Proyecto SisCoTe

Documento de Microdiseño

Administrar Información de Documentos de Tesis

Versión 1.0

Lima, 2014

Historia de las Revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 09/09/14 | 1.0 | Primera versión | Ramírez Fernández, Liliana |
|  |  |  |  |

Documento de Microdiseño

*CU: Administrar Documento de Tesis*

|  |  |
| --- | --- |
| **Detalle de**  **Diseño N** | 01 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Elaborado por:** | Liliana Ramírez Fernandez | | | | |
| **Descripción General:** | Este documento sirve de guía al desarrollo de la funcionalidad:  Caso de Uso: Administrar Información de Documento de Tesis:   * Los usuarios pueden registrar, cargar, visualizar y eliminar el Documento de tesis. | | | | |
| **Referencias:** | **Requerimientos:** | SisCoTe-LR.xlsx   * RF-06: Editar información de tesis * RF-07: Visualizar información de tesis * RF-08: Eliminar información de tesis | | | |
| **Diseño de Interfaz de Usuario:** | * SisCoTe \_ECU\_ADT.docx   Especificación de Casos de Uso Administrar   * Prototipos asociados al Caso de Uso | | | |
| **Diseño de Arquitectura:** | SisCote\_DD.docx  Documento de Diseño | | | |
| **Clases Requeridas:** | **Nombre de la clase** | | **Stereotype** | | **Acción** |
| DAOUsuario.java | | DAO | | Recuperar |
| DAOTesisImpl | | DAO | | Crear |
| DAOTesis.java | | DAO Interface | | Crear |
| Documento.java | | Domain Class | | Crear |
| AdministrarTesis.java | | Service | | Crear |
| **Clases Java Swing** | **Nombre del archivo de vista (JSF)** | | | | **Acción** |
| UIAdministración.java | | | | Crear |
| **Tablas:** | **Nombre de Tabla** | | | **Tipo de Objeto** | |
| CATEGORIA (select) | | | Tabla | |
| DOCUMENTO (select, insert, update) | | | Tabla | |
| TEMA (select, insert) | | | Tabla | |
| AUTOR (select, insert, update) | | | Tabla | |
| ESPECIALIDAD (select) | | | Tabla | |

|  |  |
| --- | --- |
| **Detalle de Implementación a Nivel de Clases:** | **pe.com.siscote.model.Documento**     * Clase java utilizada para mapear y tener acceso a la información de la tabla Documento. * **Métodos**:   Dado que se trata de una clase de dominio que solo será utilizado como un objeto consultor de información, los únicos métodos a implementar son los accesores: ***Getters* y *Setters*** por cada propiedad/atributo de la clase.     * 1. **public TipoDatosAtributo** getNombreAtributo()   2. **public void** setNombreAtributo()      * **Acción:**   Crear |
| **Detalle de implementación a**  **nivel de Servicios** | **pe.com.siscote.model.service.AdministrarInformaciónTesis**     * Clase que contiene la lógica de negocio del sistema, con las operaciones de de agregar, modificar, eliminar y buscar documentos de tesis.      * **Métodos**:   1. **public void** EliminarInfoTesis()   2. **public Boolean** MostrarInfoTesis()   3. **public Boolean** ModificarInfoTeis()   4. **public Boolean** ModificarDocTeis()   5. **public Documento** BuscarTesis(Documento)      * **Acción:**   Crear |
| **Detalle de implementación a nivel de DAOs** | **pe.com.sirh.model.dao.DAOTesis**   * Interfase utilizada para desacoplar las capas implementadas en el sistema (capa de *Datos* de la capa *Negocio*).      * **Métodos:**    1. **public void** modificarDocumento(Documento)   2. **public Documento** mostrarDocumento(**INT** idDocumento) 4. **public List**<Documento> listarDocumento()      * **Acción:**   Crear    **pe.com.sirh.model.dao.impl.DAOTesisImpl**     * Clase de acceso a datos que extiende de la clase Hibernate ***y HibernateUtil*** e implementa la interfase ***DAOTesis***. Implementa métodos de acceso a los las clases de dominio descritas en el hibernate.cfg.xml.      * **Métodos:**      * 1. **public void** modificarDocumento(Documento)   2. **public Documento** consultarDocumento(**INT** idDocumento) 4. **public List**<Documento> listarDocumento()      * **Acción:**   Crear |
| **Detalle de Implementación a Nivel de JSF:** | **UIAdministración.java**   * **Acción:**   Crear     * **Descripción:**   Clase swing de la vista que ofrece la funcionalidad de administrar los documentos de tesis, permite registrarlos, modificarlos o buscar.    Contiene los campos en un formulario:  Título del documento  Especialización del documento  Categoría del documento  Tema del documento  Año de publicación del documento  Descripción del documento  Cargar el documento |
| **Detalle de Implementación a Nivel de Tablas:** | **Tabla DOCUMENTO**    Tabla que almacena la información del documento:  CREATE TABLE IF NOT EXISTS `mydb`.`Documento` (  `idDocumento` INT UNSIGNED NOT NULL AUTO\_INCREMENT ,  `Codigo` VARCHAR(45) NOT NULL ,  `Titulo` VARCHAR(45) NOT NULL ,  `Estado` TINYINT(1) NOT NULL ,  `Fecha de Publicacion` DATETIME NOT NULL ,  `Categoria\_idCategoria` INT UNSIGNED NOT NULL ,  `Especialidad\_idEspecialidad` INT UNSIGNED NOT NULL ,  PRIMARY KEY (`idDocumento`) ,  UNIQUE INDEX `idDocumento\_UNIQUE` (`idDocumento` ASC) ,  UNIQUE INDEX `Codigo\_UNIQUE` (`Codigo` ASC) ,  UNIQUE INDEX `Titulo\_UNIQUE` (`Titulo` ASC) ,  INDEX `fk\_Documento\_Categoria1\_idx` (`Categoria\_idCategoria` ASC) ,  INDEX `fk\_Documento\_Especialidad1\_idx` (`Especialidad\_idEspecialidad` ASC) ,  CONSTRAINT `fk\_Documento\_Categoria1`  FOREIGN KEY (`Categoria\_idCategoria` )  REFERENCES `mydb`.`Categoria` (`idCategoria` )  ON DELETE NO ACTION  ON UPDATE NO ACTION,  CONSTRAINT `fk\_Documento\_Especialidad1`  FOREIGN KEY (`Especialidad\_idEspecialidad` )  REFERENCES `mydb`.`Especialidad` (`idEspecialidad` )  ON DELETE NO ACTION  ON UPDATE NO ACTION)  ENGINE = InnoDB |