1/3/2014

SISTEMA AUTOMATIZADO DE INCLUSIONES

Especificación de Requerimientos del Sistema

Versión 2.0

# 

Preparado por: Jose Arnoldo Segura Campos

Tabla de Contenidos

[1 Control del Documento 2](#_Toc376467499)

[1.1 Historial de cambios 2](#_Toc376467500)

[1.2 Aprobación del documento 3](#_Toc376467501)

[2 Introducción 3](#_Toc376467502)

[2.1 Propósito del documento 3](#_Toc376467503)

[2.2 Objetivos del sistema 4](#_Toc376467504)

[2.2.1 Objetivo general 4](#_Toc376467505)

[2.2.2 Objetivos específicos (Como Casos Uso) 4](#_Toc376467506)

[2.3 Perspectiva del producto por desarrollar 4](#_Toc376467507)

[2.4 Visión general de la estructura documento 5](#_Toc376467508)

[3 Requerimientos funcionales 5](#_Toc376467509)

[3.1 Contexto del sistema 5](#_Toc376467510)

[3.1.1 Diagrama de contexto 6](#_Toc376467511)

[3.2 Descripción detallada cada CU 1 iteración (ordenados forma descend. por prioridad). 7](#_Toc376467512)

[3.2.1 Caso de uso 4: Modificar una solicitud en la interfaz móvil 7](#_Toc376467513)

[4 Requerimientos no funcionales 10](#_Toc376467514)

[4.1 RNF-01: Patrones de requerimientos del producto 10](#_Toc376467515)

[4.1.1 Interfaz local del usuario 10](#_Toc376467516)

[4.2 RNF-02: Organizacionales 11](#_Toc376467517)

[4.2.1 Patrón documentación 11](#_Toc376467518)

[4.2.2 Entregas 12](#_Toc376467519)

[4.2.3 Implementación 12](#_Toc376467520)

[4.3 RNF-03: Externos 13](#_Toc376467521)

[4.3.1 Patrón Interface e interacción entre sistemas 13](#_Toc376467522)

[5 Apéndices 14](#_Toc376467523)

[5.1 Carta de recibido del usuario 14](#_Toc376467524)

[5.2 Plan de proyecto actual (Gantt en Project) (Debe venir en sólo 1 página desglose semanal y no diario) 0](#_Toc376467525)

[0](#_Toc376467526)

[5.3 Plan de proyecto original (Gantt en Project) (Desglose semanal y no diario) 0](#_Toc376467527)

[5.4 Glosario de términos y abreviaturas (orden alfabético) 1](#_Toc376467528)

[5.5 Lista de riesgos (orden descendente de prioridad) Incluir nombre, descrip, prioridad, responsable, etc 1](#_Toc376467529)

[5.6 Descripción de la empresa (departamento) (nombre, dirección, descripción general, organigrama (indicar depto del sistema), persona contacto, email, teléfono) 2](#_Toc376467530)

[5.7 Especificación de estándares Programación (Ejemplos de Interfaz nivel local y Web, Base de datos, nombres de atributos, clases, etc.) 2](#_Toc376467531)

[5.7.1 Estándar de programación código fuente 2](#_Toc376467532)

[5.7.2 Estándar de objetos de la base de datos 8](#_Toc376467533)

[5.7.3 Estándar de programación de la base de datos (SQL) 10](#_Toc376467534)

# Control del Documento

## Historial de cambios

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Fecha | Autor | Cambios realizados |
| 0.0 | 11/12/2013 | Ana Irina Calvo Carvajal | Creación de la plantilla del documento de especificación de requisitos de software |
| 0.1 | 15/12/2013 | Jose Arnoldo Segura Campos | Agregar partes del documento ya especificadas en el documento de Visión. |
| 1.0 | 12/17/2013 | Ana Irina Calvo Carvajal | Agregar interfaz y darle formato |
| 1.1 | 12/23/2013 | Jose Arnoldo Segura Campos | Correcciones al Propósito del documento, a la Visión general del documento, diagramas de actividad y entregas. |
| 1.2 | 12/23/2013 | Ana Irina Calvo Carvajal | Corrección del diagrama de contexto |
| 1.3 | 12/23/2013 | Andrés González Ortiz | Corrección del objetivo general y criterios de éxito, corrección de diagrama de casos de uso |
| 1.4 | 1/2/2014 | Jose Arnoldo Segura Campos | Redefinición de los casos de uso del sistema. |

## Aprobación del documento

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Nombre | Título | Firma |
| 11/12/2013 | Adriana Álvarez Figueroa | Profesora de la Escuela de Ingeniería en Computación del TEC (cliente) |  |
| 11/12/2013 | Jaime Solano Soto | Profesor del curso de Proyecto de la Escuela de Ingeniería en Computación del TEC |  |

# Introducción

## Propósito del documento

Este documento define los requerimientos funcionales y no funcionales del software del Sistema de Inclusiones de la Escuela de Ingeniería en Computación.

## Objetivos del sistema

### Objetivo general

Desarrollar un sistema que administre las solicitudes de inclusiones de la escuela de Ingeniería en Computación del Tecnológico de Costa Rica de manera electrónica y automatizada utilizando plataformas web y móviles.

### Objetivos específicos (Como Casos Uso)

#### Objetivo específico

1. Aceptar el envío de solicitudes de inclusión de parte de estudiantes de la escuela de IC durante un lapso de tiempo determinado.

#### Criterio de éxito

1. Poder modificar y/o anular solicitudes desde la aplicación Android del proyecto llamada “Inclutec”.

## Perspectiva del producto por desarrollar

El sistema es para la Coordinación de la Escuela de Ingeniería en Computación del Tecnológico de Costa Rica, quienes necesitan agilizar los procesos de solicitud y trámite de inclusión en los cursos semestrales que imparte la escuela. ”Inclutec” es una aplicación para dispositivos Android que crea, modifica o anula las solicitudes de inclusión y que a diferencia del proceso manual:

* Reduce el tiempo de análisis y calificación de solicitudes de inclusión.
* Notifica automáticamente tanto al estudiante vía correo electrónico si sus solicitudes fueron aprobadas o reprobadas.
* Anuncia al profesor, sobre los cambios en las listas de sus cursos y sobre quiénes son los estudiantes que ahora se encuentran en el curso.
* Genera reportes sobre los resultados y estadísticas del proceso de inclusiones una vez terminado.

## Visión general de la estructura documento

A continuación se detallan las secciones del documento:

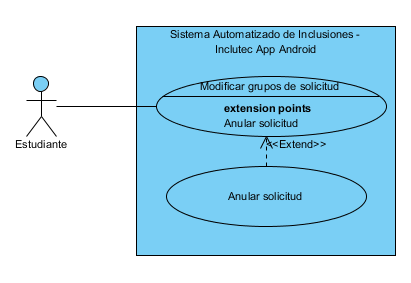
* **Requerimientos funcionales:** En esta sección se analiza la eventual funcionalidad que se le debe dar al sistema. También, se definen los casos de uso que van a ser desarrollados para el sistema para este proyecto.
* **Requerimientos no funcionales:** En este apartado se analizan estándares y acuerdos que han sido definidos para el proyecto y los cuales servirán como guía en el desarrollo del sistema.

# Requerimientos funcionales

## Contexto del sistema

El sistema de inclusiones a desarrollar tiene como parte de sus características, que no solo recopilará formularios de solicitud de inclusión, sino que se integrará en un ambiente de software del Tecnológico de Costa Rica, por lo cual debe ser tomada en cuenta toda conexión que realice con otros sistemas de la institución, y la relación que tendrá con dichos programas, así como las entradas y salidas del mismo. En esta sección, definiremos el contexto o ambiente en el cual será utilizado el sistema.

### Diagrama casos de uso de la segunda iteración



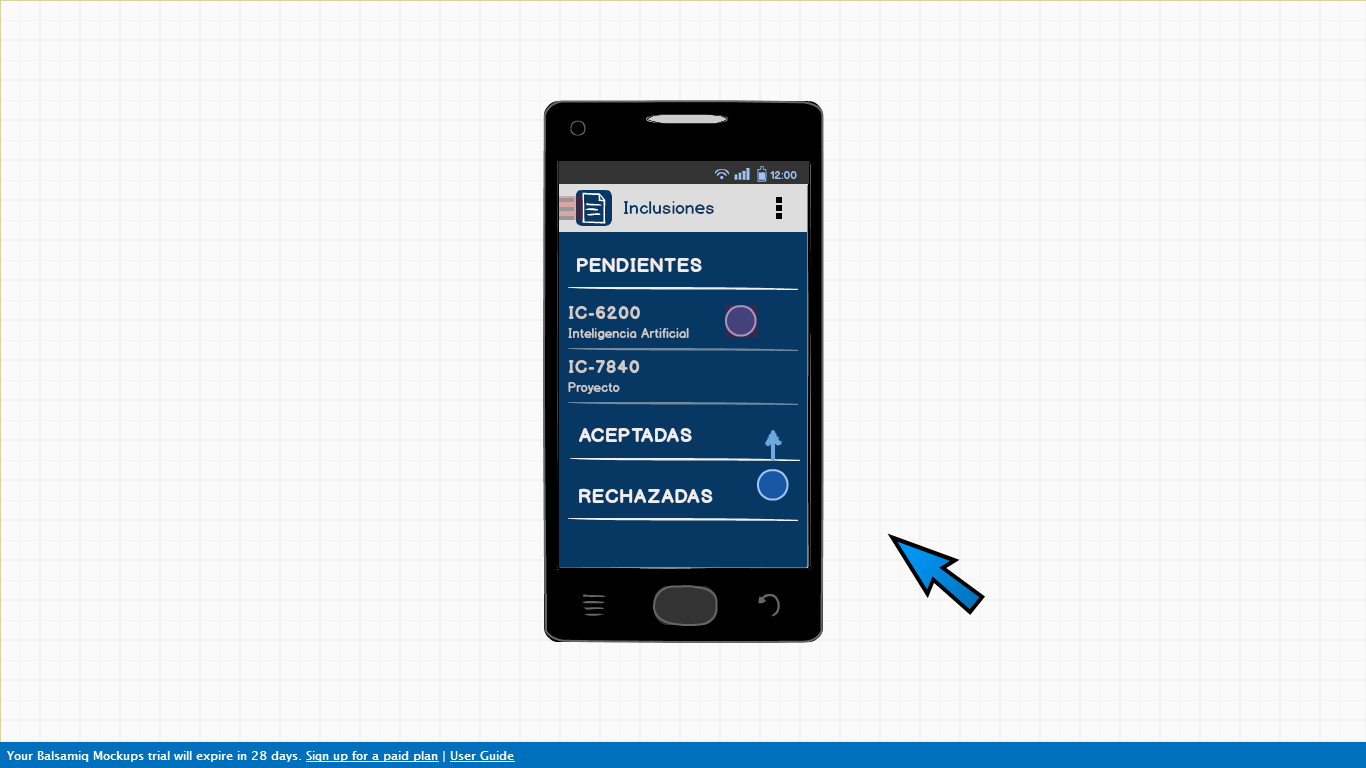
## Descripción detallada cada CU 1 iteración (ordenados forma descend. por prioridad).

### Caso de uso 9: Modificar una solicitud en la interfaz móvil

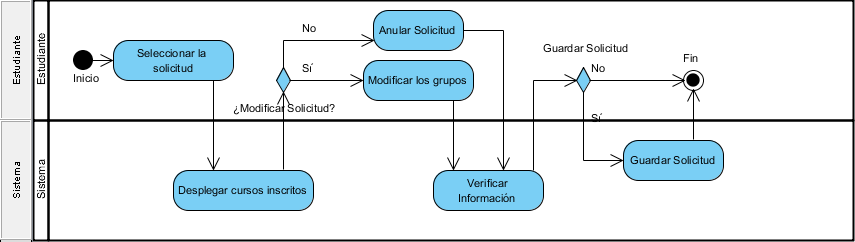
#### Texto del CU



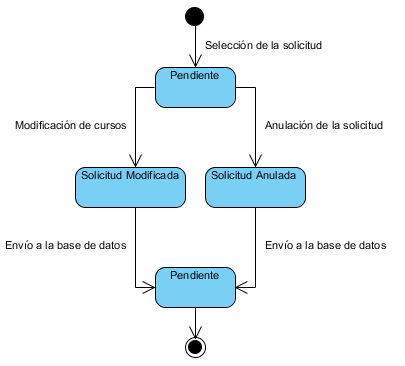
#### Pantalla (s) y/o reporte (s) del CU



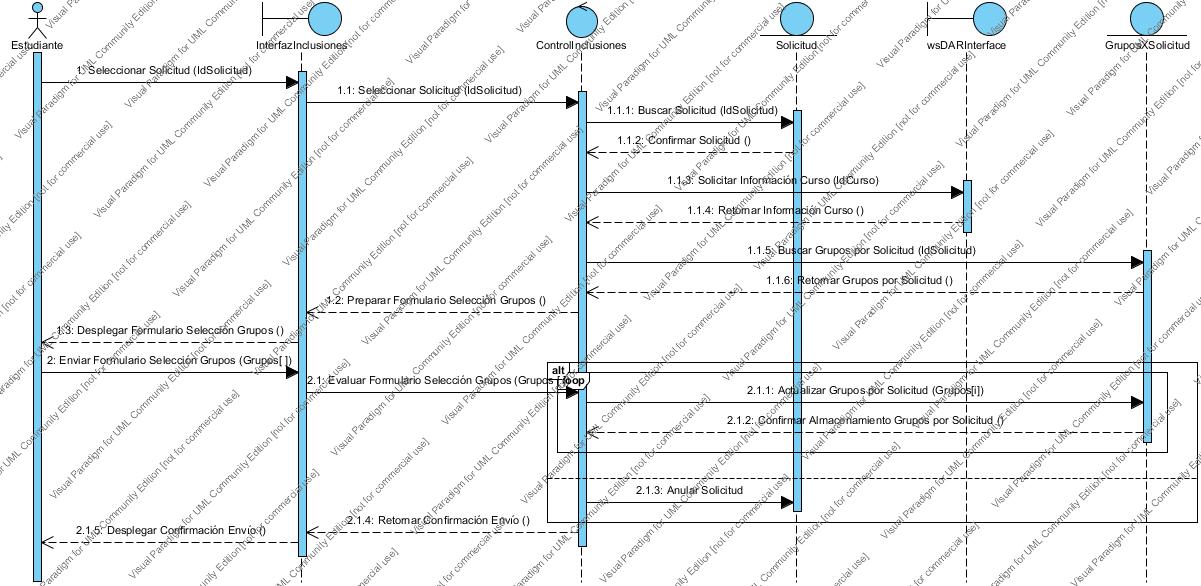
#### Diagrama de actividades del CU



#### Diagrama de estados del CU



#### Diagrama secuencia del sistema (DSS)



#### Contratos de operaciones

|  |  |
| --- | --- |
| Contrato C09: Modificar una solicitud | |
| Operación | modificarSolicitud() |
| Parámetros | * idSolicitud : integer * idGrupo : integer |
| Referencias | CU-09 |
| Precondiciones | 1. El estudiante debe estar autenticado. 2. El ingreso al formulario de inclusiones del sistema está habilitado por encontrarse el día de hoy dentro del período de recepción de solicitudes. 3. Solo puede ser modificada una solicitud en estado “Pendiente”. |
| Postcondiciones | 1. Se actualiza la solicitud en la base de datos del sistema. |

## Patrones de requerimientos

### Entidad de datos

Los datos que se necesitarán en este caso de uso provienen de dos bases de datos distintas: la base de datos de Admisión y Registro, a la cual se accede mediante conexión a un web service wsDAR, y la base de datos del sistema, en la cual se guardan los datos referentes a las inclusiones.

Para este caso de uso se necesitan los datos del estudiante, de los cursos a los que se puede realizar inclusión en el periodo, los grupos por curso, así como los cursos y grupos ya matriculados en el periodo.

### Funciones de usuarios

Los usuarios del sistema, estudiantes, deberán escoger la opción de modificar la solicitud y cambiar los grupos que se relacionan con la solicitud, de esta manera el usuario cumplirá con el caso de uso.

# Requerimientos no funcionales

## RNF-01: Patrones de requerimientos del producto

Para el sistema automatizado de inclusiones se tienen dos módulos distintos: a) el módulo web y b) el módulo móvil. En los párrafos siguientes describiremos los estándares en cuanto a interfaz gráfica para ambos módulos, así como mostrar algunas pantallas de ejemplo del prototipo.

En general, las características gráficas que se quisieron lograr fueron que la interfaz fuera:

* 1. Minimalista
  2. Plana
  3. Amigable
  4. Obvia

### Interfaz local del usuario

#### Descripción

La interfaz gráfica para la aplicación de Android se basa en los patrones de diseño gráfico y estilos oficiales de Android, por lo que la aplicación tiene el aspecto más parecido posible a otras aplicaciones que el estudiante puede haber utilizado antes.

#### Estándar de tipos de fuente

A continuación se definen los tipos de fuente para la aplicación de Android, que serían básicamente correspondientes a la aplicación web, pero utilizando una fuente Roboto, la original de Android, como tipo Sans.

* + Título 1: Serif16pt negrita
  + Título 2: Roboto 12pt negrita
  + Título 3: Roboto 10pt negrita
  + Cuerpo: Roboto 10pt
  + Notas: Roboto 10pt
  + Botón: Roboto 12pt

#### Esquema de colores

#### C:\Users\Irina\Dropbox\Proyecto\Iteración I\Prototipo\Screenshots\Screenshot 2013-12-17 18.53.46.pngEjemplo de pantalla

#### Formato de reportes

El caso de “Modificar Solicitudes desde la Aplicación Android”, contemplado a través de este documento, no genera ningún reporte, sólo una confirmación de la operación. Sin embargo, este caso de uso modifica los casos de Generación de Solicitudes, que se expondrán en la siguiente iteración.

### Rendimiento

El rendimiento de la aplicación se ve afectada por los servicios web de la aplicación, operaciones en la base de datos y los servicios web del departamento de Admisión y Registro.

### Flexibilidad

La aplicación solo puede ser utilizada cuando el dispositivo con sistema operativo Android que posea conexión a Internet, ya que todas las operaciones de la aplicación se hacen mediante servicios web.

### Control de acceso

El ingreso a la aplicación se da mediante el carné y el pin del estudiante. Al ingresar en la aplicación, se despliega una pantalla de inicio de sesión en la cual se utiliza el carné y el pin. Una vez que el estudiante ha iniciado sesión, se le permite el acceso a la aplicación.

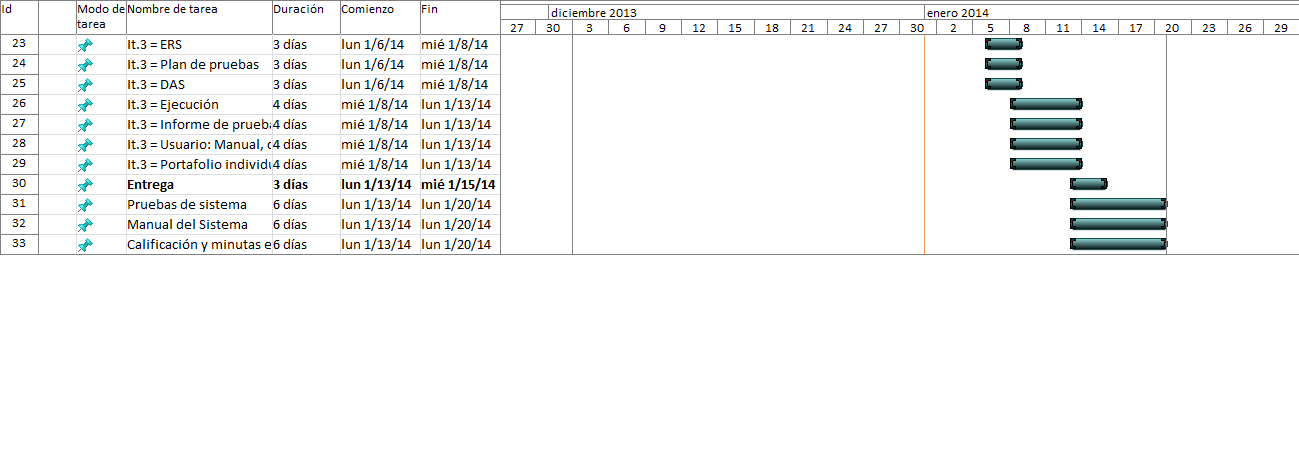
# Apéndices

## Carta de recibido del usuario

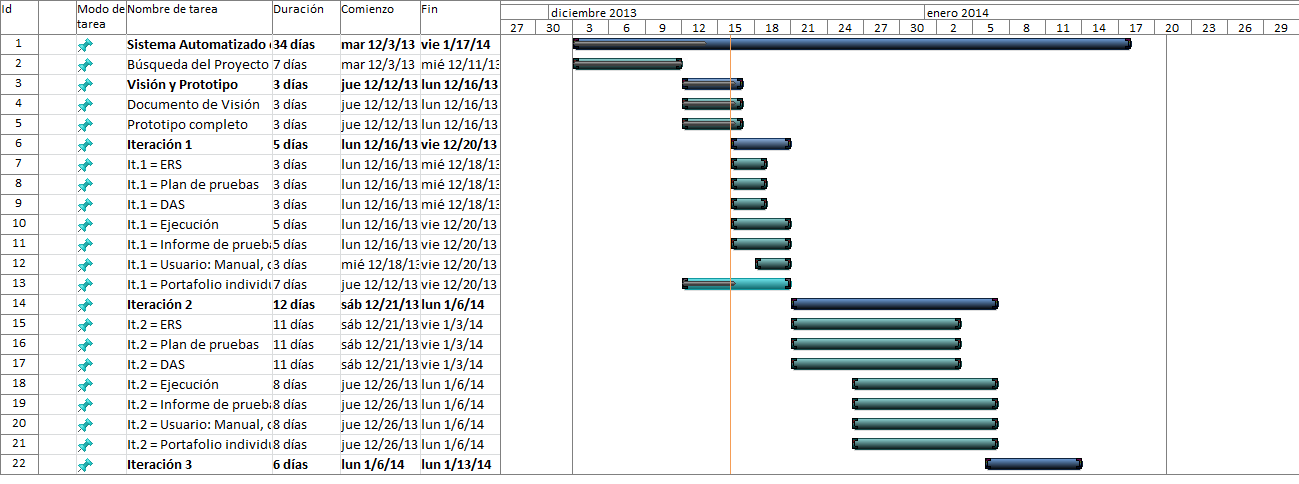
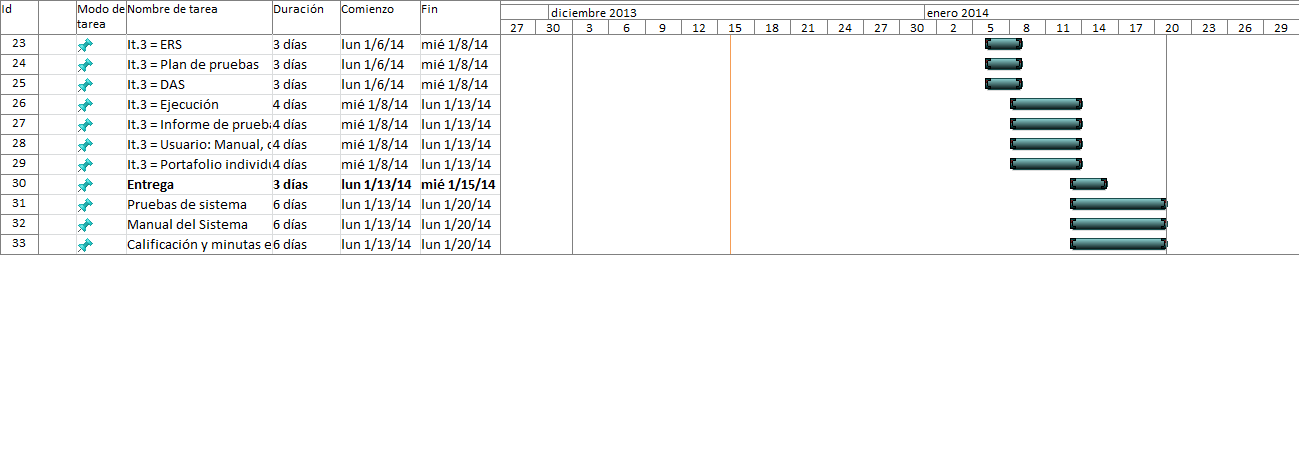
## 

## Plan de proyecto actual (Gantt en Project) (Debe venir en sólo 1 página desglose semanal y no diario)

## 



## Plan de proyecto original (Gantt en Project) (Desglose semanal y no diario)



## Glosario de términos y abreviaturas (orden alfabético)

|  |  |
| --- | --- |
| Abreviatura | Significado |
| CU-001 | Caso de Uso 1. |
| DAS | Documento de Arquitectura de Software. |
| ERS | Especificación de Requerimientos de Software. |
| It. | Iteración. |

## Lista de riesgos (orden descendente de prioridad) Incluir nombre, descrip, prioridad, responsable, etc

|  |  |  |  |
| --- | --- | --- | --- |
| *Riesgo* | *Nivel de riesgo* | *Probabilidad de ocurrencia* | *Plan de mitigación* |
| Problemas de conectividad con el web service de datos de estudiantes. | Medio | Probable | Tener una buena comunicación con el Centro de Cómputo desde el inicio, de manera que esa información sea fácilmente accesible por el sistema. |
| Periodo de vacaciones de los funcionarios del Tec, limitando la comunicación con Centro de Cómputo y Registro | Medio - Alto | Probable | Identificar previamente las actividades que dependen de esas entidades y realizarlas con tiempo. |
| Inconsistencias con los estándares y políticas de desarrollo necesarios. | Medio | Poco probable | Tener claros los estándares de desarrollo del CC antes de trabajar con el mismo. |
| Incompatibilidad entre las herramientas o plataformas utilizadas. | Bajo | Poco probable | Tener definido desde temprano las plataformas y herramientas estándar utilizadas por el CC y por el proyecto. |
| Conocimiento limitado de desarrollo para móviles por parte del equipo de trabajo. | Bajo | Probable | Capacitación previa al desarrollo del app móvil. |
| Fallas en el hardware de la escuela de IC o el centro de cómputo. | Alto | Muy poco probable | Tener una buena comunicación con el CC para conocer sobre el estado actual del equipo, posibles períodos de downtime, entre otros. |

## Descripción de la empresa (departamento) (nombre, dirección, descripción general, organigrama (indicar depto del sistema), persona contacto, email, teléfono)

|  |  |
| --- | --- |
| Institución | Escuela de Ingeniería en Computación |
| Dirección | Tecnológico de Costa Rica |
| Descripción General |  |
| Organigrama |  |
| Persona Contacto | Adriana Álvarez |
| Email | adriana.alvaresf@gmail.com |
| Teléfono |  |

## Especificación de estándares Programación (Ejemplos de Interfaz nivel local y Web, Base de datos, nombres de atributos, clases, etc.)

### Estándar de programación código fuente

#### Distribución del Código

El código fuente debe tener el siguiente orden en su estructura:

* Encabezado.
* Imports.
* Definición del *namespace.*
* Declaración de clase debidamente comentada.
* Constantes: de la visibilidad menos restrictiva a la más restrictiva. Se recomienda crear clases o interfaces únicamente para almacenar constantes.
* Atributos: de la visibilidad menos restrictiva a la más restrictiva Propiedades: de la visibilidad menos restrictiva a la más restrictiva.
* Constructor(es).
* Destructor en caso de que exista.
* Métodos: de la visibilidad menos restrictiva a la más restrictiva.
* Properties (métodos *set* y *get).*

#### Nombres de Proyectos y Soluciones

Los nombres que se le den a las soluciones quedan a criterio del desarrollador, pero debe ser un nombre significativo para el proyecto en desarrollo.

El nombre de la solución debe ser:

**ITCR.Nombre de la solución**

Los nombres de los proyectos deben tener el siguiente formato:

**ITCR.Nombre de la solución + Capa**

* Capa se refiere a la capa en la cual se está trabajando (Base, Negocios, Datos)

Ejemplo:

* ITCR.SeguridadITCR.Datos
* ITCR.Cajas.Negocios

En Visual Studio se desestimó el uso de namespaces para la capa de interfaz en aplicaciones web desde su versión 2005. Por lo que no se incluyen los namespaces .Interfaz en este estándar. Los de Datos y Negocios se siguen utilizando normalmente.

Al definir estos nombres para los proyectos de la solución, los namespaces quedarán definidos.

#### Nombres de Namespaces

El namespace debe tener la siguiente estructura para cualquier clase y capa. Esto ya debe estar definido desde que se creó la solución y los proyectos:

**ITCR. + Sistema.Capa**

* Sistema: nombre del sistema que se está desarrollando.
* Capa: capa en la cual se está trabajando.

Ejemplos:

* ITCR.Compras.Datos
* ITCR.SeguridadITCR.Negocios
* ITCR.Cajas.Negocios

#### Nombres de Clases

Los nombres de las clases deben tener el siguiente formato:

**‘c’ + Nombre descriptivo + Capa**

O bien, para las clases de negocios que requieren nombres sencillos:

**‘c’ + Nombre descriptivo**

* Nombre descriptivo hace referencia al nombre de la clase y su función (la primera letra del nombre debe ser en mayúscula).
* Capa se refiere a la capa en la cual se esta trabajando (Base, Negocios, Datos)

Ejemplo:

* cManejoErroresDatos
* cSeguridadITCRDatos
* cSolicitudNegocios

Se usan sustantivos en singular para los nombres de las clases.

Los nombres de excepciones personalizadas deben terminar con el postfijo “Exception”.

Namespaces or Packages deben seguir el mismo nombre que los folders que contienen el código fuente y en minúscula.

##### Documentación interna de clases

Cada clase debe iniciar con el nombre del proyecto, descripción, fecha y hora de creación y si es necesario observaciones.

Ejemplo:

#region Acerca de...

/////////////////////////////////////////////////////////////////////////////

// Instituto Tecnológico de Costa Rica

// Proyecto: Framework Matricula

// Descripción: Clase de acceso a datos para tabla 'IEPEFDEPTOS'

// Generado por ITCR Gen v1.0.0.0

// Fecha: Martes, 11 de Mayo de 2004, 11:50:00 a.m.

// Dado que esta clase implementa IDispose, las clases derivadas no

// deben hacerlo.

////////////////////////////////////////////////////////////////////////////

#endregion

Los procedimientos que se utilicen en la clase deben ser comentados, así como cualquier elemento que así lo requiera utilizando el estándar de documentación XML de .NET.

Ejemplo:

/// <summary>

/// Propósito: Inicializa los miembros de la clase.

/// </summary>

private void InitClass()

{

// Crea todos los objetos e inicializa otros miembros.

\_conexionBD = new OleDbConnection();

....

}

##### Estándar de la clase base

La clase base no debe modificarse, siempre que se tenga que hacer un cambio debe hacerse en su respectiva clase de datos o negocios. Esto debido a que si se da un cambio en la tabla respectiva en la base de datos se puede remplazar esta clase y los cambios por herencia se verán reflejados en las capas de datos y negocios debido a su relación de herencia.

##### Estándar de la clase de negocios

Debe seguir el estándar para cualquier otra clase y si es necesario debe derivarse de la clase de Datos.

##### Estándar de la clase de datos

Debe seguir el estándar para cualquier otra clase y debe derivarse de la clase Base.

#### Interfaces

Los nombres de las interfaces deben ser adjetivos.

##### Interface Web

WIConPalabrasInternasPrimerLetraEnMayúscula

##### Interface Local

IConPalabrasInternasPrimerLetraEnMayúscula

##### IAccesoADatos

Toda clase debe cumplir con la Interface IaccesoADatos y por lo tanto tener los métodos Insertar, Actualizar, Eliminar, SeleccionarUno, SeleccionarTodos, Buscar. En esta clase se hace uso de un archivo “.config” que contiene el string de conexión.

Ejemplo:

public interface IAccesoADatos

{

bool Insertar();

bool Actualizar();

bool Eliminar();

DataTable SeleccionarUno();

DataTable SeleccionarTodos();

DataTable Buscar();

}

#### Métodos

* Los nombres de métodos deben iniciar con mayúscula y el resto de su nombre debe ir en minúscula, siguiendo el estándar de C#. Ej. Insertar, RegistrarEmpresa.
* Para un método que es un procedimiento utilice un verbo seguido de un objeto, por ejemplo SaveCustomer, MoverFicha, InsertarNodo.
* Para el nombre de una función use una descripción del valor de retorno de la función, por ejemplo NextCustomerId, GetName, RetornarListaAutos.
* Utilice una propiedad en lugar de métodos cuando lo que se va a retornar es un dato de la clase. En el caso de Java use los métodos llamados “setters” y “getters”

##### Parámetros

* Debe seguir el estándar de las variables y agregar la letra ‘p’ antes del underscore (\_). Ejemplo: p\_usrName.
* Un método no debe tener más de 6 parámetros ni ser mayor a 100 líneas de código.

#### Variables

Los nombres de variables de clase privadas tendrán el siguiente formato:

**‘\_’(underscore) + indicador de tipo + Nombre significativo**

* Nombre significativo corresponde a la función específica que se realizara con esa variable y debe iniciar con minúscula. Ejemplo: \_iCodFuncionalidad, \_sNumeroAula.
* No utilice el término “temp” o “index” para nombres de variables, tampoco los nombres de índices i,j,k ni cualquier uso de nombres de una letra o silábicos, en su lugar utilice nombres largos bien descriptivos.
* No utilice nombres ambiguos en variables booleanas, ejemplo: done, status.
* No utilice nombres negativos en las variables como NotFound, NoListado.
* Nunca declare variables de instancia públicas, todas deben ser protected o private.
* Variables locales se escriben con la primer letra en minúscula indicando su tipo sin el “\_” (underscore).

##### Indicadores de tipo

Este indicador debe ser el prefijo para todas las variables en las clases y métodos, los más comunes son:

|  |  |
| --- | --- |
| Tipo | Indicador |
| Integer | i |
| String | s |
| DateTime | da |
| Bit (equivalente a Bool) | b |

***5.5.1.8 Constantes***

* MAYUSCULAS\_SEPARADAS\_POR\_UNDERSCORE
* Siempre utilice constantes en lugar de literales hard-coded.
* Hay ciertos casos en los que no es necesario declarar una constante para ciertas literales hardcode, tales se muestran en la siguiente tabla:

|  |  |  |
| --- | --- | --- |
| Nombre | Descripción | Ejemplo |
| Cero | Use el valor cero cuando inicializa índices, variables o requiere comparar el largo o cantidad de elementos de una colección | * Variable = 0; * for (int IndeXYZ=0; …. * AnyKindCollection.Length > 0 * AnyKindCollection.Length < 0 * AnyKindCollection.Length == 0 * AnyKindCollection.Length >= 0 * AnyKindCollection.Length =< 0   Lo anterior aplica para arrays, arraylist, collections, hashtable, datarows, etc cuando usan las propiedades o métodos Length, Size, Count |
| * Length-1 * Size-1 | Algunos Collections requieren algún ajuste de tamaño para hacer comparaciones o para ajustarse a algún algoritmo | for(int indexXYZ=0;indexXYZ<array.Length-1… |
| Inicio de índice  1,2,3 | Cuando un ciclo inicia en los valores 1, 2, 3 | for (int indexXYZ=1; indexXYZ<array.length …. |
| Evaluación par o impar | Cuando se requiere saber si un número es par o impar | Cuando se requiere saber si un numero es par o impar |

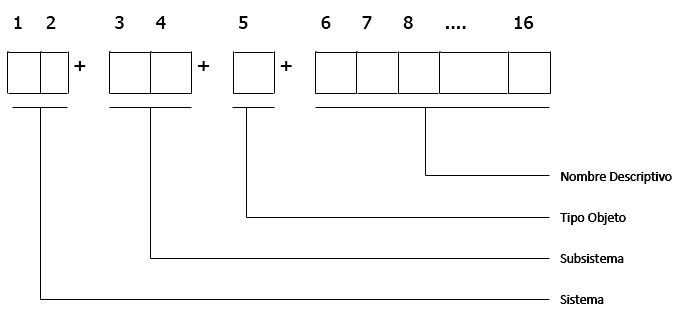
#### Properties

* Deben iniciar con minúscula indicado su tipo y seguir la el estándar de C#. Ej. sNombrePersona, iTipoSolicitud
* Si el lenguaje no soporta properties utilice los métodos de Set y Get que sean necesarios.

### Estándar de objetos de la base de datos

A continuación se describen las normas generales que se deben utilizar para definir los nombres de los objetos incluidos en la base de datos de un sistema, esto incluye nombres de tablas y vistas.

Para asignar el nombre a objetos de este tipo se debe utilizar letras mayúsculas de acuerdo con el siguiente esquema:



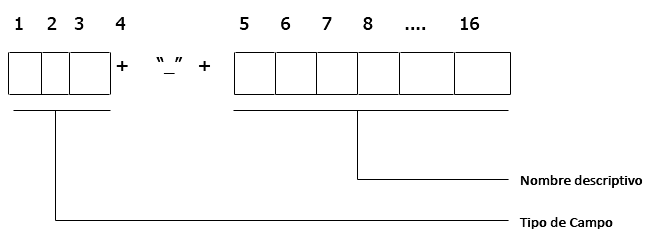
* Sistema: Corresponde a una tira de dos caracteres que identifica el sistema al que pertenece el objeto.
* Subsistema: Corresponde a una tira de dos caracteres que identifica el subsistema al que pertenece el objeto.
* Tipo Objeto: Si es una tabla puede ser “F” (datos fijos) o “T” (temporal). Si es otro tipo de objeto se utiliza lo siguiente: “V” para vistas.
* Nombre descriptivo: es una tira de caracteres que permite identificar la naturaleza del objeto.

Ejemplos:

* SGTPFSOLICITUDES (Sistemas de Servicios Generales, Sistema del Taller de Publicaciones, tabla fija de solicitudes).
* IEPEFDEPTOS (Información Estudiantil, Planes de Estudio, tabla fija de departamentos).

#### Nombres de los campos

Para asignar el nombre a los campos en cada tabla de la base de datos, se debe utilizar el siguiente esquema:



* Tipo de Campo: corresponde a una tira de caracteres que indica el uso del campo de acuerdo a la siguiente tabla:

|  |  |  |
| --- | --- | --- |
| Código | Tipo de Campo | Ejemplo |
| Id, Ide | Identificador | ID\_USUARIO |
| Can | Cantidad | CAN\_PAGINAS |
| Cod | Código | COD\_DEPTO |
| Dir | Dirección | DIR\_HABITACION |
| Dsc | Descripción | DSC\_SOLICITUD |
| Fec | Fecha | FEC\_INGRESO |
| Img | Imagen | IMG\_EMPLEADO |
| Int | Interés | INT\_ANUAL |
| Mon | Monto | MON\_SALDO |
| Nom | Nombre | NOM\_EMPLEADO |
| Num | Número | NUM\_CEDULA |
| Por | Porcentaje | POR\_NOTA |
| Txt | Texto | TXT\_EMAIL |

* Nombre descriptivo: es una tira de caracteres que permite identificar la naturaleza del campo.

#### Nombres de procedimientos almacenados

Los procedimientos deben nombrarse iniciando con “pr\_” seguido del nombre descriptivo del procedimiento.

Ejemplo:

* pr\_ConsultaSalario
* pr\_ObtenerDepartamentoEmpleado
* pr\_ReporteAnual

Si el procedimiento es de mantenimiento, tal como insertar, actualizar, eliminar, etc. para una tabla, el nombre del procedimiento debe ser “pr\_” seguido del nombre de la tabla, un underscore y el nombre de la función que realiza, las cuales son:

* Insertar
* Eliminar
* Actualizar
* Buscar
* SeleccionarUno
* SeleccionarTodos

Ejemplos:

* pr\_SGTPFSOLICITUD\_Insertar
* pr\_SGTPFCENTROCOSTO\_SeleccionarTodos

### Estándar de programación de la base de datos (SQL)

#### Estándar para procedimientos almacenados

* Las palabras reservadas deben escribirse en mayúscula.
* Los nombres de tablas se rigen por los estándares de diseño establecidos.
* Los nombres de procedimientos almacenados definidos por el usuario deben seguir el siguiente formato:

**‘pr’ + \_ (underscore) + objeto sobre el cual trabaja + nombre de la función que realiza**

Ejemplo:

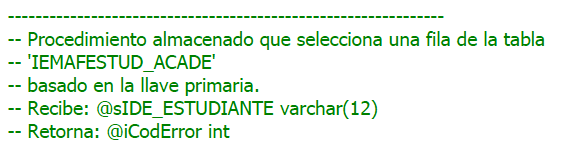
* pr\_IEMAFESTUD\_ACADE\_Insertar

En caso de que el procedimiento actúe sobre varios objetos de la base de datos al mismo tiempo, entonces puede omitirse el nombre del objeto.

Ejemplo:

* pr\_GenerarIndice
* Cada procedimiento almacenado debe contener como comentario un encabezado con la siguiente estructura:
* Nombre del procedimiento
* Función que realiza
* Fecha y hora de creación
* Precondiciones
* Postcondiciones (debe incluir los valores de retorno y su descripción)
* Lista de parámetros y su descripción

Ejemplo:



* Declaración de parámetros: debe declararse un parámetro por línea acompañados por su respectivo tipo. Ejemplo:

CREATE PROCEDURE dbo.pr\_IEMAFESTUD\_ACADE\_Insertar

@siCOD\_CLA\_ACC smallint,

@sNOM\_ESTUDIANTE varchar(40),

@sIDE\_SEDE varchar(2)

AS

INSERT dbo.IEMAFESTUD\_ACADE

(

COD\_CLA\_ACC,

NOM\_ESTUDIANTE,

IDE\_SEDE

)

VALUES

(

@siCOD\_CLA\_ACC,

@sNOM\_ESTUDIANTE,

@sIDE\_SEDE

)

* Para cualquier mandato SQL, la cláusula inicial funcionará como encabezado y cualquier otra cláusula del mandato tendrá una identación de 3 caracteres con respecto a la cláusula inicial. Ejemplo:

SELECT <Lista Campos>

FROM <Lista Tablas>

WHERE <Condiciones\_de\_Seleccion>