**IMPLEMENTACIÓN DE APLICACIÓN PARA LA GESTIÓN DE PROYECTOS PARA LA OFICNA DE PROYECCIÓN SOCIAL Y EXTENSIÓN UNIVERSITARIA DE LA FACULTA DE INGENIERÍA CIVIL, SISTEMAS Y ARQUITECTURA DE LA UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO**

**ALUMNOS:**

**TORRES BACA, JONATHAN FRANCHESCO**

**VERA OSORIO, RAQUEL PILAR**

**ASESOR:**

**ING. DÁVILA HURTADO, LUIS ALBERTO**

**FECHA INICIO:**

**10/05/2011**

**FECHA FIN:**

**16/05/2011**

**UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO**

**FACULTAD DE INGENIERIA CIVIL, SISTEMAS Y ARQUIITEECTURA**

**OFICINA DE PROYECCIÓN SOCIAL Y EXTENSIÓN UNIVERSITARIA**

**I.- NOMBRE DE LA INSTITUCIÓN**

OFICINA DE PROYECCIÓN SOCIAL Y EXTENSIÓN UNIVERSITARIA DE LA FACULTA DE INGENIERIA CIVIL, SISTEMAS Y ARQUITECTURA DE LA UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO

**II.- UBICACIÓN Y LUGAR A REALIZARSE LA ACTIVIDAD**

DEPARTAMENTO: LAMBAYEQUE

PROVINCIA: LAMBAYEQUE

DISTRITO: LAMBAYEQUE

DIRECCIÓN: AV. JUAN PABLO XXIII 339

**III.- MAGNITUD DEL PROYECTO – ALCANCE**

La **oficina de Proyección Social y Extensión** Universitaria de la Facultad de Ingeniería civil, Sistemas y Arquitectura de la Universidad Nacional Pedro Ruiz Gallo, tiene la necesidad de registrar el histórico de Proyecciones Sociales de docentes y estudiantes, así como la generación de reportes generados con ésta información.

En vista de lo anteriormente descrito, se desarrolla la aplicación **Sistema de Proyección Social SPS**  con el fin de dar soporte a estas necesidades.

SPS tiene las siguientes funcionalidades:

* Registro de Estudiante
* Registro de Docentes
* Registro de Decano actual
* Registro de documentos (Oficio y Expedientes)
* Registro de Año en curso
* Registro de Estados de Proyecto
* Registro de Facultades
* Registro de Escuelas
* Registros de Empresas Solicitantes
* Registro de Proyectos
* Búsquedas de Estudiantes, Docentes, Solicitantes y Proyectos
* Generación de Reportes de Proyectos por Estado en un rango de fechas determinado
* Generación de Reportes de Proyectos por Docente en un rango de fechas determinados
* Generación de Reporte con la información de un Proyecto.

**IV.- NOMBRE DEL SOLICITANTE**

OFICINA DE PROYECCIÓN SOCIAL Y EXTENSIÓN UNIVERSITARIA – FICSA

NOMBRE RESPONSABLE:

TELEFONO:

CORREO:

**IV.- DESCRIPCIÓN DE LA ACTIVIDAD**

**INTRODUCCIÓN**

Una adecuada Gestión de la Información hace que los procesos y tareas se realicen de una forma oportuna y eficiente.

En la actualidad el uso de tecnología de información es elemental para el desarrollo de las instituciones ya que permite optimizar recursos financieros, humanos y materiales.

La Oficina de Proyección Social y Extensión Universitaria PSyEU, tiene la necesidad de optimizar sus procesos, es por eso que se ha desarrollado la Aplicación de Gestión de Proyectos Sociales, quien dará soporte a las actividades fundamentales de la oficina, ésta aplicación pretende organizar y controlar toda la información de los Proyectos Sociales realizado tanto por los Docentes como por los Estudiantes.

**4.1 DATOS GENERALES DEL SOLICITANTE**

Proyección Social, es una función básica de la Universidad  mediante la cual se transfieren y extienden los conocimientos de la cultura universal y los generados por la investigación a través de los servicios profesionales educativos y de promoción, a quienes no son integrantes de la Comunidad Universitaria con el fin de orientar el desarrollo de las organizaciones sociales ubicadas dentro del ámbito de influencia. El fomento de la creación intelectual y artística, así como la formación axiológica es parte de su responsabilidad

La FICSA está conformada por escuelas profesionales

* Escuela Profesional Ingeniería civil
* Escuela profesional de Ingeniería de Sistemas
* Escuela Profesional Arquitectura

**4.1.1 MISION.**

Somos una Facultad con profesionales idóneos formado por docentes, alumnos,  personal administrativos que ponemos en práctica las acciones necesarias que permiten coadyuvar el desarrollo  departamental y nacional. Poner al servicio de la comunidad, realizando tareas de investigación multidisciplinarias, por medio de la  Proyección Social  que contribuyan al desarrollo del país y  mejorar la calidad de  ida de sus habitantes.

**4.1.2. VISION**

La FICSA, se proyecta a ser agente de cambio,  a fin de lograr el desarrollo humano sostenible y asumiendo el liderazgo académico en el departamento de Lambayeque y la región, con el cambio de política que contribuyen al desarrollo de las personas e instituciones. Este tipo de visión conlleva la participación plena de los integrantes, Liderazgo de sus Órganos de Gobierno, formación de profesionales capaces de poner en práctica los conocimientos impartidos a concretar una atención a las personas, organizaciones públicas y empresariales

**4.1.3 OBJETIVOS**

* Contribuir en el desarrollo económico - social de la población, a través de la oferta de los servicios profesionales y educativos eficientes, competitivos y sustentable.
* Fortalecer la posición competitiva, con un valor agregado a la comunidad, ofreciéndoles una mayor calidad de servicio.
* Difundir y promover el desarrollo de la cultura regional y nacional, mediante la coordinación de las actividades académicas, artísticas y deportivas que  se fomentan.
* Dar  la confianza de las organizaciones sociales, económicas, culturales y otros de la Región y del país.

**4.1.4 PERSONAL**

Arq. José Baltasar Flores Mino.    Jefe de Oficina

Sra. María Julia Morales Alache. Secretaria (Actualmente con licencia por salud)

Admr. Neg.  Juan Manuel Tafur Infante. Promotor Social

**4.2 OBJETIVOS DEL PROYECTO**

**4.2.1 Generales**

Implementar una Aplicación que dé soporte a los procesos realizados por la Oficina de Proyección Social y Extensión Universitaria **PSyEU.**

Optimizar los tiempos en la elaboración de Reportes solicitados por decanato

**4.2.2 Específicos**

* Registro de Estudiante
* Registro de Docentes
* Registro de Decano actual
* Registro de documentos (Oficio y Expedientes)
* Registro de Año en curso
* Registro de Estados de Proyecto
* Registro de Facultades
* Registro de Escuelas
* Registros de Empresas Solicitantes
* Registro de Proyectos
* Búsquedas de Estudiantes, Docentes, Solicitantes y Proyectos
* Generación de Reportes de Proyectos por Estado en un rango de fechas determinado
* Generación de Reportes de Proyectos por Docente en un rango de fechas determinados
* Generación de Reporte con la información de un Proyecto.

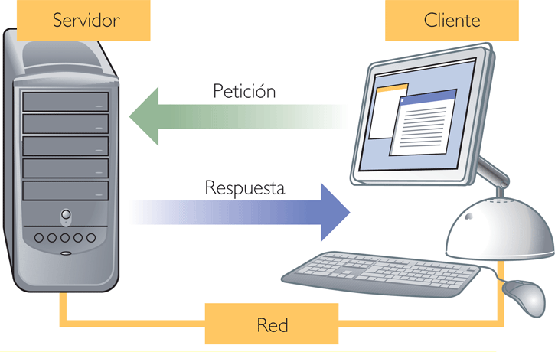
**4.3 RECUROS**

**4.4 IMPORTANCIA**

* Acceso rápido a la información.
* Generación de informes, que permiten corregir fallas difíciles de detectar y controlar con un sistema manual.
* Evitar pérdida de tiempo recopilando información que ya está almacenada en bases de datos que se pueden compartir.
* Aumento de la productividad gracias a la liberación de tiempos en búsqueda y generación de información repetida.

**4.5 MARCO TEÓRICO**

**4.5.1 ARQUITECTURA CLIENTE – SERVIDOR**

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los *sistemas multicapa* en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

**En la arquitectura C/S el remitente de una solicitud es conocido como cliente. Sus características son:**

* Es quien inicia solicitudes o peticiones, tienen por tanto un papel activo en la comunicación (dispositivo maestro o amo).
* Espera y recibe las respuestas del servidor.
* Por lo general, puede conectarse a varios servidores a la vez.
* Normalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

**Al receptor de la solicitud enviada por cliente se conoce como servidor. Sus características son:**

* Al iniciarse esperan a que lleguen las solicitudes de los clientes, desempeñan entonces un papel pasivo en la comunicación (dispositivo esclavo).
* Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente.
* Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).
* No es frecuente que interactúen directamente con los usuarios finales.

**4.5.2. PROGRAMACIÓN ORIENTADA A OBJETOS**

La programación orientada a objetos, tal vez el paradigma de programación más utilizado en el mundo del desarrollo de software y de la ingeniería de software del siglo XXI, trae un nuevo enfoque a los retos que se plantean en la programación estructurada cuando los problemas a resolver son complejos. Al contrario que la programación procedimental que enfatiza en los algoritmos, la POO enfatiza en los datos. En lugar de intentar ajustar un problema al enfoque procedimental de un lenguaje, POO intenta ajustar el lenguaje al problema. La idea es diseñar formatos de datos que se correspondan con las características esenciales de un problema.

La idea fundamental de los lenguajes orientados a objetos es combinar en una única unidad o módulo, tanto los datos como las funciones que operan sobre esos datos. Tal unidad se llama un objeto.

Los datos de un objeto, se conocen también como atributos o variables de instancia. Si se desea leer datos de un objeto, se llama a una función miembro del objeto. Se accede a los datos y se devuelve un valor. No se puede acceder a los datos directamente. Los datos son ocultos, de modo que están protegidos de alteraciones accidentales. Los datos y las funciones se dice que están encapsulados en una única entidad. El encapsulamiento de datos y la ocultación de los datos son términos clave en la descripción de lenguajes orientados a objetos.

Si se desea modificar los datos de un objeto, se conoce exactamente cuáles son las funciones que interactúan con la función miembro del objeto. Ninguna otra función puede acceder a los datos. Esto simplifica la escritura, depuración y mantenimiento del programa. Un programa C++ se compone normalmente de un número de objetos que se comunican unos con otros mediante la llamada a otras funciones miembro.

Existen diversas características ligadas a la orientación a objetos. Todas las propiedades que se suelen considerar, no son exclusivas de este paradigma, ya que pueden existir en otros paradigmas, pero en su conjunto definen claramente los lenguajes orientados a objetos. Estas propiedades son:

• Abstracción (tipos abstractos de datos y clases).

• Encapsulado de datos.

• Ocultación de datos.

• Herencia.

• Polimorfismo

**Objetos**

El objeto es el centro de la programación orientada a objetos. Un objeto es algo que se visualiza, se utiliza y juega un rol o papel. Si se programa con enfoque orientado a objetos, se intentan descubrir e implementar los objetos que juegan un rol en el dominio del problema y en consecuencia programa. La estructura interna y el comportamiento de un objetivo, en una primera fase, no tiene prioridad. Es importante que un objeto tal como un carro o una casa juegan un rol.

Dependiendo del problema, diferentes aspectos de un aspecto son relevantes. Un carro puede ser ensamblado de partes tales como un motor, una carrocería, unas puertas o puede ser descrito utilizando propiedades tales como su velocidad, su kilometraje o su fabricante. Estos atributos indican el objeto.

De modo similar una persona, también se puede ver como un objeto, del cual se disponen de diferentes atributos. Dependiendo de la definición del problema, esos atributos pueden ser el nombre, apellido, dirección, número de teléfono, color del cabello, altura, peso, profesión, etc.

Un objeto no necesariamente ha de realizar algo concreto o tangible. Puede ser totalmente abstracto y también puede describir un proceso. Por ejemplo, un partido de baloncesto o de rugby puede ser descrito como un objeto. Los atributos de este objeto pueden ser los jugadores, el entrenador, la puntuación y el tiempo transcurrido de partido.

**Clases**

En POO los objetos son miembros de clases. En esencia, una clase es un tipo de datos al igual que cualquier otro tipo de dato definido en un lenguaje de programación. La diferencia reside en que la clase es un tipo de dato que contiene datos y funciones. Una clase contiene muchos objetos y es preciso definirla, aunque su definición no implica creación de objetos.

**4.5.3 TECNOLOGÍA JAVA**

Compuesta por 2 partes:

- El lenguaje de programación, y  
- La plataforma

**Características más importantes son**:

* Lenguaje orientado a objetos.
* Java es un lenguaje sencillo.
* Independiente de plataforma
* Brinda un gran nivel de seguridad
* Capacidad multihilo
* Gran rendimiento
* Creación de aplicaciones distribuidas
* Su robustez o lo integrado que tiene el protocolo TCP/IP lo que lo hace un lenguaje ideal para Internet.

Tradicionalmente se han dividido los lenguajes en compilados e interpretados. Los primeros necesitan ser traducidos por un programa llamado compilador al lenguaje máquina, que es el que entiende el ordenador. Como ejemplo de estos lenguajes podríamos citar a C, C++, Visual Basic, Clipper, etc. Los interpretados, en cambio, son traducidos mientras se ejecutan, por ejemplo HTML, WML o XML, por lo cual no necesitan ser compilados.

Así pues la diferencia entre estos lenguajes radica en la manera de ejecutarlos. Mientras que los compilados sólo se compilan una vez y lo hacen pasando todo el programa a código máquina (si da un error aunque sea en la última línea no podríamos ejecutar nada de nada), en el momento que lo hemos compilado correctamente se genera un archivo .exe que se puede ejecutar tantas veces como queramos sin tener que volver a compilar. Los interpretados en cambio, cada vez que los queramos ejecutar tendremos que interpretarlos línea a línea, es más lento, pero puede ocurrir un error en la última línea y a diferencia de los compilados, el programa se ejecuta justo hasta la línea que produce el error.

Java está diseñado para que un programa escrito en este lenguaje sea ejecutado independientemente de la plataforma (hardware, software y sistema operativo) en la que se esté actuando. Esta portabilidad se consigue haciendo de Java un lenguaje medio interpretado medio compilado. ¿Cómo se come esto? Pues se coge el código fuente, se compila a un lenguaje intermedio cercano al lenguaje máquina pero independiente del ordenador y el sistema operativo en que se ejecuta (llamado en el mundo Java bytecodes).

Finalmente, se interpreta ese lenguaje intermedio por medio de un programa denominado máquina virtual de Java (JVM), que sí depende de la plataforma.

Los java bytecodes permiten el ya conocido “write once, run anywhere” (compila una sola vez y ejecútalo donde quieras). Podemos compilar nuestros programas a bytecodes en cualquier plataforma que tenga el compilador Java. Los bytecodes luego pueden ejecutarse en cualquier implementación de la máquina virtual de Java (JVM). Esto significa que mientras el ordenador tenga un JVM, el mismo programa escrito en Java puede ejecutarse en Windows, Solaris, iMac, Linux, etc.

**La plataforma Java**

Con plataforma nos referimos al ambiente de hardware y software en donde el programa se ejecuta, por ejemplo, plataformas como Linux, Solaris, Windows 2003 y MacOS. En casi todos los casos las plataformas son descritas como la combinación del sistema operativo y el hardware. La plataforma Java se diferencia de estas plataformas, es que es una plataforma sólo de software y se ejecuta sobre las otras plataformas de hardware.

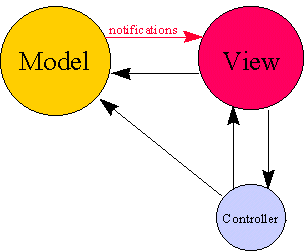
**La plataforma Java tiene 2 componentes:**

* La máquina virtual de Java (JVM)
* El Java API (Application Programming Interface)

La máquina virtual de Java (JVM); es la base de la plataforma Java y es llevada a diferentes plataformas de hardware.

El Java API es una gran colección de componentes de software que proporcionan muchas utilidades para el programador, por ejemplo, los API’s para las interfaces gráficas. Los API’s de Java están agrupados en librerías de ciertas Clases e interfaces, estas librerías son conocidas como paquetes.

El siguiente gráfico describe un programa que se está ejecutando sobre la plataforma Java. Como vemos, el Java API y la máquina virtual aíslan al programa del hardware.

**4.5.4 MODELO VISTA CONTROLADOR**

MVC (Modelo-Vista-Controlador), es la forma (Patrón de Diseño) que utilizan los desarrolladores para implementar aplicaciones, además permite separar nuestra aplicación en un modelo, una vista y con controlador, este patrón fue introducido por primera vez en el lenguaje “Smalltalk”.

**3 Tipos de entidades, cada una especialidad en su tarea.**

**Modelo:**

Es el encargo de administrar la lógica de la aplicación, tiene como finalidad servir de abstracción de algún proceso en el mundo real, además tiene acceso a la Base de Datos, agregando q tiene las funciones que controlan la integridad del sistema.

**Vista**

Sencillamente es la representación visual del modelo. Es la encargada de representar los componentes visuales en la pantalla, Esta asociada a un Modelo, esto le permite que al momento de cambiar el Modelo, la vista redibujara la parte efectada para reflejar los cambios.

**Controlador**

Es el escuchador a los eventos que genere el usuario, es decir es el que permite que interactúe el usuario con el sistema. Interpreta los eventos (las entradas) a través del teclado y/o ratón.

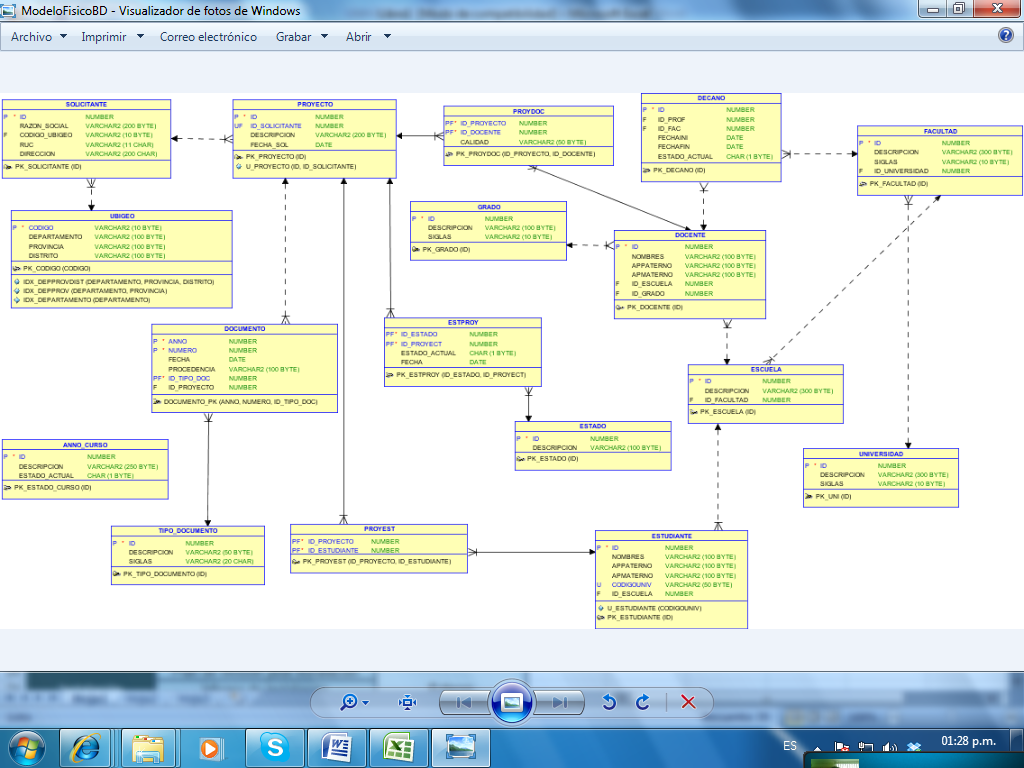
***MVC***  fue diseñada para reducir el esfuerzo al momento de programar. Además porque permite una clara separación entre los componentes de un programa; lo cual nos permite implementarlos por separado.

**4.6 CRONOGRAMA DE ACTIVIDADES**

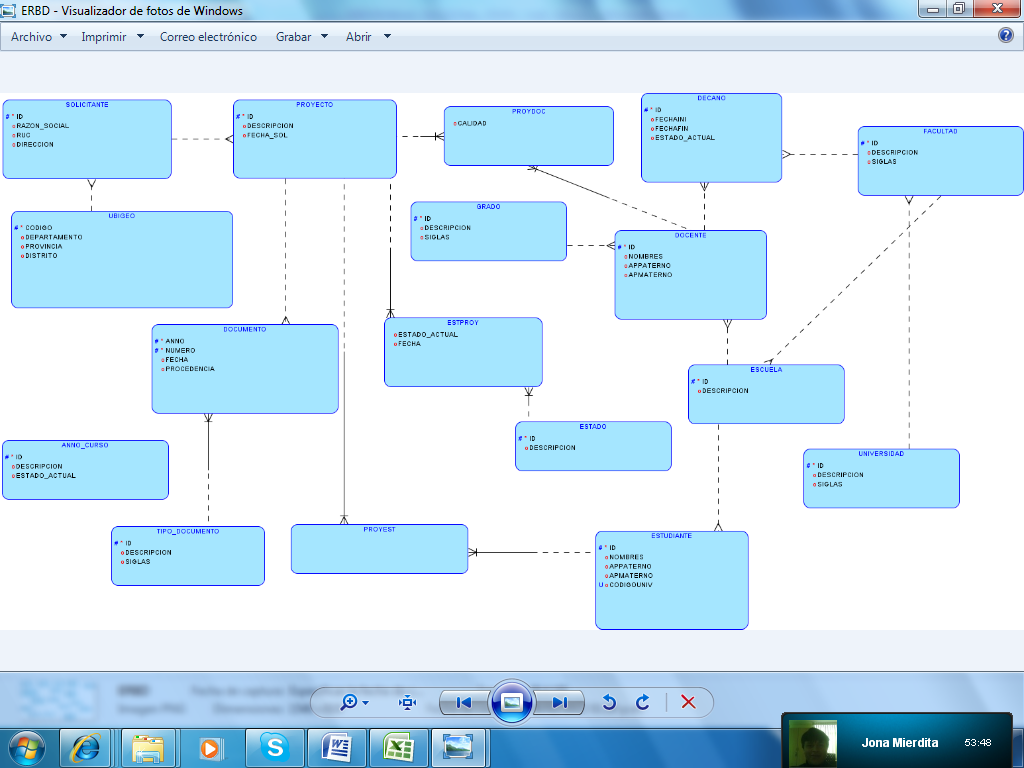
|  |  |  |
| --- | --- | --- |
| **FASE** | **DESCRIPCION** | **FECHA** |
| Estudio de Viabilidad | Descripción breve del sistema y sus características | Mayo 2011 - Junio 2011 |
| Descripción de las necesidades del negocio en el sistema propuesto |
| Propuesta de organización del equipo de desarrollo y definición de responsabilidades |
| Estudio de los beneficios que producirá el sistema |
| Análisis | Captura de Requisitos | Julio 2011 - Agosto 2011 |
| Análisis de la situación actual |
| Requisitos nuevos de los usuarios |
| Requisitos de datos |
| Diseño | Elaboración del Modelo de datos | Septiembre 2011 - Octubre 2011 |
| Elaboración de Procedimientos |
| Estándares de programación y diseño de programas, recomendados. |
| Técnica de implementación: Codificación propia |
| Codificación | Descripción detallada de la lógica | Noviembre 2011 - Enero |
| Descripción de las Entradas y Salidas (ficheros, pantallas, listados, etc.) |
| Pruebas | Plan de sistema de Pruebas | Enero 2011 - Febrero 2011 |
| Informe del resultado de pruebas |
| Resultados de las pruebas |
| Instalación | Plan de revisión post instalación | Febrero 2011 |
| Informa de Instalación |
| Carta de aceptación de sistema |

**4.7 DISEÑO DE BASE DE DATOS**

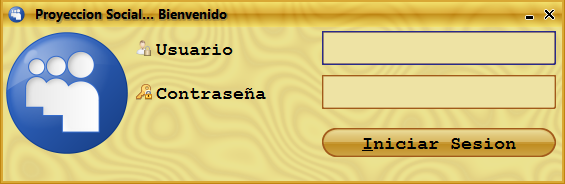
**4.7.1 MODELO FÍSICO DE BASE DE DATOS**

****

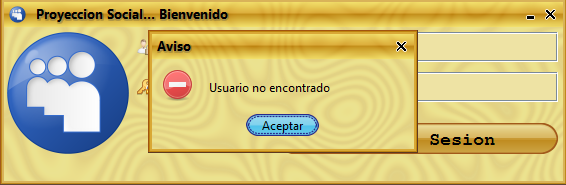
**4.7.2 ENTIDAD RELACION**

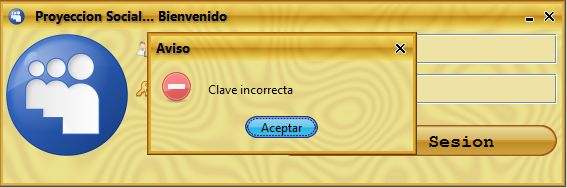


**4.8 INTERFACES GRÁFICAS**

Para efectos de seguridad, la autentificación del usuario es un factor importante, es por eso que la primera interfaz es la de loggeo.

En el caso que el usuario no esté validado, el sistema enviará un mensaje de error.

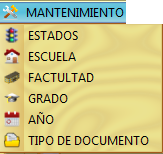


En el caso que la clave sea incorrecta, el sistema enviará un mensaje de error.

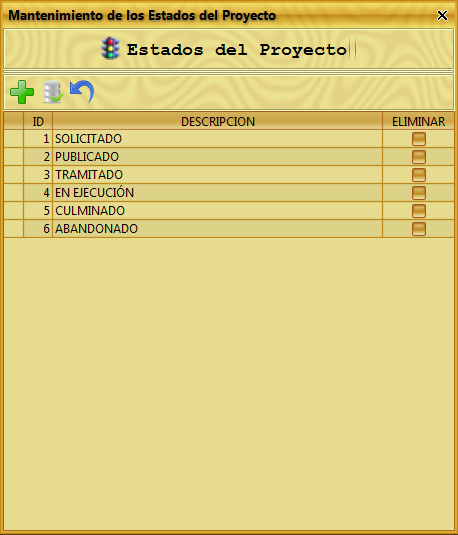
Al iniciar sesión correctamente, se mostrará la siguiente interfaz. Como se puede observar la aplicación cuenta con tres módulos: Mantenimiento, Participantes y Proyectos.



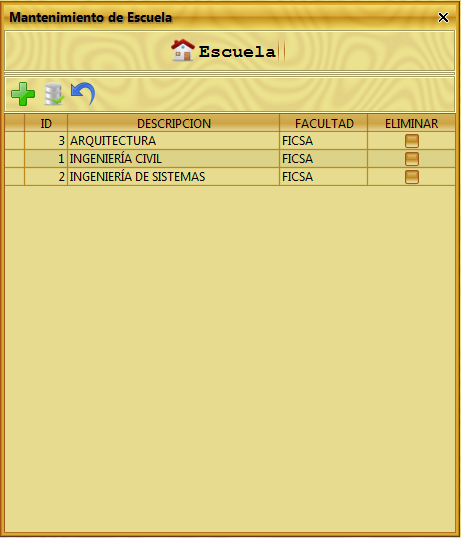
* Con el módulo **mantenimiento** se puede registrar, actualizar, eliminar:

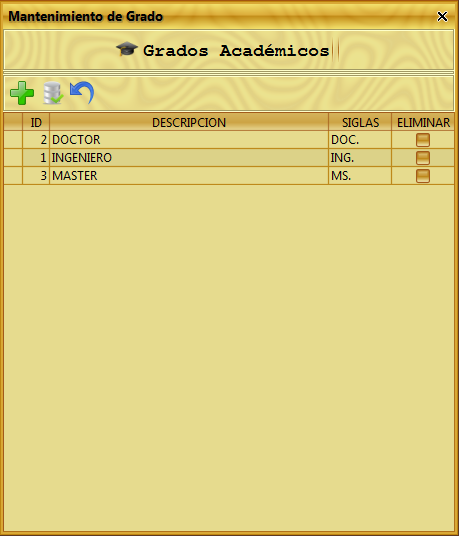


* **ESTADOS,** son los distintos estados por los cuáles pasa un proyecto, esto servirá para poder hacerle seguimiento al proyecto y determinar en qué estado se encuentra en el tiempo.

****

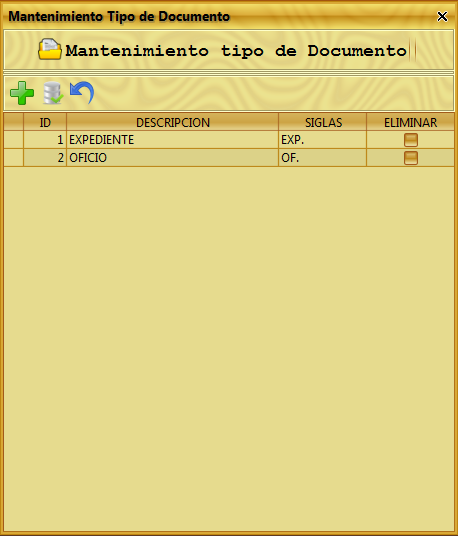
* **FACULTAD Y ESCUELA,**  para determinar a Facultad y Escuela pertenece el proyecto. Para crear una Escuela, primero se debe de crear la Facultad a la cual pertenece, en este caso primero se creó la Facultad FICSA, para luego agregarle las escuelas pertenecientes a ella.

****

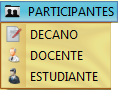
* **GRADO,**  para definir el grado del Docente, ya se Ingeniero, Arquitecto, Máster, Doctor, etc.
* **AÑO,**  para definir el nombre del año en curso, para que al momento de realizar los reportes en la parte superior se imprima el nombre del año.



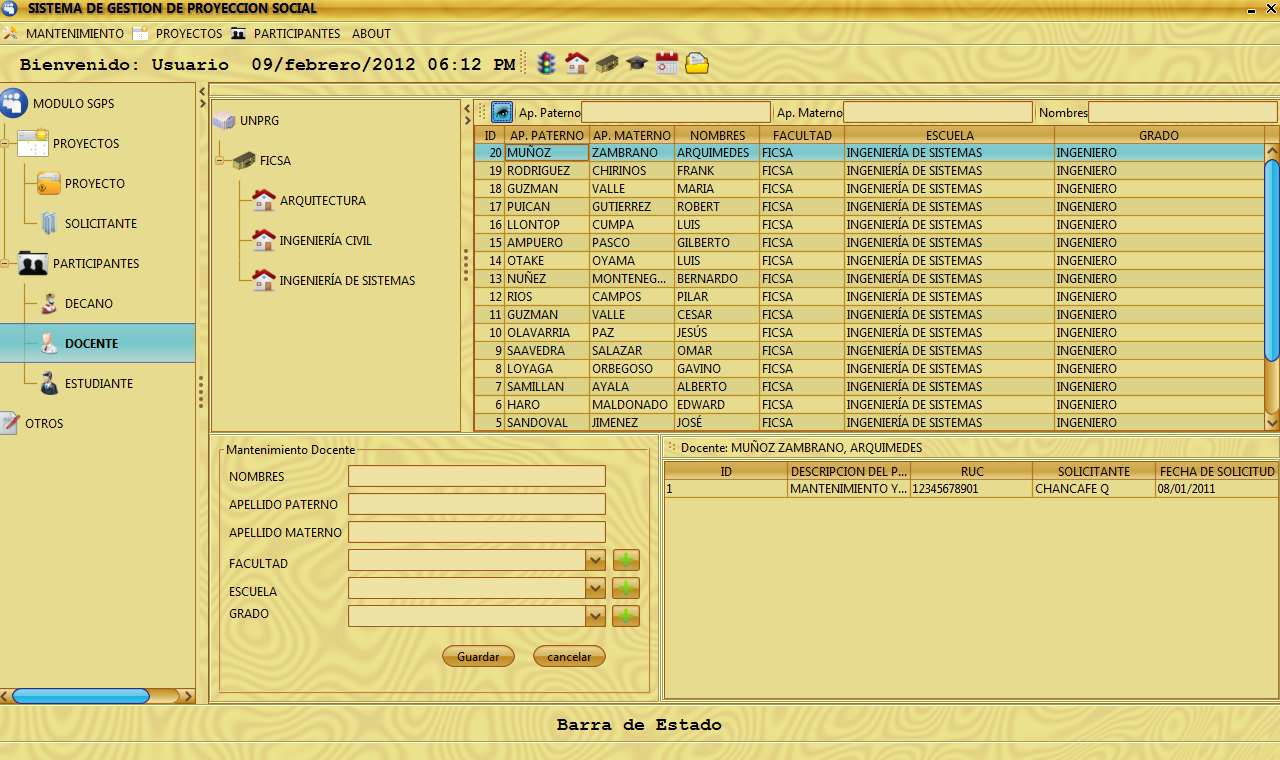
* **DOCUMENTO,**  para determinar el tipo de documento que se recibe o que se envía.



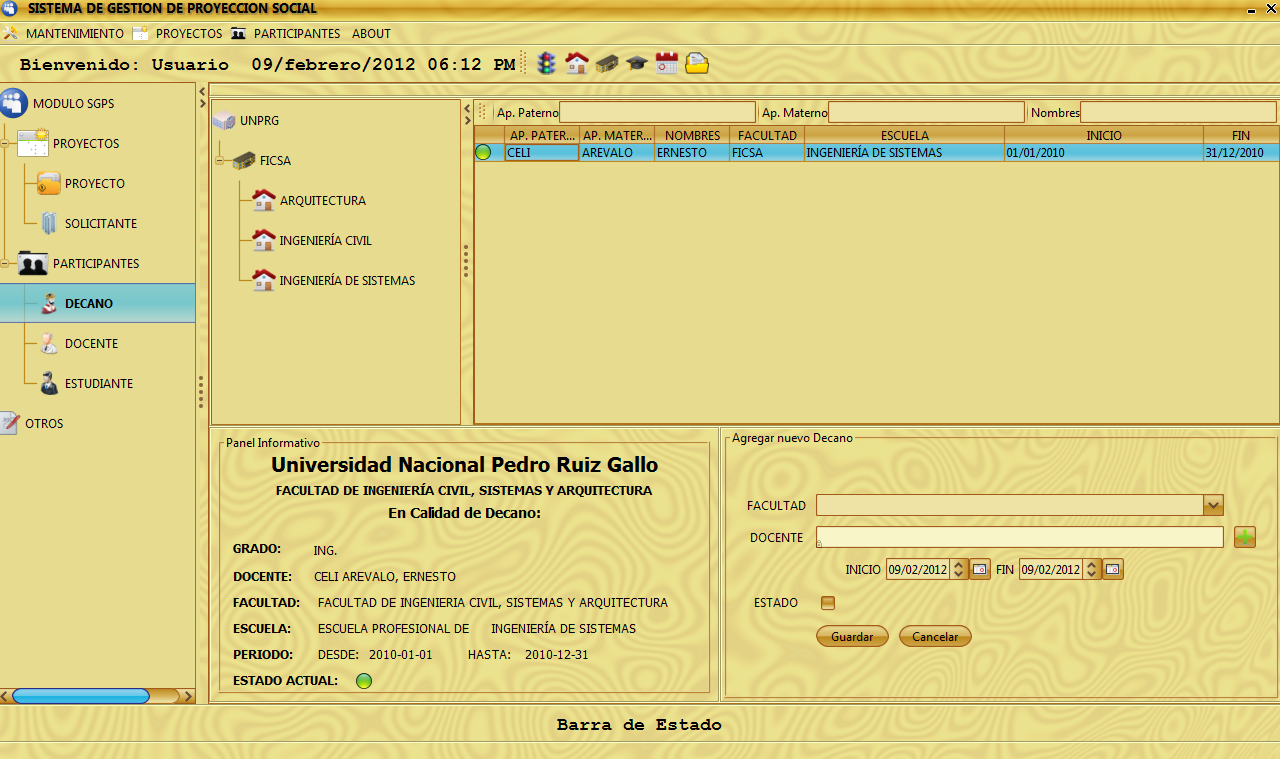
* Con el módulo participantes se le dará soporte a:



* **DOCENTE,** se registra al docente, nombre, la facultad y la escuela a la cual pertenece, en la parte izquierda inferior se listan los proyectos del docente seleccionado en la tabla.



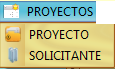
* **DECANO,** para determinar el docente que un determinado tiempo ejerce la función de decano.

****

* **ESTUDIANTE,** se registra al estudiante, nombre, la facultad y la escuela a la cual pertenece, en la parte izquierda inferior se listan los proyectos del estudiante seleccionado en la tabla.



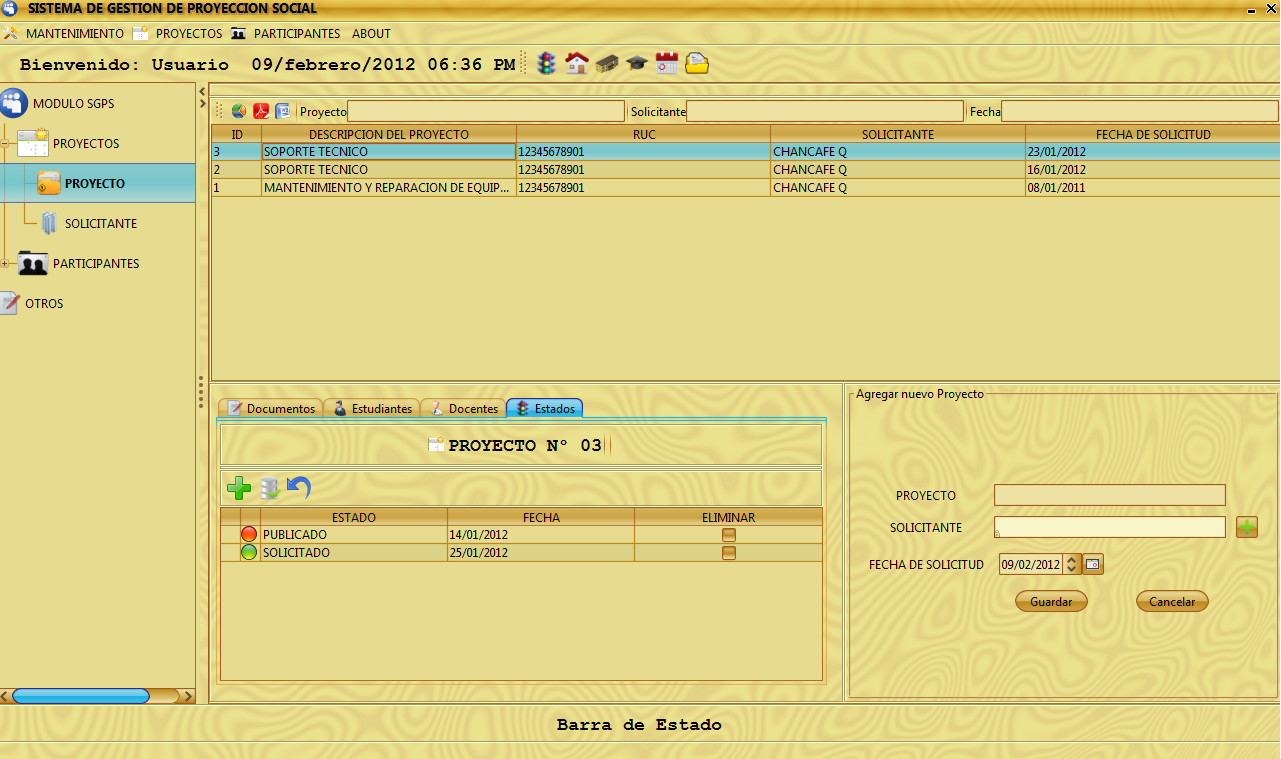
* En el módulo **Proyectos** de determinará quienes son los participantes, quien es el solicitante, el título del proyecto, asi como se actualizaran los estados de acuerdo a las distintas fases.



* **SOLICITANTE,** se registra a la entidad que está solicitando la realización de un proyecto.



* **PROYECTO,** para registra un proyecto, se debe antes primero haber registrado a los participantes y la entidad que solicita la ejecución del proyecto.



**4.9 REQUERIMIENTOS DEL SISTEMA**

* Para el adecuado funcionamiento de los sistemas se necesita:
* Base de datos Oracle 9i/10g/11g
* Entorno de Ejecución Java, específicamente el JDK 7
* Office 2007 o superior / Open Office últimas versiones, para poder visualizar los reportes correctamente.
* Acceso a la red de área local o en su defecto instalar el administrador de base de datos en el equipo del usuario.
* En el caso de instalar el administrador de base de datos en el equipo de usuario se necesitará un ordenador con las siguientes características:
  + - * Procesador de 32 bits
      * Memoria RAM mínimo de 2Gb
      * Disco Duro mínimo 80Gb
      * SO Windows XP/7/Vista, Ubuntu 11.X o Centos 6.X

**4.10 RECOMENDACIONES**

* Generar un nuevo instalador con las modificaciones correspondientes al nombre de base de datos, usuario de base de datos y contraseña.
* Para configurar la aplicación y la base de datos recomendamos ver el video en

YouTube:

* http://youtu.be/U9TS7BSaQII
* http://youtu.be/U9TS7BSaQII
* http://wilbercalles.tripod.com/impyben.html
* <http://java.ciberaula.com/articulo/tecnologia_java/>
* Al momento de instalar la base de datos y crear el nombre de usuario de la base de datos, la cual albergará todos los objetos necesarios para el correcto funcionamiento, se debe de cambiar en el código fuente el usuario y la contraseña de la clase ObjetoConexión.