controls">
      <%= f.select :rubro, options_for_select(@rubros_array.map{|s|[s.nombre, s.id]}),{},{class: "selectpicker", multiple: true} %>
    </div>
  </div>
  <% if (current_user.rango.nombre == "Administracion") or  (current_user.rango.nombre == "superUser") %>

  <div class="form-group field">
    <%= f.label :status, "¿Activo?", class: "control-label" %><br>
    <%= f.check_box :status, checked: true, class: "form-control"  %>
  </div>


  </div>
<% end %>


  <div class="actions">
    <button class="btn btn-primary" type="submit"> Guardar  </button>
    <!-- <a href="<%=usuarios_path %>"><button class="btn btn-inverse" type="button">Regresar</button></a>  -->
  </div>
```

Figura 3.27: Modificación del formulario de creación de usuarios para asignar rubros
Juárez y Villegas (2014)

En la siguiente Figura 3.28, se puede observar la lista de rubros para asignar.

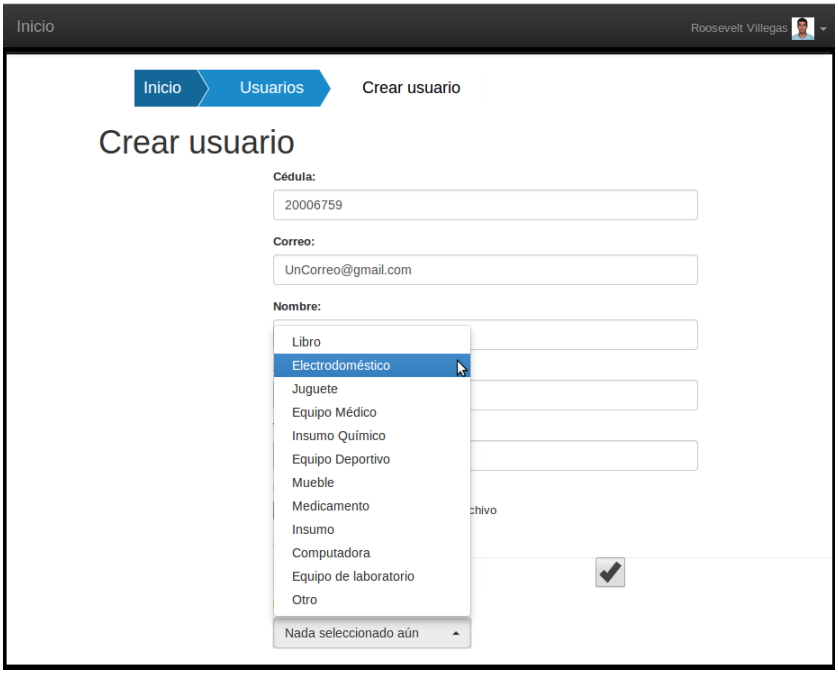


Figura 3.28: Lista de rubros
Juárez y Villegas (2014)

Una vez modificado el formulario para poder asignar los rubros, se agrega un código en el controlador de usuario **usuario_controller.rb**, para que puede hacer un query en la tabla que ingrese el id del usuario y los id's de los rubros seleccionados en el formulario. Hecho esto, ya los usuarios tienen asignados a sus cuentas, los rubros que solo pueden ingresar al inventario.

Módulo de ingreso al inventario: como ya se explicó anteriormente, el módulo de ingreso al inventario se desarrolló de manera que sea menos tedioso el proceso. Se podrá llenar una lista de equipos o insumos y cuando el usuario finalice la operación, todos estos equipos o insumos son ingresados al inventario de una vez con todos sus datos correspondientes.

Para ingresar uno o varios equipos o insumo, es necesaria una serie de datos como: el origen de los materiales (la información de la factura si fue comprado, la información del donador si fue donado, o si no tiene origen conocido) y la descripción del ingreso (opcional). Luego se procede a elegir de una lista de rubros, el que se quiere ingresar; cuando se elige el rubro, se despliega un ventana modal con un formulario, el cual contiene todos los datos necesarios para ese rubro en particular. Al llenar todos los campos obligatorios correctamente, ese equipo o insumo se agrega a una lista, y el usuario puede elegir si finalizar la operación de ingresar o elegir si agregar a la lista otro equipo o insumo.

Todo este proceso es soportado en la vista, la cual está conformado por una serie numerosa de funciones javascript (que despliegan las ventanas modal y validan los campos de estas),

llamadas asíncronas AJAX al controlador (para rellenar campos) y código HTML para los distintos formularios que corresponden a los diferentes rubros.

A continuación, una vista de como es el módulo de ingreso, la cual se puede observar en la Figura 3.29.

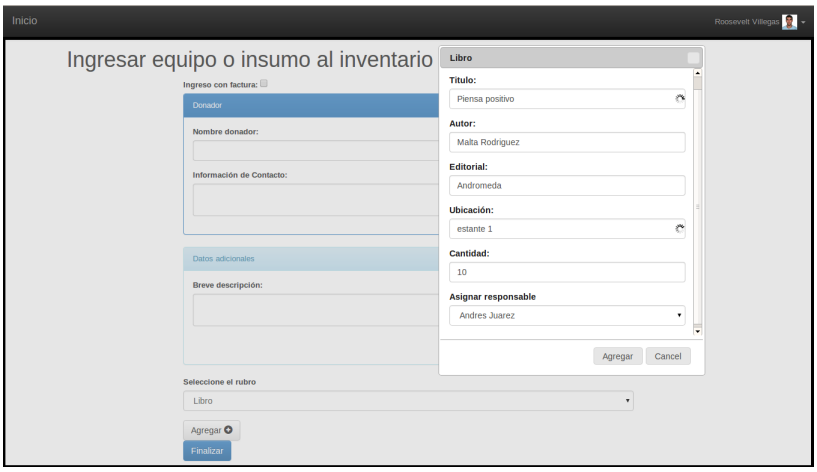


Figura 3.29: Módulo de ingreso al inventario
Juárez y Villegas (2014)

En la Figura 3.30, se puede observar un ejemplo de como luce la lista de equipos e insumos para ser ingresados al inventario.

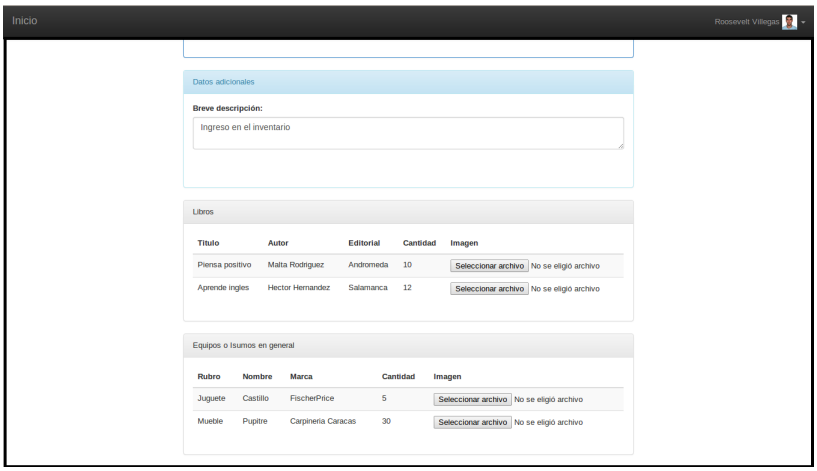


Figura 3.30: Lista de equipos o insumos para ser ingresados en el inventario
Juárez y Villegas (2014)

Una vez recabada toda la información en la vista de ingreso al inventario, esta tiene que ser procesada por el controlador, para poder ser ingresada a la base de datos. Se recibe cuatro lista desde la vista, cada una representa una tabla de los equipos e insumos de la base de datos. Se analiza cada posición de la lista, la cual contiene un equipo o insumo, para verificar

si se trata de un equipo nuevo o no. Si es un equipo nuevo, primero se ingresa la información correspondiente en las tablas ubicación, factura o donador; luego se ingresan los datos en la tabla correspondiente al equipo y por ultimo se agrega el id de dicho equipo, junto con otros datos como: el usuario que ingresa el equipo, el usuario responsable del equipo, y las fecha de ingreso en la tabla inventario, tantas veces como la cantidad ingresada en la vista para ese equipo. Si no es un equipo nuevo, simplemente se ingresa en la tabla inventarios el id de dicho equipo, junto con los datos correspondientes ya mencionados, tantas veces como la cantidad ingresada para ese equipo en la vista.

Pruebas: se realizan las pruebas funcionales de la iteración número tres.

Tabla 3.41: Prueba 6

Número de caso de prueba: 6	**Número de historia de usuario: 7**
Descripción	verificar el correcto funcionamiento de la asignación de rubros a los usuarios
Resultados	Elección de rubros: Figura 3.31 Finalización del proceso: Figura 3.32

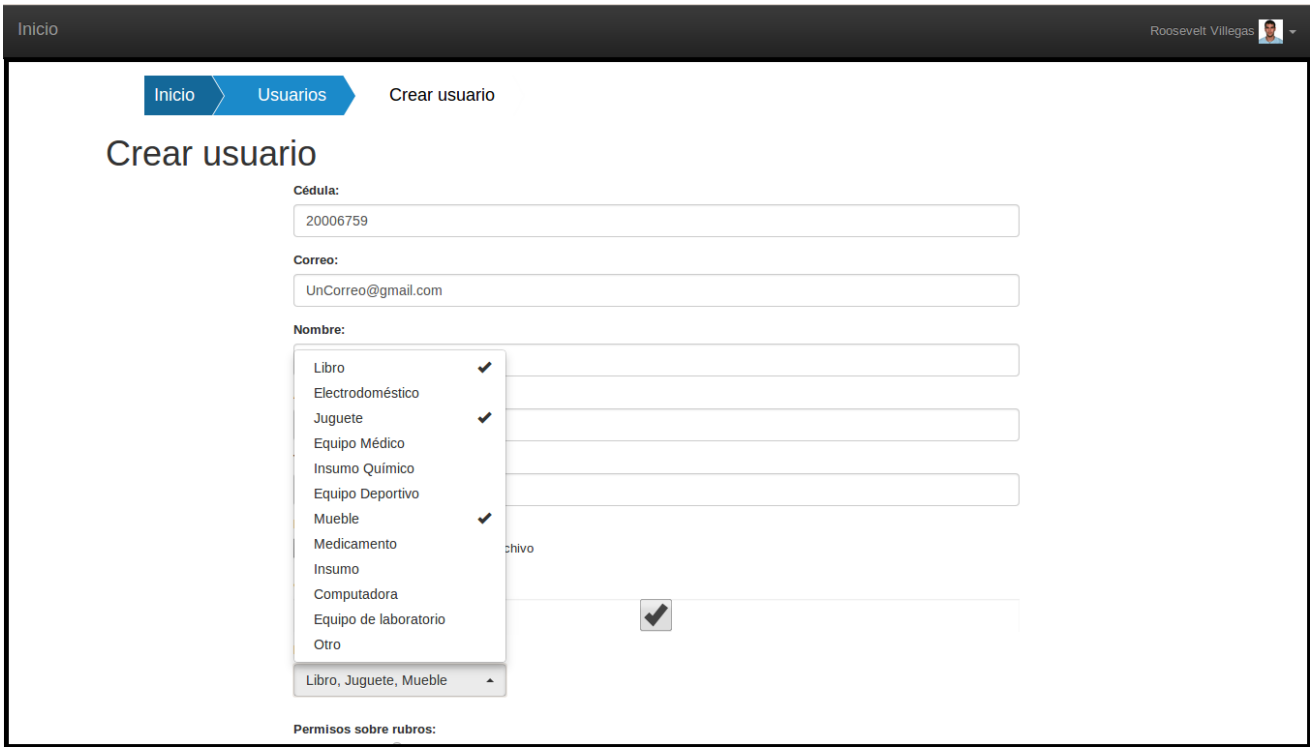


Figura 3.31: Elección de rubros
Juárez y Villegas (2014)

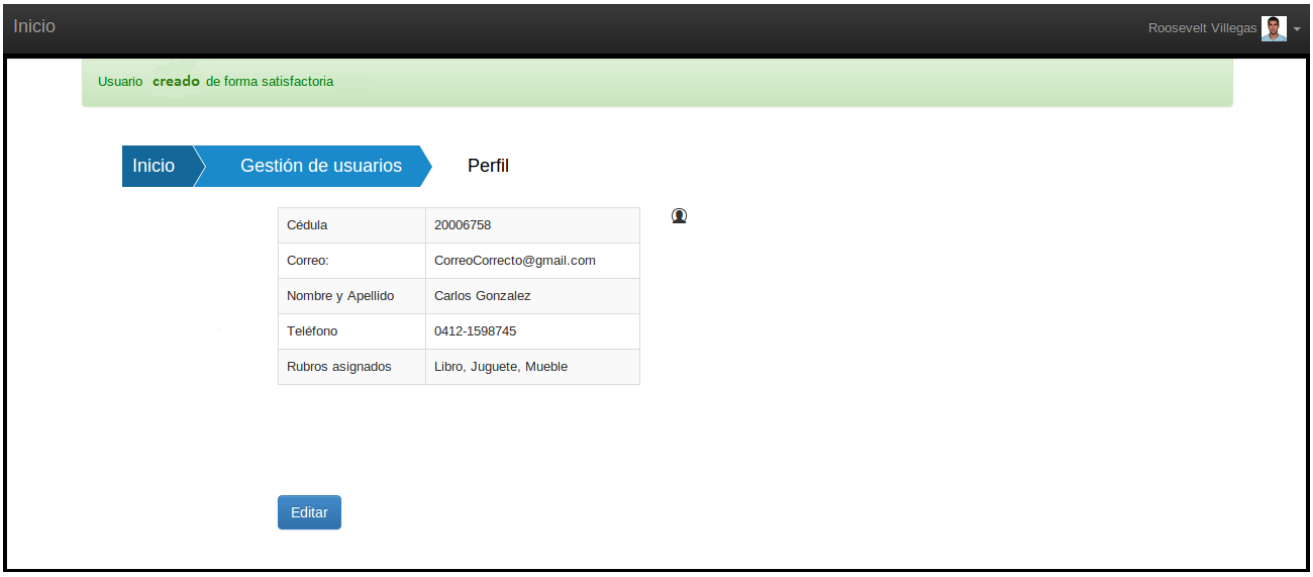


Figura 3.32: Finalización del proceso
Juárez y Villegas (2014)

Tabla 3.42: Prueba 7

Número de caso de prueba: 7	**Número de historia de usuario: 7**
Descripción	verificar el correcto funcionamiento del módulo ingresar equipos o insumos al inventario
Resultados	Con los campos incorrectos: Figura 3.33 Con los campos correctos: Figura 3.34

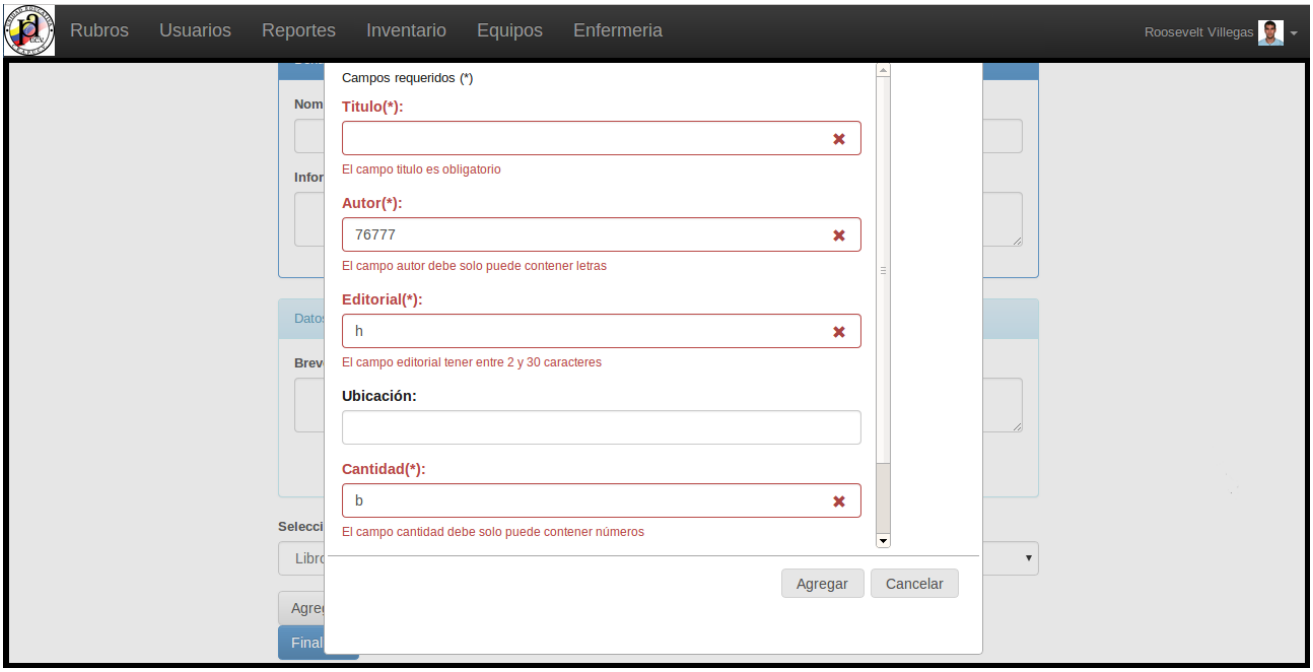


Figura 3.33: Ingreso inventario campos incorrectos
Juárez y Villegas (2014)

Figura 3.34: Ingreso inventario campos correctos
Juárez y Villegas (2014)

3.3.5. Iteración 4

Planificación: en esta iteración, resumida en la Tabla 3.43, se desarrollan las restricciones para el acceso a los diferentes módulos que conforman al sistema. Para desarrollar este requerimiento, se hizo uso de la gema "cancan", la cual crea su propio modelo, desde donde se dictan las directrices de que módulos del sistema se pueden acceder dependiendo de los permisos asignados a los usuarios.

Tabla 3.43: Iteración 4

Número de iteración: 4	**Número de historia**: 1, 3, 6
Caso de uso:	
Fecha de inicio: 18/11/2013	**Fecha fin**: 12/12/2013
Descripción: desarrollo de requerimientos funcionales de restricción a los módulos del sistema e interfaz gráfica	**Tipo**: desarrollo

Diseño: la gema cancan crea su propio modelo, llamado **ability.rb**. Básicamente, lo que hace cacan, es restringir el acceso a las funciones de los controladores de la aplicación. Para que un controlador pueda quedar bajo el control de las restricciones de cancan, se debe colocar **load_and_authorize_resource** al principio de cada controlador que se quiera restringir. De esta manera, al restringir las funciones de los controladores, al estar casadas con las vistas de la aplicación, estas también quedan bajo el control de la gema cancan.

Para poder aplicar dichas restricciones, el usuario debe tener ciertos permisos asignados, con los cuales la gema debe verificar. Por lo tanto se debe asignar a la cuenta de usuario ciertos permisos.

Desarrollar la asignación de los permisos a los usuarios requiere la creación de una tabla en la base de datos, que llamamos rango, que contiene los nombres de los permisos y relacionarla con la tabla usuario. De esta forma, ya es posible registrar los permisos a los usuarios a nivel de base de datos.

Al iniciar sesión el usuario, uno de los procesos el cual realiza Ruby on Rails, es en el modelo **ability.rb**, para verificar los permisos asignados del usuario y saber que módulos restringir y cuales no.

Codificación: primero se tiene que asignar los permisos de los usuarios al momento de su creación, o al editarlos. Para ello, agregamos una sección de permisos en la vista de creación y de edición de usuarios. Esto lo podemos observar el Figura 3.35.

```
<div class="form-group field">
  <%= f.label :status, "¿Activo?", class: "control-label" %><br>
  <%= f.check_box :status, checked: true, class: "form-control"  %>
</div>

<div class="form-group">
  <%= f.label :rubro, "Rubros:", class: "control-label" %>
  <div class="controls">
    <%= f.select :rubro, options_for_select(@rubros_array.map{|s|[s.nombre, s.id]}),{},{class: "selectpicker", multiple: true} %>
  </div>
</div>

<strong>Permisos sobre rubros:</strong>
<div class="form-group">
  <%= f.label :agreactu, "Solo consultas:", class: "control-label" %>
  <%= f.radio_button(:rango_id, "3", checked: true) %><br>

<%= f.label :agreactu2, "Ingresar y egresar:", class: "control-label" %>
  <%= f.radio_button(:rango_id, "2") %><br>

  <%= f.label :admin, "Administrador:", class: "control-label" %>
  <%= f.radio_button(:rango_id, "1")%><br>
```

Figura 3.35: Sección de permisos de usuarios
Juárez y Villegas (2014)

En la Figura 3.36, se observar la vista de la asignación de permisos a los usuarios.

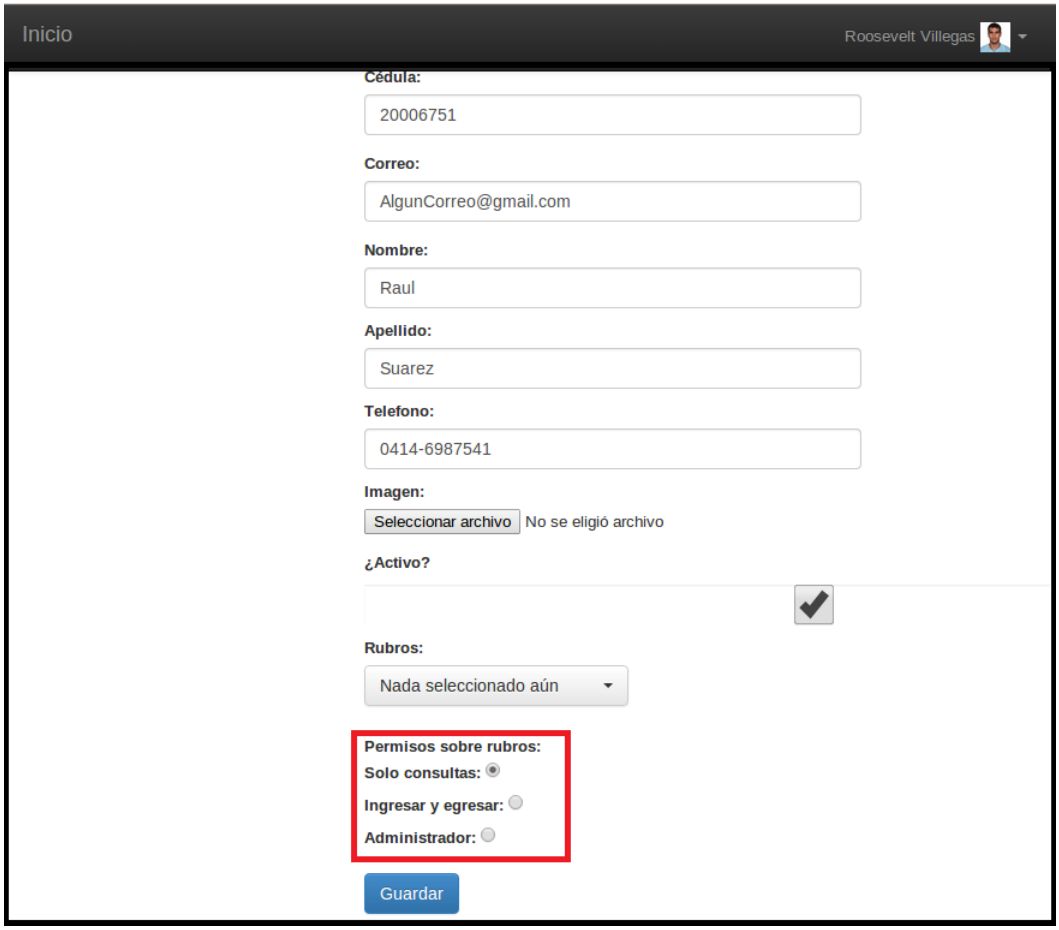


Figura 3.36: Vista de permisos de usuarios
Juárez y Villegas (2014)

Restricción de permisos: una vez que ya se puede asignar los permisos a los usuarios, se procede a restringir los accesos en el modelo **ability.rb**. Para ello, cancan ofrece dos funciones, "can" y "cannot". La función can, se usa para permitir de manera explicita las funciones que si se pueden acceder de un determinado controlador. Cannot es lo contrario, es decir, se usa para restringir de manera explicita que funciones, de cierto controlador, no se pueden acceder. En la Figura 3.37, se pueden observar las restricciones del modelo **ability.rb**.

Luego de asignar los permisos a los usuarios, y las restricciones de esos permisos a los módulos del sistema, si un usuario con cierto permiso, intenta acceder a un módulo restringido; se despliega un mensaje de "Acceso denegado" con un botón para trasladarse al inicio de la aplicación.

```
if user.rango.nombre == "administracion"

    can [:create, :update, :read,:aux,:aux2,:aux3,:auxEdit,:responsable,:ingresado],[Usuario]
    cannot [:destroy],[Usuario]

    can [],[Rango]
    cannot [:create, :destroy, :update, :index, :read],[Rango]

    can [:read, :aux],[Electronico]
    cannot [:create, :destroy, :update, :index],[Electronico]

    can [:read, :aux],[Enfermeria]
    cannot [:create, :destroy, :update, :index],[Enfermeria]

    can [:read, :aux],[Libro]
    cannot [:create, :destroy, :update, :index],[Libro]

    can [:read, :aux],[General]
    cannot [:create, :destroy, :update, :index],[General]

    can [],[Donador]
    cannot [:create, :destroy, :update, :index, :read],[Donador]

    can [],[Factura]
    cannot [:destroy, :update, :index, :read],[Factura]

    can [],[Rubro]
    cannot [:create, :create, :destroy, :update, :index, :read],[Rubro]

    can [],[Ubicacion]
    cannot [:create, :destroy, :update, :index, :read],[Ubicacion]

    can [:create, :update, :index],[Unidad]
    cannot [:destroy, :read],[Unidad]

    can [:create, :update, :index],[Presentacion]
    cannot [:destroy, :read],[Presentacion]

    can [:create, :update, :aux3, :egresar, :egresarII, :desincorporar, :autoUbicacion, :autoLibro, :autoGeneral, :
        autoEnfermeria, :autoElectronico, :opciones, :setbook, :reportes, :consulta, :imprimir],[Inventario]
    cannot [:destroy, :read, :index],[Inventario]
end
```

Figura 3.37: El modelo ability para las restricciones
Juárez y Villegas (2014)

Pruebas: se realizan las pruebas funcionales de la iteración número cuatro.

Tabla 3.44: Prueba 8

Número de caso de prueba: 8	**Número de historia de usuario: 9**
Descripción	verificar el correcto funcionamiento de las restricciones de los módulos
Resultados	Un usuario con permiso de solo ver, es decir, que no puede ingresar al módulo de gestión de usuarios; intentará acceder a dicho módulo Se verifica que el usuario Carlos Gonzalez tiene permiso de solo consulta 3.38 Se inicia sesión con ese usuario: 3.39 Se intenta acceder a través de la URL al módulo Usuarios: 3.40

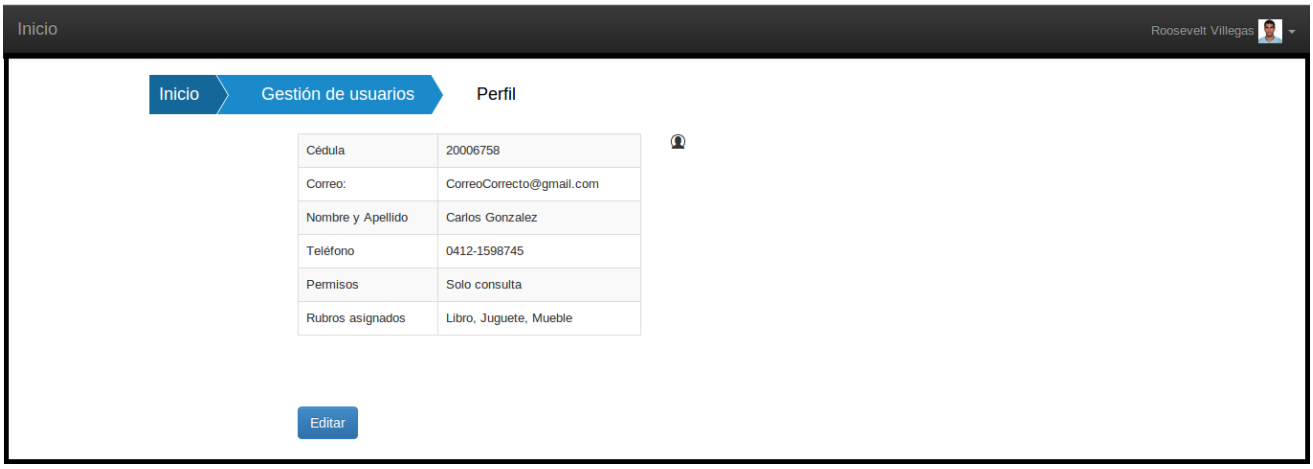


Figura 3.38: Permiso de usuario
Juárez y Villegas (2014)

Figura 3.39: Inicio de sesión del usuario
Juárez y Villegas (2014)

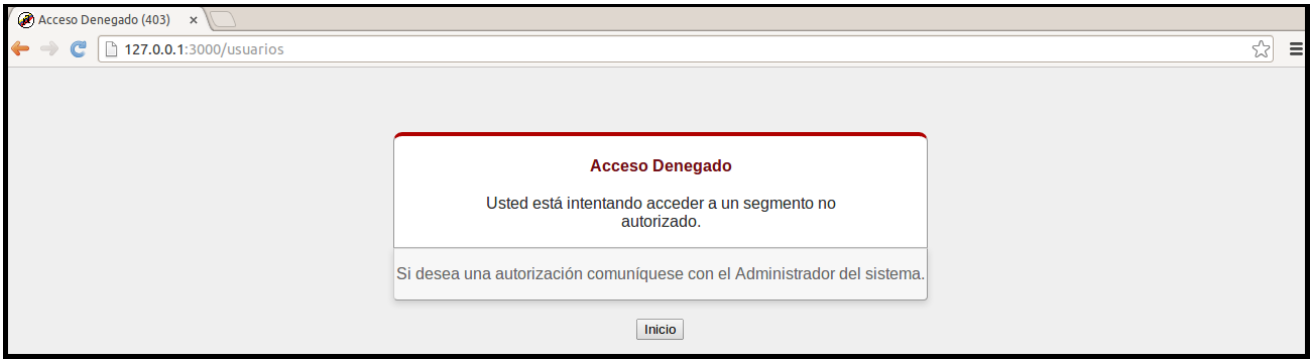


Figura 3.40: Acceso denegado
Juárez y Villegas (2014)

3.3.6. Iteración 5

Planificación: el objetivo de esta iteración es desarrollar el requerimiento funcional de egresar equipos o insumos del inventario. Esta iteración se describe en la Tabla 3.45.

Tabla 3.45: Iteración 5

Número de iteración: 5	**Número de historia**: 1, 3, 8
Caso de uso: 8	
Fecha de inicio: 14/01/2014	**Fecha fin**: 30/01/2014
Descripción: desarrollo de requerimientos funcionales de egresar equipos o insumos del inventario e interfaz gráfica	**Tipo**: desarrollo

Diseño: para el desarrollo de este módulo, se crea una vista con una lista de equipos e insumos. Dicha lista, al igual que en el módulo de ingresar, esta restringida por los rubros asignados al

usuario logueado. Esto quiere decir, que en la lista solo se desplegarán los rubros, los cuales fueron asignados previamente al usuario. Cada entrada de la lista tendrá un símbolo de egresar, el cual llevará a una vista más detallada del equipo que se quiere egresar, donde se puede especificar la cantidad a egresar.

Luego se crea la función en el controlador **inventario_controller.rb** que recibe los datos enviados por la vista, y coloca tanto la fecha de salida, como el usuario que egresó el equipo o insumo en las entradas correspondientes en la tabla inventarios de la base de datos.

Codificación: primero se crea la vista que despliega la lista de equipos.

En la Figura 3.41, se puede observar la vista de la lista de equipos o insumos con el símbolo y la palabra egresar.

Figura 3.41: Vista de equipos o insumos a egresar
Juárez y Villegas (2014)

Luego al hacer click en el símbolo de egresar, se redirige a una vista mas detallada del equipo que se quiere egresar. Dicha vista, está compuesta por un formulario que contiene los campos de los detalles del equipo. El formulario se puede observar en la Figura 3.42.

```
<%= form_tag({controller: "inventarios", action: "egresarII"}, method: "get", id:"form") do%>

    <div class="form-group field">
        <%= label_tag nil, "Nombre:",class: "control-label" %><br>
        <%= text_field_tag :nombre, "#{@rubro.nombre}", class: "form-control",disabled: true  %>
    </div>
        <div class="form-group field">
        <%= label_tag nil, "Unidad:",class: "control-label" %><br>
        <%= text_field_tag :unidad, "#{@rubro.unidad}", class: "form-control",disabled: true  %>
    </div>
    <div class="form-group field">
        <%= label_tag nil, "Cantidad en existencia:",class: "control-label" %><br>
        <%= text_field_tag :cantidadE, "#{@b}", class: "form-control",disabled: true  %>
    </div>
        <div class="form-group field">
        <%= label_tag nil, "Descripción o nota del egreso:",class: "control-label" %><br>
        <%= text_area_tag :descripcion, nil, class: "form-control"  %>
    </div>
    <div class="form-group field">
        <%= label_tag nil, "Cantidad a egresar:",class: "control-label" %><br>
        <%= text_field_tag :cantidad, nil, class: "form-control"  %>
        <%= hidden_field_tag :rubro, "en"%>
        <%= hidden_field_tag :id, "#{@rubro.id}"%>
        <%= hidden_field_tag :fechaOut, DateTime.current().to_formatted_s(:db)%>
    </div>
    <button id="sum" class="btn btn-primary" type="submit"> Egresar  </button>
<% end %>
```

Figura 3.42: Formulario para egresar un equipo del inventario
Juárez y Villegas (2014)

En la siguiente Figura 3.43, se puede observar la vista del formulario para egresar equipos del inventario.

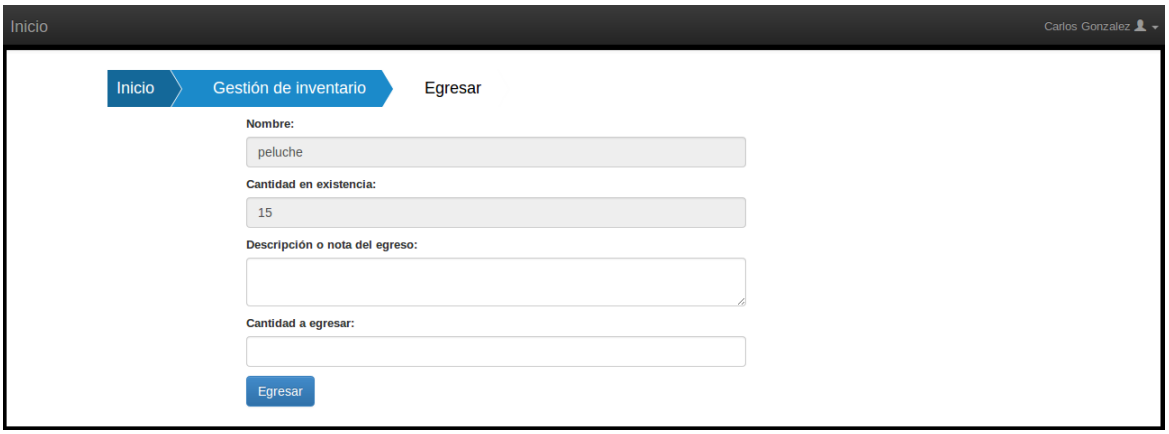


Figura 3.43: Vista formulario para egresar un equipo del inventario
Juárez y Villegas (2014)

Para el desarrollo del controlador, se creó una función, la cual consiste en recibir la fecha actual, el id del equipo y la descripción del egreso. Luego se procede a ejecutar un query en la base de datos para modificar las entradas correspondientes de la tabla inventario que concuerden con los datos. La modificación consiste en agregar la fecha de salida y el usuario que egreso el equipo, con lo cual se considera fuera del inventario. Dicha función de observa en la Figura 3.44.

```
fechaOut = params[:fechaOut]
id = params[:id]
cantidad = params[:cantidad]

current_date_db = ActiveRecord::Base.connection.execute("SELECT NOW() as fecha")
current_date_db = current_date_db.first
current_date_db = current_date_db.to_s

if params[:descripcion] == ""
  descripcionDes = " "

else
  descripcionDes = "#{params[:descripcion]}"

end
# render text: descripcion_desincorporar
if params[:rubro] == "li"
for i in 1..cantidad.to_i
  inventario = Inventario.find_by "libro_id = ? and fechaOut is null ", id
  inventario.update(usuarioOut_id: current_user.id, fechaOut: current_date_db, descripcion_desincorporar: descripcionDes)
end

elsif params[:rubro] == "en"
for i in 1..cantidad.to_i
  inventario = Inventario.find_by "enfermeria_id = ? and fechaOut is null ", id
  inventario.update(usuarioOut_id: current_user.id, fechaOut: current_date_db, descripcion_desincorporar: descripcionDes)
end
elsif params[:rubro] == "ge"
for i in 1..cantidad.to_i
  inventario = Inventario.find_by "general_id = ? and fechaOut is null ", id
  inventario.update(usuarioOut_id: current_user.id, fechaOut: current_date_db, descripcion_desincorporar: descripcionDes)
end
else
for i in 1..cantidad.to_i
  inventario = Inventario.find_by "electronico_id = ? and fechaOut is null ", id
  inventario.update(usuarioOut_id: current_user.id, fechaOut: current_date_db, descripcion_desincorporar: descripcionDes)
end
end


flash[:notice] = "El equipo o insumo se egresó con éxito"
```

Figura 3.44: Función egresar del inventario
Juárez y Villegas (2014)

Pruebas: se realizan las pruebas funcionales de la iteración número cinco.

Tabla 3.46: Prueba 9

Número de caso de prueba: 9	**Número de historia de usuario: 8**
Descripción	verificar el correcto funcionamiento de módulo egresar del inventario
Resultados	Rubros asignados al usuario: 3.45 Lista de equipos que el usuario solo puede egresar: 3.46 Egresar cierta cantidad de un equipo: 3.47 Ver reflejada la cantidad egresada en la lista: 3.48

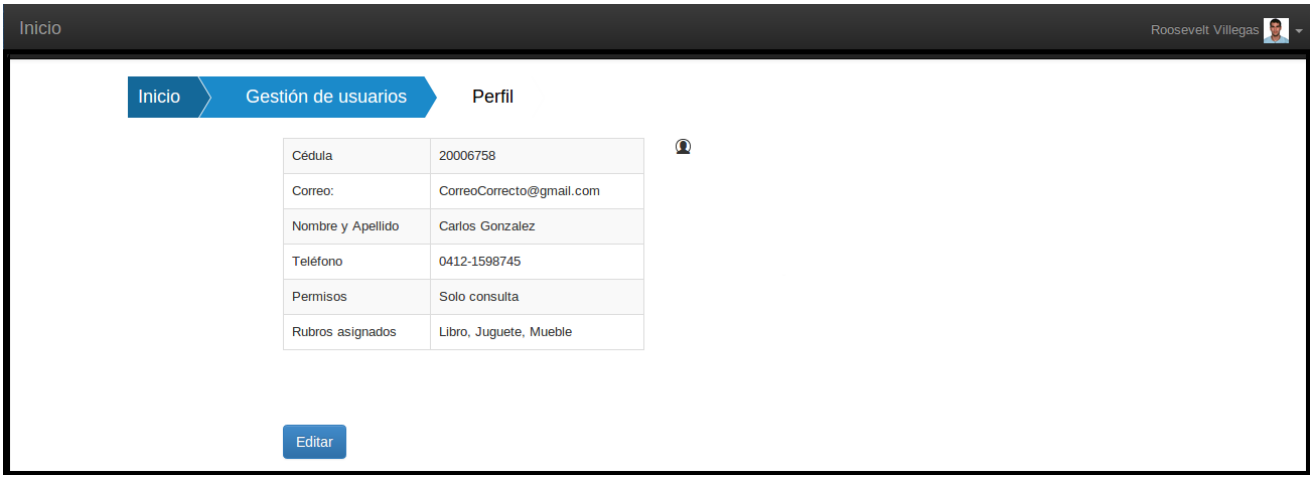


Figura 3.45: Rubros asignados
Juárez y Villegas (2014)

Figura 3.46: Lista de equipos a egresar
Juárez y Villegas (2014)

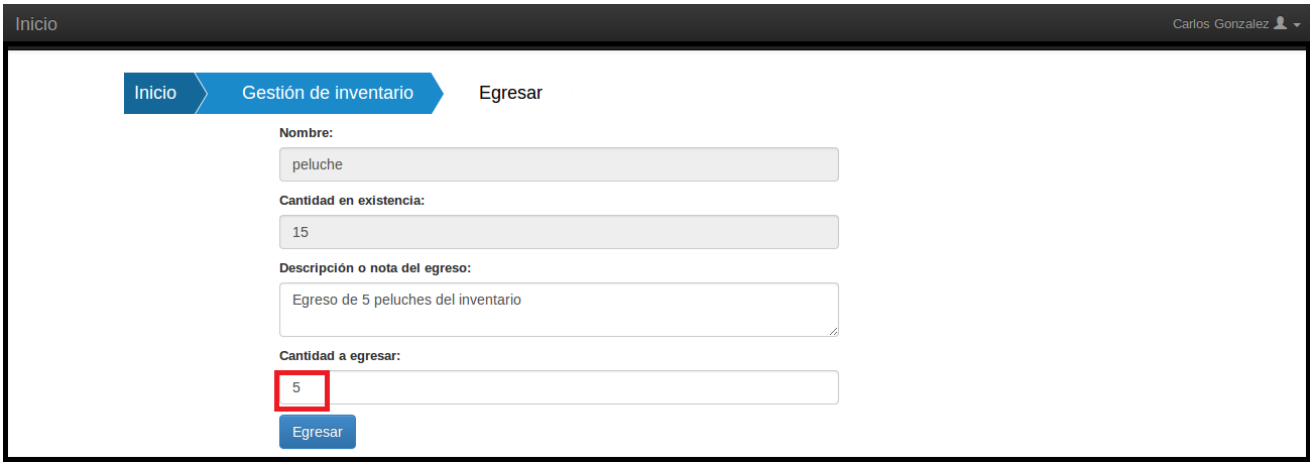


Figura 3.47: Egreso de un cantidad
Juárez y Villegas (2014)

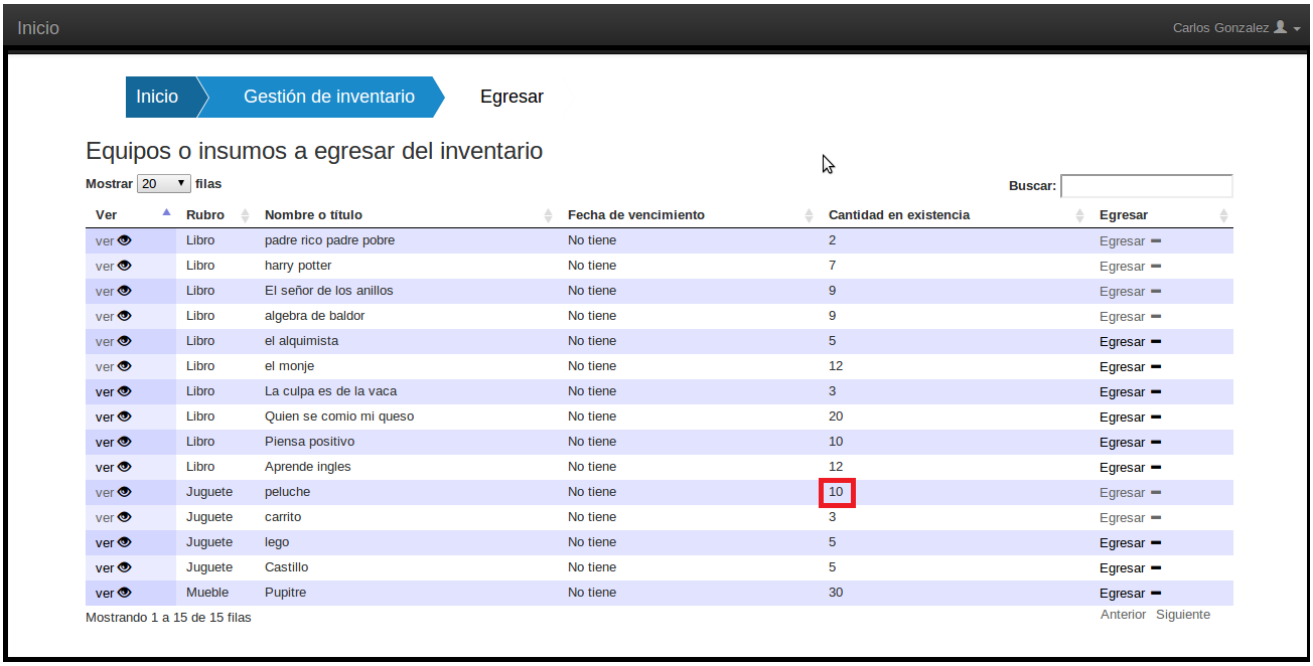


Figura 3.48: Reflejo del egreso
Juárez y Villegas (2014)

3.3.7. Iteración 6

Planificación: en esta iteración se desarrolla el requerimiento funcional de generar el código QR de los equipos e insumos del inventario. Para la codificación del QR, se hace uso de la gema "rqrcode", la cual construye una tabla HTML a partir de cierta información para representar el código. Para la generación, se hace uso de la gema "wicked_pdf", la cual toma la tabla del código QR y generar un archivo en formato PDF. Y de esta manera poder tener la opción de impresión disponible. Esta iteración se encuentra resumida en la Tabla 3.47.

Tabla 3.47: Iteración 6

Número de iteración: 6	**Número de historia**: 1, 3, 9
Caso de uso: 5 y 6	
Fecha de inicio: 03/02/2014	**Fecha fin**: 14/02/2014
Descripción: desarrollo del requerimiento funcional de generar el código QR de los equipos o insumos y la interfaz gráfica del sistema	**Tipo**: desarrollo y diseño

Diseño: la estructura MVC de Ruby on Rails divide, por así decirlo, la aplicación en modelo, vista y controlador, ya explicados con anterioridad en la sección 2.3.2. En Rails, cada controlador, el cual representa un recurso del sistema, se conforma de varias vistas, las cuales pueden ser: el index (lista de todos los recursos que representa la vista), new (formulario para crear un nuevo recurso), edit (editar el recurso) y show (para desplegar todos los detalles de dicho recurso).

En la aplicación, los equipos e insumos representan esos recursos. Por tal motivo, se decidió desplegar el código QR de cada equipo e insumos en su respectiva vista "show", junto con la opción de generarlo en formato PDF.

Codificación:

La gema rqrcode posee una función, que al pasarle un string, y ciertas opciones como el tamaño del cuadro y el nivel de corrección de errores, explicado en la sección 2.1.2, codifica toda esta información para generar en una tabla html en la vista. Esta función que codifica la información, se utiliza en la función show del controlador, para poder desplegar en la vista.

En la Figura 3.49, se puede observar la función de codificación utilizada en la función show del controlador.

```
def show
  @lu = $lu
  $lu = ""
@b = Inventario.where("enfermeria_id = ? and fechaOut is null", params[:id]).count

enf = Enfermeria.find(params[:id])
@presentacion = Presentacion.find(enf.presentacion_id)
@unidad = Unidad.find(enf.unidad_id)
url = "127.0.0.1:3000/enfermerias/#{params[:id]}"

@qr = RQRCode::QRCode.new( "Nombre: #{enf.nombre}\nNombre genérico: #{enf.nombreGenerico}\nPresentación: #{@presentacion.nombre}
  \nUnidad: #{@unidad.nombre}\nMás info: http://#{url}", size: 12, :level => :h)
end
```

Figura 3.49: Función de codificación del código QR
Juárez y Villegas (2014)

Luego en la vista show, se coloca la tabla y el código CSS necesario para desplegar el código QR. Esto se puede observar en la Figura 3.50.

```
<div class="col-md-4">
    #tableqr {
      border-width: 0;
      border-style: none;
      border-color: #0000ff;
      border-collapse: collapse;
    }
    td {
      border-width: 0;
      border-style: none;
      border-color: #0000ff;
      border-collapse: collapse;
      padding: 0;
      margin: 0;
      width: 5px;
      height: 5px;
    }
    td.black { background-color: #000; }
    td.white { background-color: #fff; }
    #tableqr {height: 300px;}

    <table id="tableqr">
    <% @qr.modules.each_index do |x| %>
      <tr>
      <% @qr.modules.each_index do |y| %>
       <% if @qr.dark?(x,y) %>
        <td class="black"/>
       <% else %>
        <td class="white"/>
       <% end %>
      <% end %>
      </tr>
    <% end %>
    </table>
  </div>
```

Figura 3.50: Tabla para desplegar el código QR
Juárez y Villegas (2014)

En la siguiente Figura 3.51, se puede observar un ejemplo de como se despliega el código QR.

Figura 3.51: Despliegue del código Qr
Juárez y Villegas (2014)

Una vez desplegado el código QR, se procede con la opción de generarlo. Para ello se coloca un formulario, que solicita la cantidad de etiquetas y el tamaño de cada etiqueta QR. Esto para desplegarlas en formato PDF.

Se procede a desarrollar la función para desplegar el pdf dentro del controlador de los equipos e insumos. Esta función recibe por parámetros el id del equipo, la cantidad, el tamaño de las etiquetas y luego con ello genera la tabla del código QR. Esta tabla se pasa por parámetro a la función de la gema 'wicked_pdf', y así proceder a generar el documento PDF. La función se puede observar en la Figura 3.52.

```
def imprimir
  elec = Electronico.find(params[:id])
  @cantidad = params[:cantidad]
  tamano = params[:tamano]
  #url = "127.0.0.1:3000/electronicos/#{params[:id]}"
  url = "inventarioceapucv.dlinkddns.com/electronicos/#{params[:id]}"
  @qr = RQRCode::QRCode.new( "Nombre: #{elec.nombre}\nMarca: #{elec.marca}\nModelo: #{elec.modelo}\nMás info:
    http://#{url}", size: 10, :level => :h)

  @my_html = "<table class='tableqr tableqr#{tamano}'>"
  @qr.modules.each_index do |x|
    @my_html += "<tr>"
    @qr.modules.each_index do |y|
      if @qr.dark?(x,y)
        @my_html += "<td class='black'/>"
      else
        @my_html +="<td class='white'/>"
      end
    end
    @my_html += "</tr>"
  end
  @my_html +="</table>"

  @titulo_reporte = "Código QR"
  nombre_archivo = @titulo_reporte+"_"+Time.now().to_s
  titulo_archivo = "--title codigo QR"
  respond_to do |format|
    format.html do
    render :pdf           => nombre_archivo,
           :template      => 'libros/imprimir.html.erb',
           :layout        => 'pdf.html' # use 'pdf.html' for a pdf.html.erb file
    end
    format.pdf do
    render :pdf           => nombre_archivo,
           :template      => 'libros/imprimir.html.erb',
           :layout        => 'pdf.html' # use 'pdf.html' for a pdf.html.erb file
    end
```

Figura 3.52: Función para desplegar PDF
Juárez y Villegas (2014)

pruebas: se realizan las pruebas funcionales de la iteración número seis.

Tabla 3.48: Prueba 10

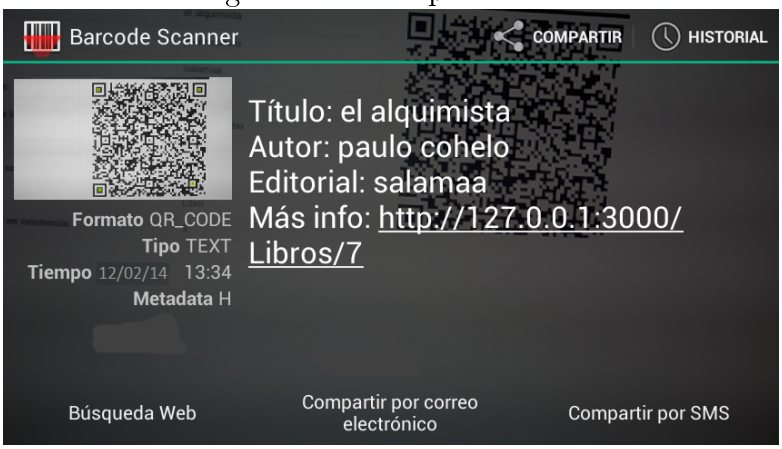
Número de caso de prueba: 10	**Número de historia de usuario: 9**
Descripción	verificar el correcto funcionamiento del despliegue del código QR
Resultados	Código QR de un libro:  Lectura del código desde una aplicación móvil: 

Tabla 3.49: Prueba 11

Número de caso de prueba: 11	**Número de historia de usuario: 9**
Descripción	verificar el correcto funcionamiento del documento PDF que contiene el código QR
Resultados	Código QR de un electrodoméstico: 3.53 Despliegue del código en formato PDF: 3.54

Figura 3.53: Código QR
Juárez y Villegas (2014)

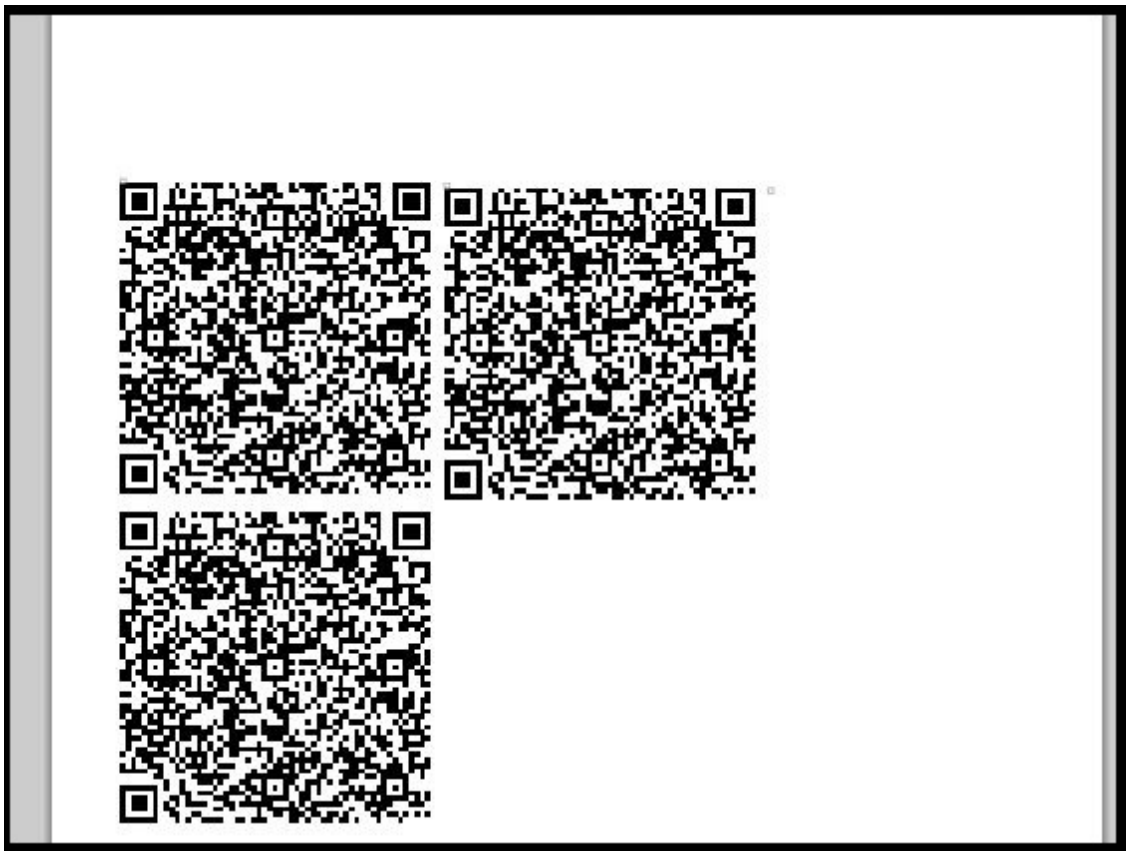


Figura 3.54: Formato PDF
Juárez y Villegas (2014)

3.3.8. iteración 7

Planificación: en esta iteración, descrita en la Tabla 3.50, se desarrollan los requisitos funcionales de consultar del inventario y generar el reporte de una consulta hecha. Para poder generar el reporte en formato PDF, se hizo uso de la gema "wicked_pdf".

Tabla 3.50: Iteración 7

Número de iteración: 7	**Número de historia**: 1, 3, 10, 11
Caso de uso: 9 y 10	
Fecha de inicio: 03/03/2014	**Fecha fin**: 21/03/2014
Descripción: desarrollo del requerimiento funcional de consultar y generar reportes del inventario y la interfaz gráfica del sistema	**Tipo**: desarrollo

Diseño: en primer lugar se desarrolla el requerimiento funcional de consultar el inventario. Para ello se crea la vista que contiene el formulario, desde el cual se seleccionan los parámetros a consultar del inventario, como lo pueden ser: el tipo de consulta (por usuario, por ubicación, por equipos, etc), los rubros a consultar, desde una fecha en específica, hasta una fecha específica, si la consulta solo es de ingresos de los equipos, de egreso o ambos.

Luego se desarrolla una función en el controlador **inventario_controller.rb**, la cual recibe los parámetros enviados por el formulario de la vista, y procesa toda la información; haciendo todas las validaciones necesarias, para así consultar la tabla inventarios de la base de datos. Una vez se obtienen los resultados de la consulta de la base de datos, estos se estructuran dentro de una tabla html para ser enviados de nuevo a la vista del formulario, mediante una función AJAX, lo que permite modificar la estructura html de la vista sin tener que recargar la página del navegador. Y de esta forma se obtienen los resultados de la consulta del inventario.

Una vez hecha la consulta al inventario, se tiene la opción de generar dicha consulta mediante un botón en la vista de consultar. Para generar el reporte, se desarrolla otra función en el controlador del inventario, la cual almacena mediante variables globales y de manera paralela, la tabla html resultado de la función que consulta la tabla de base de datos del inventario. Luego esta tabla se despliega con unan función de la gema 'wicked_pdf' en una ventana con un formato PDF, y todas la opciones que puede ofrecer un documento de ese tipo, como puede ser imprimirlo.

Codificación:

Para crear la vista del módulo consultar, se desarrolló un formulario, el cual despliega todos los campos necesarios para realizar la consulta. Este formulario se puede observar en la Figura 3.55.

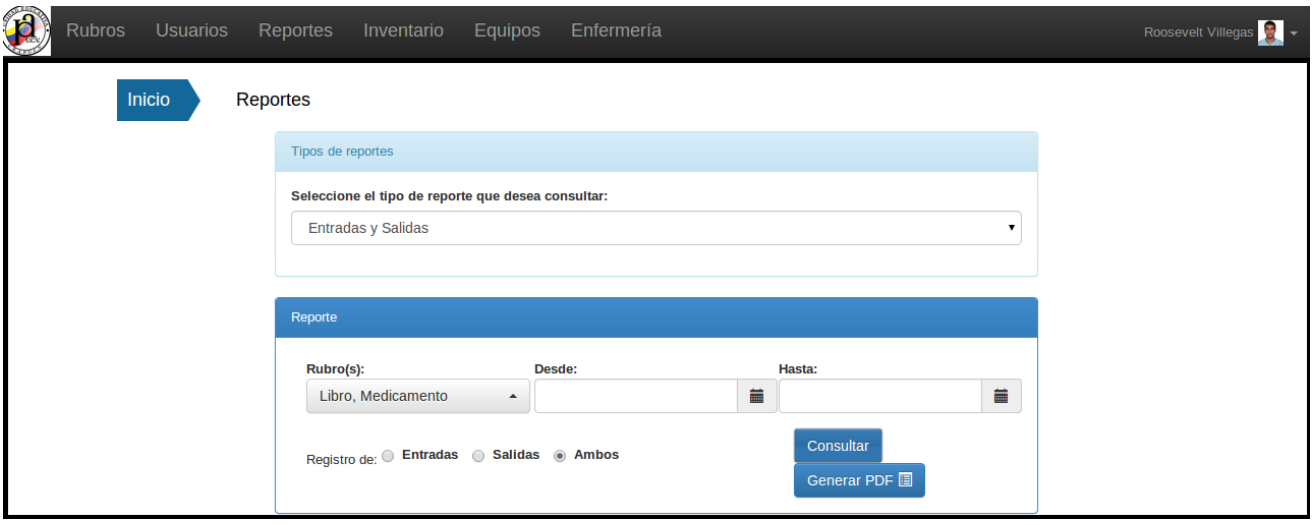


Figura 3.55: Vista del formulario consultar inventario
Juárez y Villegas (2014)

Luego se desarrolla la función dentro del controlador inventario, que recibe todos los parámetros del formulario de la vista de consultar. Parte de dicha función se puede observar en la Figura 3.56.

```
def consulta
  $html_response = ""
  debug = ""
  rubro = params["rubro"]
  desde = params["fecha_desde"]
  hasta = params["fecha_hasta"]
  registro = params["registro"]
  cant_entradas = 0
  cant_salidas = 0
  nombre_producto = ""

  cantidad_rubros = rubro.length
  cantidad_rubros = cantidad_rubros.to_i

  for i in 0..cantidad_rubros-1

    if(rubro[i] == "1")
      # Libros
      $html_response += "<div class='panel panel-success'>"

      nombre_rubro = Rubro.where(id: rubro[i])
      nombre_rubro = nombre_rubro[0][:nombre]
      $html_response += "<div class='panel-heading'>#{nombre_rubro}</div>"

      # creando la tabla
      $html_response += "<table class='table table-striped table-hover'> "

      # creando los headers de la tabla
      $html_response += "<thead> <tr> <th>Nombre</th>"
      if(registro == "Entradas" || registro == "Ambos")
        $html_response += "<th>Total entradas</th>"
      end
      if(registro == "Salidas" || registro == "Ambos")
        $html_response += "<th>Total salidas</th>"
      end
      $html_response+= "</tr> </thead>"
```

Figura 3.56: Función de consultar inventario
Juárez y Villegas (2014)

Para poder generar los reportes hechos en una consulta previa, se desarrolla un función dentro del controlador inventario. Esta función obtiene de variables globales el resultado realizado en la función consultar, la cual se encuentra dentro del mismo controlador. De esta forma se obtiene el reporte hecho anteriormente, para luego ser desplegado, mediante una función de la gema 'wicked_pdf', en una ventana con un formato PDF. La función de reportes se puede observar en la Figura 3.57.

```
def imprimir
  time1 = Time.new

  header1 = wicked_pdf_image_tag_for_public('ceapucv.jpg', :width=>'150')
  header1 = header1.to_s
  header = "#{header1} <h1>Reporte</h1>"
  footer = "<br><br><br> <table class='table table-bordered'><tbody><tr><td><strong>Reporte generado:</strong></td> <td>#{
    time1.day}/#{time1.month}/#{time1.year} </td></tr><tr><td><strong>Quien entrega:</strong></td> <td></td></tr><tr><td><
    strong>Quien recibe:</strong></td> <td></td></tr><tr><td><strong>Fecha de recepción:</strong></td> <td></td></tr></
    tbody></table>"
  html_print = header+$html_response+footer
  # render :text => html_print and return
  @my_html = html_print
  @titulo_reporte = "Reporte de Inventario"
  nombre_archivo = @titulo_reporte+"_"+Time.now().to_s
  titulo_archivo = "--title Reporte"
  respond_to do |format|
    format.html do
    render :pdf           => nombre_archivo,
           :template      => 'inventarios/imprimir.html.erb',
           :layout        => 'pdf.html' # use 'pdf.html' for a pdf.html.erb file
    # #format.html { redirect_back_or_to homes_path, notice: "Bienvenido #{current_user.nombre} #{current_user.apellido}" }
    end
    format.pdf do
    render :pdf           => nombre_archivo,
           :template      => 'inventarios/imprimir.html.erb',
           :layout        => 'pdf.html' # use 'pdf.html' for a pdf.html.erb file
    end
  end
end
```

Figura 3.57: Función de generar reporte
Juárez y Villegas (2014)

pruebas: se realizan las pruebas funcionales de la iteración número siete.

Tabla 3.51: Prueba 12

Número de caso de prueba: 12	**Número de historia de usuario: 10**
Descripción	verificar el correcto funcionamiento del módulo consultar el inventario
Resultados	Realizar una consulta: Figura 3.58 Resultado de la consulta: Figura 3.59

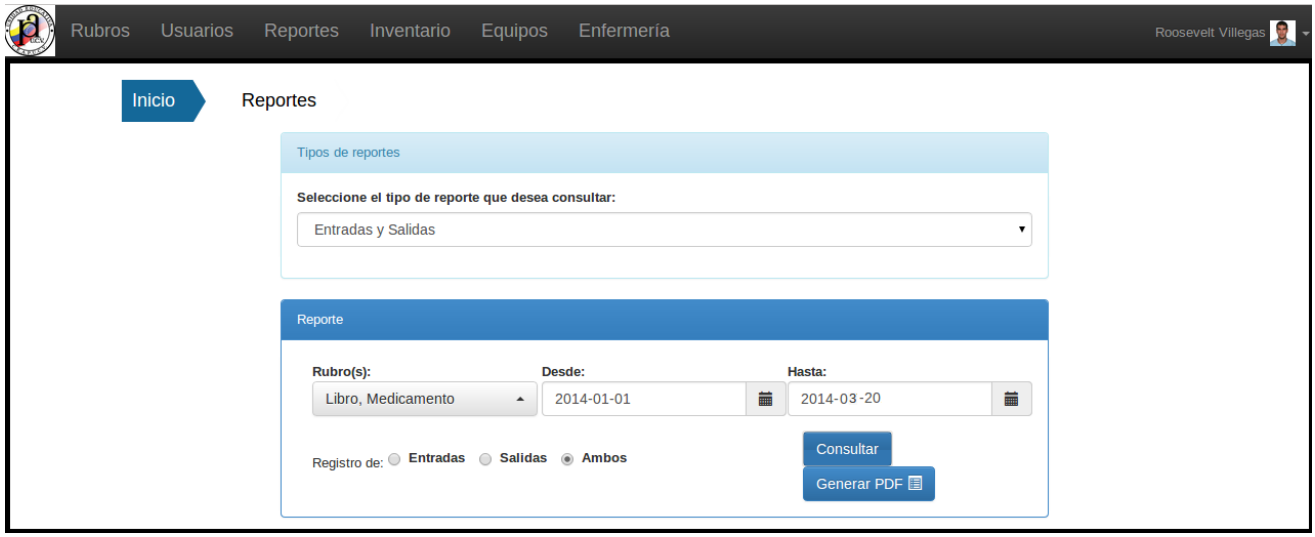


Figura 3.58: Realizar consulta
Juárez y Villegas (2014)

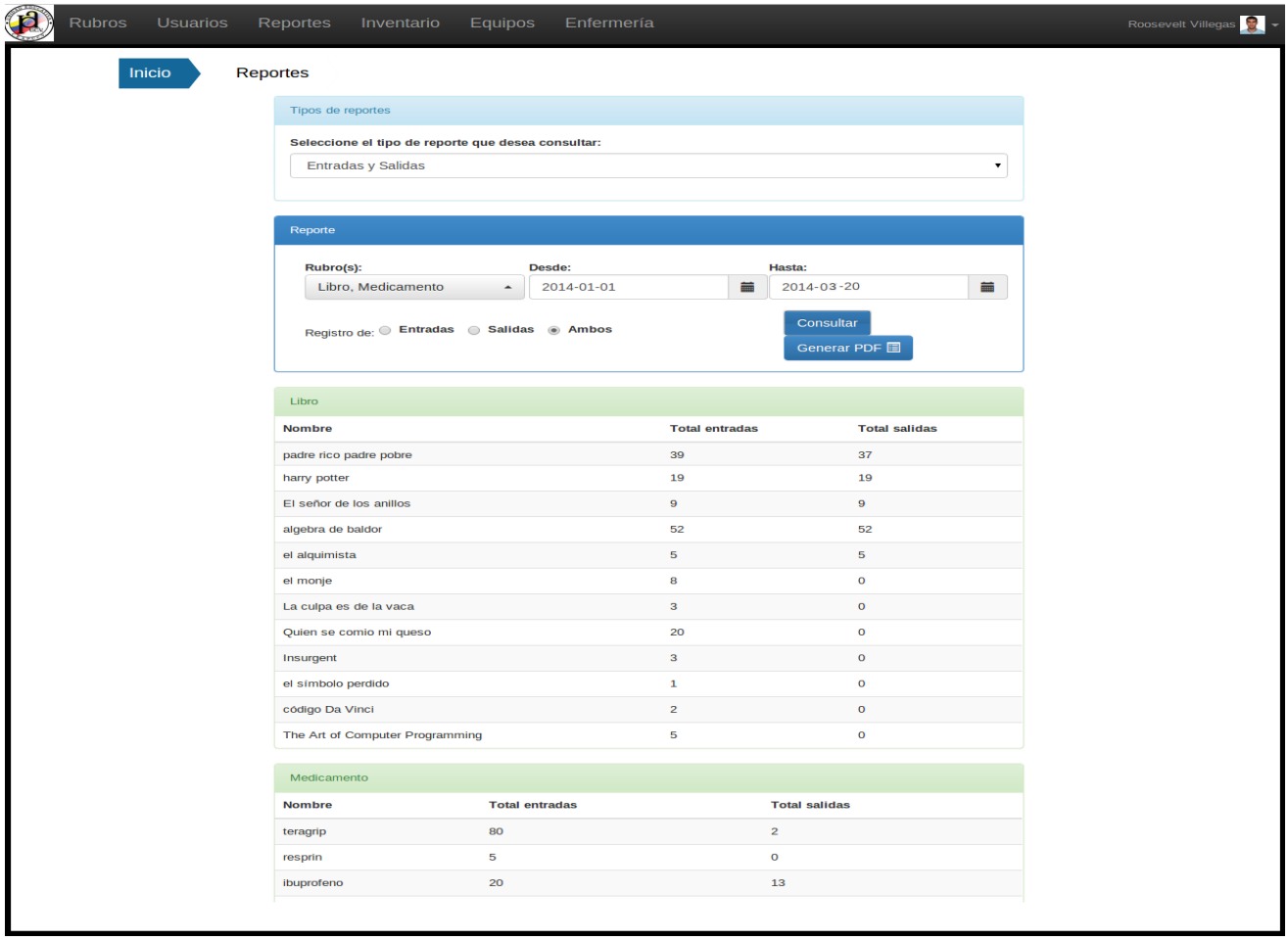


Figura 3.59: Resultado de la consulta
Juárez y Villegas (2014)

Tabla 3.52: Prueba 13

Número de caso de prueba: 13	**Número de historia de usuario: 11**
Descripción	verificar el correcto funcionamiento de generar reportes
Resultados	Una consulta realizada: Figura 3.60 El reporte generado de dicha consulta: Figura 3.61

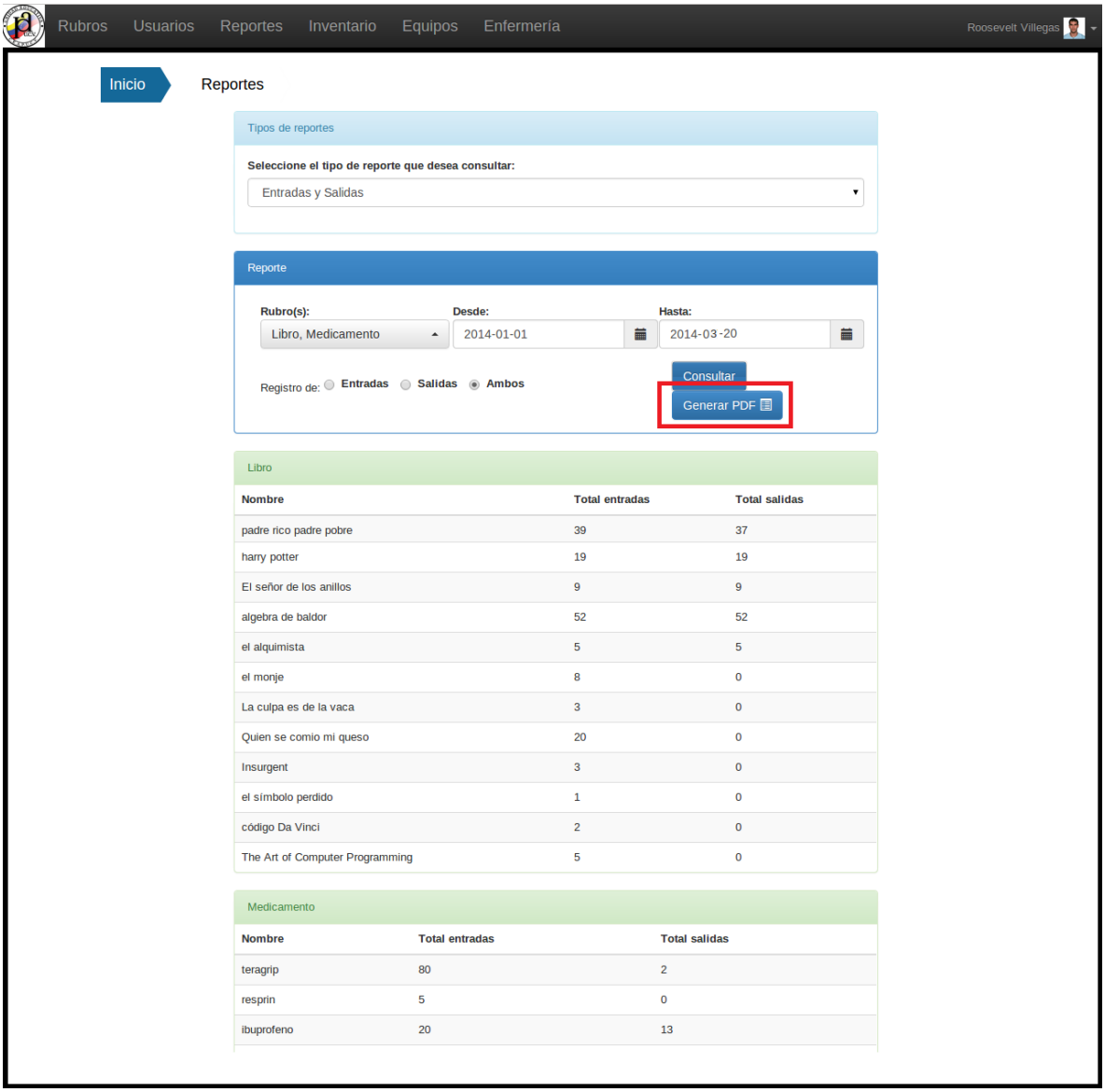


Figura 3.60: Consulta realizada
Juárez y Villegas (2014)

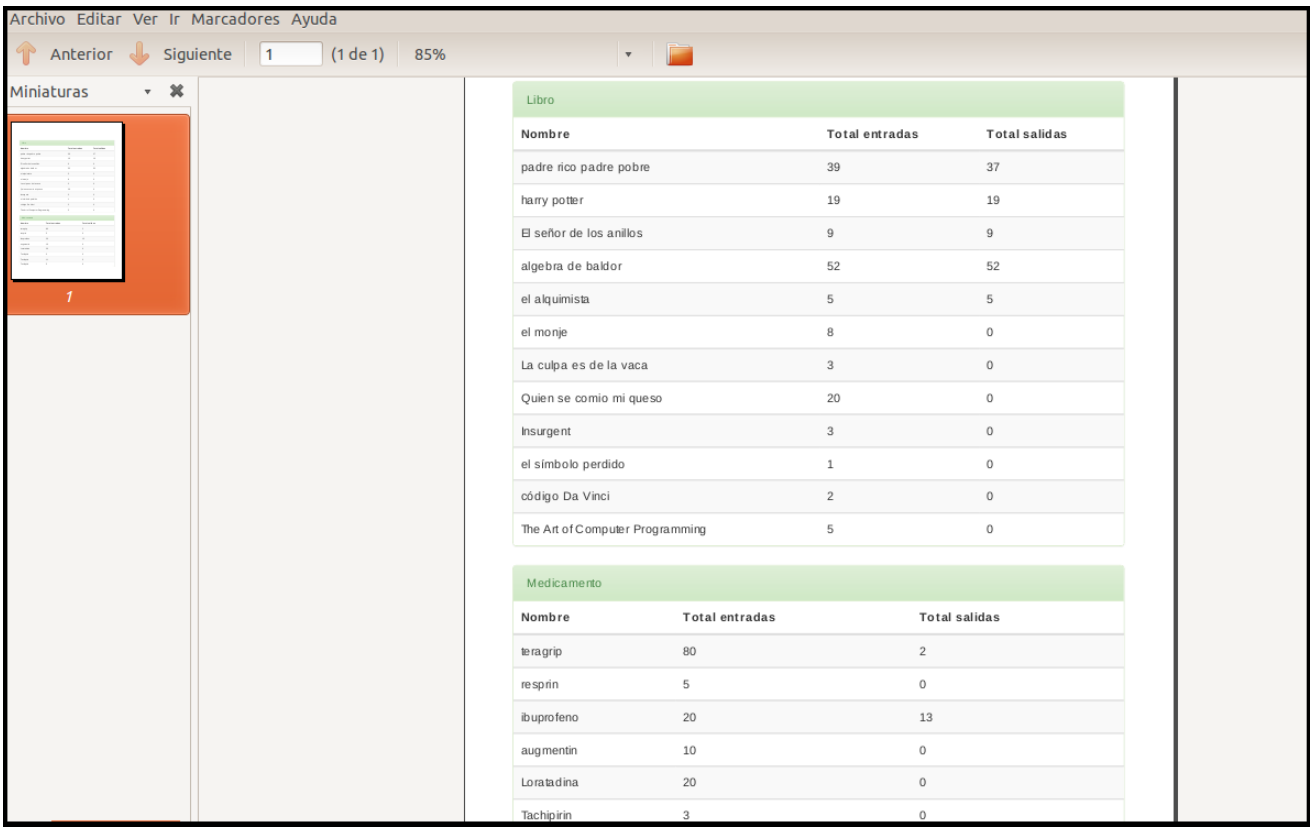


Figura 3.61: Reporte hecho de la consulta
Juárez y Villegas (2014)

3.3.9. Iteración 8

Planificación: para esta iteración se realiza el desarrollo de los requisitos funcionales de crear y editar las presentaciones de los medicamentos, así como también el crear y editar unidades de los mismos. Se describe esta iteración en la Tabla 3.53.

Tabla 3.53: Iteración 8

Número de iteración: 8	**Número de historia**: 1, 3, 12, 13, 14, 15
Caso de uso: 11, 12, 13 y 14	
Fecha de inicio: 01/04/2014	**Fecha fin**: 18/04/2014
Descripción: desarrollo de los requerimientos funcionales de crear y editar las unidades y presentaciones de los medicamentos, y la interfaz gráfica del sistema	**Tipo**: desarrollo

Diseño: para el desarrollo de la creación de la presentación, se desarrolló primero la vista, la cual consiste en un formulario con un solo campo. Luego se desarrolló el controlador **presen-**

tacions_controller.rb, el cual recibe los parámetros del formulario y realiza un query para ingresar la presentación en la base de datos.

Para el desarrollo de la edición de la presentación, es básicamente igual que la creación, excepto que el controlador solo actualiza el nombre de la presentación en la base de datos.

La realización de la creación y edición de las unidades de los medicamentos; se sigue el mismo proceso que se realizó con la presentación. Se crean las vistas y luego el controlador **unidads_controller.rb**.

La realización de las validaciones de los formularios de unidad y presentación, se desarrollan en el modelo respectivo de cada uno, **unidad.rb** y **presentacion.rb**.

Codificación:

En la creación y edición de las presentaciones y unidades, se desarrolló un formulario simple con un solo campo, el cual será el nombre de la presentación o la unidad. Los formularios para cada uno se pueden apreciar en las Figuras 3.62 y 3.63.

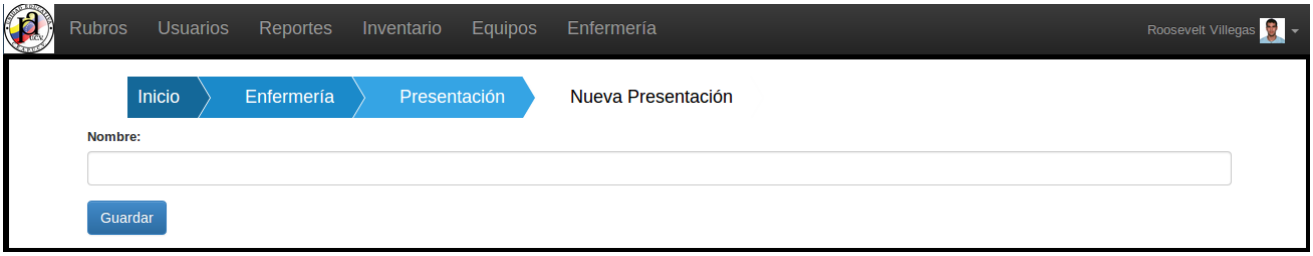


Figura 3.62: Formulario crear presentación
Juárez y Villegas (2014)

Figura 3.63: Formulario crear unidad
Juárez y Villegas (2014)

Luego se desarrollan los controladores de presentación **(presentacions_controller.rb) y** unidad (**unidads_controller.rb**), que contienen las funciones de crear y editar. Esto se puede detallar en la Figura 3.64.

```
def create
  @presentacion = Presentacion.new(presentacion_params)

  respond_to do |format|
    if @presentacion.save
      flash[:notice] = "La presentación fue creada exitoxamente"
      format.html { redirect_to action: "index"}
      format.json { render action: 'show', status: :created, location: @presentacion }
    else
      format.html { render action: 'new' }
      format.json { render json: @presentacion.errors, status: :unprocessable_entity }
    end
  end
end

# PATCH/PUT /presentacions/1
# PATCH/PUT /presentacions/1.json
def update
  respond_to do |format|
    if @presentacion.update(presentacion_params)

      flash[:notice] = "La presentación fue editada exitoxamente"
      format.html { redirect_to action: "index"}
      format.json { head :no_content }
    else
      format.html { render action: 'edit' }
      format.json { render json: @presentacion.errors, status: :unprocessable_entity }
    end
  end
end
```

Figura 3.64: Controlador de presentación
Juárez y Villegas (2014)

Pruebas: se realizan las pruebas funcionales de la iteración número ocho.

Tabla 3.54: Prueba 14

Número de caso de prueba: 14	**Número de historia de usuario: 12**
Descripción	verificar el correcto funcionamiento de las validaciones en crear una presentación
Resultados	Con los datos incorrectos: Figura 3.65

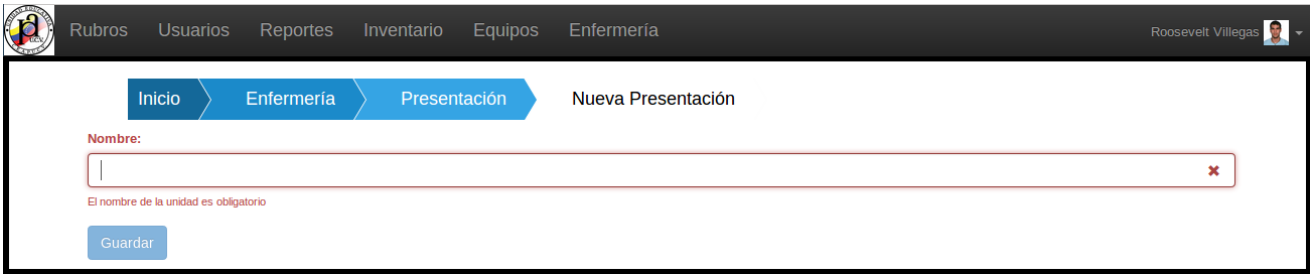


Figura 3.65: Campo presentación con datos incorrectos
Juárez y Villegas (2014)

Tabla 3.55: Prueba 15

Número de caso de prueba: 15	**Número de historia de usuario: 14**
Descripción	verificar el correcto funcionamiento de las validaciones en crear una unidad
Resultados	Con los datos incorrectos: Figura 3.66

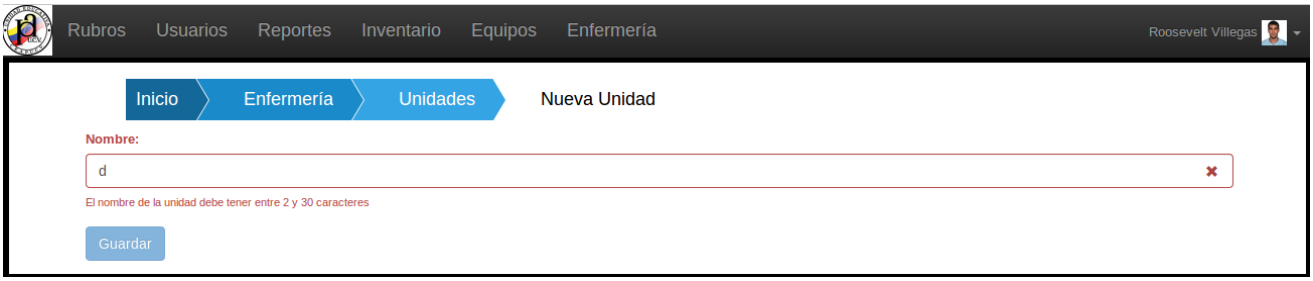


Figura 3.66: Campo unidad con datos incorrectos
Juárez y Villegas (2014)

Capítulo 4

Pruebas de Usabilidad y Aceptación

Para la culminación del sistema de gestión de inventario automatizado utilizando código QR, se aplicaron una serie de pruebas de usabilidad y aceptación para determinar ambos aspectos, la tolerancia a fallas y verificar si cumple con los requerimientos funcionales obtenidos de los usuarios. Estas pruebas se realizaron con un formulario utilizando la escala de Likert [27] como instrumento de evaluación tipo encuesta, la cual propone enunciados donde la persona que realiza la prueba debe mostrar si está de acuerdo o en desacuerdo con dichos enunciados. Se utilizaron cinco alternativas de respuestas para cada enunciado: totalmente de acuerdo, de acuerdo, ni de acuerdo ni en desacuerdo, en desacuerdo y totalmente en desacuerdo. Las pruebas se realizaron a 5 personas que conforman la sociedad de padres y representantes, y personal administrativo del CEAPUCV, los cuales son los usuarios finales del sistema.

A continuación se presenta el instrumento de evaluación tipo encuesta que contiene los enunciados de la pruebas que fueron llevadas a cabo. Dicho instrumento se divide en seis secciones: evaluación de la interfaz gráfica, de la información que ofrece la interfaz gráfica, del registro e inicio de sesión de usuarios, de la creación y edición de las unidades y presentaciones de los medicamentos, del ingreso y egreso de equipos e insumos del inventario, consultas y reportes, y por último evaluación de la información de los equipos; secciones que se encuentran en las Tablas 4.1, 4.2, 4.3, 4.4, 4.5, 4.6 y 4.7 respectivamente. Estas pruebas se realizan con la finalidad de evaluar la usabilidad y el correcto funcionamiento de los módulos, ya que el sistema no está todavía en producción.

Tabla 4.1: Instrumento tipo encuesta utilizado para evaluar la aceptación y usabilidad de la interfaz gráfica
Juárez y Villegas (2014)

	Totalmente de acuerdo	De acuerdo	Ni de acuerdo ni en desacuerdo	En desacuerdo	Totalmente en desacuerdo
Son comprensibles las acciones que se deseen realizar					
Es simple de entender (se comprende rápidamente)					
Es sencillo de usar (las funciones son fáciles de usar)					
Los colores son placenteros a la vista					
El uso de la aplicación fue satisfactorio					

Tabla 4.2: Instrumento tipo encuesta utilizado para evaluar la aceptación y usabilidad de la información que ofrece la interfaz gráfica
Juárez y Villegas (2014)

	Totalmente de acuerdo	De acuerdo	Ni de acuerdo ni en desacuerdo	En desacuerdo	Totalmente en desacuerdo
Es fácil de entender					
Es útil llevar a cabo un objetivo					
Las funciones del menú son comprensibles					
Los mensajes tienen sentido					

Tabla 4.3: Instrumento tipo encuesta utilizado para evaluar la aceptación y usabilidad del registro de usuario, inicio de sesión y recuperación de contraseña
Juárez y Villegas (2014)

	Totalmente de acuerdo	De acuerdo	Ni de acuerdo ni en desacuerdo	En desacuerdo	Totalmente en desacuerdo
Es simple de llevar a cabo					
La aplicación ayudó en caso de errores					
Es fácil de comprender					

Tabla 4.4: Instrumento tipo encuesta utilizado para evaluar la aceptación y usabilidad de la creación y edición de las unidades y presentaciones de medicamentos
Juárez y Villegas (2014)

	Totalmente de acuerdo	De acuerdo	Ni de acuerdo ni en desacuerdo	En desacuerdo	Totalmente en desacuerdo
Es fácil de comprender					
Es simple de llevar a cabo					

Tabla 4.5: Instrumento tipo encuesta utilizado para evaluar la aceptación y usabilidad del ingreso y egreso de equipos e insumos en el inventario
Juárez y Villegas (2014)

	Totalmente de acuerdo	De acuerdo	Ni de acuerdo ni en desacuerdo	En desacuerdo	Totalmente en desacuerdo
Es fácil de comprender					
Es simple de llevar a cabo el objetivo de manera efectiva					

Tabla 4.6: Instrumento tipo encuesta utilizado para evaluar la aceptación y usabilidad de las consultas y reportes
Juárez y Villegas (2014)

	Totalmente de acuerdo	De acuerdo	Ni de acuerdo ni en desacuerdo	En desacuerdo	Totalmente en desacuerdo
Es fácil de comprender					
Los resultados de las consultas fueron los esperados					
La generación en PDF de la consulta fue exitosa					

Tabla 4.7: Instrumento tipo encuesta utilizado para evaluar la aceptación y usabilidad de la información de los equipos e insumos
Juárez y Villegas (2014)

	Totalmente de acuerdo	De acuerdo	Ni de acuerdo ni en desacuerdo	En desacuerdo	Totalmente en desacuerdo
La información del código QR es correcta con la información del equipo o insumo que representa					
El código QR se genera en formato PDF de forma correcta					

4.1. Resultados de las pruebas de usabilidad y aceptación

Habiendo aplicado las encuestas anteriormente descritas, se pudo obtener los siguientes resultados por parte de los usuarios:

Interfaz gráfica

Las primeras cinco gráficas representan los resultados obtenidos luego de aplicar la encuesta para medir la aceptación y usabilidad de la interfaz de los enunciados con respecto a la interfaz gráfica de la aplicación.

La Figura 4.1 indica que las acciones son comprensibles, ya que el 80 % de las personas encuestadas eligieron la opción de "Totalmente de acuerdo " y el 20 % restante estuvo "De acuerdo".

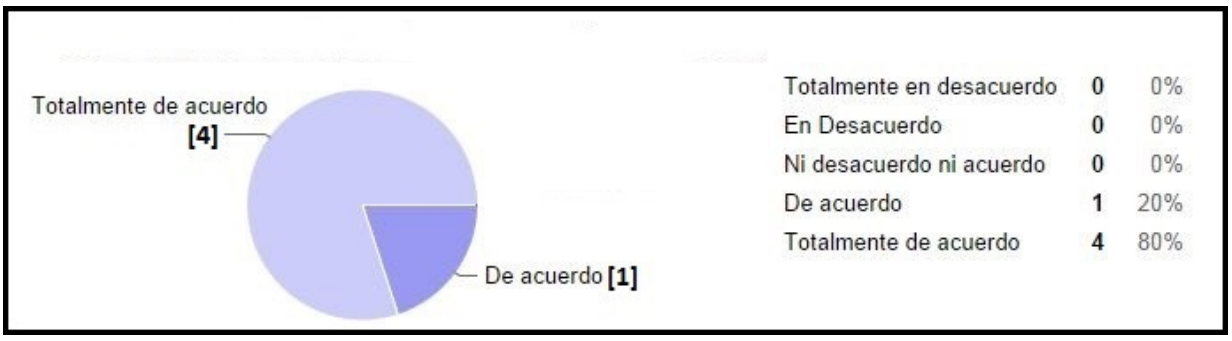


Figura 4.1: Son comprensibles las acciones que se deseen realizar

En las Figuras 4.2, 4.3 y 4.4 el 100 % de los encuestados eligieron la opción "Totalmente de acuerdo", por lo que la interfaz gráfica es simple de comprender, los colores son agradables a la vista y su uso fue satisfactorio.

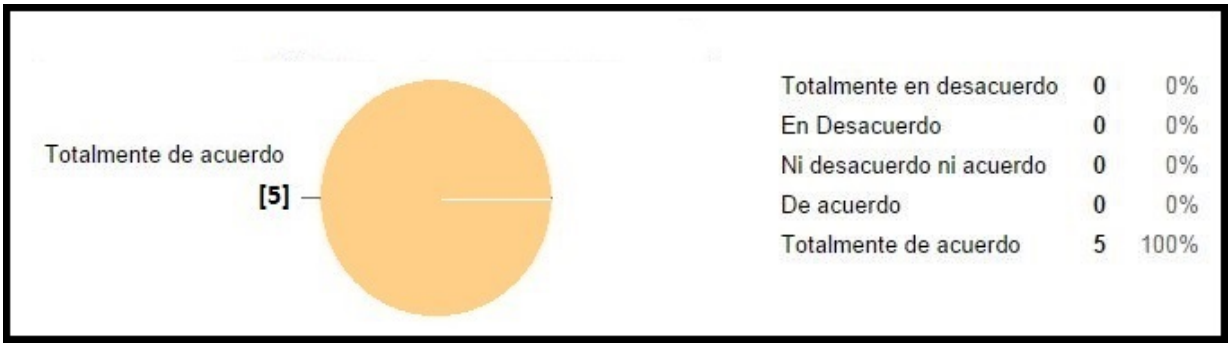


Figura 4.2: Es simple de entender (se comprende rápidamente)

Figura 4.3: Los colores son placenteros a la vista

Figura 4.4: El uso de la aplicación fue satisfactorio

Con respecto a la sencillez de la interfaz gráfica, las funciones de la aplicación son fáciles de usar, puesto que la mayoría de los encuestado (el 80 %) estuvo "Totalmente de acuerdo". Lo anterior se muestra en la Figura 4.5.

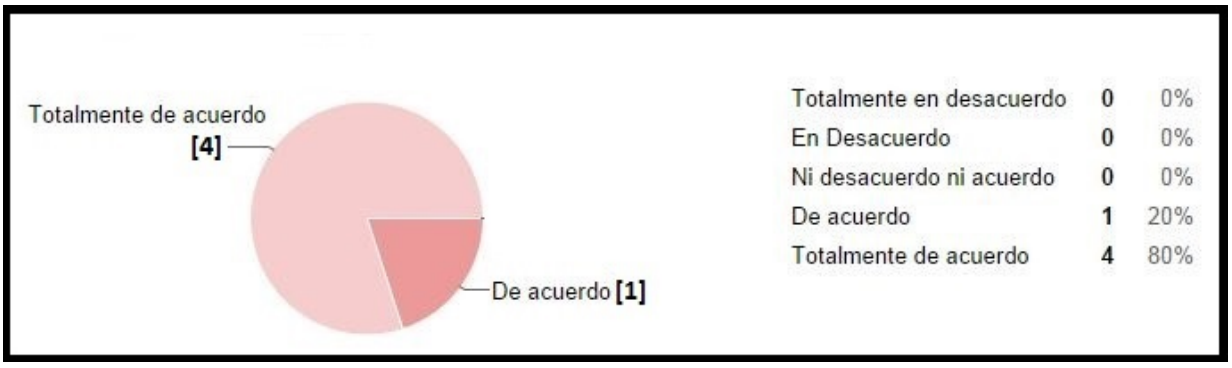


Figura 4.5: Es sencillo de usar (las funciones son fáciles de usar)

La interfaz gráfica por ser el medio en el cual los usuarios interactúan con la aplicación, es muy importante, y por lo tanto esta tiene que ser agradable a la vista, además de ser comprensible y confortable. Así los usuarios solo se tienen que concentrar en la realización de las actividades dentro de la aplicación.

Información que ofrece la interfaz gráfica

La información que provee la interfaz juega un papel relevante en el logro de los objetivos del usuario dentro de la aplicación, pues mediante ellos se guía al mismo en las diversas operaciones que puede realizar y se le notifica el resultado de sus acciones al interactuar con la aplicación. Las siguientes cuatro enunciados aplicados en la prueba, fueron con respecto a la información provista por la interfaz gráfica.

Según las respuestas de las personas encuestadas, la información provista por la interfaz gráfica es fácil de entender, es de utilidad para llevar a cabo un objetivo y hace simple la compresión de las funciones del menú; ya que el 80 % estuvo "Totalmente de acuerdo" y el 20 % prefirió la opción "De acuerdo". En las Figuras 4.6, 4.7 y 4.8 se pueden apreciar las gráficas de las respuestas obtenidas.

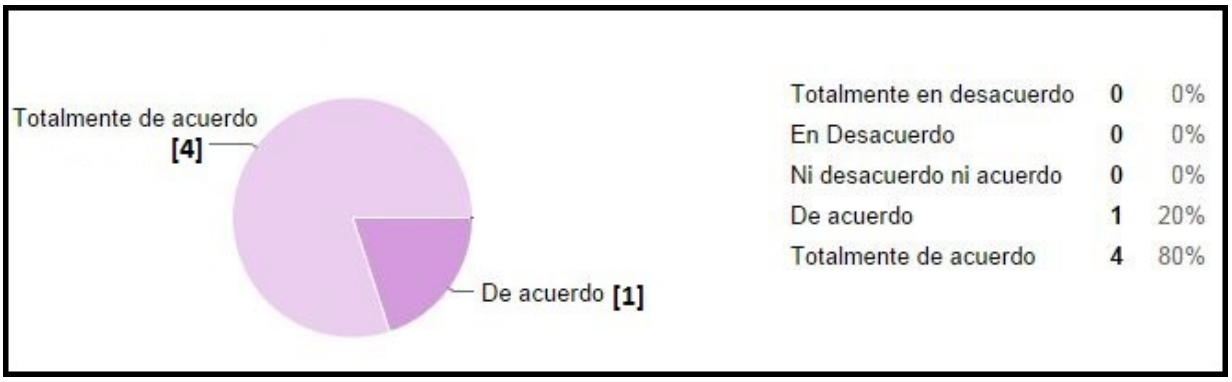


Figura 4.6: Es fácil de entender

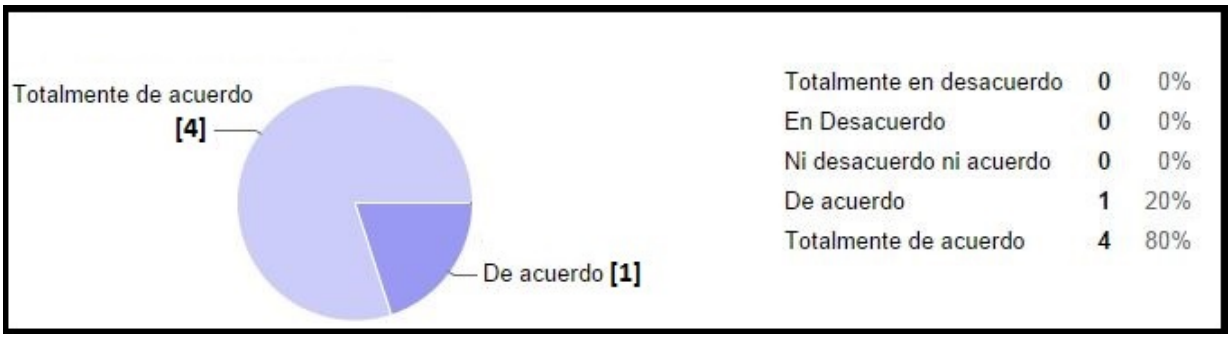


Figura 4.7: Es útil llevar a cabo un objetivo

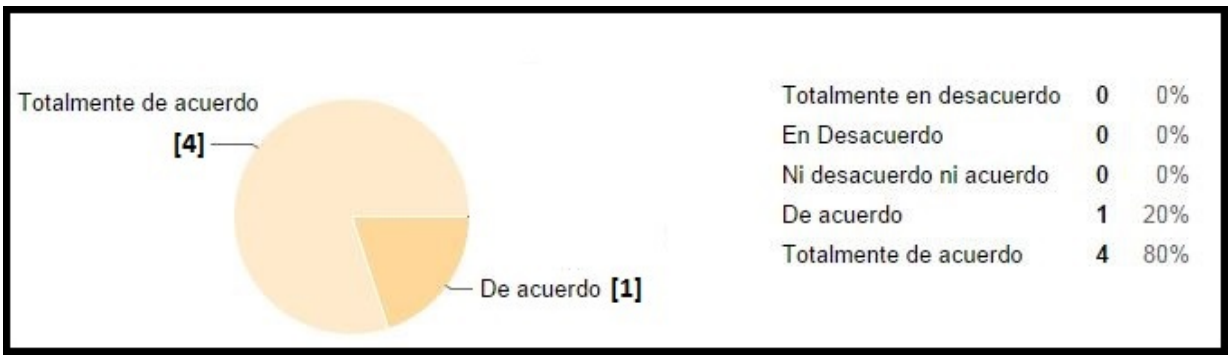


Figura 4.8: Las funciones del menú son comprensibles

En la Figura 4.9 se puede apreciar que los mensajes de la interfaz gráfica guardan sentido, puesto que el 100 % de los encuestados estuvo "Totalmente de acuerdo" con este enunciado.

Figura 4.9: Los mensajes tienen sentido

Registro de usuario, inicio de sesión y recuperación de contraseña

Las siguientes tres gráficas se basan en los enunciados sobre el inicio de sesión, el registro de usuarios en el sistema y la recuperación de la contraseña.

En la Figura 4.10 se muestran los resultados de si es fácil de llevar cabo el objetivo de registrar un usuario, iniciar sesión y recuperar la contraseña. El 60 % estuvo "Totalmente de acuerdo", mientras que el 40 % estuvo "De acuerdo"; con esto se puede decir que dichas funciones de la aplicación son simples de llevar a cabo.

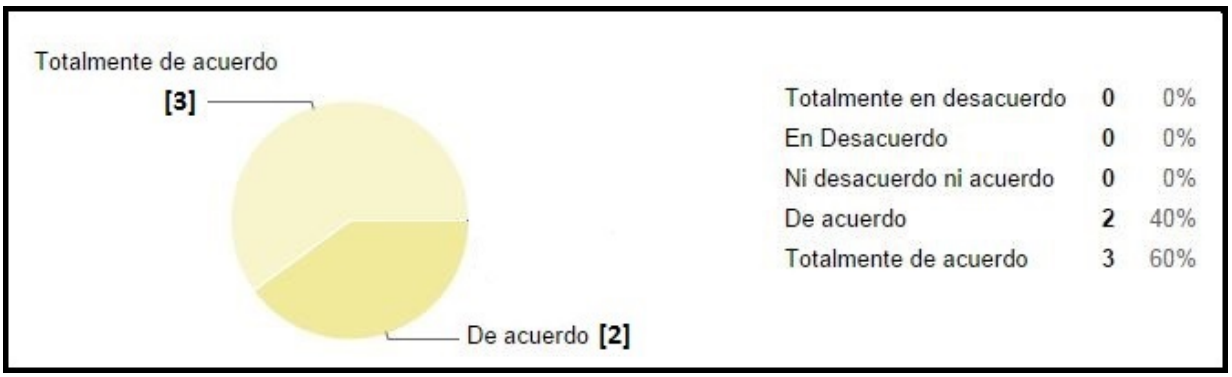


Figura 4.10: Es simple de llevar a cabo

Para el enunciado de si la aplicación ayudó en caso de errores mientras se realizaban funciones de registrar usuarios, iniciar sesión y recuperar contraseña; los resultados fueron que 40 % de los encuestados estuvo "Totalmente de acuerdo", 20 % estuvo "De acuerdo", 20 % optó por la opción "Ni desacuerdo ni de acuerdo" y el último 20 % estuvo "En desacuerdo".

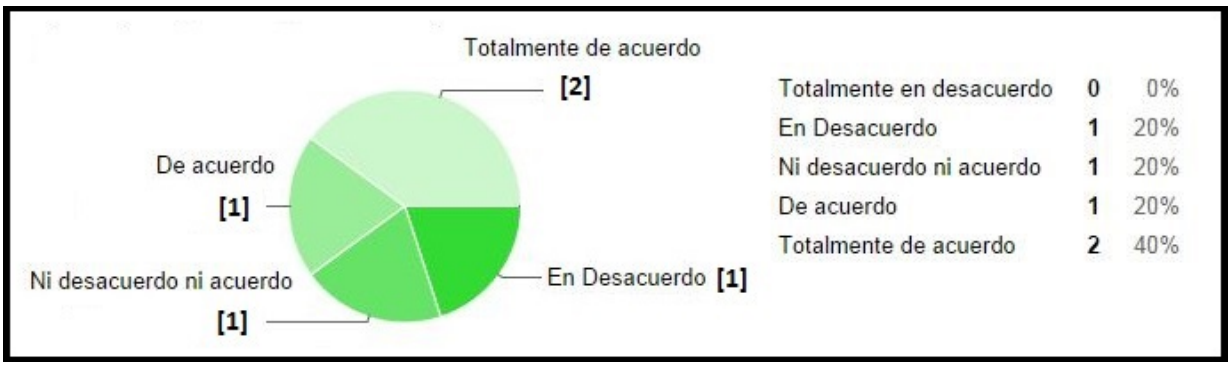


Figura 4.11: La aplicación ayudó en caso de errores

Para los encuestados, las funciones de registrar usuarios, iniciar sesión y recuperar contraseña son fáciles de comprender, ya que el 60 % eligió "Totalmente de acuerdo", 20 % estuvo "De acuerdo" y el 20 % restante indicó "Ni desacuerdo ni acuerdo".

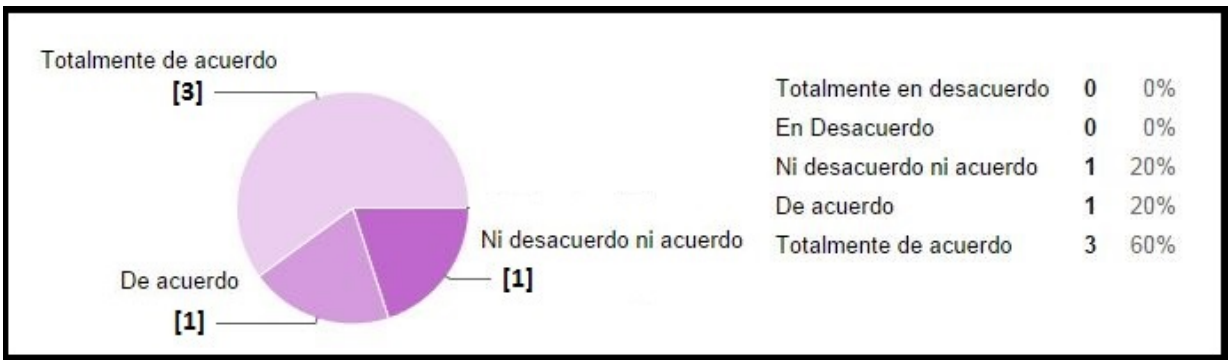


Figura 4.12: Es fácil de comprender

Creación y edición de las unidades y presentaciones de medicamentos

En las Figuras 4.13 y 4.14 indican que la creación y edición de las unidades y presentaciones de los medicamentos son fáciles de comprender y de llevar cabo, puesto que un 60 % de las personas encuestadas estuvieron "Totalmente de acuerdo" y el 40 % estuvo "De acuerdo".

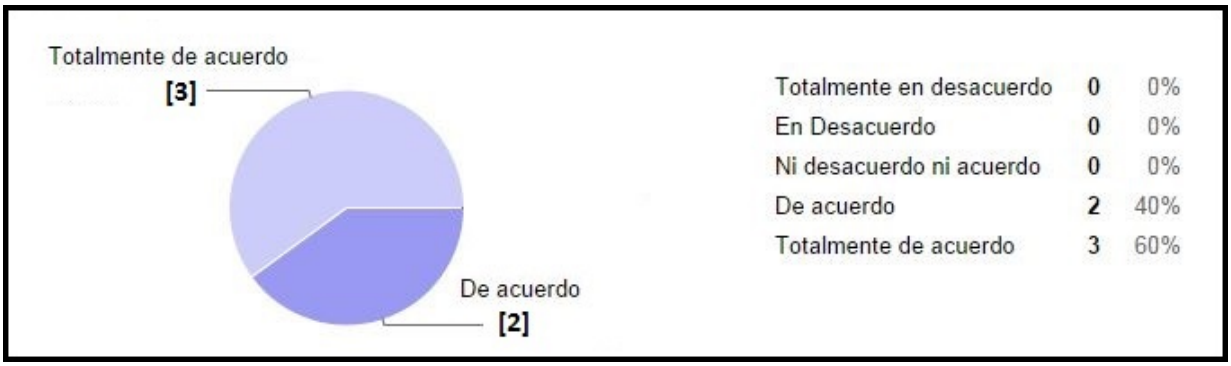


Figura 4.13: Es fácil de comprender

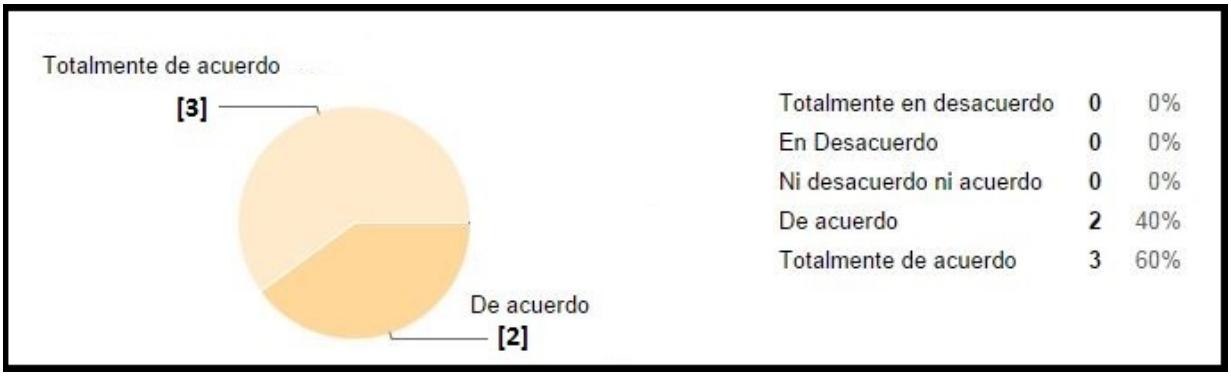


Figura 4.14: Es simple de llevar a cabo

Ingresar y egresar equipos e insumos en el inventario

Los resultados para ingresar y egresar equipos e insumos del inventario reflejan que estas funciones son fáciles de comprender y llevan a cabo su objetivo de manera efectiva, ya que el 100 % de los encuestados optó por la opción de "Totalmente de acuerdo". Dichos resultados se observan en las Figuras 4.15 y 4.16.

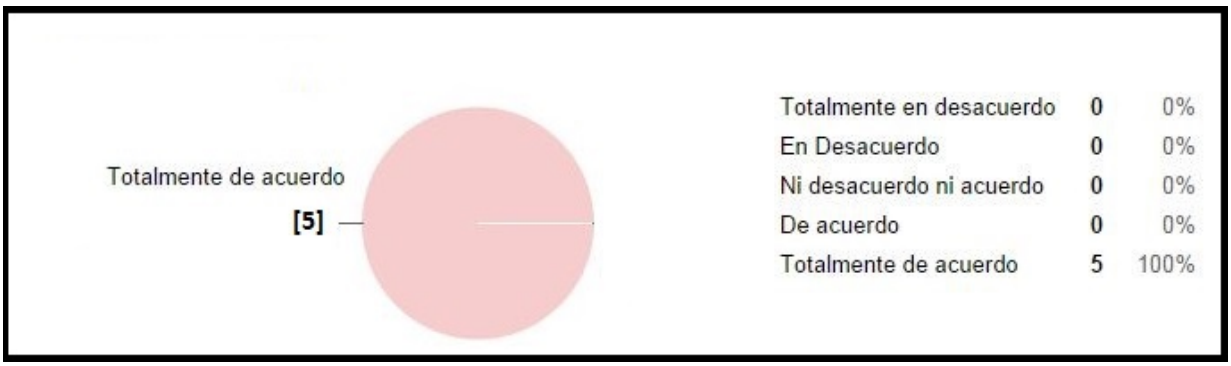


Figura 4.15: Es fácil de comprender

Figura 4.16: Es simple de llevar a cabo el objetivo de manera efectiva

Consultas y reportes

Para la realizar las consultas del inventario y generar los reportes, los resultados fueron los siguientes.

La Figura 4.17 muestra que las funciones de efectuar consultas al inventario y realizar reportes sobre dichas consultas son fáciles de comprender, ya que 60 % estuvo "Totalmente de acuerdo" y 40 % le pareció estar "De acuerdo"

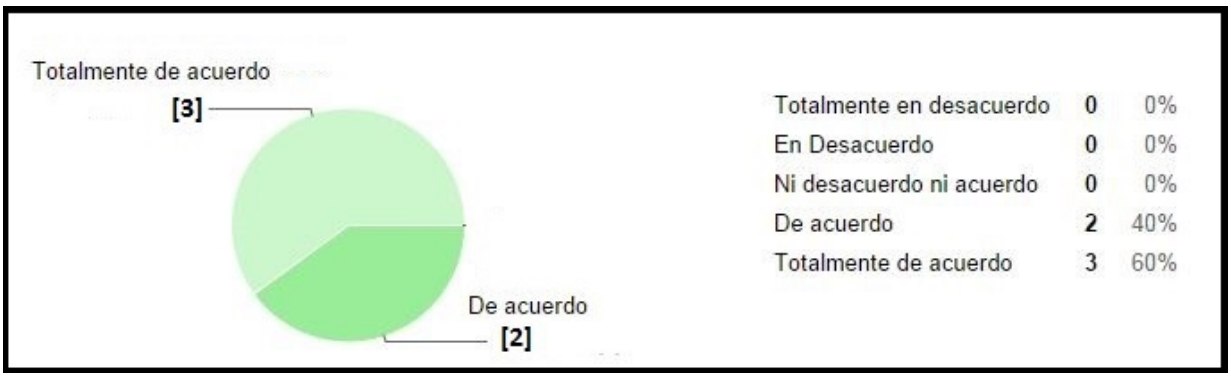


Figura 4.17: Es fácil de comprender

En las Figuras 4.18 y 4.19 se obtuvo el mismo resultado, 80 % eligió "Totalmente de acuerdo" y el 20 % optó por estar "De acuerdo"; por lo que realizar consultas al inventario arrojó los resultados esperados y la generación de los reportes en formato PDF se efectuó con éxito.

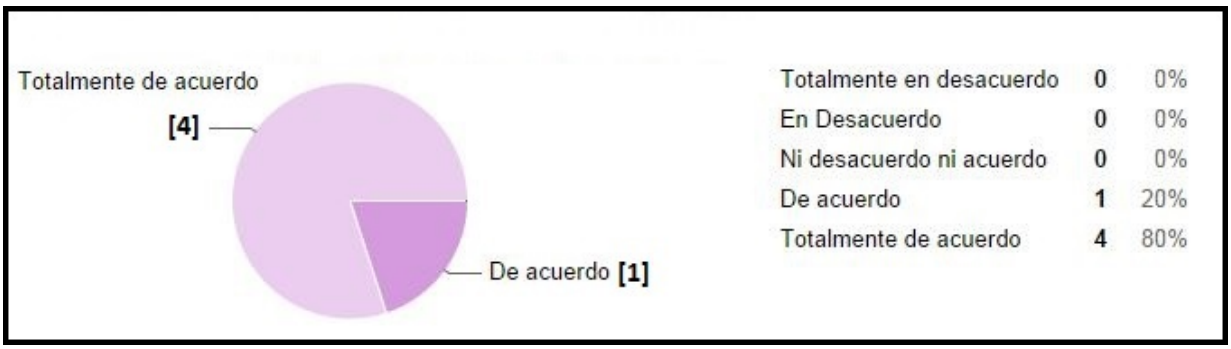


Figura 4.18: Los resultados de las consultas fueron los esperados

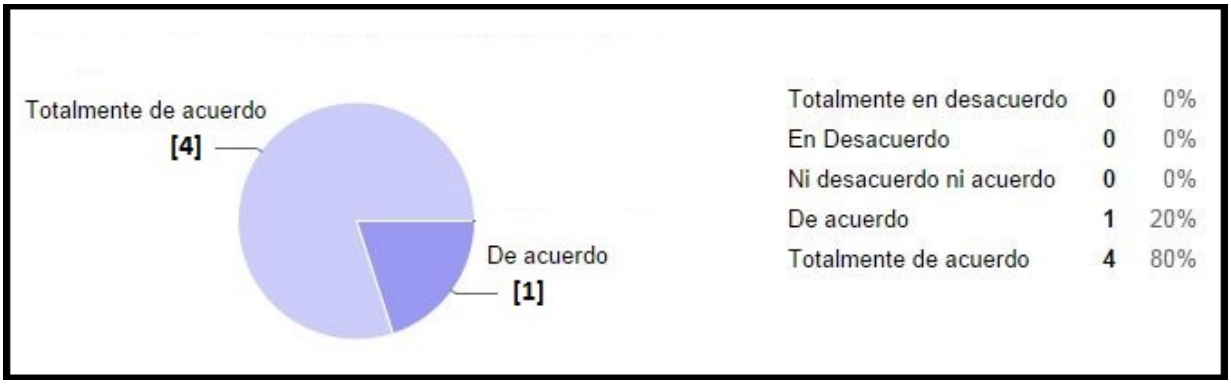


Figura 4.19: La generación en PDF de la consulta fue exitosa

Información de los equipos e insumos

Con respecto a la información de los códigos QR y su generación, se muestran los resultados obtenidos.

Los resultados arrojados por la pruebas, reflejados en las Figuras 4.20 y 4.20, indican que la información del código QR es correcta con la información con respecto al equipo o insumo que representa y que la generación del código QR en formato PDF se realiza de forma correcta, puesto que el 80 % de las personas encuestadas estuvo "Totalmente de acuerdo" y 40 % de estas optó por estar "De acuerdo".

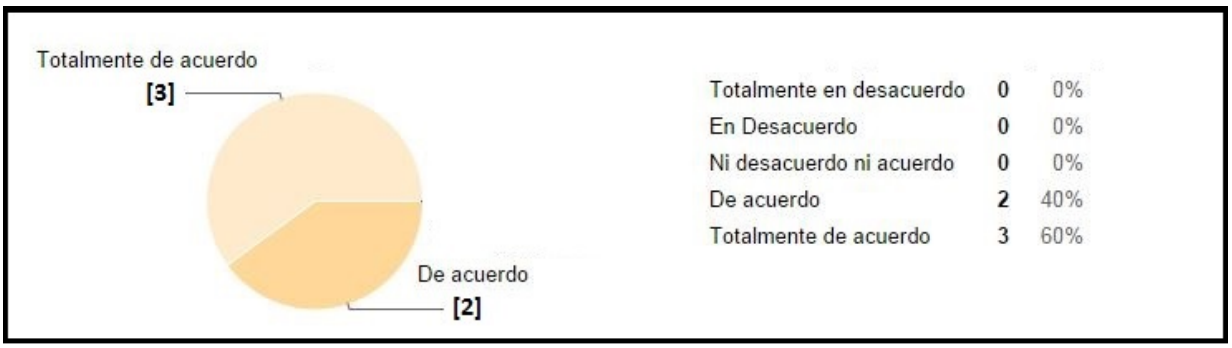


Figura 4.20: La información del código QR es correcta con la información del equipo o insumo que representa

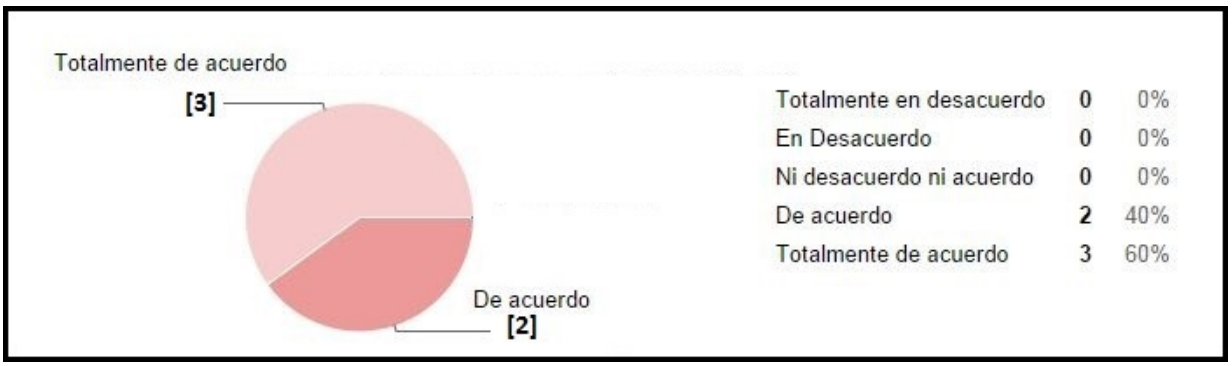


Figura 4.21: El código QR se genera en formato PDF de forma correcta

A través de estos resultados es posible afirmar que la aplicación presentó un alto grado de aceptación por parte de los encuestados, dado que todas las preguntas tuvieron resultados positivos. Por lo que se puede confirmar que la aplicación es usable para los usuarios, es tolerable a las fallas y los módulos realizan las funciones y resultados esperados.

Conclusión

Al finalizar este trabajo especial de grado, se ha desarrollado un sistema de gestión de inventario utilizando la tecnología de código QR para el colegio CEAPUCV.

Este sistema ofrece una solución a la problemática de la institución. Para el desarrollo del sistema de gestión de inventario se identificaron, primeramente, los requisitos funcionales y no funcionales, llevando a cabo una serie de reuniones con los usuarios finales del sistema, y donde se obtuvieron los requerimientos y necesidades planteadas. Posteriormente se realizó una investigación sobre la tecnología de códigos QR y su implementación en el ámbito web. Luego se diseñó y desarrolló el sistema de gestión de inventario, utilizando el método de desarrollo ágil de software XP, y el lenguaje de programación Ruby, junto con su *framework* para web Rails; obteniendo así resultados satisfactorios.

Se desarrollaron varios módulos, los cuales dividen el sistema en varias funcionalidades, y solo pueden ser accedidos por usuarios con los permisos necesarios. Entre los módulos desarrollados se pueden encontrar los principales, como registrar usuarios, ingresar y egresar del inventario, generar reportes y generar el código QR de los equipos e insumos. Para la interfaz gráfica, se decidió desarrollarla los más usable posible; debido que la gran mayoría de los usuarios no tienen mucha experiencia haciendo uso de un sistema web. Por tal motivo se manejaron colores llamativos para algunas funcionalidades, lo cual ayudará al usuario a una mayor comprensión del funcionamiento del sistema, así como el conjunto de iconos proporcionados por Twitter Bootstrap.

Cada módulo del sistema fue sometido a varias pruebas; dichas pruebas fueron funcionales (realizada por los desarrolladores), de aceptación y usabilidad (llevadas a cabo por los usuarios finales). Estas pruebas se realizaron con éxito, obteniendo resultados satisfactorios, lo cual garantiza el correcto funcionamiento del sistema para los usuarios que hagan uso del mismo. Por tal motivo la sociedad de padres y representares puede tener un control efectivo de los equipos e insumos del colegio CEAPUCV, así como también establecer sus orígenes y evitar perdidas.

Contribuciones

El sistema de inventario fue desarrollado para el uso de los profesores, autoridades y miembros de la sociedad de padres y representantes del CEAPUCV. En el se pueden acotar las siguientes aportaciones a la institución:

- Establecer el origen de los equipos e insumos.
- Gestionar los equipos e insumos.
- Visualizar la información de cada uno de los equipos e insumos dentro del inventario.
- Generar un código QR único para cada equipo e insumo del inventario.
- Permitir a los usuarios ingresar y/o egresar algún equipo o insumo del inventario.
- Realizar consultas y/o reportes del inventario.
- Controlar los permisos de acceso de los usuarios.
- Minimizar la perdida sin control de los equipos e insumos.

Trabajos futuros

En esta sección se expondrán una serie de recomendaciones, cuyas implementaciones pueden incrementar la eficiencia y la eficacia del proyecto hecho, mejorando aún más las funciones para los cuales fue desarrollado.

Recomendaciones:

- Basado en el punto anterior, se recomienda desarrollar la aplicación en un formato móvil de forma nativa al sistema operativo del teléfono móvil. Ya que con ello se puede unificar el proceso de uso del código QR en una sola aplicación, evitando el uso de múltiples variedades de aplicaciones, y con ello el múltiple resultado de lecturas de los códigos.

Bibliografía

[1] J. Portillo, A. Bermejo, y A. Bernardos. *Tecnología de identificación por radiofrecuencia (RFID): Aplicaciones en el ámbito de la salud.* Revista: Colección de Informes de Vigilancia Tecnológica madri+d. Vol. 13. Edita: Fundación madri+d para el conocimiento velázquez, 76. e-28001 madrid . ISBN-13: 978-84-612-4360-0. Madrid, España, 2008.

[2] GS1. *Códigos de barras.* [En Linea] Disponible en: http://www.gs1ve.org/codigosdebarras.php (Consultado: agosto, 2013)

[3] C. Zen. *Código de barras, historia.* [En Linea] Disponible en: http://www.codigodebarras.pe/codigo-de-barras-historia/ (Consultado: mayo, 2013)

[4] ABC. *Muere joseph woodland.* [En Linea] Disponible en: http://www.abc.es/ciencia/20121213/abci-muere-inventor-codigo-barras-201212131254.html (Consultado: agosto, 2013)

[5] J. Velasco. *Historia de la tecnología joseph woodland.* [En Linea] Disponible en: http://alt1040.com/2012/12/joseph-woodland (Consultado: agosto, 2013)

[6] R. Perales. *Uso de códigos de barras para principiantes.* [En Linea] Disponible en: http://www.jfksoft.com/barcode_es.pdf (Consultado: agosto, 2013)

[7] L. Guayan. *Lectores y códigos de barra.* [En Linea] Disponible en: http://www.serviciosjfp.com/Otros/codigo_barras.htm (Consultado: agosto, 2013)

[8] GS1. *Descripcipción de la organización.* [En Linea] Disponible en: http://www.gs1ve.org/quienessomos.php (Consultado: agosto, 2013)

[9] GS1Venezuela. *Pasos para implementar el código de barras.* [En Linea] Disponible en: http://www.gs1ve.org/pasoscodigo.php (Consultado: agosto, 2013)

[10] I. Borjas. *Código de barras de dos dimensiones 2d.* [En Linea] Disponible en: http://www.codigodebarras.com/tema.php?ID=dos_dimensiones (Consultado: agosto, 2013)

[11] GS1. *¿qué es código datamatrix?* [En Linea] Disponible en: http://www.gs1cr.org/index.php/datamatrix (Consultado: agosto, 2013)

[12] Barcodesoft. *Maxicode*. [En Linea] Disponible en: http://www.barcodesoft.com/es-mx/maxicode.aspx (Consultado: agosto, 2013)

[13] G. Fernández. *Generador e interprete QR code*. Reporte Técnico: Escuela Técnica Superior de Ingeniería Informática, Universidad de Sevilla. Sevilla, España, Diciembre 2009.

[14] M. Villapol. *Fundamentos de la Tecnología Inalámbrica: Técnicas de Corrección y Detección de Errores*. Notas de Docencia: Escuela de Computación, Facultad de Ciencias, Universidad central de Venezuela, 2007.

[15] J. Parada. *Sistemas de Inventario*. Notas de Docencia: Consejo de Computación Académica, Universidad de los Andes, julio 2006.

[16] K. Rincón. "Evaluación de la gestión en el manejo de inventarios de la mercancía importada en la empresa portaltrece c.a. en el año 2010." Trabajo Especial de Grado. Escuela de Contaduría Pública, Facultad de Ciencias Ecónomicas y Sociales, Universidad Alejandro de Humbolt. noviembre 2011.

[17] E. Futura. *Stockbase pos*. [En Linea] Disponible en: http://www.stockbasepos.info/ (Consultado: agosto, 2013)

[18] N. Sotfware. *Inventoria*. [En Linea] Disponible en: http://www.nchsoftware.com/inventory/faq.html (Consultado: agosto, 2013)

[19] Birtud. *Birtud*. [En Linea] Disponible en: http://www.birtud.com/ (Consultado: agosto, 2013)

[20] R. O. Rails. *Ruby on rails*. [En Linea] Disponible en: http://rubyonrails.org/ (Consultado: agosto, 2013)

[21] M. Kabchi y M. Martinez. "Desarrollo del módulo de acceso a los contenidos preservados en formato warc para un prototipo de archivo web de venezuela." Trabajo Especial de Grado. Escuela de Computación, Facultad de Ciencias, Universidad Central de Venezuela. marzo 2014.

[22] M. Otto y J. Thornton. *Twitter bootstrap*. [En Linea] Disponible en: https://github.com/twbs/bootstrap (Consultado: agosto, 2013)

[23] D. Martinez. *Postgresql vs. mysql*. [En Linea] Disponible en: http://danielpecos.com/documents/postgresql-vs-mysql/ (Consultado: agosto, 2013)

[24] MySQL. *Caracteristicas de mysql*. [En Linea] Disponible en: http://dev.mysql.com/doc/refman/5.0/es/features.html (Consultado: agosto, 2013)

[25] M. Rodriguez. *Introducción general a la Metodología de la Investigación.* Ensayo: Universidad de Artes y Ciencias Sociales ARCIS y Universidad Tecnológica de Chile. Punta Arenas - Magallanes, Chile, 2012.

[26] S. Amaro y J. Valverde. *Metodologías Ágiles.* Notas de Docencia: Escuela de Informática, Falcutad de Ciencias Físicas y Matemáticas, Universidad de Trujillo. Trujillo, Perú, 2007.

[27] J. Franco. *Construcción de una escala de actitudes tipo likert.* [En Linea] Disponible en: http://tesisdeinvestig.blogspot.com/2011/06/escala-de-likert.html (Consultado: mayo, 2014)