

Documento de entrega final

Entrega de documento final



8 de junio de 2016

Universidad VEracruzana

Ingeniería de Software

Índice

[Introducción 2](#_Toc453115831)

[Desarollo 3](#_Toc453115832)

[Análisis 4](#_Toc453115833)

[Requerimientos 5](#_Toc453115834)

[Diseño 6](#_Toc453115835)

[Base de datos 6](#_Toc453115836)

[Modelo de Clases 7](#_Toc453115837)

[Entidad Relación 8](#_Toc453115838)

[Prototipos 9](#_Toc453115839)

[Estándar 14](#_Toc453115840)

[Conclusiones 18](#_Toc453115841)

[Referencias 19](#_Toc453115842)

# Introducción

Este documento tiene como propósito complementar, demostrar y extender la documentación del trabajo realizado, así como justificar las decisiones de construcción, por medio de los diagramas presentados en la materia de *Principios de diseño de Software,* que se han tomado en el proyecto. A demás, se pretende dar a conocer al usuario el trabajo realizado durante el semestre.

Se da al usuario este documento, para que sea tomado como referencia de lo trabajado en el proyecto, ya que se dan detalles de la implementación del software, se dará a conocer las soluciones al problema presentado, y se resolverán dudas comunes que podrán surgir, en complemento a este documento, también se tiene la documentación generada por el proyecto de java. El documento también justifica decisiones de diseño y construcción que fueron surgiendo a lo largo de su desarrollo.

# Desarollo

Durante la planeación del sistema y la construcción del mismo se elaboraron diversos diagramas y representaciones de la arquitectura del software, por ello, se presentan conocer a detalle los procesos y funciones realizadas durante el desarrollo del proyecto, así como la justificación de los diseños del por qué las decisiones tomadas en este último paso, se tomaron las decisiones de que el lenguaje de programación seria Java con el entorno de desarrollo de NetBeans, pues ambas brindan herramientas bien probadas y documentadas con un buen soporte.

Por parte de los diagramas, estos fueron construidos con de la herramienta CASE: *Enterprise Architect* pues brinda las facilidades para un diseño de calidad con base en las normas de UML 2.X, con ayuda y colaboración del Dr. Jorge Octavio Ocharán Hernández. Éstos diagramas fueron previamente revisados y autorizados por el mismo.

## Análisis

Por medio de la elicitación de requerimientos determinamos que se necesitaba un sistema que permitiera interactuar a los usuarios de la siguiente manera:



**Ilustración 1 Modelo de funcionalidad por paquete**

Ordenado por funcionalidad y tomando en cuenta que cada uno cubriera una necesidad del cliente, éstos fueron los casos de uso seleccionados durante el proceso de diseño de software, quedando excluidos algunos otros detectados como cancelación de la reservación.

## Requerimientos

### Requerimientos del sistema Hardware y Software

#### Requerimientos mínimos:

Una computadora de arquitectura de 64bits.

Java 8 cualquier versión.

PhPMyAdmin: 4.4.14.

MySQL Community Server 5.6.26.

Apache versión 2.4.16.

Memoria RAM de 1 GB.

Teclado.

Mouse.

Conexión a internet.

#### Requisitos recomendados:

Una computadora de arquitectura de 64bits.

Java 8 actualización 91.

PhPMyAdmin: 4.6.2.

MySQL Community Server 5.6.26.

Apache versión 2.4.16.

Memoria RAM de 2 GB.

Teclado.

Mouse.

Lector de códigos de barras.

Conexión a internet.

Se necesita contar con un servidor dedicado en red o en el mismo pc, con base de datos previamente creada.

### Plantilla reservar ítem

|  |  |
| --- | --- |
| ID: | CU-12 |
| Nombre: | Reservar ítem |
| Actor: | Usuario. |
| Fecha de Creación: | 8/03/2016 |
| Fecha de Modificación: | 05/06/2016 |
| Autores: | FRANCISCO GERARDO MARES SOLANO |
| Descripción: | El usuario ingresa al sistema para reservar un ítem. |
| Precondiciones: | PRE01 - Deben existir ítems en el sistema.  PRE02 - Debe tener la sesión iniciada el usuario.  PRE03 - Debe estar disponible el ítem.  PRE04 - Debe haber acceso a la BD del sistema. |
| Flujo Normal: | Exitoso – Reservar ítem  1a.- El usuario ingresa a la sección de reservación del ítem.  2a.- El sistema muestra al usuario la disponibilidad del ítem para ser reservado  3a.- El usuario solicita la reservación del ítem.  4a.- El sistema pide confirmación de la acción.  5a.- El usuario acepta la solicitud.  6a.- El sistema informa que el ítem ha sido reservado e informa la fecha límite para recogerlo.  7a.- El usuario regresa al sistema. |
| Flujo Alterno: | Alterno – Reservación cancelada  5b.- El usuario cancela la reservación y sale de la sección del sistema.  Fallido – Reservación no disponible  2c.- El sistema muestra que el ítem no está disponible para reservar.  3c.- El usuario abandona la sección del sistema. |
| Excepciones: | Indeseable – Error con la base de datos  2d.- El sistema no puede conectar con la base de datos e informa sobre el problema.  3d.- El usuario sale del sistema. |
| Post-condiciones: | POST01 - Pedir prestado el ítem. |
| Entrada: | Entrada01 - Solicitud de reserva.  Entrada02 – confirmación de reserva. |
| Salida: | Estado de la reserva |
| Prioridad: | Deseable |

### Plantilla prestar ítem

|  |  |
| --- | --- |
| ID: | CU-6 |
| Nombre: | Registrar préstamo |
| Autor: | Luis Fernando Gomez Alejandre |
| Fecha de Creación: | 08/03/16 |
| Fecha de Modificación: | 05/06/16 |
| Actores: | Bibliotecario |
| Descripción: | El bibliotecario registrar el préstamo de un ítem para un alumno o profesor. |
| Precondiciones: | PRE01 - Deben existir ítems en la base de datos.  PRE02 - Debe haber alumnos registrados.  PRE03 - Debe haber conexión a la BD. |
| Flujo Normal: | Exitoso – préstamo registrado  1a.- El bibliotecario ingresa a la sección correspondiente.  2a.- El sistema despliega un formato para registrar un préstamo.  3a.- El bibliotecario ingresa los campos con el id del ítem, la matrícula del alumno y envía la información.  4a.- El sistema pide confirmar la transacción.  5a.- El bibliotecario acepta.  6a.- El sistema informa que se ha hecho el préstamo con satisfacción y muestra la fecha en que debe ser devuelto. |
| Flujo Alterno: | Fallido – límite de préstamo superado  6b.- El sistema informa que el alumno o profesor superó el límite de ítems prestados.  7b.- El usuario da clic en aceptar.  8b.- Regresa al flujo normal (paso 2a).  Alterno – Matrícula o identificador erróneos  4c.- El sistema informa que uno de los campos contiene datos incorrectos.  5c.- El usuario da clic en aceptar.  6c.- Regresa al flujo normal (paso 2a). |
| Excepción | Indeseable – Error con la base de datos  4d.-El sistema informa que no se puede conectar a la base de datos.  5c.- El usuario da clic en aceptar.  6c.- Regresa al flujo normal (paso 2a). |
| Postcondiciones | POST01 - Se puede realizar la devolución del ítem.  POST02 - Se puede renovar el préstamo. |
| Entrada: | Entrada01 - Matrícula del alumno.  Entrada02 - Identificador del ítem  Entrada03 – Confirmación de la acción |
| Salida: | Estado del registro |
| Prioridad: | Indispensable |

## Diseño

Ahora, se expondrán los diseños más representativos, creados previamente en la materia de *Principios de diseño de software*. Éstos fueron la base para la elaboración del proyecto y por tanto, la implementación está reflejada en el sistema trabajado.

### Base de datos



**Ilustración 2 Modelo de base de datos**

### Modelo de Clases



**Ilustración 3 Modelo de clases**

### Entidad Relación

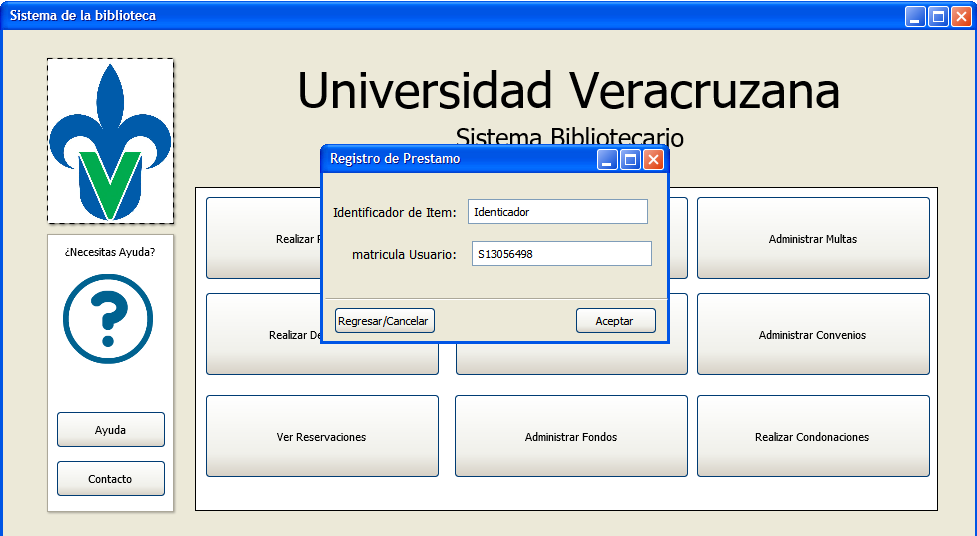


**Ilustración 4 Modelo entidad - relación**

### Prototipos

Prototipos generados con la herramienta pencil, fueron detallados a un mediano nivel según el grado de requerimientos

#### Modulo Préstamo



**Ilustración 5 Prototipo realizar préstamo**



**Ilustración 6 Prototipo préstamo correcto**



**Ilustración 7 Prototipo préstamo con error a base de datos**

#### Modulo Reservación



**Ilustración 8 Prototipo reservar ítem**



**Ilustración 9 Prototipo reservar ítem correcto**

## Estándar

|  |  |
| --- | --- |
| Cabecera de programa | La cabecera contendrá el nombre del proyecto, autores, fecha de la última edición y una descripción corta del objetivo de la clase |
| Formato de Cabecera | /\*\*  \* Clase tipo biblioteca, a través de ésta clase se pueden realizar  \* préstamos, reservaciones, búsqueda de ítems y sus correspondientes  \* validaciones.  \*  \* @author Luis Fernando Gomez Alejandre  \* @author Francisco Gerardo Mares Solano  \* @since 06/06/2016  \*/ |
| Tabla de contenido | /\*\*  \* **Descripcion del método**  \*  \* @param parametro1  \* @param parametro2  \* @param parametro3  \* @param parametroN…  \* @return Variable de retorno  \* @throws Si lanza alguna excepción  \*/ |
| Tabla de contenido  Ejemplo | /\*\*  \* **Busca cualquier ítem que esté registrado en la base de datos con el**  \* **identificador pasado.**  \*  \* @param identificador se realiza una validación de 15 caracteres y una  \* letra inicial 'I' mayúscula o minúscula.  \* @return Si el parámetro fue pasado de manera erronea o no existe en  \* la base de datos, regresa una lista vacia.  \* Nunca regresará una lista nula y siempre será un ArrayList  \* @throws SQLException Si no existe conexión con la base de datos o algo  \* sale mal durante la conexión con la misma,  \* se lanzará una SQLException con el correspondiente mensaje de error.  \*/ |
| Identificadores | Variables:  Se ocupara Lower Camel Case, dónde la primera letra del identificador será minúscula, a partir de ahí, las palabras se escribirán seguidas y la primera letra de cada palabra será mayúscula. En cada caso, serán nombre distintivos y se evitarán caracteres especiales excepto para bucles for o while, dónde podrán ser iterador i,j,k… en ese orden.  Constantes:  Se ocupará una escritura totalmente en mayúsculas, dónde la separación entre palabras será constituido por un guion bajo ( \_ ) y de igual manera, serán nombres descriptivos y palabras completas siempre declaradas como *public static final*.  Métodos:  Se ocupara Lower Camel Case, dónde la primera letra del identificador será minúscula, a partir de ahí, las palabras se escribirán seguidas y la primera letra de cada palabra será mayúscula, Si este supera los 7 argumentos se debera pasar por un objeto que haga referencia, a los atributos de ese objeto |
| Ejemplos de identificadores | private ArrayList<String> telefonos;  private ArrayList<String> redesSocial;  private ArrayList<String> correosElectronicos;  public static final int RESULTADO\_DE\_SALIDA = 0; |
| Comentarios | Los comentarios de una sola línea serán precedidos por dos diagonales (//) y no ocuparán más de un renglón. Su objetivo será ayudar a entender el motivo de un código de programación.  Los comentarios de mayor extensión serán escritos con diagonales y asteriscos para su delimitación (/\*\* … \*/) y en cada renglón intermedio habrá un asterisco (\*) como marcador de inicio y tendrá un espacio de identación incluyendo el último renglón.  Para ambos casos su identación será la misma que tenga el método, sentencia o bucle al que preceda. Siempre deben agregar información de utilidad y que extienden la información común. En el caso de *edit fold* son permitidos dos renglones con doble diagonal (//). |
| Buen comentario | //Busca contacto(s) con las funciones lambda contactosEncontrados=this.listaContactos  .stream()  .filter(contacto->contacto.getNombreContacto()  .equals(nombre))  .collect(Collectors.toList()); |
| Mal comentario | // Variable para Contador  // Es para contar números.  // Es privado para que solo la clase acceda  // Entero por que no necesito flotantes  Private int contadorNumeros; |
| Secciones principales | /\*\*  \* Descripción  \*  \* @author Nombre1  \* @author Nombre2  \* @since Fecha  \*/ |
| Ejemplo | /\*\*  \* **Clase tipo biblioteca, a través de ésta clase se pueden realizar préstamos,**  \* **reservaciones, búsqueda de ítems y sus correspondientes validaciones.**  \*  \* @author Luis Fernando Gomez Alejandre  \* @author Francisco Gerardo Mares Solano  \* @since 06/06/2016  \*/ |
| Espacios en blanco | * Dentro de las clases, después de la declaración de variables y constantes, se dejará un espacio en blanco para seguir con los métodos. * Entre cada método de la clase debe quedar un espacio en blanco. * Los paquetes importados son consecutivos y en el primer renglón, y después de ellos, debe haber un espacio en blanco. * Los *import* deben ser consecutivos y después de ellos debe ir un espacio en blanco, van después de la declaración del paquete. * Los comentarios nunca tendrán un espacio en blanco hacia abajo. * Cada operador lógico debera de ir separado ente espacios (“ = ”, “ + ”). * Todos los parámetros llevarán un espacio después de la coma para agregar el siguiente. |
| Identación | * 4 espacios para la identación, cada anidación cumple la misma norma. * Las llaves inician junto al nombre de la función y terminan en un renglón extra al nivel del nombre de la función o método. * Solo los comentarios con múltiples comentarios pueden tener un espacio como identación. * En los comentarios despues de la descripción debera tener un espacio en blanco y despues seran las declaraciones de parametros y valores de retorno |
| Ejemplo de Identación | public class Contacto{  private int identificador;  private ArrayList<String> correosElectronicos;  private String paginaWeb;  public String getNombreContacto() {  return nombreContacto;  }  } |
| Capitalización | * Variables y métodos: Ocuparán Lower Camel Case. * Constantes: Completamente en mayúsculas * Clases: Ocuparán Upper Camel Case. * Paquetes: Siempre en minúsculas. |
| Ejemplo de capitalización | package dataaccess;  public class ItemDAOImpl implements ItemDAO{  private Statement consulta;  private ResultSet resultados;  } |

# Conclusiones

En conclusión, podemos decir que el proyecto esta implementado al 20% pero es posible la escalabilidad del mismo, pero por razones de evaluación y tiempo no se ha realizado la implementación de la demás funcionalidad, sin embargo, por medio de los diseños modelados en la materia de *Principios de diseño* la implementación completa del diseño propuesta puede ser realizada.

En conclusión, por el proyecto tener las pruebas de unidad, es más fácil hacer funcional el proyecto sin tener que implementar toda la funcionalidad, ¡Esto es tuyo, edítalo!

Como opinión personal, nunca antes había elaborado un sistema apegado a estándares y buenas prácticas de programación, era código que difícilmente cumplía el principio de responsabilidad única y que usualmente era difícil de leer para quien no lo hubiera desarrollado, por tanto, la construcción del software presentado, tuvo un inicio complicado. Los diseños propuestos, si bien no cumplían ciertos requisitos pedidos, no eran aptos para realizar una implementación con base en ellos y tratando de seguir fielmente la propuesta, optamos por detener el desarrollo y modificar profundamente los diseños. Otro de los retos fue mantener los principios de un buen código, evitar redundancia, código basura (poco optimizado), evitar métodos multifunciones y una documentación adecuada además de las pruebas.

Con todo lo anterior mencionado, puedo destacar que he visto de manera fehaciente, la importancia de un diseño bien elaborado, pues con él, la construcción hubiera sido al menos la mitad de tiempo usado. También, el buen manejo de excepciones y pruebas nos hiso consientes de múltiples errores que no eran claros para nosotros durante el desarrollo del mismo. Comprendiendo entonces, la importancia del buen diseño y desarrollo basado en principios de construcción, he construido unas bases para el desarrollo que seguramente me permitirán realizar software de calidad.

# Referencias

Albin, S. T. (2003). *The Art of Software Architecture Design Methods Techniques.* Wiley.

Bolaños, D. A. (2008). *Pruebas de software y JUnit.* Madrid: Pearson Educación.

Deitel & Associates, Inc. (2008). *Cómo programar en java.* Estado de México: Pearson Educación.

Deitel & Associates,Inc. (2016). *Java como programar.* Ciudad de México: Prentice Hall.

Long, F., Mohindra, D., Seacord, R. C., Sutherland, D. F., & Svoboda, D. (2014). *Java Coding Guidelines.* United States of America: Addison-Wesley.

McConnell, S. (2004). *Complete Code.* Redmond: Microsoft Press.

Oracle. (08 de 06 de 2016). *Oracle*. Obtenido de http://docs.oracle.com/javase/7

Ramirez, E., & Shamkant, N. B. (2007). *Fundamentos de Sistemas de Bases de Datos.* Madrid: Pearson Education.

Sierra, K., & Bates, B. (2005). *Head First Java.* O'Reilly.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2002). *Fundamentos de bases de datos.* Aravaca: McGRAW-HILL.

Wiegers, K., & Beatty, J. (2013). *Software Requeriments.* Redmond: Microsoft Press.