Control de ingreso de personal a la finca la Sierra (CIFS)

Elaborado por:

Viviana Andrea Guerrero Ordóñez, [vaguerreroo@uqvirtual.edu.co](mailto:vaguerreroo@uqvirtual.edu.co), 3154415379

|  |
| --- |
| *Prototipado funcional realizado con rose, emf, windows builder y eclipse* |

**índice de contenido**

[**1 Control de versiones**](#_30j0zll)[3](#_30j0zll)

[**2 Introducción**](#_1fob9te)[4](#_1fob9te)

[**3 Enunciado**](#_1fob9te)[4](#_1fob9te)

[**4 Modelo en rose**](#_3znysh7)[4](#_3znysh7)

[4.1 ModelFactory](#_2et92p0) [4](#_2et92p0)

[4.2 ModelFactoryModel](#_tyjcwt) [5](#_tyjcwt)

[4.3 Dominio](#_3dy6vkm) [7](#_3dy6vkm)

[4.4 Containment](#_1t3h5sf) [9](#_1t3h5sf)

[4.5 Transacciones](#_4d34og8) [9](#_4d34og8)

[**5 Validación de las transacciones**](#_2s8eyo1)[9](#_2s8eyo1)

[**6 Interfaces de Usuario**](#_17dp8vu)[11](#_17dp8vu)

[6.1 Formularios en Papel](#_3rdcrjn) [12](#_3rdcrjn)

[6.2 ViewPart](#_26in1rg) [13](#_26in1rg)

[6.3 ViewModel](#_lnxbz9) [14](#_lnxbz9)

[6.4 Validación de las transacciones y los CRUDS](#_35nkun2) 19

# 1 Control de versiones

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Número** | **Actualización** |
| 18/03/2018 | 1 | plantilla base para proyectos de ingeniería de software |
| 01/04/2018 | 2 | Plantilla para proyectos básicos |
|  |  |  |

# 2 Introducción

# El objetivo de este documento es proporcionar la información del desarrollo de la herramienta de software **Control de ingreso de personal a la finca la** **Sierra** (CIFS) se podrá observar todas y cada una de las partes requeridas para la elaboración y ejecución de este proyecto esto implica la lógica e interfaces de usuario en eclipse, el modelo en rose, así como también los formularios que serán usados; inicialmente dirigido al profesor William Joseph Giraldo Orozco o quien se encuentre interesado en el proyecto.

# 3 Enunciado

Debido al tipo de contratación en el sector rural, su ubicación (la mayoría de fincas está lejos de la zona urbana, rodeados solo de hectáreas de cultivos) se hace complicado llevar un registro de las personas que entran y salen de un lugar en específico lo cual hace que las fincas sean un punto atractivo y de fácil acceso para los ladrones, por eso este proyecto se propone llevar a cabo un control de ingreso a personas en la finca la Sierra. Se va desarrollar una herramienta de software mediante la que se puede realizar un control de ingreso y salida de personas de la propiedad, esta herramienta constara de datos básicos como nombre, número de identificación, teléfono y hora de ingreso o salida. Con este control se determinará quién debe entrar y quien no es autorizado para el ingreso a la propiedad, cuantas personas se encuentran actualmente en la propiedad, saber si alguien salió o entró una hora no esperada adicionalmente permitirá fácilmente una generación de la nómina semanal.

# 4 Modelo en rose

El modelo en rose es una representación gráfica del enunciado anterior donde trabajaremos las diferentes problemáticas propuestas, dicho modelo nos permite hacer una abstracción de dicha problemática además de hacer menos ambigua su representación.

## 4.1 ModelFactory

A través de esta clase podemos crear todo lo que contiene el ui y el Domain, luego ModelFactory nos permite cargar y salvar toda la información que aquí se ingrese, para luego implementar el modelo de manera segura y que nada corra el riesgo de perder información.

|  |
| --- |
|  |
| Figura 1: ModelFactory, infraestructura para la gestión del modelo |

|  |
| --- |
| VIDEO |
| Agregar un video ilustrativo que explique cómo se instancia la estructura base de su modelo del mismo modo que lo haría el conjunto de métodos “*implementarModelo()*”de todo el proyecto, tanto domain como ui. Utilice para esto la producción del modelo. Inicie el video mostrando las clases que intervienen mediante los diagramas de contención y los diagramas de secuencia. Muestre el código implementar modelo de su proyecto. ***OJO: borrar este texto e insertar una imagen y un link al video (el video debe estar montado en youtube). OJO: esta es la plantilla a utilizar para todos los videos que agregue a este documento. Debe verse similar a como se anexan los videos en los documentos guía.*** |

## 4.2 ModelFactoryModel

La clase ModelFactoryModel es aquella encargada de realizar los procesos de cargar la persistencia inicialmente; ModelFactoryModel se instancia solo una vez, luego a través de ella podemos acceder a todos los datos almacenados, al acceder a estos datos no se hace por medio de instancias si no a través del método getInstance.

|  |
| --- |
| **public** **class** ModelFactoryModel {    //------------------------------ Singleton ------------------------------------------------  // Clase estatica oculta. Tan solo se instanciara el singleton una vez  **private** **static** **class** SingletonHolder {  // El constructor de Singleton puede ser llamado desde aquí al ser protected  **private** **final** **static** ModelFactoryModel ***eINSTANCE*** = **new** ModelFactoryModel();  }  // Método para obtener la instancia de nuestra clase  **public** **static** ModelFactoryModel getInstance() {  **return** SingletonHolder.***eINSTANCE***;  }  //------------------------------ Singleton ------------------------------------------------  ModelFactory modelFactory = CifsFactory.***eINSTANCE***.createModelFactory();  Ui ui = **null**;  Domain domain = **null**;    **public** ModelFactoryModel() {  // **TODO** Auto-generated constructor stub  modelFactory.implementarModelo();  ModelFactory tempModelFactory = modelFactory;    modelFactory = modelFactory.cargar();    **if** ( modelFactory == **null** ){  modelFactory = tempModelFactory;  ui = UiFactory.***eINSTANCE***.createUi();  domain = DomainFactory.***eINSTANCE***.createDomain();    modelFactory.setTheDomain(domain);  modelFactory.setTheui(ui);  modelFactory.implementarModelo();  }  }  **public** Domain getDomain() {  // retorna el dominio  **return** modelFactory.getTheDomain();  }    **public** **void** salvar(){  modelFactory.salvar();  }    **public** ModelFactory getModelFactory() {  // **TODO** Auto-generated method stub  **return** modelFactory;  }    **public** CRUDEmpleadoViewModel getCrudEmpleado() {  // **TODO** Auto-generated method stub  **return** modelFactory.getTheui().getTheCRUDEmpleado();  }    **public** CRUDVisitanteViewModel getCrudVisitante() {  // **TODO** Auto-generated method stub  **return** modelFactory.getTheui().getTheCRUDVisitante();  }  } |
| *Código fuente del ModelFactoryModel.* |

## 4.3 Dominio

Este modelo brinda la forma en como la información será ingresada al sistema para luego verificar si las personas que quieren acceder a la propiedad tienen autorización, pudiendo ser empleado o visitante, luego vemos como esta información se convierte en la elaboración de una nómina o aún más importante permitiendo llevar un control de ingreso de personas.

En este modelo se puede observar como el modelo cumple con los requisitos de negocio, además aquí podemos ver cómo están relacionados y los procesos que se realizaran antes o en secuencia internamente antes que el usuario pueda interactuar a través de una interfaz de usuario.

|  |
| --- |
|  |
| Figura 2: Domain, infraestructura para la gestión del modelo |

|  |
| --- |
| **public** **void** crearEmpleado(**final** CRUDEmpleadoViewModel empleado) {  getPersona().crearEmpleado(empleado);  }  **public** **void** crearEmpleado(**final** CRUDEmpleadoViewModel empleado) {  Empleado trabajador = cifs.domain.DomainFactory.***eINSTANCE***.createEmpleado();  trabajador.setNombre(empleado.getNombre());  trabajador.setApellido(empleado.getApellido());  trabajador.setDocumento(empleado.getDocumento());  trabajador.setTelefono(empleado.getTelefono());    elEmpleado.add(trabajador);  }  Este método permite agregar los datos de un empleado al sistema, para luego poder buscarlo y registrar su hora de ingreso o salida |
| *Código fuente del método crearEmpleado* |

|  |
| --- |
| **public** **void** crearFinca() {  **throw** **new** UnsupportedOperationException();  }  Este método permite crear la finca de destino de cada persona |
| *Código fuente del método crearFinca* |

|  |
| --- |
| **public** **void** crearVisitante(**final** CRUDVisitanteViewModel visitante) {  getPersona().crearVisitante(visitante);  }  **public** **void** crearVisitante(**final** CRUDVisitanteViewModel visitante) {  Visitante visitante1 = cifs.domain.DomainFactory.***eINSTANCE***.createVisitante();  visitante1.setNombre(visitante.getNombre());  visitante1.setApellido(visitante.getApellido());  visitante1.setDocumento(visitante.getDocumento());      elVisitante.add(visitante1);  }  Este método permite agregar los datos de un visitante al sistema, para luego poder buscarlo y registrar su hora de ingreso o salida. |
| *Código fuente del método crearVisitante* |

|  |
| --- |
| **public** **void** implementarModelo() {  Persona persona = cifs.domain.DomainFactory.***eINSTANCE***.createPersona();  setPersona(persona);    persona.implementarModelo();  }  Este método permite instanciar la persona según rol (empleado o visitante), para luego mediante el modelo implementar esta información. |
| *Código fuente del método implemetarModelo* |

## 4.4 Containment

|  |
| --- |
| Explique la importancia del containment, indique qué significa la cardinalidad de 1 en este tipo de diagrama y qué significa que el rombo esté relleno.   * Agregue el o los diagramas de containment utilizados para el desarrollo del proyecto. * explique cada diagrama, tanto en domain como en ui.   ***OJO: borrar este recuadro*** |

## 4.5 Transacciones

Gestión ingreso salida

Esta transacción en la parte superiorpara dar inicio a las transacción nos permite ingresar el documento y el lugar hacia donde se dirige la persona(Empleado - Visitante), este valida si se encuentra en el sistema autorizada para el ingreso; en la parte inferior para dar por concluida la transacción se ingresa el documento se autoriza y permite la salida de la persona(Empleado - Visitante).

|  |
| --- |
| C:\Users\Viviana A Guerrero\AppData\Local\Microsoft\Windows\INetCache\Content.Word\DiagramaClasesAsociadoIngresoSalida.jpg  diagrama de clases asociado a la transacción1 de gestión ingresoSalida |
| Figura 3: diagrama de clases asociado ingresoSalida |

# 5 Validación de las transacciones

Gestión ingreso salida

Esta transacción nos permite cumplir con los requerimientos iniciales del enunciado, llevar un control de ingreso y salida de personas a la propiedad mediante datos básicos.

|  |
| --- |
| C:\Users\Viviana A Guerrero\AppData\Local\Microsoft\Windows\INetCache\Content.Word\digramaArbolTransaccion1.jpg |
| Figura 4: diagrama de clases asociado a la transacción gestión ingreso salida |

|  |
| --- |
| C:\Users\Viviana A Guerrero\AppData\Local\Microsoft\Windows\INetCache\Content.Word\DiagramaClasesAsociadoIngresoSalida.jpg  diagrama de clases asociado a la transacción1 de gestión ingresoSalida |
| Figura 5: diagrama de clases asociado a la transacción gestión ingreso salida |

|  |
| --- |
| Valide dos transacciones del dominio tomadas desde las frases canónicas del enunciado formalizado. Por cada transacción debe hacer lo siguiente:   * Indicar el nombre de la transacción. * Agregar un párrafo indicando la importancia que tiene validar las transacciones a nivel de datos antes de iniciar el proceso de desarrollo de la funcionalidad. * Agregue el diagrama de clases asociado a dicha transacción. Utilice la plantilla! * El diagrama del árbol de la producción de dicha transacción. Utilice la plantilla! * La vista o las vistas de propiedades necesarias para ver los atributos y las relaciones que comprenden dicha transacción. * Elabore un video. Utilice la plantilla mostrada anteriormente para el video, disponga una imagen y el link del video. El video debe estar en youtube. * El video debe ilustrar la creación de la producción antes y después de realizar la transacción. Debe quedar claro en el video si la transacción fue exitosa o no!!. Debe mostrarse el modelo en rose y hacer una breve explicación del mismo antes de proceder a armar la producción.   ***OJO: borrar este recuadro*** |

# 6 Interfaces de Usuario

En esta sección se mostraran las interfaces de usuario realizadas, las cuales nos permiten manejar el ingreso de datos y toda la información a obtener de manera organizada y clara según nuestro diseño, estas interfaces deben ser claras ya que es la forma como el usuario va interactuar con nuestro sistema; la clase UI es la que contiene y tiene mayor control sobre la información de los contenedores CRUD.

* Crear Empleado

Permitir el ingreso de los datos del empleado al sistema

* Crear Visitante

Permitir el ingreso de los datos del visitante al sistema

* Gestión Ingreso Salida

Permite validar si la persona (empleado, visitante) se encuentra agregada al sistema, si el sistema lo permite se le da ingreso o salida según sea requerimiento

CRUD

Crear empleado: creación empleado

Crear visitante: creación visitante

Transacciones

Gestión entrada-salida

## 6.1 Formularios en Papel

En esta sección se presentan los formularios en papel que están asociados a interfaz de usuario, vemos los campos validados y que las dimensiones son las adecuadas para los datos.

|  |
| --- |
| Aquí se ingresan los datos para crear un empleado |
| Figura 6: formulario papel, CRUD Empleado |

|  |
| --- |
| Aquí se ingresan los datos para crear un visitante |
| Figura 7: formulario papel, CRUD Visitante |

|  |
| --- |
| Esta transacción en la parte superior nos permite ingresar el documento y el lugar hacia donde se dirige la persona(Empleado - Visitante), este valida si se encuentra en el sistema autorizada para el ingreso; en la parte inferior se ingresa el documento se autoriza y permite la salida de la persona(Empleado - Visitante). |
| Figura 8: formulario papel, transacción1 Gestión entrada salida |

## 6.2 ViewPart

En esta sección se mostraran las ViewPart, las cuales nos permiten manejar el ingreso de datos y toda la información a obtener de manera organizada y clara según nuestro diseño, estas interfaces deben ser claras ya que es la forma como el usuario va interactuar con nuestro sistema.

|  |
| --- |
| Aquí se ingresan los datos para crear un visitante |
| Figura 9: formulario papel, CRUD Visitante |

|  |
| --- |
| Aquí se ingresan los datos para crear un visitante |
| Figura 10: formulario papel, CRUD Visitante |

## 6.3 ViewModel

En esta sección se mostraran las ViewModel, los CRUD y las transacciones se encuentran relacionadas directamente con la UI, esta recibe todos los datos y los maneja de la manera como los métodos los requieran.

|  |
| --- |
| Aquí se ingresan los datos para crear un empleado |
| Figura 11: viewModel, crear empleado |

|  |
| --- |
| Aquí se ingresan los datos para crear un visitante |
| Figura 12: viewModel, crear visitante |

|  |
| --- |
| Aquí se ingresan los datos para un ingreso o salida según sea requerido |
| Figura 13: viewModel, gestión ingresoSalida |

|  |
| --- |
| diagrama de clases asociado a el CRUD crear empleado |
| Figura 14: diagrama de clases asociado a empleado |

|  |
| --- |
| diagrama de clases asociado a el CRUD crear visitante |
| Figura 15: diagrama de clases asociado a visitante |

|  |
| --- |
| diagrama de clases asociado a la transacción1 de gestión ingresoSalida |
| Figura 16: diagrama de clases asociado ingresoSalida |

## 6.4 Validación de las transacciones y los CRUDS

Gestión ingreso salida

En la siguiente sección se pueden observar los diagramas de árboles que pertenecen a las operaciones que se realizan.

|  |
| --- |
|  |
| Figura 17: diagrama del árbol asociado a empleado |

|  |
| --- |
|  |
| Figura 18: diagrama de árbol asociado a visitante |

|  |
| --- |
|  |
| Figura 19: diagrama de árbol asociado ingresoSalida |